Neural Network Potentials

Mark DelloStritto

1 Introduction

Machine learning refers to a diverse range of methods which focus on discovering free parameters of a model using automated, typically gradient-based methods. While machine learning encompasses methods as simple as least-squares fitting of linear functions, colloquially it generally refers to more advanced methods of fitting nonlinear functions with an over-determined number of parameters. An important subset of machine learning methods which will be the focus of this study are artificial neural networks (ANNs). The term "neural network" (NN) refers to any arrangement of neurons, which are simple units of computation, which are joined together in a network to facilitate computing functions which are much more complex than a single neuron. NNs can therefore refer to both biological NNs in animals as well as artificial NNs used to model biological systems or fit mathematical functions. We will focus on artificial NNs which, while originally designed as a model for cognition, have proven useful as a powerful tool for automated fitting of high dimensional functions.

Recently, both machine learning in general and ANNs specifically have seen wide interest in the physical sciences. In chemistry it has been shown that ANNs can be used to fit the potential energy surface (PES) of atomic systems, and in particular can fit so-called ab-initio PES. The resulting fits to the PES, known as neural network potentials (NNPs), can then be used to compute energies and forces of atomic systems, thereby allowing for ab-initio quality molecular dynamics simulations often at a small fraction of the cost. In this review we will discuss the use of NNs to fit ab-initio potential energy surfaces of chemical systems to create neural network potentials (NNPs) which can then be used to simulate chemical systems with high affinity to the ab-initio results while at a greatly reduced cost.

2 Neural Networks

2.1 Basic Construction

In order to discuss neural network potentials, we must first discuss the construction and training of neural networks themselves. In this section we will address the basic construction of neural networks and how we can label and manipulate their different components. We show a simple example of a NN in Figure 1, where each node in the network is represented by a disk and each edge is represented by an arrow connecting the disks. The first layer in

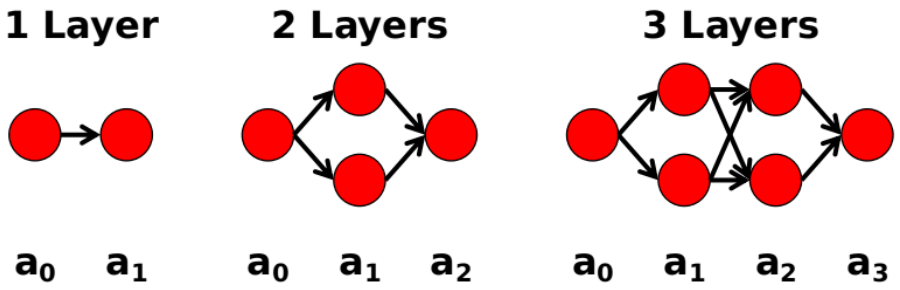

Figure 2: Illustration of the number of layers of a network, best thought of as the number of "connections" between each node. The inputs are thus the "zeroth" layer, the outputs are the "n^{th}" layer, and the hidden layers begin with the 1^{st} layer.

Figure 1 is the input data, the second layer is a "hidden layer" where the node values don't have a physical meaning, and the third layer is the output. Each node in the NN which is not in the input layer has a value which is determined by a transfer function. The argument of the transfer function is the product of the values in the previous layer and a series of weights associated with a given layer, represented by the weights in Figure 1. The transfer function is generally non-linear and a NN can have any number of layers, such that NNs are therefore compositions of nonlinear functions which can be trained to fit virtually any function. The dimensions of the function one is fitting are then determined by the inputs; that is, a NN with n inputs and m outputs can fit a function with a domain: $f : R^n \rightarrow R^m$.

The size and complexity of NNs are mostly determined by the type of transfer function and the number of layers. While one can use any number of weights in each layer of a NN, it has generally been found that increasing the number of layers while maintaining a modest number of nodes per layer is the best approach. We therefore label each network by the number of layers, that is a "3-layer" network is fundamentally different than a "4-layer" network. When it comes to labeling networks, the number of "layers" of a network are best thought of as the number of "connections" between each set of nodes, rather than the total number of sets of nodes. We illustrate this concept in Figure 2, where we show that the input layer can be thought of as the "zeroth" layer, the hidden layers begin with the first layer, and the outputs are the n^{th} layer of an n-layer network.

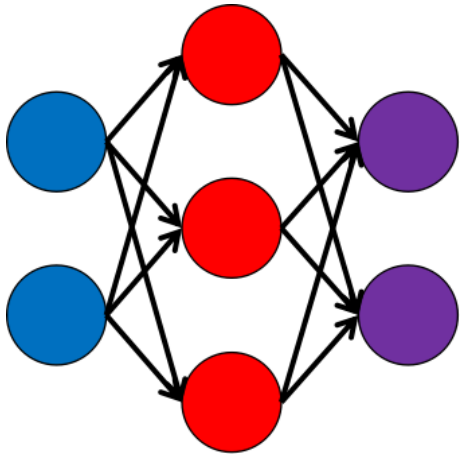

Figure 1: A basic example of a neural network. The disks all represent floating point numbers, and the arrows represent multiplication by a weight associated with that connection. The blue disks are the inputs, the red disks are a hidden layer, and the purple disks are the outputs.

With consistent labeling of each layer of a network, we can now more precisely define

the mathematical operations represented by a NN. We label the value or activation of the n^{th} node of the l^{th} layer by a_n^l. The activation of a given node is determined by the transfer function f, such that the activation can be written as:

$$a_i^l \equiv f(w_{ij}^l a_j^{l-1} + b_i^l) \tag{1}$$

where w_{ij}^l is weight connecting the j^{th} node in the l^{th} layer with the i^{th} node in the $(l+1)^{th}$ layer, and we are using Einstein summation notation where repeated indices are implicitly summed. We also define the argument of the transfer function as:

$$z_i^l \equiv w_{ij}^l a_j^{l-1} + b_i^l \tag{2}$$

such that $a_i^l = f(z_i^l)$, as this will simplify notation later. Note that the input for the zeroth later z_i^0 is not well defined as this is the input layer and thus has no associated activation function. In equation 1 we note that the set of activations and biases of a given layer have the form of a vector, and the weights of a given layer has the form of a matrix. We can therefore rewrite equation 1 using matrix notation to simplify the expression and reduce the number of indices.

$$a^l = f(w^l a^{l-1} + b^l) \tag{3}$$

2.2 Training - Backpropagation

Given a set of N input and output pairs, we wish to train a network such that for every input the values of the final layer of the network closely match the associated output. That is, we wish to minimize the following cost function:

$$C \equiv \frac{1}{N}\frac{1}{2}\sum_x |a^L(x) - y(x)|^2 \tag{4}$$

where $y(x)$ is the given output y as a function of a given input x and $a_L(x)$ is the output of an L-layer network for the same input x. In order to train the network, we must therefore compute the gradient of the cost function with respect to the values of the weights and biases: $\partial C/\partial w_{ij}^l$ and $\partial C/\partial b_k^l$. This is a difficult problem as the value of every layer in the network depends upon the values of all previous layers, and a similar dependence therefore applies to the gradients of the network. Fortunately, owing the structure of the network, there exist recursive algorithms that allow one to compute gradients for increasingly larger networks. We can thus start with the simplest possible case, a NN with one layer, and then compute gradients for NNs with two layers, three layers, etc. This procedure of computing gradients for the layers closest to the output and then recursively computing gradients for adjacent layers is known as the "backpropagation" algorithm, as it involves propagating gradients backwards from the output layer to the input layer.

Rather than focus on the averaged cost function in equation 4, we instead focus on the cost associated with a single datum:

$$C(x) \equiv \frac{1}{2}|a^L - y|^2 \tag{5}$$

where y is the target value and a^L is the activation of the output layer of the network. In order to compute the gradients of the cost function with backpropagation, we need to find a generating function which will allow us to relate gradients in one layer of the network to gradients in the adjacent layer. Rather than focusing on gradients with respect to the parameters ω_{ij}^l and b_k^l, we instead take the gradient with respect to the input z_i^l. We focus on these gradients as the chain rule can be used to relate the gradient with respect to z_i^l to the gradient with respect to z_i^{l+1}, and it's relatively simple to relate gradients with respect to z_i^l to gradients with respect to ω_{ij}^l and b_k^l.

We begin by connecting the gradient with respect to z^l to z^{l+1}, thereby allowing us to move backwards through the network if we already know the gradient with respect to z^{l+1}.

$$\frac{\partial C}{\partial z_i^l} = \frac{\partial C}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial z_i^l} \tag{6}$$

We thus need to compute $\partial z_j^{l+1}/\partial z_i^l$.

$$\begin{aligned} \frac{\partial z_i^{l+1}}{\partial z_j^l} &= \frac{\partial}{\partial z_i^l}(\omega_{ik}^{l+1} a_k^l + b_i^{l+1}) \\ &= \omega_{ik}^{l+1} \frac{\partial a_k^l}{\partial z_j^l} = \omega_{ik}^{l+1} \frac{\partial}{\partial z_j^l} f(z_k^l) = \omega_{ik}^{l+1} \dot{f}(z_k^l) \delta_{jk} \\ \frac{\partial z_i^{l+1}}{\partial z_j^l} &= \omega_{ij}^l \dot{f}(z_j^l) \end{aligned} \tag{7}$$

We can therefore write

$$\begin{aligned} \frac{\partial C}{\partial z_i^l} &= \frac{\partial C}{\partial z_j^{l+1}} \frac{\partial z_j^{l+1}}{\partial z_i^l} = \frac{\partial C}{\partial z_j^{l+1}} \omega_{ji}^l \dot{f}(z_i^l) \\ \nabla_{z^l} C &= \dot{f}(z^l)(\omega^l)^T \nabla_{z^{l+1}} C \end{aligned} \tag{8}$$

Now that we have a generating function for backpropagation in equation 8, we can compute the gradients of the cost function with respect to the biases and weights.

The bias gradients follow immediately from the definition of the inputs.

$$\begin{aligned} \frac{\partial C}{\partial b_i^l} &= \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_i^l} = \frac{\partial C}{\partial z_j^l} \delta_{ij} \\ \frac{\partial C}{\partial b_i^l} &= \frac{\partial C}{\partial z_i^l} \end{aligned} \tag{9}$$

The weight gradients are also simple to compute once we have the gradients with respect to z^l.

$$\begin{aligned}\frac{\partial C}{\partial \omega_{ij}^l} &= \frac{\partial C}{\partial z_k^l}\frac{\partial z_k^l}{\partial \omega_{ij}^l} = \frac{\partial C}{\partial z_k^l}\delta_{ik}a_j^{l-1} \\ \frac{\partial C}{\partial \omega_{ij}^l} &= \frac{\partial C}{\partial z_i^l}a_j^{l-1}\end{aligned} \tag{10}$$

Equations 8, 9, and 10 together yield the backpropagation algorithm.

In order to initialize the algorithm we need $\partial C/\partial z_L$. This follows simply from the definition of the cost function and the activation function.

$$\begin{aligned}\frac{\partial C}{\partial z_i^L} &= \frac{\partial}{\partial z_i^L}\left[\frac{1}{2}(a_j^L - y_j)^2\right] \\ &= (a_j^L - y_j)\frac{\partial a_j^L}{\partial z_i^L} \\ &= (a_j^L - y_j)\dot{f}(z_j^L)\delta_{ij} \\ &= \frac{\partial C}{\partial a_j^L}\dot{f}(z_j^L)\delta_{ij} \\ \frac{\partial C}{\partial z_i^L} &= \frac{\partial C}{\partial a_i^L}\dot{f}(z_i^L)\end{aligned} \tag{11}$$

2.3 Gradients

While the gradient of the cost function with respect to the parameters of a NN is essential for training, it is often that other gradients of a NN are required. In particular, although the mathematics of a NN are couched in terms of linear algebra, we can write a NN as a single function returning an m-dimensional vector of values y for any given n-dimensional vector of arguments x. Thus, for a NN represented by the function $\psi(x)$ if is often necessary to compute the gradient of the function. That is, in this section we will derive the gradients of the output of a NN with respect to its inputs.

We will first derive an expression for the first derivative of the outputs of a NN with respect to its inputs. Note that for a given NN with n inputs and m outputs the gradient will be a (m, n) matrix ρ, such that ρx yields a linear approximation of the output for a given input x. The quantity we wish to compute is thus $\partial a_i^L/\partial a_j^0$. We will take the same approach as the backpropagation algorithm and compute the gradients layer by layer, moving from those layers closest to the output to the inputs.

$$\begin{aligned}\frac{\partial a_i^L}{\partial a_j^l} &= \frac{\partial a_i^L}{\partial a_k^{l+1}}\frac{\partial a_k^{l+1}}{\partial a_j^l} \\ \frac{\partial a_i^L}{\partial a_j^l} &= \frac{\partial a_i^L}{\partial a_k^{l+1}}\frac{\partial a_k^{l+1}}{\partial z_k^{l+1}}\omega_{kj}^l \\ A^l &= A^{l+1}\dot{f}(z^{l+1})\omega^l\end{aligned} \tag{12}$$

We can thus use equation 12 to compute the gradient of the output for every layer, beginning with $L-1$ and decrementing until we reach the input layer. In order to start this process, we need $\partial a_i^L/\partial a_j^L$, but this quantity is trivially equivalent to δ_{ij}.

In addition to the first derivative, we also often are required to compute second derivatives of the NN. In this case, we wish to compute $\partial^2 a_i^L/\partial a_j^0 \partial a_k^0$, such that the second derivative is a tensor of rank 3.

$$\begin{aligned}
\frac{\partial a_i^L}{\partial a_j^l \partial a_k^l} =& \frac{\partial}{\partial a_k^l}\frac{\partial a_i^L}{\partial a_j^l} \\
\frac{\partial a_i^L}{\partial a_j^l \partial a_k^l} =& \frac{\partial}{\partial a_k^l}\frac{\partial a_i^L}{\partial a_n^{l+1}}\frac{\partial a_n^{l+1}}{\partial z_n^{l+1}}\omega_{nj}^l \\
\frac{\partial a_i^L}{\partial a_j^l \partial a_k^l} =& \frac{\partial}{\partial a_m^{l+1}}\frac{\partial a_m^{l+1}}{\partial a_k^l}\frac{\partial a_i^L}{\partial a_n^{l+1}}\frac{\partial a_n^{l+1}}{\partial z_n^{l+1}}\omega_{nj}^l \\
\frac{\partial a_i^L}{\partial a_j^l \partial a_k^l} =& \frac{\partial a_m^{l+1}}{\partial a_k^l}\frac{\partial a_i^L}{\partial a_m^{l+1}\partial a_n^{l+1}}\frac{\partial a_n^{l+1}}{\partial z_n^{l+1}}\omega_{nj}^l \\
\frac{\partial a_i^L}{\partial a_j^l \partial a_k^l} =& \frac{\partial a_m^{l+1}}{\partial z_m^{l+1}}\omega_{mk}^l\frac{\partial a_i^L}{\partial a_m^{l+1}\partial a_n^{l+1}}\frac{\partial a_n^{l+1}}{\partial z_n^{l+1}}\omega_{nj}^l
\end{aligned} \tag{13}$$

We thus have an equation relating the second derivative at each layer to the next layer in the process.

Another set of gradients which are important in NNs are the gradients of the outputs with respect to the NN parameters, namely the biases and weights. Just as in the backpropagation algorithm, we require a generating function relating the gradients of the output with respect to the inputs for each layer.

$$\begin{aligned}
\frac{\partial a_i^L}{\partial z_j^l} &= \frac{\partial a_i^L}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_j^l} \\
\frac{\partial a_i^L}{\partial z_j^l} &= \frac{\partial a_i^L}{\partial z_k^{l+1}}\omega_{kj}^l \dot{f}(z_j^l) \\
\alpha_{ij}^l &= \alpha_{ik}^{l+1}\omega_{kj}^l \dot{f}(z_j^l)
\end{aligned} \tag{14}$$

We can then easily find the gradient of the output with respect to the bias:

$$\frac{\partial a_i^L}{\partial b_j^l} = \frac{\partial a_i^L}{\partial z_j^l} \tag{15}$$

Similarly, we can find the gradient of the output with respect to the weights:

$$\begin{aligned}
\frac{\partial a_i^L}{\partial w_{jk}^l} &= \frac{\partial a_i^L}{\partial z_x^l}\frac{\partial z_x^l}{\partial \omega_{jk}^l} = \frac{\partial a_i^L}{\partial z_x^l}\delta_{xj}a_k^{l-1} \\
\frac{\partial a_i^L}{\partial w_{jk}^l} &= \frac{\partial a_i^L}{\partial z_j^l}a_k^{l-1}
\end{aligned} \tag{16}$$

To initialize the recursion, we need to compute $\partial a_i^L / \partial z_j^L$.

$$\frac{\partial a_i^L}{\partial z_j^L} = \dot{f}(z_i^L)\delta_{ij} \tag{17}$$

3 Neural Network Potentials

In this section we will discuss the use of neural networks to fit the potential energy surface of a set of atomic coordinates.

3.1 NN Hamiltonian

If we are to use a NNP to simulate a given system, we require that the NNP obey the same symmetry operations as the underlying ab-initio Hamiltonian. More formally, given a potential energy surface $E(\mathbf{R})$ as a function of the atomic coordinates $\mathbf{R}$, we require the NNP to obey the following symmetries:

$$\begin{aligned} E(\mathbf{R}) &= E(\mathbf{R} + \mathbf{dr}) && \text{(uniform translation)} \\ E(\mathbf{R}) &= E(R(\theta)\mathbf{R}) && \text{(uniform rotation)} \\ E(\mathbf{R}) &= E(\mathbf{R} + \mathbf{nR}) && \text{(periodicity)} \\ NE(\mathbf{R}) &= \sum_n^N E(\mathbf{R}) && \text{(intensive)} \end{aligned} \tag{18}$$

The simplest way to enforce the symmetries in equation 18 is to write the total energy of a system as a sum over atomic energies.

$$E(\mathbf{R}) = \sum_{\mathbf{i}} \mathbf{E_i}(\mathbf{r}) \tag{19}$$

where $E_i(r)$ is the energy of the i^{th} atom. We can then write the atomic energies as the output of a neural network, where the inputs are measures of the local symmetry around the atom. While we cannot write the energy as an analytic function of the inputs, we simply state that we can compute the value and gradients of the unknown function. The energy of the i^{th} atom can then be written as:

$$E_i = \psi_i(G) \tag{20}$$

where G is the vector of inputs to the neural network and $\psi_i(G)$ is the unknown function represented by the neural network which yields the energy E_i. Note that ψ_i tends to be identical for each species of atom while the inputs can differ for each atom, though the function generating the inputs will tend to be identical for all atoms. The inputs G can be any measure of the local symmetry around the atom. There are a large number of different strategies for generating G, but we choose to use the approach of Behler and Parinello, where

each input is an average of a given symmetry function over the positions of neighboring atoms up to a cutoff distance. This approach has been shown to yield accurate potentials for a wide variety of systems while also allowing for a relatively simple implementation and a more intuitive understanding of how the local symmetries impact the energy and forces. A hallmark of Behler's approach is that the symmetry functions are split between radial functions, which are averages over pairs of atoms, and angular functions, which are averages over triples of atoms. We can thus rewrite equation **??** to reflect this:

$$E_i = \psi_i(G_i^R, G_i^A) \tag{21}$$

where G_i^R is a vector of all radial inputs and G_i^A is a vector of all angular inputs.

3.2 Forces

Given a NNP trained to ab-initio energies, we wish to compute the force on a given atom in addition to the energy. For any potential energy surface we can compute the force on the n^{th} atom by taking the gradient of the energy with respect to the n^{th} position.

$$F_n = -\partial_{r_n} E \tag{22}$$

where F_n denotes the total force on the n^{th} atom and ∂_{r_n} is the gradient with respect to the n^{th} position. We can then expand the gradient using equation 21:

$$\begin{aligned} F_n = -\partial_{r_n} E &= -\sum_i \frac{\partial}{\partial r_n} E_i \\ &= -\sum_i \frac{\partial}{\partial r_n} \psi_i(G_i^R, G_i^A) \\ &= -\sum_{i\alpha} \frac{\partial \psi_i}{\partial G_{i\alpha}^R} \frac{\partial G_{i\alpha}^R}{\partial r_n} - \sum_{i\alpha} \frac{\partial \psi_i}{\partial G_{i\alpha}^A} \frac{\partial G_{i\alpha}^A}{\partial r_n} \end{aligned} \tag{23}$$

We have split the gradient into two terms, using the chain rule to express the force as a product of the gradient of the network $\partial\psi_i/\partial G_{i\alpha}$ and the gradients of the inputs to the network $\partial\psi_i/\partial r_n$. The index α now denotes the index of the radial and angular inputs. The gradient of the network is simple to compute using standard algorithms, and so we will simplify by our notation by writing the gradient as $\dot{\psi}_{i\alpha}^R \equiv \partial\psi_i/\partial G_{i\alpha}^R$ for the radial inputs and $\dot{\psi}_{i\alpha}^A \equiv \partial\psi_i/\partial G_{i\alpha}^A$ for the angular inputs.

We next turn our attention to the gradients of the inputs. Here we assume each of the inputs can be written as sums of functions of the nearest neighbor positions around the i^{th} atom.

$$\begin{aligned} G_{i\alpha}^R &= \sum_{j\neq i} \rho_{ij}^{(\alpha)}(r_{ij}) \\ G_{i\alpha}^A &= \sum_{j\neq i} \sum_{k\neq i,j} \gamma_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk}) \end{aligned} \tag{24}$$

Here each function is labeled by the pair (i, j) or triple (i, j, k) as the type of function will often change with the species of the pair or triple of atoms.

In order to simplify our notation and the operations on sums, we will make use of the Kronecker delta function. We will define an indicator function $\zeta_{ij} \equiv 1 - \delta_{ij}$ which is zero if the two indices (i, j) are equal and one otherwise. We can thus rewrite equation 24 as:

$$\begin{aligned} G^R_{i\alpha} &= \sum_j \zeta_{ij} \rho^{(\alpha)}_{ij}(r_{ij}) \\ G^A_{i\alpha} &= \sum_{j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk} \gamma^{(\alpha)}_{ijk}(r_{ij}, r_{ik}, r_{jk}) \end{aligned} \tag{25}$$

The simpler form of the sums in equation 25 will make it much easier to affirm that the equations for the forces follow Newton's third law and will aid in constructing efficient algorithms to compute the forces in a simulation.

3.3 Radial Force

The radial force on the n^{th} atom is given as:

$$F^R_n = -\sum_{i\alpha} \dot{\psi}^R_{i\alpha} \partial_{r_n} G^R_{i\alpha} \tag{26}$$

We expand the definition of the radial inputs to find:

$$F^R_n = -\sum_{i,j} \sum_{\alpha} \dot{\psi}^R_{i\alpha} \partial_{r_n} \zeta_{ij} \rho^{(\alpha)}_{ij}(r_{ij}) \tag{27}$$

Since $\rho^{(\alpha)}_{ij}(r_{ij})$ is a scalar function with a scalar argument, we can separate the derivative of the symmetry function and the gradient of the position. We thus define the following notation for the derivative of the symmetry function $\dot{\rho}^{(\alpha)}_{\eta}(x) \equiv \partial_x \rho^{(\alpha)}_{\eta}(x)$. We can then compute the gradient in equation 27 without specifying the exact form of the radial symmetry functions.

$$
\begin{aligned}
F_n^R &= -\sum_{i,j}\sum_{\alpha} \dot{\psi}_{i\alpha}^R \zeta_{ij} \dot{\rho}_{ij}^{(\alpha)}(r_{ij}) \partial_{r_n} r_{ij} \\
&= -\sum_{i,j}\sum_{\alpha} \dot{\psi}_{i\alpha}^R \zeta_{ij} \dot{\rho}_{ij}^{(\alpha)}(r_{ij}) (\hat{r}_{ij}\delta_{ni} - \hat{r}_{ij}\delta_{nj}) \\
&= -\left[\sum_{i,j}\sum_{\alpha} \dot{\psi}_{i\alpha}^R \zeta_{ij} \dot{\rho}_{ij}^{(\alpha)}(r_{ij}) \hat{r}_{ij}\delta_{ni} - \sum_{i,j}\sum_{\alpha} \dot{\psi}_{i\alpha}^R \zeta_{ij} \dot{\rho}_{ij}^{(\alpha)}(r_{ij}) \hat{r}_{ij}\delta_{nj}\right] \\
&= -\left[\sum_{j}\sum_{\alpha} \dot{\psi}_{n\alpha}^R \zeta_{nj} \dot{\rho}_{nj}^{(\alpha)}(r_{nj}) \hat{r}_{nj} - \sum_{i}\sum_{\alpha} \dot{\psi}_{i\alpha}^R \zeta_{in} \dot{\rho}_{in}^{(\alpha)}(r_{in}) \hat{r}_{in}\right] \\
&= -\left[\sum_{j}\sum_{\alpha} \dot{\psi}_{n\alpha}^R \zeta_{nj} \dot{\rho}_{nj}^{(\alpha)}(r_{nj}) \hat{r}_{nj} + \sum_{j}\sum_{\alpha} \dot{\psi}_{j\alpha}^R \zeta_{jn} \dot{\rho}_{jn}^{(\alpha)}(r_{jn}) \hat{r}_{nj}\right] \\
&= -\sum_{j}\sum_{\alpha} \zeta_{nj}\left[\dot{\psi}_{n\alpha}^R \dot{\rho}_{nj}^{(\alpha)}(r_{nj}) + \dot{\psi}_{j\alpha}^R \dot{\rho}_{jn}^{(\alpha)}(r_{jn})\right]\hat{r}_{nj}
\end{aligned}
\tag{28}
$$

Thus, the final result for the radial force using more standard notation is:

$$
\begin{aligned}
F_i^R &= -\sum_{j}\sum_{\alpha} \zeta_{ij}\left[\dot{\psi}_{i\alpha}^R \dot{\rho}_{ij}^{(\alpha)}(r_{ij}) + \dot{\psi}_{j\alpha}^R \dot{\rho}_{ji}^{(\alpha)}(r_{ji})\right]\hat{r}_{ij} \\
F_i^R &= -\sum_{j\neq i}\sum_{\alpha} \left[\dot{\psi}_{i\alpha}^R \dot{\rho}_{ij}^{(\alpha)}(r_{ij}) + \dot{\psi}_{j\alpha}^R \dot{\rho}_{ji}^{(\alpha)}(r_{ji})\right]\hat{r}_{ij}
\end{aligned}
\tag{29}
$$

While this is a perfectly valid equation to use to compute the radial force, it is cumbersome in practice. Note that, in order to compute the force on atom i, one must compute the inputs and gradient of the neural network for atom j. Thus, to compute the force on all atoms, one must store the inputs, network, and network gradients of all atoms, which requires a significant amount of memory. We can simplify our calculations by noting that the term within the brackets obeys Netwon's third law, i.e. switching the indices i, j does not change the value of the expression.

$$
\begin{aligned}
F_T^R &= -\sum_{i,j}\sum_{\alpha} \zeta_{ij} f_{ij} \hat{r}_{ij} \\
2F_T^R &= -\sum_{i,j}\sum_{\alpha} \zeta_{ij} f_{ij} \hat{r}_{ij} - \sum_{i,j}\sum_{\alpha} \zeta_{ij} f_{ij} \hat{r}_{ij} \\
2F_T^R &= -\sum_{i,j}\sum_{\alpha} \zeta_{ij} f_{ij} \hat{r}_{ij} - \sum_{j,i}\sum_{\alpha} \zeta_{ji} f_{ji} \hat{r}_{ji} \\
2F_T^R &= -\sum_{i,j}\sum_{\alpha} \zeta_{ij} f_{ij} \hat{r}_{ij} + \sum_{j,i}\sum_{\alpha} \zeta_{ij} f_{ij} \hat{r}_{ij} \\
2F_T^R &= -\sum_{i,j}\sum_{\alpha} \zeta_{ij} f_{ij} \hat{r}_{ij} + \sum_{i,j}\sum_{\alpha} \zeta_{ij} f_{ij} \hat{r}_{ij} \\
F_T^R &= 0
\end{aligned}
\tag{30}
$$

The best algorithm for computing the radial force is thus to compute the first half of term in brackets in equation 29 and add its value to the force on atom i and the negated value to the force on atom j. More formally we can describe the algorithm thusly:

$$\begin{aligned}
&\forall_i \forall_{j \neq i} \forall_\alpha \\
&f_{ij}^{R(\alpha)} \equiv -\dot{\psi}_{i\alpha}^R \dot{\rho}_{ij}^{(\alpha)}(r_{ij}) \\
&F_i^R \to F_i^R + f_{ij}^{R(\alpha)} \hat{r}_{ij} \\
&F_j^R \to F_j^R - f_{ij}^{R(\alpha)} \hat{r}_{ij}
\end{aligned} \tag{31}$$

3.4 Angular Force

The angular force on the n^{th} atom is given as:

$$F_n^A = -\sum_{i\alpha} \dot{\psi}_{i\alpha}^A \partial_{r_n} G_{i\alpha}^A \tag{32}$$

We expand the definition of the angular inputs to find:

$$F_n^A = -\sum_{i,j,k} \sum_\alpha \dot{\psi}_{i\alpha}^R \zeta_{ij} \zeta_{ik} \zeta_{jk} \partial_{r_n} \gamma_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk}) \tag{33}$$

We next assume that we can write the function $\gamma_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk})$ as a product of two functions, one which depends upon the angle between the vectors $\vec{r}_{ij}$ and $\vec{r}_{ik}$ and the other which depends upon the trio of distances (r_{ij}, r_{ik}, r_{jk}).

$$\gamma_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk}) \equiv f_{ijk}^{(\alpha)}(\cos(\theta_{ijk})) h_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk}) \tag{34}$$

where we define $\cos(\theta_{ijk}) \equiv \vec{r}_{ij} \cdot \vec{r}_{ik} / r_{ij} r_{ik}$. We can then compute the gradient of γ by separately computing the gradient of the functions f and h.

We first address the gradient of h, which depends only upon distances. We can therfore use the product rule to compute the gradient as a product of the derivative of h and the gradient of the distances. Is is therefore not necessary to specify the exact form of h, we need only to define a notation for the gradients of the different arguments of h.

$$\begin{aligned}
h_{ijk}^{(\alpha,1)}(x_1, x_2, x_3) &\equiv \partial_{x_1} h_{ijk}^{(\alpha)}(x_1, x_2, x_3) \\
h_{ijk}^{(\alpha,2)}(x_1, x_2, x_3) &\equiv \partial_{x_2} h_{ijk}^{(\alpha)}(x_1, x_2, x_3) \\
h_{ijk}^{(\alpha,3)}(x_1, x_2, x_3) &\equiv \partial_{x_3} h_{ijk}^{(\alpha)}(x_1, x_2, x_3)
\end{aligned} \tag{35}$$

We next address the gradient of f, where we once again simplify the expression by taking the derivative of the function first and then taking the gradient of the cosine argument. We denote the derivative of f as $\dot{f}_\eta^{(\alpha)}(x) \equiv \partial_x f_\eta^{(\alpha)}(x)$. We can then compute the gradient of the cosine argument using the above definition in terms of interatomic distances.

$$
\begin{aligned}
\partial_{r_i}\cos(\theta_{ijk}) &= (r_{ik}^{-1} - \cos(\theta_{ijk})r_{ij}^{-1})\hat{r}_{ij} + (r_{ij}^{-1} - \cos(\theta_{ijk})r_{ik}^{-1})\hat{r}_{ik} \\
\partial_{r_j}\cos(\theta_{ijk}) &= -r_{ij}^{-1}\hat{r}_{ik} + \cos(\theta_{ijk})r_{ij}^{-1}\hat{r}_{ij} \\
\partial_{r_k}\cos(\theta_{ijk}) &= -r_{ik}^{-1}\hat{r}_{ij} + \cos(\theta_{ijk})r_{ik}^{-1}\hat{r}_{ik}
\end{aligned} \tag{36}
$$

We can now use equations 35 and 36 to expand the angular force in equation 33.

$$
\begin{aligned}
F_n^A &= -\sum_{i,j,k}\sum_{\alpha} \dot{\psi}_{i\alpha}^A \zeta_{ij}\zeta_{ik}\zeta_{jk}\partial_{r_n}\gamma_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk}) \\
&= -\sum_{\alpha,i,j,k} \dot{\psi}_{i\alpha}^A \zeta_{ij}\zeta_{ik}\zeta_{jk}\partial_{r_n}\left[f_{ijk}^{(\alpha)}(\cos(\theta_{ijk}))h_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk})\right] \\
&= -\sum_{\alpha,i,j,k} \dot{\psi}_{i\alpha}^A \zeta_{ij}\zeta_{ik}\zeta_{jk}\left[h_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk})\partial_{r_n}f_{ijk}^{(\alpha)}(\cos(\theta_{ijk})) + f_{ijk}^{(\alpha)}(\cos(\theta_{ijk}))\partial_{r_n}h_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk})\right]
\end{aligned} \tag{37}
$$

We then write out each term in the gradient, leading to a sum of seven distinct terms.

$$
\begin{aligned}
F_n^A = &-\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk})\dot{f}_{ijk}^{(\alpha)}(\cos(\theta_{ijk}))\delta_{ni}(r_{ik}^{-1} - \cos(\theta_{ijk})r_{ij}^{-1})\hat{r}_{ij} \\
&-\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk})\dot{f}_{ijk}^{(\alpha)}(\cos(\theta_{ijk}))\delta_{ni}(r_{ij}^{-1} - \cos(\theta_{ijk})r_{ik}^{-1})\hat{r}_{ik} \\
&-\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk})\dot{f}_{ijk}^{(\alpha)}(\cos(\theta_{ijk}))\delta_{nj}(-r_{ij}^{-1}\hat{r}_{ik} + \cos(\theta_{ijk})r_{ij}^{-1}\hat{r}_{ij}) \\
&-\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)}(r_{ij}, r_{ik}, r_{jk})\dot{f}_{ijk}^{(\alpha)}(\cos(\theta_{ijk}))\delta_{nk}(-r_{ik}^{-1}\hat{r}_{ij} + \cos(\theta_{ijk})r_{ik}^{-1}\hat{r}_{ik}) \\
&-\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A f_{ijk}^{(\alpha)}(\cos(\theta_{ijk}))\delta_{ni}(h_{ijk}^{(\alpha,1)}(r_{ij}, r_{ik}, r_{jk})\hat{r}_{ij} + h_{ijk}^{(\alpha,2)}(r_{ij}, r_{ik}, r_{jk})\hat{r}_{ik}) \\
&-\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A f_{ijk}^{(\alpha)}(\cos(\theta_{ijk}))\delta_{nj}(-h_{ijk}^{(\alpha,1)}(r_{ij}, r_{ik}, r_{jk})\hat{r}_{ij} + h_{ijk}^{(\alpha,3)}(r_{ij}, r_{ik}, r_{jk})\hat{r}_{jk}) \\
&-\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A f_{ijk}^{(\alpha)}(\cos(\theta_{ijk}))\delta_{nk}(-h_{ijk}^{(\alpha,2)}(r_{ij}, r_{ik}, r_{jk})\hat{r}_{ik} - h_{ijk}^{(\alpha,3)}(r_{ij}, r_{ik}, r_{jk})\hat{r}_{jk})
\end{aligned} \tag{38}
$$

Once we have expanded the gradients we can simplify the expression by making the arguments for h and f implicit.

$$
\begin{aligned}
F_n^A = &- \sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} \delta_{ni}(r_{ik}^{-1} - \cos(\theta_{ijk})r_{ij}^{-1})\hat{r}_{ij} \\
&- \sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} \delta_{ni}(r_{ij}^{-1} - \cos(\theta_{ijk})r_{ik}^{-1})\hat{r}_{ik} \\
&- \sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} \delta_{nj}(-r_{ij}^{-1}\hat{r}_{ik} + \cos(\theta_{ijk})r_{ij}^{-1}\hat{r}_{ij}) \\
&- \sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} \delta_{nk}(-r_{ik}^{-1}\hat{r}_{ij} + \cos(\theta_{ijk})r_{ik}^{-1}\hat{r}_{ik}) \\
&- \sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A f_{ijk}^{(\alpha)} \delta_{ni}(h_{ijk}^{(\alpha,1)}\hat{r}_{ij} + h_{ijk}^{(\alpha,2)}\hat{r}_{ik}) \\
&- \sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A f_{ijk}^{(\alpha)} \delta_{nj}(-h_{ijk}^{(\alpha,1)}\hat{r}_{ij} + h_{ijk}^{(\alpha,3)}\hat{r}_{jk}) \\
&- \sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}_{i\alpha}^A f_{ijk}^{(\alpha)} \delta_{nk}(-h_{ijk}^{(\alpha,2)}\hat{r}_{ik} - h_{ijk}^{(\alpha,3)}\hat{r}_{jk})
\end{aligned} \tag{39}
$$

We then remove the delta functions to obtain:

$$
\begin{aligned}
F_n^A = &- \sum_{\alpha,j,k} \zeta_{nj}\zeta_{nk}\zeta_{jk}\dot{\psi}_{n\alpha}^A h_{njk}^{(\alpha)} \dot{f}_{njk}^{(\alpha)} (r_{nk}^{-1} - \cos(\theta_{njk})r_{nj}^{-1})\hat{r}_{nj} \\
&- \sum_{\alpha,j,k} \zeta_{ij}\zeta_{nk}\zeta_{jk}\dot{\psi}_{n\alpha}^A h_{njk}^{(\alpha)} \dot{f}_{njk}^{(\alpha)} (r_{nj}^{-1} - \cos(\theta_{njk})r_{nk}^{-1})\hat{r}_{nk} \\
&- \sum_{\alpha,i,k} \zeta_{in}\zeta_{ik}\zeta_{nk}\dot{\psi}_{i\alpha}^A h_{ink}^{(\alpha)} \dot{f}_{ink}^{(\alpha)} (-r_{in}^{-1}\hat{r}_{ik} + \cos(\theta_{ink})r_{in}^{-1}\hat{r}_{in}) \\
&- \sum_{\alpha,i,j} \zeta_{ij}\zeta_{in}\zeta_{jn}\dot{\psi}_{i\alpha}^A h_{ijn}^{(\alpha)} \dot{f}_{ijn}^{(\alpha)} (-r_{in}^{-1}\hat{r}_{ij} + \cos(\theta_{ijn})r_{in}^{-1}\hat{r}_{in}) \\
&- \sum_{\alpha,j,k} \zeta_{nj}\zeta_{nk}\zeta_{jk}\dot{\psi}_{n\alpha}^A f_{njk}^{(\alpha)} (h_{njk}^{(\alpha,1)}\hat{r}_{nj} + h_{njk}^{(\alpha,2)}\hat{r}_{nk}) \\
&- \sum_{\alpha,i,k} \zeta_{in}\zeta_{ik}\zeta_{nk}\dot{\psi}_{i\alpha}^A f_{ink}^{(\alpha)} (-h_{ink}^{(\alpha,1)}\hat{r}_{in} + h_{ink}^{(\alpha,3)}\hat{r}_{nk}) \\
&- \sum_{\alpha,i,j} \zeta_{ij}\zeta_{in}\zeta_{jn}\dot{\psi}_{i\alpha}^A f_{ijn}^{(\alpha)} (-h_{ijn}^{(\alpha,2)}\hat{r}_{in} - h_{ijn}^{(\alpha,3)}\hat{r}_{jn})
\end{aligned} \tag{40}
$$

We then reverse the unit vectors and change the summation indices to make the terms look similar.

$$
\begin{aligned}
F_n^A = &- \sum_{\alpha,j,k} \zeta_{nj}\zeta_{nk}\zeta_{jk}\dot{\psi}^A_{n\alpha} h^{(\alpha)}_{njk} \dot{f}^{(\alpha)}_{njk} (r_{nk}^{-1} - \cos(\theta_{njk}) r_{nj}^{-1}) \hat{r}_{nj} \\
&- \sum_{\alpha,j,k} \zeta_{ij}\zeta_{nk}\zeta_{jk}\dot{\psi}^A_{n\alpha} h^{(\alpha)}_{njk} \dot{f}^{(\alpha)}_{njk} (r_{nj}^{-1} - \cos(\theta_{njk}) r_{nk}^{-1}) \hat{r}_{nk} \\
&- \sum_{\alpha,j,k} \zeta_{jn}\zeta_{jk}\zeta_{nk}\dot{\psi}^A_{j\alpha} h^{(\alpha)}_{jnk} \dot{f}^{(\alpha)}_{jnk} (r_{jn}^{-1}\hat{r}_{kj} - \cos(\theta_{jnk}) r_{jn}^{-1} \hat{r}_{nj}) \\
&- \sum_{\alpha,j,k} \zeta_{jk}\zeta_{jn}\zeta_{kn}\dot{\psi}^A_{j\alpha} h^{(\alpha)}_{jkn} \dot{f}^{(\alpha)}_{jkn} (r_{jn}^{-1}\hat{r}_{kj} - \cos(\theta_{jkn}) r_{jn}^{-1} \hat{r}_{nj}) \\
&- \sum_{\alpha,j,k} \zeta_{nj}\zeta_{nk}\zeta_{jk}\dot{\psi}^A_{n\alpha} f^{(\alpha)}_{njk} (h^{(\alpha,1)}_{njk}\hat{r}_{nj} + h^{(\alpha,2)}_{njk}\hat{r}_{nk}) \\
&- \sum_{\alpha,j,k} \zeta_{jn}\zeta_{jk}\zeta_{nk}\dot{\psi}^A_{j\alpha} f^{(\alpha)}_{jnk} (h^{(\alpha,1)}_{jnk}\hat{r}_{nj} + h^{(\alpha,3)}_{jnk}\hat{r}_{nk}) \\
&- \sum_{\alpha,j,k} \zeta_{jk}\zeta_{jn}\zeta_{kn}\dot{\psi}^A_{j\alpha} f^{(\alpha)}_{jkn} (h^{(\alpha,2)}_{jkn}\hat{r}_{nj} + h^{(\alpha,3)}_{jkn}\hat{r}_{nk})
\end{aligned} \tag{41}
$$

We then write the force in a more standard notation as:

$$
\begin{aligned}
F_i^A = &- \sum_{\alpha,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}^A_{i\alpha} h^{(\alpha)}_{ijk} \dot{f}^{(\alpha)}_{ijk} (r_{ik}^{-1} - \cos(\theta_{ijk}) r_{ij}^{-1}) \hat{r}_{ij} \\
&- \sum_{\alpha,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}^A_{i\alpha} h^{(\alpha)}_{ijk} \dot{f}^{(\alpha)}_{ijk} (r_{ij}^{-1} - \cos(\theta_{ijk}) r_{ik}^{-1}) \hat{r}_{ik} \\
&- \sum_{\alpha,j,k} \zeta_{ji}\zeta_{jk}\zeta_{ik}\dot{\psi}^A_{j\alpha} h^{(\alpha)}_{jik} \dot{f}^{(\alpha)}_{jik} (r_{ji}^{-1}\hat{r}_{kj} - \cos(\theta_{jik}) r_{ji}^{-1} \hat{r}_{ij}) \\
&- \sum_{\alpha,j,k} \zeta_{jk}\zeta_{ji}\zeta_{ki}\dot{\psi}^A_{j\alpha} h^{(\alpha)}_{jki} \dot{f}^{(\alpha)}_{jki} (r_{ji}^{-1}\hat{r}_{kj} - \cos(\theta_{jki}) r_{ji}^{-1} \hat{r}_{ij}) \\
&- \sum_{\alpha,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}^A_{i\alpha} f^{(\alpha)}_{ijk} (h^{(\alpha,1)}_{ijk}\hat{r}_{ij} + h^{(\alpha,2)}_{ijk}\hat{r}_{ik}) \\
&- \sum_{\alpha,j,k} \zeta_{ji}\zeta_{jk}\zeta_{ik}\dot{\psi}^A_{j\alpha} f^{(\alpha)}_{jik} (h^{(\alpha,1)}_{jik}\hat{r}_{ij} + h^{(\alpha,3)}_{jik}\hat{r}_{ik}) \\
&- \sum_{\alpha,j,k} \zeta_{jk}\zeta_{ji}\zeta_{ki}\dot{\psi}^A_{j\alpha} f^{(\alpha)}_{jki} (h^{(\alpha,2)}_{jki}\hat{r}_{ij} + h^{(\alpha,3)}_{jki}\hat{r}_{ik})
\end{aligned} \tag{42}
$$

While equation 42 does allow us to compute the angular force, it is rather impractical to implement and it is not obvious that it obeys Newton's third law. We thus group terms in 42 which are equal and opposite in order to exploit the symmetry of the equation and to more clearly illustrate Newton's third law. Note that in some cases we permute the indices (j, k) in order to group terms since the order of summation does not impact the result.

$$
\begin{aligned}
F_i^A = &- \sum_{\alpha,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}^A_{i\alpha} h^{(\alpha)}_{ijk} \dot{f}^{(\alpha)}_{ijk} r^{-1}_{ik} \hat{r}_{ij} - \sum_{\alpha,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\dot{\psi}^A_{i\alpha} h^{(\alpha)}_{ijk} \dot{f}^{(\alpha)}_{ijk} r^{-1}_{ij} \hat{r}_{ik} \\
&- \sum_{\alpha,j,k} \zeta_{ji}\zeta_{jk}\zeta_{ik}\dot{\psi}^A_{j\alpha} h^{(\alpha)}_{jik} \dot{f}^{(\alpha)}_{jik} r^{-1}_{ji} \hat{r}_{kj} - \sum_{\alpha,j,k} \zeta_{jk}\zeta_{ji}\zeta_{ki}\dot{\psi}^A_{j\alpha} h^{(\alpha)}_{jki} \dot{f}^{(\alpha)}_{jki} r^{-1}_{ji} \hat{r}_{kj} \\
&- \sum_{\alpha,j,k} -\zeta_{ij}\zeta_{ik}\zeta_{jk} \left[\dot{\psi}^A_{i\alpha} h^{(\alpha)}_{ijk} \dot{f}^{(\alpha)}_{ijk} \cos(\theta_{ijk}) r^{-1}_{ij} + \dot{\psi}^A_{j\alpha} h^{(\alpha)}_{jik} \dot{f}^{(\alpha)}_{jik} \cos(\theta_{jik}) r^{-1}_{ji} \right] \hat{r}_{ij} \\
&- \sum_{\alpha,j,k} -\zeta_{ij}\zeta_{ik}\zeta_{jk} \left[\dot{\psi}^A_{i\alpha} h^{(\alpha)}_{ijk} \dot{f}^{(\alpha)}_{ijk} \cos(\theta_{ijk}) r^{-1}_{ik} + \dot{\psi}^A_{k\alpha} h^{(\alpha)}_{kji} \dot{f}^{(\alpha)}_{kji} \cos(\theta_{kji}) r^{-1}_{ki} \right] \hat{r}_{ik} \\
&- \sum_{\alpha,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk} \left[\dot{\psi}^A_{i\alpha} f^{(\alpha)}_{ijk} h^{(\alpha,1)}_{ijk} + \dot{\psi}^A_{j\alpha} f^{(\alpha)}_{jik} h^{(\alpha,1)}_{jik} \right] \hat{r}_{ij} \\
&- \sum_{\alpha,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk} \left[\dot{\psi}^A_{i\alpha} f^{(\alpha)}_{ijk} h^{(\alpha,2)}_{ijk} + \dot{\psi}^A_{k\alpha} f^{(\alpha)}_{kji} h^{(\alpha,2)}_{kji} \right] \hat{r}_{ik} \\
&- \sum_{\alpha,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk} \left[\dot{\psi}^A_{j\alpha} f^{(\alpha)}_{jik} h^{(\alpha,3)}_{jik} + \dot{\psi}^A_{j\alpha} f^{(\alpha)}_{jki} h^{(\alpha,3)}_{jki} \right] \hat{r}_{ik}
\end{aligned}
\tag{43}
$$

Note that all lines in 43 except the first two have terms in brackets which are invariant with respect to permutation of the indices of the associated unit vector, thereby clearly illustrating Newton's third law. To make this discussion more concrete, we show an example where it's assumed that $f_{ijk} = f_{jik}$, as in the third and fifth lines above. While we will not explicitly show examples for other cases, the proof of zero total force is similar.

$$
\begin{aligned}
f^A_T &= - \sum_{ijk} \zeta_{ij}\zeta_{ik}\zeta_{jk} f_{ijk} \hat{r}_{ij} \\
2f^A_T &= - \sum_{ijk} \zeta_{ij}\zeta_{ik}\zeta_{jk} f_{ijk} \hat{r}_{ij} - \sum_{ijk} \zeta_{ij}\zeta_{ik}\zeta_{jk} f_{ijk} \hat{r}_{ij} \\
2f^A_T &= - \sum_{ijk} \zeta_{ij}\zeta_{ik}\zeta_{jk} f_{ijk} \hat{r}_{ij} - \sum_{jik} \zeta_{ji}\zeta_{jk}\zeta_{ik} f_{jik} \hat{r}_{ji} \\
2f^A_T &= - \sum_{ijk} \zeta_{ij}\zeta_{ik}\zeta_{jk} f_{ijk} \hat{r}_{ij} + \sum_{jik} \zeta_{ij}\zeta_{ik}\zeta_{jk} f_{ijk} \hat{r}_{ij} \\
2f^A_T &= - \sum_{ijk} \zeta_{ij}\zeta_{ik}\zeta_{jk} f_{ijk} \hat{r}_{ij} + \sum_{ijk} \zeta_{ij}\zeta_{ik}\zeta_{jk} f_{ijk} \hat{r}_{ij} \\
f^A_T &= 0
\end{aligned}
\tag{44}
$$

The first two lines of equation 43 are more difficult to interpret, and so below we find the total force from these terms.

$$
\begin{aligned}
f_T^A =& -\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\Big[\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ik}^{-1}\hat{r}_{ij} + \dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ij}^{-1}\hat{r}_{ik} \\
&\qquad + \dot{\psi}_{j\alpha}^A h_{jik}^{(\alpha)} \dot{f}_{jik}^{(\alpha)} r_{ji}^{-1}\hat{r}_{kj} + \dot{\psi}_{j\alpha}^A h_{jki}^{(\alpha)} \dot{f}_{jki}^{(\alpha)} r_{ji}^{-1}\hat{r}_{kj}\Big] \\
f_T^A =& -\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\Big[\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ik}^{-1}\hat{r}_{ij} + \dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ij}^{-1}\hat{r}_{ik} \\
&\qquad + \dot{\psi}_{k\alpha}^A h_{kji}^{(\alpha)} \dot{f}_{kji}^{(\alpha)} r_{kj}^{-1}\hat{r}_{ik} + \dot{\psi}_{j\alpha}^A h_{jik}^{(\alpha)} \dot{f}_{jik}^{(\alpha)} r_{jk}^{-1}\hat{r}_{ij}\Big] \\
f_T^A =& -\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\Big[\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ik}^{-1}\hat{r}_{ij} + \dot{\psi}_{j\alpha}^A h_{jik}^{(\alpha)} \dot{f}_{jik}^{(\alpha)} r_{jk}^{-1}\hat{r}_{ij} \\
&\qquad + \dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ij}^{-1}\hat{r}_{ik} + \dot{\psi}_{k\alpha}^A h_{kji}^{(\alpha)} \dot{f}_{kji}^{(\alpha)} r_{kj}^{-1}\hat{r}_{ik}\Big] \\
f_T^A =& -\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\Big[\big(\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ik}^{-1} + \dot{\psi}_{j\alpha}^A h_{jik}^{(\alpha)} \dot{f}_{jik}^{(\alpha)} r_{jk}^{-1}\big)\hat{r}_{ij} \\
&\qquad + \big(\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ij}^{-1} + \dot{\psi}_{k\alpha}^A h_{kji}^{(\alpha)} \dot{f}_{kji}^{(\alpha)} r_{kj}^{-1}\big)\hat{r}_{ik}\Big] \\
f_T^A =& -\sum_{\alpha,i,j,k} \zeta_{ij}\zeta_{ik}\zeta_{jk}\Big[f_{ijk}^{(1)}\hat{r}_{ij} + f_{ijk}^{(2)}\hat{r}_{ik}\Big] \\
f_T^A =& 0
\end{aligned}
\tag{45}
$$

where we note that $f_{ijk}^{(1)} = f_{jik}^{(1)}$ and $f_{ijk}^{(1)} = f_{kji}^{(1)}$, leading to a zero total force using similar arguments as in previous examples. We thus see in equation 45 that we can pair terms that cancel out when summed over all atoms, even if they do not immediately appear to do so in equation 43. For clarity, we write the pairs of terms below:

$$
\begin{aligned}
\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ik}^{-1}\hat{r}_{ij} &\leftrightarrow \dot{\psi}_{j\alpha}^A h_{jki}^{(\alpha)} \dot{f}_{jki}^{(\alpha)} r_{ji}^{-1}\hat{r}_{kj} \\
\dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ij}^{-1}\hat{r}_{ik} &\leftrightarrow \dot{\psi}_{j\alpha}^A h_{jik}^{(\alpha)} \dot{f}_{jik}^{(\alpha)} r_{ji}^{-1}\hat{r}_{kj}
\end{aligned}
\tag{46}
$$

We thus must add these terms on the right hand side to atoms j and k when computing the force on atom i. We can't directly apply these terms to the atoms j and k however, due to the $\dot{\psi}_j^A$ term. We thus must permute the indices in order to get a $\dot{\psi}_i^A$ term and indices in the order ijk for the h and f functions.

$$
\begin{aligned}
\dot{\psi}_{j\alpha}^A h_{jik}^{(\alpha)} \dot{f}_{jik}^{(\alpha)} r_{ji}^{-1}\hat{r}_{kj} &\rightarrow \dot{\psi}_{i\alpha}^A h_{ikj}^{(\alpha)} \dot{f}_{ikj}^{(\alpha)} r_{ik}^{-1}\hat{r}_{ji} \\
&\rightarrow \dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ij}^{-1}\hat{r}_{ki} \\
\dot{\psi}_{j\alpha}^A h_{jki}^{(\alpha)} \dot{f}_{jki}^{(\alpha)} r_{ji}^{-1}\hat{r}_{kj} &\rightarrow \dot{\psi}_{i\alpha}^A h_{ijk}^{(\alpha)} \dot{f}_{ijk}^{(\alpha)} r_{ik}^{-1}\hat{r}_{ji}
\end{aligned}
\tag{47}
$$

Finally, we must first permute the indices in the last line of equation 43 so that we can get a term of the form $\dot{\psi}^A_{i\alpha}$.

$$\begin{aligned}\dot{\psi}^A_{j\alpha} f^{(\alpha)}_{jik} h^{(\alpha,3)}_{jik} \hat{r}_{ik} &\rightarrow \dot{\psi}^A_{i\alpha} f^{(\alpha)}_{ikj} h^{(\alpha,3)}_{ikj} \hat{r}_{kj} \\ &\rightarrow \dot{\psi}^A_{i\alpha} f^{(\alpha)}_{ijk} h^{(\alpha,3)}_{ijk} \hat{r}_{jk}\end{aligned} \tag{48}$$

Now that we have determined the terms which cancel out in equation 43, we can more easily develop an algorithm for computing the force on each atom.

$$\begin{aligned}&\forall_i \forall_{j\neq i} \forall_{k\neq i,j} \forall_\alpha \\ &\phi^{(\alpha)}_{ijk} \equiv -\dot{\psi}^A_{i\alpha} h^{(\alpha)}_{ijk} \dot{f}^{(\alpha)}_{ijk} \\ &\eta^{(\alpha,n)}_{ijk} \equiv -\dot{\psi}^A_{i\alpha} f^{(\alpha)}_{ijk} h^{(\alpha,n)}_{ijk} \\ &F^A_i \rightarrow F^A_i + (\phi^{(\alpha)}_{ijk}(r^{-1}_{ik} - \cos(\theta_{ijk}) r^{-1}_{ij}) + \eta^{(\alpha,1)}_{ijk}) \hat{r}_{ij} \\ &F^A_i \rightarrow F^A_i + (\phi^{(\alpha)}_{ijk}(r^{-1}_{ij} - \cos(\theta_{ijk}) r^{-1}_{ik}) + \eta^{(\alpha,2)}_{ijk}) \hat{r}_{ik} \\ &F^A_j \rightarrow F^A_j - (-\phi^{(\alpha)}_{ijk} \cos(\theta_{ijk}) r^{-1}_{ij} + \eta^{(\alpha,1)}_{ijk}) \hat{r}_{ij} - \phi^{(\alpha)}_{ijk} r^{-1}_{ij} \hat{r}_{ik} \\ &F^A_k \rightarrow F^A_k - (-\phi^{(\alpha)}_{ijk} \cos(\theta_{ijk}) r^{-1}_{ik} + \eta^{(\alpha,2)}_{ijk}) \hat{r}_{ik} - \phi^{(\alpha)}_{ijk} r^{-1}_{ik} \hat{r}_{ij} \\ &F^A_j \rightarrow F^A_j + \eta^{(\alpha,3)}_{ijk} \hat{r}_{jk} \\ &F^A_k \rightarrow F^A_j - \eta^{(\alpha,3)}_{ijk} \hat{r}_{jk}\end{aligned} \tag{49}$$

3.5 Training - Energy

In order to use a NNP to simulate a system moving on a given ab-initio potential energy surface, we must first train the NNP for a given set of trajectories and ab-initio energies. While this training procedure mostly follows the procedure for a single neural network, there are some subtle differences which need to be addressed.

We write the energy of a given system in the following manner:

$$\begin{aligned}E_T &= \sum_i \epsilon^{(\alpha)}_i \\ E_T &= \sum_i \psi^{(\alpha)}(x, G^{(\alpha)}_i)\end{aligned} \tag{50}$$

where $\epsilon^{(\alpha)}_i$ is the energy of the i^{th} atom of species α which is determined by the output of the neural network $\psi^{(\alpha)}(x, G^{(\alpha)}_i)$ which is a function of a vector of weights and biases x and a set of inputs $G^{(\alpha)}_i$. We can then write a cost function for a given trajectory:

$$\begin{aligned}C &= \frac{1}{2}\left[\frac{1}{N}\big(E_T - E^{(0)}\big)\right]^2 \\ &= \frac{1}{2}\left[\frac{1}{N}\big(\sum_i \epsilon^{(\alpha)}_i - E^{(0)}\big)\right]^2\end{aligned} \tag{51}$$

where N is the number of atoms in the trajectory and $E^{(0)}$ is the ab-initio energy of the structure. We note that the cost function is defined in terms of the energy per atom rather than the total energy as the total energy, being an intensive property, is not uniquely defined for all trajectories. For example, a given crystal structure and its supercell have identical symmetries and therefore identical symmetry function inputs, but they will have different total energies and therefore different gradients during the training procedure. In particular, supercells with many atoms will tend to dominate the error, leading to cells with fewer atoms having less weight in the training process.

In order to train the neural network $\psi^{(\gamma)}$ we need to find the gradient of the cost function with respect to the parameters $\omega^{(\gamma)}$ associated with the γ species. For each atom we have a unique input and therefore a unique output, and thus for each atom we need the gradient of the cost function with respect to the output $\partial C/\partial a^L$ in order to compute the gradient with respect to the parameters. Since the output for the j^{th} of species γ is the atomic energy $\epsilon_j^{(\gamma)}$, we need the quantity $\partial C/\partial \epsilon_j^{(\gamma)}$.

$$\begin{aligned}
\frac{\partial C}{\partial \epsilon_j^{(\gamma)}} &= \frac{\partial}{\partial \epsilon_j^{(\gamma)}} \frac{1}{2}\left[\frac{1}{N}\big(\sum_i \epsilon_i^{(\alpha)} - E^{(0)}\big)\right]^2 \\
&= \frac{1}{N}\big(\sum_i \epsilon_i^{(\alpha)} - E^{(0)}\big)\frac{1}{N}\frac{\partial}{\partial \epsilon_j^{(\gamma)}}\sum_k \epsilon_k^{(\alpha)} \\
&= \frac{1}{N^2}\big(\sum_i \epsilon_i^{(\alpha)} - E^{(0)}\big)\sum_k \frac{\partial \epsilon_k^{(\alpha)}}{\partial \epsilon_j^{(\gamma)}} \\
&= \frac{1}{N^2}\big(\sum_i \epsilon_i^{(\alpha)} - E^{(0)}\big)\sum_k \delta_{j,k}\delta_{\alpha,\gamma} \\
\frac{\partial C}{\partial \epsilon_j^{(\gamma)}} &= \frac{1}{N^2}\big(\sum_i \epsilon_i^{(\alpha)} - E^{(0)}\big)\delta_{\alpha,\gamma}
\end{aligned} \tag{52}$$

We can thus treat each atom of species γ as an independent training sample for the neural network for species γ, each of which however has the same $\partial C/\partial a^L$ given in equation 52.

3.6 Training - Forces

In addition to training with respect to the energy we can also define a cost function which is a function of the energies and forces of a given structure:

$$C = \frac{1}{2}\left[\frac{1}{N}(E_T - E^{(0)})\right]^2 + \sum_i \frac{1}{2}\left[\frac{\beta}{N}\big(F_i^{(\alpha)} - F_i^{(0)}\big)\right]^2 \tag{53}$$

where β is a parameter which reduces the impact of the forces as they will contribute $3N$ terms to the error as opposed to the single term contributed by the total energy.

Rather than use the typical backpropagation algorithm, we will use an alternative approach where we will take the derivative of the cost function with respect to the NN pa-

rameters. The reason for this is that the error function is no longer a direct function of the output, and so we cannot express the gradient of the parameters in terms of the gradient of the cost function.

$$\begin{aligned}
\frac{\partial C}{\partial \omega_k^{(\gamma)}} &= \frac{\partial}{\partial \omega_k^{(\gamma)}} \frac{1}{2}\left[\frac{1}{N}(E_T - E^{(0)})\right]^2 + \frac{\partial}{\partial \omega_k^{(\gamma)}} \sum_i \frac{1}{2}\left[\frac{\beta}{N}\big(F_i^{(\alpha)} - F_i^{(0)}\big)\right]^2 \\
&= \frac{\partial}{\partial \omega_k^{(\gamma)}} \frac{1}{2}\left[\frac{1}{N}(E_T - E^{(0)})\right]^2 + \frac{\partial}{\partial \omega_k^{(\gamma)}} \sum_i \frac{1}{2}\left[\frac{\beta}{N}\big(\frac{\partial E_T}{\partial_{r_i}} - F_i^{(0)}\big)\right]^2 \\
&= \frac{1}{N^2}(E_T - E^{(0)})\frac{\partial E_T}{\partial \omega_k^{(\gamma)}} + \sum_i \frac{\beta^2}{N^2}\big(\frac{\partial E_T}{\partial_{r_i}} - F_i^{(0)}\big)\frac{\partial^2 E_T}{\partial \omega_k^{(\gamma)} \partial_{r_i}} \\
&= \frac{1}{N^2}(E_T - E^{(0)})\sum_i \frac{\partial \epsilon_i^{(\alpha)}}{\partial \omega_k^{(\gamma)}} + \sum_i \frac{\beta^2}{N^2}\big(F_i^{(\alpha)} - F_i^{(0)}\big)\sum_j \frac{\partial^2 \epsilon_j^{(\alpha)}}{\partial \omega_k^{(\gamma)} \partial_{r_i}} \\
&= \sum_i \left[\frac{1}{N^2}\big(E_T - E^{(0)}\big)\frac{\partial \epsilon_i^{(\alpha)}}{\partial \omega_k^{(\gamma)}} + \frac{\beta^2}{N^2}\big(F_i^{(\alpha)} - F_i^{(0)}\big)\sum_j \frac{\partial^2 \epsilon_j^{(\alpha)}}{\partial \omega_k^{(\gamma)} \partial_{r_i}}\right]
\end{aligned} \tag{54}$$

We take special note of the last term:

$$\sum_j \frac{\partial^2 \epsilon_j^{(\alpha)}}{\partial \omega_k^{(\gamma)} \partial_{r_i}} = \frac{\partial}{\omega_k^{\gamma}} \frac{\partial}{\partial r_i} \sum_j \epsilon_j^{(\alpha)} = \frac{\partial}{\omega_k^{\gamma}} \frac{\partial}{\partial r_i} E_T \tag{55}$$

Thus, we are essentially taking the gradient with respect to $\omega_k^{(\gamma)}$ of the net force on the i^{th} atom. Note that for each parameter $\omega_k^{(\gamma)}$ of the network the gradient of the gradient of the output with respect ot the inputs will produce a vector with the same dimension as the input, similar to the first derivative of the gradient of the output with respect to the input. We can thus use the same methods and algorithms derived for computing forces to efficiently calculate the term in equation 55. For the gradient with respect to each individual parameter, the math is exactly the same, except the first derivative of the neural network with respect to the input is replaced by the gradient of this quantity with respect to the given parameter.