

Radial Density Encoding of Molecular Environments:
Neural Networks and Spherical DFT

Undergraduate Senior Thesis
The University of New Mexico

by
Sol Samuels
Department of Physics and Astronomy
University of New Mexico
May 2025

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Susan R. Atlas, for all the time spent guiding me toward becoming a confident researcher. Your care and encouragement kept me writing, far more than I thought I was capable of doing.

I would like to thank those in the Atlas research group, including Chance Baxter, Surya Bijjala, Aung Zaw Htut, and Saul Hernandez. I acknowledge aid in conducting electronic structure calculations for electron densities. I want to extend thanks to Logan Ross for conversations which inspired a solution to the hydrogen cation dilemma in this work. I would also like to thank Godwin Amo-Kwao for a great body of work that inspired much in this thesis and for inherited code used in this work.

I am grateful for the UNM Department of Physics and Astronomy and the UNM Department of Chemistry and Chemical Biology for research support. I also gratefully acknowledge internship support from NSF REU grant #PHY-1659618 (PIs: Prof. Jim Thomas, Prof. David Dunlap) and support from the 2023 UNM Rayburn Reaching Up Fund which enabled me to present at the 2024 APS March Meeting. I would like to thank the UNM Center for Advanced Research Computing, supported in part by the NSF, for providing the research computing resources used in this work.

Finally, I want to thank all of my friends and family who have helped me while conducting this research and pursuing my degree. I especially want to thank Leif Nelson for all the time spent waiting for me to get out of late meetings so I could get a safe ride home.

Radial Density Encoding of Molecular Environments: Neural Networks and Spherical DFT

Abstract

The molecular environment is defined by complex quantum chemical relationships that are challenging to model, especially for large systems. This work introduces a scalable neural network-based framework for representing molecular environments using compact, physically-derived, radially symmetric atomic electron densities. Motivated by an ensemble atom-in-molecule (AIM) formalism and built on the foundation of density functional theory (DFT), we develop a radial basis function neural network (RBF-NN) that characterizes the electron density of a molecule as a superposition of spherical, atom-centered basis functions derived from isolated atomic charge and excited states. We demonstrate that the RBF-NN can resolve charge transfer in diatomic systems, including LiF and HF, tracking changes in bonding character as a function of internuclear separation, and highlighting how quantum entanglement between atoms is encoded within the global electron density. Finally, we introduce a formal extension to the spherical DFT analog of the Hohenberg-Kohn theorem of density functional theory, by taking advantage of mathematical methods of distance geometry.

TABLE OF CONTENTS

1	Introduction and Background	1
1.1	Density Functional Theory (DFT)	2
1.2	Atom-in-Molecule Decompositions	5
1.2.1	Bader's Topological Approach	6
1.2.2	The Hirshfeld and Iterative Hirshfeld Atoms-in-Molecule	8
1.2.3	The Ensemble Atom-in-Molecule Framework	9
2	Radial Basis Function Neural Networks (RBF-NN)	13
2.1	Theoretical Foundation	13
2.2	Radial Basis Functions for Representing Radial Densities	16
2.3	Physical Constraints in the Loss Function	19
2.4	Hydrogen and the Special Case of H^+	21
3	Computational Implementation of the RBF-NN	24
3.1	Design of the RBF-NN	24
3.2	Computational Methods and Tools	26
3.3	Training the RBF-NN	27
3.3.1	Loss Function Design	29
3.3.2	Implementing the ADAM Optimizer	31
3.3.3	Batching in RBF-NN Training	32
3.3.4	Early Stopping Routine	33
3.4	Training as a Function of Internuclear Separation	34
4	Application to Various Molecular Systems	36
4.1	LiF	36
4.1.1	Comparison with SWE Results of Varandas	39
4.1.2	LiF RBF-NN Analysis at Various Levels of Theory	42
4.2	HF	44
4.2.1	Ladder Implementation of Hydrogen Excited States	45
4.2.2	Evaluation of the H^+ State Implementation	50
5	Spherical DFT and Relation to an AIM formulation	53
5.1	Nuclear Location Reconstruction From Sphericalized Densities	55
5.2	A Sphericalized Set of Densities as an AIM	59
5.3	Sphericalization Used in the Ensemble AIM Formulation	62
6	Conclusions and Future Work	63
	References	66
A	Python Code for Implementing the RBF-NN	75
A.1	Neural Network Configuration	75
A.2	Loss and Early Stopping Functions	80
A.3	Calling the RBF-NN	82

B Python Code for Evaluating the Analytic Forms of the Radial Basis Functions	83
B.1 Analytical Fitted Models	83
B.2 Exact Hydrogen Radial Densities	84

1 Introduction and Background

It is often the case in describing physical phenomena that there is a tradeoff between the accuracy of a model and the generality with which it can be applied. This is especially true when dealing with quantum mechanical systems. For small chemical systems, such as isolated atoms and small molecules, wavefunction-based methods—i.e. solving the N -particle Schrödinger wave equation, Eq. (1)—can yield highly accurate results.

$$\mathcal{H}\Psi(\vec{r}_1, \vec{r}_2 \dots \vec{r}_N) = E\Psi(\vec{r}_1, \vec{r}_2 \dots \vec{r}_N). \quad (1)$$

However, these methods become too computationally costly for large systems due to how they scale. This comes down to the fact that to solve for the wavefunction $\Psi(\vec{r}_1, \vec{r}_2 \dots \vec{r}_N)$ of a system of N electrons, the solution of a $3N$ -dimensional equation is required (e.g., 30 dimensions are required to represent a 10 electron system) [1]. The issue of scalability becomes even more significant when considering systems with multiple atoms, each with its associated electrons. One of the most basic wavefunction methods which does not include a representation for quantum mechanical correlation effects, single-determinant Hartree-Fock, scales as $O(N^4)$ [2]. For even more complex wavefunction methods, including the post-Hartree Fock, Møller–Plesset perturbation theory methods MP2 and MP4, which add corrections for electron correlation, this scaling becomes even more unfavorable, $O(N^7)$ [3]. As the number of electrons grows in a system, the number of determinants that need to be included in the wavefunction representation increases exponentially [3]. Thus, if modeling a protein is desired, a system composed of tens of thousands of atoms, wavefunction methods are no longer feasible. Alternative, more scalable methods are required.

One pathway toward improving scalability is to reconsider the structure of the representation itself. In atomic systems, the electron density exhibits a high degree of radial symmetry around each nucleus, with asymptotic constraints applicable at both short and

long range radial distances from the atomic nucleus [4, 5]. By focusing on this symmetry and incorporating angular corrections, it becomes possible to reduce the complexity of a molecular system without discarding chemically relevant information. A representation built from radially symmetric components can compress the essential features of a molecular system. This not only lessens the computational burden but also opens the door to more interpretable models. In contrast to methods that treat the electron density as a general function of three spatial dimensions [6], the radial framework allows for localized descriptions that naturally scale with the number of atoms rather than the size of a surrounding grid. As system size increases, this shift in perspective becomes increasingly advantageous. An analysis must then be conducted in order to determine if such radial representations can be as successful as traditional, 3D wavefunction methods, in representing the molecular environment.

To carry out this analysis, we explore the use of an ensemble atom-in-molecule formulation, founded in density functional theory, which decomposes a total molecular density distribution into a sum of radially symmetric atomic state densities. We also investigate spherical density functional theory and show how, through established methods in distance geometry, 3D nuclear location information is retained despite sphericalization of the total molecular density distribution.

1.1 Density Functional Theory (DFT)

Density functional theory (DFT) is a quantum mechanical method used to study the electronic structure of many-electron systems, including atoms, molecules and solids. Unlike wavefunction methods, its computational complexity scales as $O(N^3)$ [7]. DFT relies on the Hohenberg-Kohn (HK) theorem which states that the ground-state energy of a many-electron system is uniquely determined by its electron density, $\rho(\vec{\mathbf{r}})$ [8]. This replaces the solution of the Schrödinger wave equation (SWE) with the minimization of a *universal*

energy functional $E_v[\rho]$ with respect to the density [8]. In practice, DFT is implemented by solving the coupled, single-particle Kohn-Sham equations [9], which are formally equivalent to the original HK energy functional minimization:

$$\left[-\frac{\hbar^2 \nabla^2}{2m} + v_{\text{eff}}(\mathbf{r}) \right] \phi_i(\mathbf{r}) = \epsilon_i \phi_i(\mathbf{r}), \quad i = 1, \dots, N, \quad (2)$$

where $v_{\text{eff}}(\mathbf{r})$ is given by:

$$v_{\text{eff}}(\mathbf{r}) = v_{\text{ext}}(\mathbf{r}) + v_{\text{H}}(\mathbf{r}) + v_{\text{xc}}(\mathbf{r}). \quad (3)$$

Though the coupled equations resemble the SWE, the single-particle equations involve N Kohn-Sham orbitals, $\phi_i(\mathbf{r})$, with their corresponding orbital energies, ϵ_i . The total density of the N -particle system is related to the Kohn-Sham orbitals as:

$$\rho(\mathbf{r}) = \sum_{i=1}^N |\phi_i(\mathbf{r})|^2. \quad (4)$$

As seen in Eq. (3), the Kohn-Sham potential, $v_{\text{eff}}(\mathbf{r})$, consists of three terms: the external potential, $v_{\text{ext}}(\mathbf{r})$, the Hartree potential, $v_{\text{H}}(\mathbf{r})$, and the exchange-correlation potential, $v_{\text{xc}}(\mathbf{r})$. The external potential is defined by the configuration of the atoms in the system, each with nuclear charge Z_i and nuclear location \mathbf{R}_i , according to:

$$v_{\text{ext}}(\mathbf{r}) = \sum_{i=1}^{N_A} \frac{-Ze^2}{|\mathbf{r} - \mathbf{R}_i|}. \quad (5)$$

The Hartree potential is the classical electron electrostatic potential:

$$v_{\text{H}}(\mathbf{r}) = \int \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}'. \quad (6)$$

The exchange-correlation potential includes statistical exchange effects due to the Pauli exclusion principle and electron correlation effects, reflecting quantum spatial interactions due to the electron-electron Coulomb interaction. Unlike the external poten-

tial and the Hartree potential, the exchange-correlation potential is not known explicitly. The exchange-correlation potential is related to the exchange-correlation energy functional, $E_{xc}[\rho]$, which is a component of the total energy functional in Kohn-Sham density functional theory. It is defined as the functional derivative of $E_{xc}[\rho]$ with respect to the electron density:

$$v_{xc}(\mathbf{r}) = \frac{\delta E_{xc}[\rho]}{\delta \rho(\mathbf{r})}. \quad (7)$$

The exchange-correlation potential is precisely the component of the original SWE which describes the quantum mechanical behavior of the interacting electrons that DFT must replace. DFT as a formal theory provides an exact quantum mechanical solution for the properties of a many-electron molecular system just as the SWE does; however, in practice, approximation of the exchange-correlational potential is required.

The earliest approximation to $E_{xc}[\rho]$, proposed in the original HK paper [9], was the local density approximation (LDA), which models $E_{xc}[\rho]$ as a locally homogeneous electron gas. Numerous refinements and alternative approximations have been proposed, including the celebrated *Jacob's ladder* family of functionals introduced by Perdew and coworkers [10]. These functionals are based on successively higher-order gradients of the electron density and may include a fraction of some exact, Hartree-Fock exchange [11]. Functionals of this type are referred to as ‘hybrid’ [12]. The functionals utilized in the present work include B3LYP [13, 14], PBE0 [15], and wB97XD [16, 17].

Despite the unavoidable approximation of the exchange-correlation potential, DFT has been shown to yield remarkably accurate results for diverse, many-electron systems at much lower computational cost than the solution of the SWE. Unlike solving the SWE, DFT enables the accurate calculation of properties of both large molecules and periodic solids. However, despite its widespread use and constant improvements, current approximations for the exchange-correlation potential still struggle to accurately describe certain classes of systems and properties, including charge transfer and bond formation and breaking [12]. SWE methods that are more successful at modeling these properties do not scale

to large chemical systems with more than 10 atoms. Work done in this thesis seeks to address this issue through the application of an atom-in-molecule ensemble representation [18] to molecular system, with radial basis functions motivated in part by spherical DFT.

1.2 Atom-in-Molecule Decompositions

Fundamental to chemistry and materials science is the knowledge that molecules are comprised of atoms. Understanding a molecule’s properties depends on the electronic context of the atoms that constitute it. However, there is no universally accepted definition for what defines an atom within a molecule [19–21]. Different atom-in-molecule decomposition methods have been proposed for decomposing molecules into chemically-reasonable atomic subsystems, as illustrated schematically for a heteronuclear diatomic in Fig. 1.

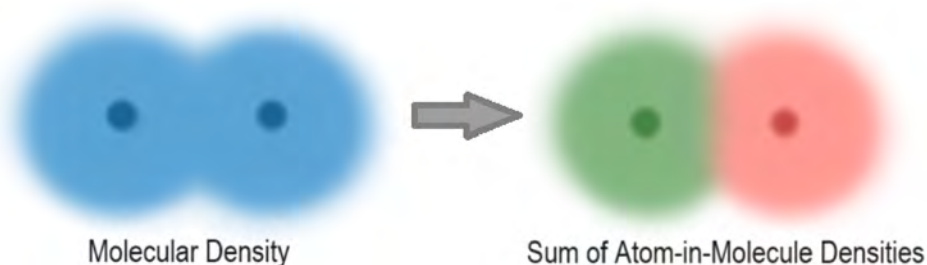


Figure 1: Schematic of an atom-in-molecule decomposition for a simple diatomic.

In general, implementing an atom-in-molecule decomposition involves partitioning the total molecular density, $\rho_{\text{mol}}(\vec{\mathbf{r}})$, into multiple *pseudoatoms* in such a way that no density is double-counted. The sum over all psuedoatom densities $\rho_i^*(\vec{\mathbf{r}})$ must equal the original total molecular density $\rho_{\text{mol}}(\vec{\mathbf{r}})$ at every point $\vec{\mathbf{r}}$ in space [20]:

$$\rho_{\text{mol}}(\vec{\mathbf{r}}) = \sum_{i=1}^{N_{\text{atoms}}} \rho_i^*(\vec{\mathbf{r}}). \quad (8)$$

In this way, the total electron density and total charge of the molecular system are

conserved, while allowing density to be assigned to a specific pseudoatom. The decomposition into pseudoatom components provides insight into quantum mechanical information contributed by each atom in the overall system.

It is important to note that this partitioning is not unique, and the properties by which the pseudoatoms are defined varies depending on the choice of atom-in-molecule method. Since an atom-in-molecule is not a physically-observed object but a conceptual tool, certain partitioning schemes may be more useful in specific contexts [20]. Some well-known atom-in-molecule decompositions base the partitioning scheme on atomic orbitals [22, 23], spatial positioning [24], or an analysis of the electron density distribution [25, 26]. The following subsections review atom-in-molecule methods which utilize the electron density as the foundation for their pseudoatom partitioning. As the electron density distribution determines the ground state energy and all quantum mechanical observables in DFT [8], the partitioning scheme described in this work uses the electron density as its foundation.

1.2.1 Bader’s Topological Approach

One commonly-used atom-in-molecule decomposition is the Bader formulation, which defines the atom-in-molecule using a *topological* approach [25, 27]. Bader’s atom-in-molecule implements a hard spatial partitioning that defines electron density subsystems for each atom in the molecule. These partitions, known as *basins of attraction*, are shaped by the pull of the various nuclear charges which direct the surrounding electron density toward each nucleus. As is illustrated in Fig. 2(b), the molecule is partitioned with hard dividing lines which trace the valleys of the density’s topology. Given a molecular density $\rho_{\text{mol}}(\vec{\mathbf{r}})$, atoms are assigned to subsystems by partitioning 3D space according to *zero-flux surfaces* at which $\nabla\rho_{\text{mol}}(\vec{\mathbf{r}}) \cdot \hat{\mathbf{n}} = 0$ [19]. A 2D example of this partitioning along a zero-flux surface is seen in Fig. 2(a).

Once the zero-flux surfaces have been used to assign each atom i to its unique region

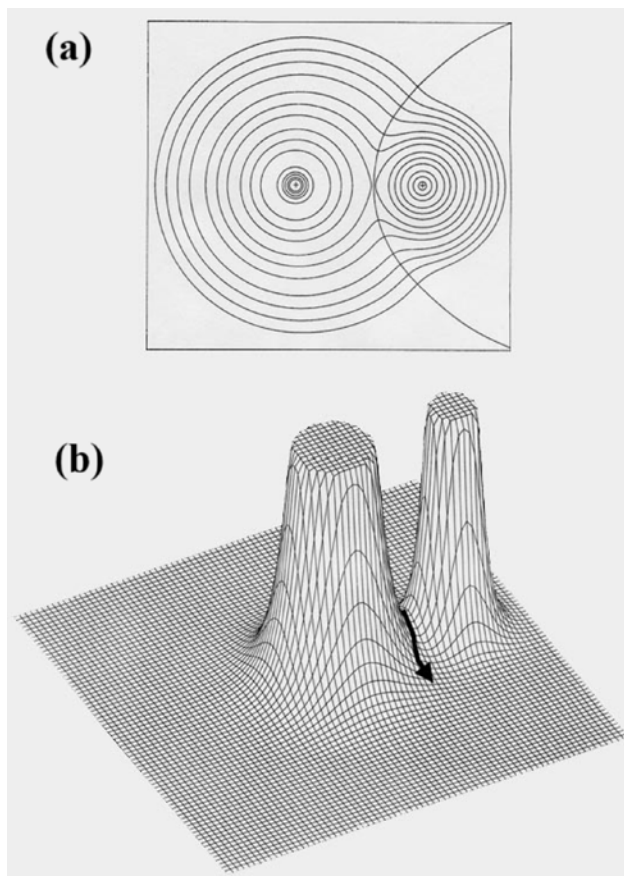


Figure 2: Bader’s topological atom-in-molecule decomposition of the electron density of LiF, with 2D (a) and 3D (b) perspectives. The zero-flux surface is depicted by a black solid line in (a). From [27].

of space, Ω_i , the partial charge, q_i , on the atom can be computed as:

$$q_i = Z_i - \int_{\Omega_i} \rho_i^*(\vec{\mathbf{r}}) d\vec{\mathbf{r}}, \quad (9)$$

where Z_i is the nuclear charge of the atom. Since the zero-flux surfaces are physically-based and determined from the true electron density distribution, Bader’s atoms-in-molecule definition has been found to be successful in modeling charge transfer [28]. It also results in more chemically-reasonable charge assignments than other methods, taking chemical electronegativity into account [29].

Although visually elegant and physically-based, the mathematical implementation of Bader’s technique [30] is computationally intensive. Calculating the Laplacian at every

point in space becomes unfeasible for very large systems. In general, due to the global molecular perspective needed to perform the spatial partitioning, Bader’s method can only be applied to relatively small systems [30]. Additionally, by assigning electrons to ‘belong’ to specific atoms in the molecule, Bader’s method is inconsistent with our physical understanding of electron indistinguishability and entanglement in the larger molecular system. In this respect, the hard spatial partitioning imposed by Bader’s method is limited in its representation of a chemical environment.

1.2.2 The Hirshfeld and Iterative Hirshfeld Atoms-in-Molecule

The Hirshfeld partitioning method [26] functions by assigning a fraction of the density to each atom at each point in space in a molecular system. This approach is often referred to as the “stockholder” partitioning method because it distributes the total electron density based on the proportion each atom contributes — analogous to a shareholder receiving a fraction of a company’s total assets. For a bonded atom i in a molecule with total density $\rho(\mathbf{r})$, the i th Hirshfeld atom-in-molecule density, or *proatom density*, is defined as:

$$\rho_i(\mathbf{r}) = \left[\frac{\rho_i^0(\mathbf{r})}{\rho^0(\mathbf{r})} \right] \rho(\mathbf{r}), \tag{10}$$

where $\rho^0(\mathbf{r})$ is the *promolecular density*, the sum of the isolated atom densities $\rho_i^0(\mathbf{r})$ at each point in space:

$$\rho^0(\mathbf{r}) = \sum_{i=1}^{N_A} \rho_i^0(\mathbf{r}). \tag{11}$$

Hirshfeld is a relatively simple atom-in-molecule method and thus can scale to large systems. However, the use of isolated, neutral atomic densities in the promolecular reference means that the method cannot capture charge transfer effects between atoms [31]. To address this limitation, Bultinek, Ayers and co-workers proposed the *Iterative Hirshfeld* (Hirshfeld-I) method [20, 32]. In this approach, the initial Hirshfeld atomic densities are

iteratively refined by including a, charge-dependent reference state in the proatom density representation.

Iterative Hirshfeld models charge transfer by allowing atomic densities to adjust in response to their environment, providing more physically meaningful charge distributions compared to the original method. However, like Bader’s decomposition, it requires a holistic decomposition of the total density at each iteration, thus limiting scalability. In addition, the method has been found to exhibit instabilities and yield unphysical atom-in-molecule densities in test systems [20, 33]. Consequently, Heidar-Zadeh and co-workers have recently proposed a modified, *variational* Hirshfeld partitioning approach [33]. In this method, the proatoms are constructed as a linear combination of basis densities, with weights optimized by minimizing the divergence between the promolecular and true molecular densities. In the initial instantiation of the model, the authors utilize the Kullback-Liebler divergence measure, as in the derivation of the original Hirshfeld method by Nalewajski and Parr [34]. This has the effect of minimizing the distortion of the proatom densities away from their isolated atom counterparts [33], in contrast to the DFT ensemble approach utilized here.

1.2.3 The Ensemble Atom-in-Molecule Framework

To address the goal of representing charge transfer and charge distortion with an atom-in-molecule formulation that can be scaled to large systems an *ensemble DFT framework* has been proposed by Atlas, Valone, and co-workers as the basis for an atom-in-molecule decomposition [4, 18, 35]. This involves constructing weighted superpositions of isolated electron densities for individual ground, excited and charge states of all constituent atoms of the molecule [5]. This is visualized in Fig. 3 for an exemplar atom A , where the density contributions of individual states, like the neutral (A^0), cation (A^{1+}), anion (A^{1-}), and excited states (A'), are combined to create an ensemble superposition. All of these states

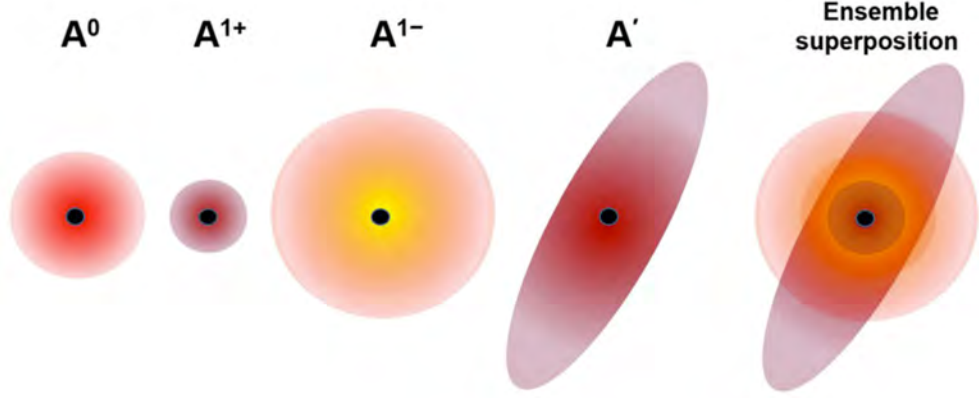


Figure 3: Schematic representation of the superposition of charge and excited states contributing to the pseudoatom density in the ensemble atom-in-molecule framework. Figure from [18].

belong to the same atom, making the ensemble formulation an atomistic view of a molecular system.

The ensemble atom-in-molecule satisfies the defining atom-in-molecule constraint given in Eq. (8), where the i th atom's contributing pseudoatom density, $\rho_i^*(\vec{\mathbf{r}})$, is given by [18]:

$$\rho_i^*(\vec{\mathbf{r}}) = \sum_{j=-\infty}^{Z_i-1} \alpha_{ij} \varrho_{ij}(\vec{\mathbf{r}}), \quad (12)$$

where the i th pseudoatom density $\rho_i^*(\vec{\mathbf{r}})$ is determined by the contributions of the j th charge state density, $\varrho_{ij}(\vec{\mathbf{r}})$, with corresponding weight α_{ij} . The excited state ensemble densities $\varrho_{ij}(\vec{\mathbf{r}})$ for each ion are defined as:

$$\varrho_{ij}(\vec{\mathbf{r}}) = \sum_{k=1}^{\infty} \beta_{ijk} \rho_{ijk}(\vec{\mathbf{r}}), \quad (13)$$

where $\rho_{ijk}(\vec{\mathbf{r}})$ is the density of the k th eigenstate of the j th ion of atom i . Thus, $\rho_i^*(\vec{\mathbf{r}})$ can be expressed as:

$$\rho_i^*(\vec{\mathbf{r}}) = \sum_{j,k} \omega_{ijk} \rho_{ijk}(\vec{\mathbf{r}}), \quad (14)$$

where

$$\omega_{ijk} = \alpha_{ij}\beta_{ijk}. \quad (15)$$

By optimizing the ω_{ijk} in the superposition [21], the total density $\tilde{\rho}(\vec{\mathbf{r}})$ of the molecular system in the ensemble formulation is given by:

$$\tilde{\rho}(\vec{\mathbf{r}}) = \sum_{i=1}^{N_A} \rho_i^*(\vec{\mathbf{r}}). \quad (16)$$

Note that, in contrast to the Bader and Hirshfeld decompositions, the ensemble AIM formulation can only approximate the total density, $\tilde{\rho}(\vec{\mathbf{r}}) \approx \rho(\vec{\mathbf{r}})$. This is because it relies on fixed atomic basis densities, $\rho_{ijk}(\vec{\mathbf{r}})$, combined through ensemble weights. As a consequence, interatomic electric correlations are not explicitly described by the superposition [18]. Nevertheless, as will be seen in this present work, the ensemble formulation has proven remarkably successful in representing bond formation and breaking in the initial systems to which it has been applied.

The approximation for the total density does allow for the calculation of the effective charges of individual atoms in the molecule, which can then be compared across atom-in-molecule partitioning methods. Analogous to Eq. (9), the effective charge is calculated as:

$$q_i = Z_i - \int \rho_i^*(\vec{\mathbf{r}}) d\mathbf{r}, \quad (17)$$

where the atom-in-molecule density is now integrated over all space. Note that the ensemble formulation allows electron density to be ‘shared’ among all atoms in the molecule, rather than being subject to strict spatial assignment, as is the case in Bader’s approach.

In general, the calculation of the effective charge can be used in the assessment of the ability for an atom-in-molecule method to model charge transfer. The effective charges on each atom can be calculated and compared to effective charges predicted by other atom-in-molecule formulations.

In practice, the basis densities used in the ensemble formulation are spherically averaged, resulting in densities that are simple functions of the radial distance from each nuclear center. The superposition of states that defines the total molecular density, $\tilde{\rho}(\vec{\mathbf{r}})$, is therefore composed of a sum of radial functions. This is a further approximation to the total density in addition to the ensemble representation of Eq. (14). The rationale for introducing this approximation is discussed in Sections 2 and 5 below.

The motivation for modeling the electron density extends beyond evaluating effective charges. The electron density, $\rho(\vec{r})$, is a fundamental observable in quantum mechanics that determines the ground-state energy of a many-electron system [8, 36]. It is directly related to the many-electron wavefunction through the relation:

$$\rho(\vec{r}) = N \int |\Psi(\vec{r}, \vec{r}_2, \dots, \vec{r}_N)|^2, d\vec{r}_2 \dots d\vec{r}_N, \quad (18)$$

where the integration is performed over all electron coordinates except one to yield the probability density distribution for finding an electron at position \vec{r} [36]. This density contains all the information needed to determine the chemical and physical properties of a system composed of atoms and electrons, including structure, reactivity, and bonding. As such, it is important for modeling dynamical force fields [18]. Additionally, because $\rho(\vec{r})$ is a measurable quantity in experiments such as nuclear magnetic resonance (NMR) [37], building accurate, interpretable models of the density has relevance well beyond theory. In this context, the ensemble AIM framework provides a way to represent and interpret molecular electron densities across systems, offering both physical insight and practical utility.

2 Radial Basis Function Neural Networks (RBF-NN)

2.1 Theoretical Foundation

In order to construct the ensemble representation of a given molecular density, determine the corresponding ensemble weights of each contributing atom, and compute effective charges according to Eq. (17), we utilize a *Radial Basis Function neural network*. Neural networks (NNs) are computational models composed of interconnected nodes organized in layers, originally inspired by the structure and function of neurons in the human brain [38]. Each artificial neuron in a neural network receives input signals, processes them through weighted connections and an *activation function*, and produces an output that is transmitted to other neurons, thereby mimicking the way biological neurons transmit information. The movement of information in a feedforward NN design is illustrated in Fig. 4, where input data is propagated sequentially through successive layers toward the output. During the training process, the network learns by adjusting the weights on each connection, typically using *backpropagation* [39, 40], a method that computes an error function, or *loss function*, with respect to the network weights, and propagates these errors backward through the network layers to update the weights accordingly [38]. This loss function evaluates the accuracy of optimized weights by determining the difference between a current prediction and a target, *training set* of data. NNs are widely used for tasks including classification and regression, where traditional algorithms struggle to capture nonlinear and high-dimensional relationships [41].

Radial Basis Function neural networks (RBF-NNs) are a special type of neural network that utilize a single hidden layer. As their name suggests, RBF-NNs employ radial functions as the foundation for their node activation functions, which define the output of each node. RBF-NNs have been shown to be *universal function approximators*: a feedforward network with a single hidden layer of non-constant, bounded, and continuous ac-

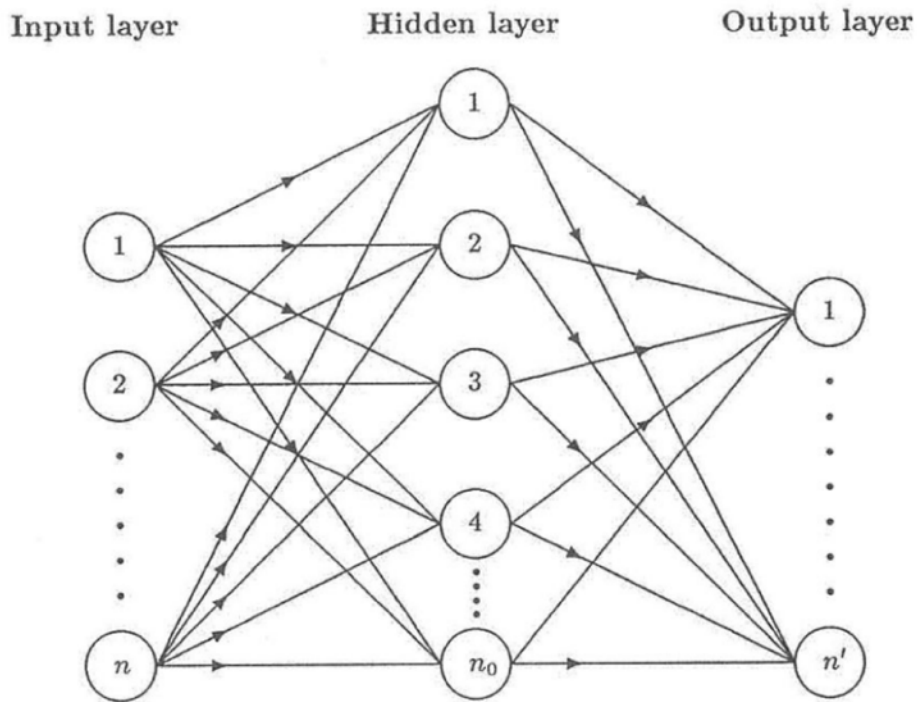


Figure 4: Schematic of a feedforward neural network. Information is input through the node layer on the left and processed in a hidden layer(s) with nodes $1 \dots n_0$ and weighted interconnections indicated by arrows. The output is then given through an output layer. After [41].

tivation functions can approximate any continuous function on a compact subset of \mathbb{R}^n to arbitrary precision, provided that a sufficient number of neurons are used [42, 43]. This result implies that RBF-NNs, despite their architectural simplicity, are capable of learning highly nonlinear mappings by effectively projecting input data onto a *latent space*.

Most recently, when machine learning is used or discussed in the literature, the subject is so-called *deep* neural networks. Deep neural networks (DNNs), including modern architectures such as transformer networks [44], consist of many hidden layers with nonlinear activation functions [45]. In DNNs, latent spaces are constructed progressively across many layers, with each layer transforming its input into more abstract representations that capture complex patterns and relationships [46]. This depth enables DNNs to model highly nonlinear and global structures in data but comes at the cost of increased computational complexity and a large number of parameters that require extensive training. These

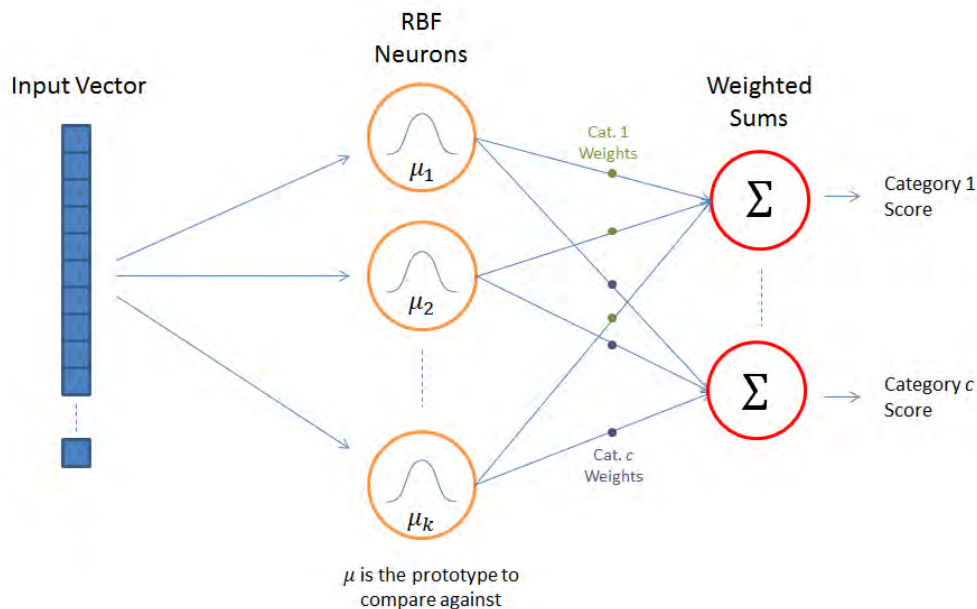


Figure 5: Example of an RBF-NN structure. Here Gaussians are used as the activation functions for the RBF neurons, representing radial information. Figure from [50].

DNNs also require extremely large datasets for training, as exemplified by the transformer-based AlphaFold model of protein folding [47, 48].

In contrast, RBF-NNs approach both network design and latent space construction in a fundamentally different way. Instead of stacking multiple layers to create hierarchical representations, RBF-NNs employ a single hidden layer made up of radial basis functions centered at specific 3D points defined by the input data. In this way, RBF-NNs involve localized latent spaces, or embedding spaces, that encode distance-based relationships. Each neuron in an RBF-NN activates according to the distance between the input and the node centers [49]. The use of a single layer significantly reduces computational and architectural complexity of the NN design in comparison to DNNs. Most commonly, RBF-NN utilize Gaussian basis functions at their nodes, as illustrated in Fig. 5. However, other radial basis functions, including exponential decay functions of the form $\alpha \exp(-\beta r)$ can also be used as activation functions.

Radial basis functions also differ from DNNs in how training and testing are used to determine information regarding a data set. Classically, DNNs are used in research by

first *training* the network, and calibrating its hyperparameters, on an extensive training set. After the network has learned from this training set, and the neural network hyperparameters have been fixed, it can be applied to data outside the training set. This way, the DNN learns underlying patterns and structure in the training set data and can then apply this understanding to any new, unseen data. This differs from an RBF-NN working as a universal function approximator, as is the case in this work. The focus is on training the RBF-NN to analyze and extract information from the training set itself, rather than developing a model for generalization beyond it. Unlike DNNs, where the primary goal is to optimize a representation that can extend beyond the training data, the purpose of training an RBF-NN is to construct an explicit functional mapping that encapsulates the information contained in the training set.

2.2 Radial Basis Functions for Representing Radial Densities

The RBF-NN architecture provides a compact, physics-based representation of a molecule. Each node’s activation function is analytically fitted to the spherically-averaged quantum mechanical electron density distribution of a particular atomic state [4] — neutral, cation, anion, or excited — thus embedding prior knowledge of atomic behavior directly into the network architecture. For instance, Fig. 6 displays the spherically-averaged electron density distribution for the charge states of lithium and fluorine. The procedure required to generate these radial distribution functions will be detailed in Section 3.2.

The spherically-averaged electron density distributions, as illustrated in Fig. 6, are fit to radial distribution functions of the form [4, 5]:

$$M(r) = A_0 e^{-2Zr} + B_0 r^{2(\beta+\zeta r)} e^{-2\alpha r} + C_0 (m - Zr)^2 e^{-2\gamma r} + D_0 Z^2 r^2 e^{-2\eta r}. \quad (19)$$

These analytical fits to the radial distributions of the electron densities for each state of each atom in a molecule are then used as the activation functions at each node in the

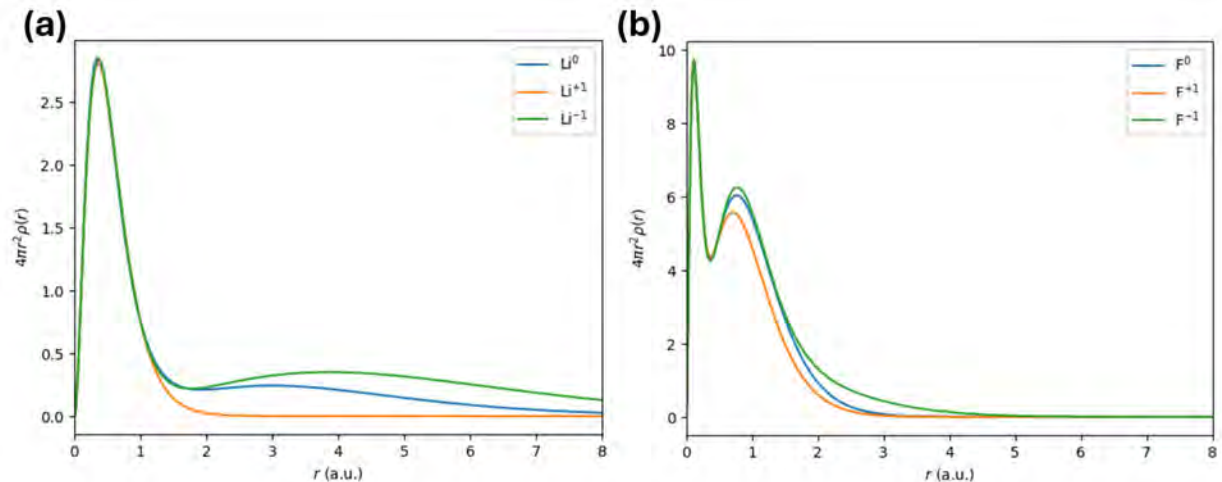


Figure 6: Radial distribution functions of the neutral and charge states of (a) lithium and (b) fluorine. From [4, 5].

RBF-NN. They are shifted spatially to be centered on the nuclear locations of each atom in the molecule. The centering of the RBF activation functions arises naturally due to the physical nature of the problem, eliminating the need for additional RBF center identification techniques commonly required in standard RBF-NNs. In conventional applications, RBF-NNs rely on methods such as *k-means clustering* to determine the optimal placement of the basis function centers, as their locations are not inherently known [51, 52]. Poor center selection can lead to inefficient function approximation and requires additional processing of training data. However, in the present case, the radial basis functions are by definition centered on the nucleus of each atom. This removes the ambiguity associated with center placement and ensures that the activation functions are physically meaningful representations of the molecular system.

The activation functions can then be used to form the pseudoatom density, $\rho_i^*(\mathbf{r})$, through the weighted superposition (Eq. (14)). This is done for each atom in the molecular system. For a heteronuclear diatomic AB, the pseudoatom densities for each atom can be ex-

pressed as:

$$\begin{aligned}\rho_A^*(r_A) &= w_{neutral}^{A^0} M^{A^0}(r_A) + w_{cation}^{A^{+1}} M^{A^{+1}}(r_A) + w_{anion}^{A^{-1}} M^{A^{-1}}(r_A) \\ \rho_B^*(r_B) &= w_{neutral}^{B^0} M^{B^0}(r_B) + w_{cation}^{B^{+1}} M^{B^{+1}}(r_B) + w_{anion}^{B^{-1}} M^{B^{-1}}(r_B),\end{aligned}\tag{20}$$

where r_A and r_B are the radial distances from the nucleus of atom A and B respectively to a point \vec{r} in space corresponding to the molecular density $\rho_{AB}(\vec{r})$. Each atom has a set of radial distribution functions, $M^{A^0}, M^{A^{+1}}, M^{A^{-1}}$, associated with each state of the atom, i.e. A^0, A^{+1}, A^{-1} . These functions are of the form of Eq. (19) but with different, optimized parameters that define each state’s radial density distribution. The radial distribution functions are each associated with a weight, $M^{A^0}, M^{A^{+1}}, M^{A^{-1}}$ that specify the contribution of each state’s electron density distribution to the overall atomic electron density, $\rho_A^*(r_A)$, as designated by Eq. (14).

The weights, $\{\omega_i\}$, determining the optimal superposition of states for each pseudoatom density that provide the best approximation to a given molecular density are found by training the neural network. Once these weights have been determined, the ensemble atom-in-molecule estimate of the molecular electron density is given by (Eq. (16)):

$$\rho_{AB}(r) \approx \rho_A^*(r_A) + \rho_B^*(r_B).\tag{21}$$

By constraining the activation functions of nodes to take the form of ensemble atom-centered sphericalized density distributions, the radial distance functions serve as a compact, chemically-relevant basis for approximating molecular electron densities. Centering these radial functions on the nuclear locations of atoms in the molecule creates a latent representation that inherently reflects the spatial and electronic structure of the molecular system, thereby constraining the network to generate outputs that are chemically reasonable. Consequently, instead of learning arbitrary latent features through high-dimensional layers trained on massive datasets, as is required in deep neural networks [53], the RBF-

NN utilizes a pre-organized latent space where each dimension corresponds to an atomic charge or excitation state, facilitating a direct and interpretable mapping from the molecular geometry to the total molecular electron density. Thus, despite the RBF-NN being remarkably simple in its architecture, a strong physical foundation is built into the design of the NN.

2.3 Physical Constraints in the Loss Function

In order for the weights to be updated successfully during the process of backpropagation, a *loss function* (or *cost function*) must be implemented in order to quantify how well the neural network is performing. The loss function measures the difference between the predicted output of the network and the known reference, or ‘true’, data [54]. During training, an optimizer minimizes the loss by adjusting the weights in Eq. (20), so that the predicted electron density more closely matches a given, true molecular electron density. The optimizer is an algorithm that determines how the weights should be changed based on the value of the loss, typically by following the gradient of the loss function with respect to each weight [38, 39, 54, 55]. In this way, the loss function serves as the guiding metric for learning, and the optimizer provides the mechanism for improving the network’s predictions over successive training steps.

A traditional loss function determines the difference between a predicted output and the known training data through the use of the mean squared error (MSE) or root mean squared error (RMSE). The MSE, used in this work due to its smooth differentiability and sensitivity to larger errors [54], is defined as:

$$\text{MSE} = \sum_{k=1}^N \frac{[\rho_k^{\text{true}}(\vec{r}) - \rho_k^{\text{pred}}(\vec{r})]^2}{N}, \quad (22)$$

where, for each point in space k of the total N 3D gridpoints, $\rho_k^{\text{true}}(\vec{r})$ is the predicted

atomic density distribution according to Eq. (20) and $\rho_k^{\text{pred}}(\vec{r})$ is the given reference, or training, density at that location. This ensures that the neural network learns to produce electron densities that closely approximate the quantum mechanically-derived reference densities, thus providing a direct measure of accuracy.

The loss function also provides an opportunity to enforce physical constraints on the weights. These constraints ensure that the learned ensemble density is consistent with fundamental chemical and physical principles, rather than being an arbitrary mathematical fit. The two key constraints applied in this work, in addition to those used in defining the analytic form of the basis densities [4, 5], are *charge conservation* and *normalization of ensemble weights*.

The charge constraint enforces the requirement that the total charge predicted by the ensemble weights matches the total molecular charge, q_{total} . Since each atomic state corresponds to a specific integer charge, the sum of all weighted charges over the atoms must equal q_{total} . In the present work, $q_{\text{total}} = 0$ for all studied system as we only consider neutral molecules. This constraint ensures that the electron density predicted by the network corresponds to a chemically valid charged or neutral system. The constraint can be expressed as:

$$q_{\text{total}} = \sum_{i=1}^{N_A} \sum_{j=0}^{N_{\text{ens}}} w_i^j q_i^j, \quad (23)$$

where w_i^j is the weight and q_i^j is the integer charge associated with the j th state (neutral, cation, anion) of the i th atom of the molecular system.

The normalization constraint ensures that, for each atom, the weights over all possible atomic charge states sum to one. This condition enforces the requirement that all the pseudoatom partitions sum to the total molecular density at all points in space, Eq. (16), and that the atom-in-molecule decomposition physically corresponds to a true, statistical,

DFT ensemble [4, 18]. This condition is written as:

$$1 = \sum_{i=1}^{N_A} \sum_{j=0}^{N_{ens}} w_i^j, \tag{24}$$

where the total sum of all weights w_i^j , taken over all N_A atoms and their N_{ens} ensemble states per atom, is constrained to be one.

Together, these constraints ensure that the neural network produces electron densities that are not only mathematically accurate but also chemically consistent. The charge constraint prevents unphysical electron distributions that would correspond to the wrong total charge, while the normalization constraint guarantees that the atomic states combine in a way that reflects a valid DFT statistical ensemble description of the molecule [5, 18]. By embedding these physical requirements directly into the loss function, the RBF-NN is guided to learn representations that honor both the numerical training data and the underlying chemical theory theoretical DFT foundations.

2.4 Hydrogen and the Special Case of H⁺

Including hydrogen within the ensemble atom-in-molecule formulation is essential for its application to realistic chemical systems. Considering that it is the most abundant element in the universe, hydrogen is a critical component of a vast range of molecules. In particular, amino acids, the building blocks of proteins, contain hydrogen atoms that interact with solvating H₂O molecules. Moreover, hydrogen is significant in its capacity to participate in a range of bonding environments, including covalent, ionic, and hydrogen bonding interactions. The ability to model charge transfer and bonding interactions involving hydrogen is therefore required in order to extend the ensemble AIM framework to more complex and diverse systems.

The ensemble of ground and excited states for hydrogen are easily constructed because

analytical forms of the hydrogen radial density distributions are well-known [56] and can be implemented directly as activation functions within the RBF-NN framework. These radial distributions are encoded through closed-form expressions corresponding to specific quantum numbers (n, l) . The hydrogen radial density functions take the form:

$$\rho_{n,l}(r) = \frac{|R_{n,l}(r)|^2}{4\pi}, \quad (25)$$

where $R_{n,l}(r)$ is the hydrogenic radial wavefunction written as a product of a polynomial and an exponential decay factor. These exact expressions [56] have been implemented here for $n = 1, 2, 3, 4$ and corresponding l values. The radial electron densities are plotted in Fig. 7. Because these radial densities are defined analytically, they can be used directly as the radial activation functions in the RBF-NN model without the need for analytic fitting.

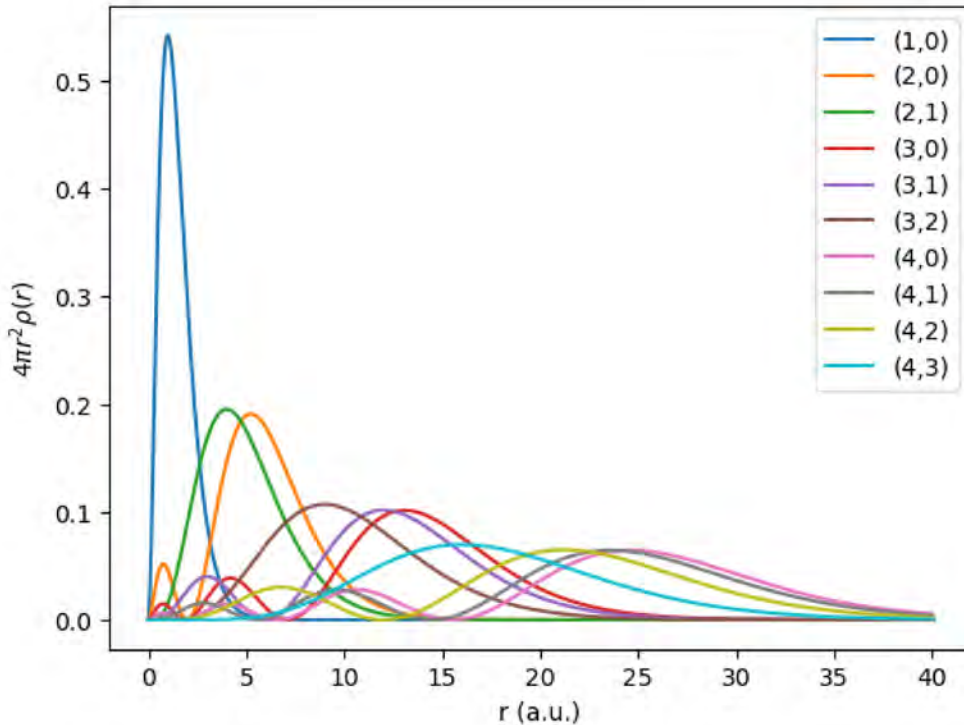


Figure 7: Radial density distributions for the ground and excited states of hydrogen, Eq. (25). States are labeled by quantum numbers (n, l) .

For charge states of hydrogen, a different methodology is required. For the hydrogen

anion, H^- , the standard analytic fit of the sphericalized, radial density can be performed according to the model in Eq. (19). However, the hydrogen cation state, H^+ , presents a glaring issue. Because H^+ lacks any electrons — it is just a proton — there is no electron density distribution to analytically fit.

Previous work in this project [4] attempted multiple strategies to represent H^+ within the ensemble representation, including using a negative ground state density distribution and as difference in density distributions, $\rho_{\text{H}^+} = \rho_{\text{H}_2^+} - \rho_{\text{H}_0}$. These proved unsuccessful.

Fortunately, the use of an RBF-NN formulation provides an alternative and natural way to include a H^+ state into the hydrogen ensemble. Although a density distribution cannot be directly added into the RBF ensemble for H^+ , its contribution can still be effectively incorporated through the loss function. In particular, the ensemble weights, which determine the linear combination of basis functions, are involved in satisfying the charge and normalization constraints imposed on the system, Eqs. (23) and (24). Since H^+ contains no electrons, its density is effectively represented as a zero distribution within the ensemble. However, the weight corresponding to H^+ remains a tunable parameter that participates in the optimization process via the constrained loss. This means that while H^+ has no explicit radial density to serve as an RBF activation function, its influence is embedded within the global ensemble properties that must satisfy physical constraints such as total number of electrons and charge neutrality. Thus, H^+ is implicitly included in the model by enforcing that its absence of electron density is consistent with the overall charge and normalization conditions imposed on the overall system.

3 Computational Implementation of the RBF-NN

3.1 Design of the RBF-NN

The structural design of the RBF-NN, as applied to a simple diatomic molecule such as LiF, is illustrated in Fig. 8. This depicts six RBF neurons/nodes that correspond to the charge states of the atoms that make up the molecule. Thus, for each atom, there exists nodes corresponding to the cation, anion and neutral states. Although not depicted in this example, additional states including those of higher charge or excited states can be easily added into this formulation as additional nodes. In the case of the representation of LiF in Fig. 8, rather than having just six nodes representing the charge states of each atom, more nodes can be added corresponding to either ions in higher charge states (i.e. Li^{+2} , F^{-2}) or excited states of the neutral atom or ion.

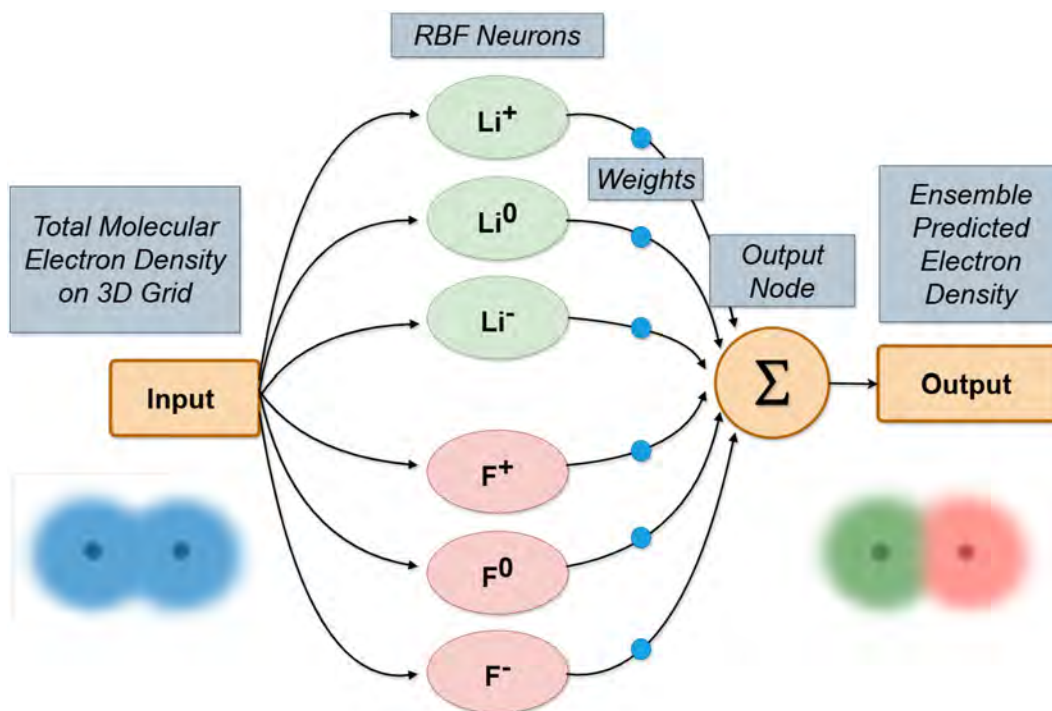


Figure 8: Structure of the RBF-NN for LiF as an exemplar.

Each of the nodes has an activation function of the form of Eq. (19). Each of these

nodes also has an associated weight, ω_{ij} , that can be adjusted during the optimization process of the neural network’s learning into order to determine the optimal superposition of radial basis function state densities (Eq. (20)).

The RBF-NN takes as input a set of spatial coordinates on a 3D grid. The 3D coordinates, which are in the form of Cartesian coordinates on a uniform grid, are converted to radial distances from the nuclear locations for each atom. For an atom A located at the Cartesian location (x_A, y_A, z_A) , the radial distance an arbitrary point (x, y, z) is away from the nucleus is:

$$r_A = \sqrt{(x - x_A)^2 + (y - y_A)^2 + (z - z_A)^2}, \quad (26)$$

and analogously for atom B.

The radial distances can be fed into the radial activation functions centered at each node in order to find each atomic state’s contribution to the density distribution. These radial functions are then summed with their corresponding weights to output the predicted electron density distribution for the entire molecule, $\rho^{\text{pred}}(\vec{r})$, equivalent to finding $\tilde{\rho}(\vec{\mathbf{r}})$ in Eq. (16).

The input spatial coordinates are also associated with a quantum mechanically-generated electron density distribution. This ‘true’ electron density distribution for the molecule, $\rho^{\text{true}}(\vec{r})$, is used to evaluate the accuracy of the predicted distribution through the loss function. As described above, this evaluation works to calibrate the weights that determine $\rho^{\text{pred}}(\vec{r})$. Through successive iterations in which the weights are updated to minimize the loss between the true and predicted electron density distributions, Eq. (22), the optimal ensemble AIM is determined for the molecule. Thus, this training of the RBF-NN is used to determine the weights allocating the superposition of atomic states of the molecule as well as to determine the predicted electron density distribution.

3.2 Computational Methods and Tools

The RBF-NN was coded in Python using the TensorFlow software library [57] with custom Keras layers enabled [58], included in Appendix A. TensorFlow is an open-source machine learning framework developed by Google. Keras, integrated within TensorFlow, is a high-level neural network application programming interface (API) that simplifies the process of building and training machine learning models. It offers an interface for defining a neural network’s architecture, while maintaining compatibility with TensorFlow’s optimization algorithms and other capabilities. The use of Keras allowed for a more straightforward implementation of the RBF-NN, as the custom activation functions that compute the radial basis function response can be implemented using Keras’ `Layer` class.

The weights ω_{ij} that define the ensemble atom-in-molecule superposition were optimized using the ADAM optimizer [59]. ADAM (Adaptive Moment Estimation) is a stochastic optimization algorithm that combines the benefits of both the Adaptive Gradient Algorithm (AdaGrad) [60] and Root Mean Square Propagation (RMSProp) [55]. It utilizes first-moment (mean) and second-moment (uncentered variance) estimates of the gradients to adaptively adjust the learning rate for each weight [61]. This optimizer was implemented through its packaging in the TensorFlow library.

The RBF-NN is trained with quantum mechanical computed molecular densities on a grid. These were generated using the `Gaussian’16` software package [62]. `Gaussian’16` is a computational chemistry software package used for electronic structure (quantum chemical) calculations. It supports various levels of theory, including Hartree-Fock (HF), density functional theory, and post-Hartree-Fock correlation techniques such as MP2 and MP4. The training data utilized in the present work consists of molecular densities generated on a uniform Cartesian grid and computed at the MP4 level of theory.

Of note, the sphericalization of the atomic densities to construct the radial distribution functions and perform the analytic fits to define the RBF-NN activation functions were

done with a custom Matlab script, as described in [4], and translated to Python for use in the present work.

3.3 Training the RBF-NN

The RBF-NN is trained with quantum-mechanically computed molecular densities on a grid generated using the Gaussian'16 electronic structure code. Over many iterations, it uses this training data to determine the optimal weights for each state in the AIM superposition. The training process follows the general cycle illustrated in Fig. 9.

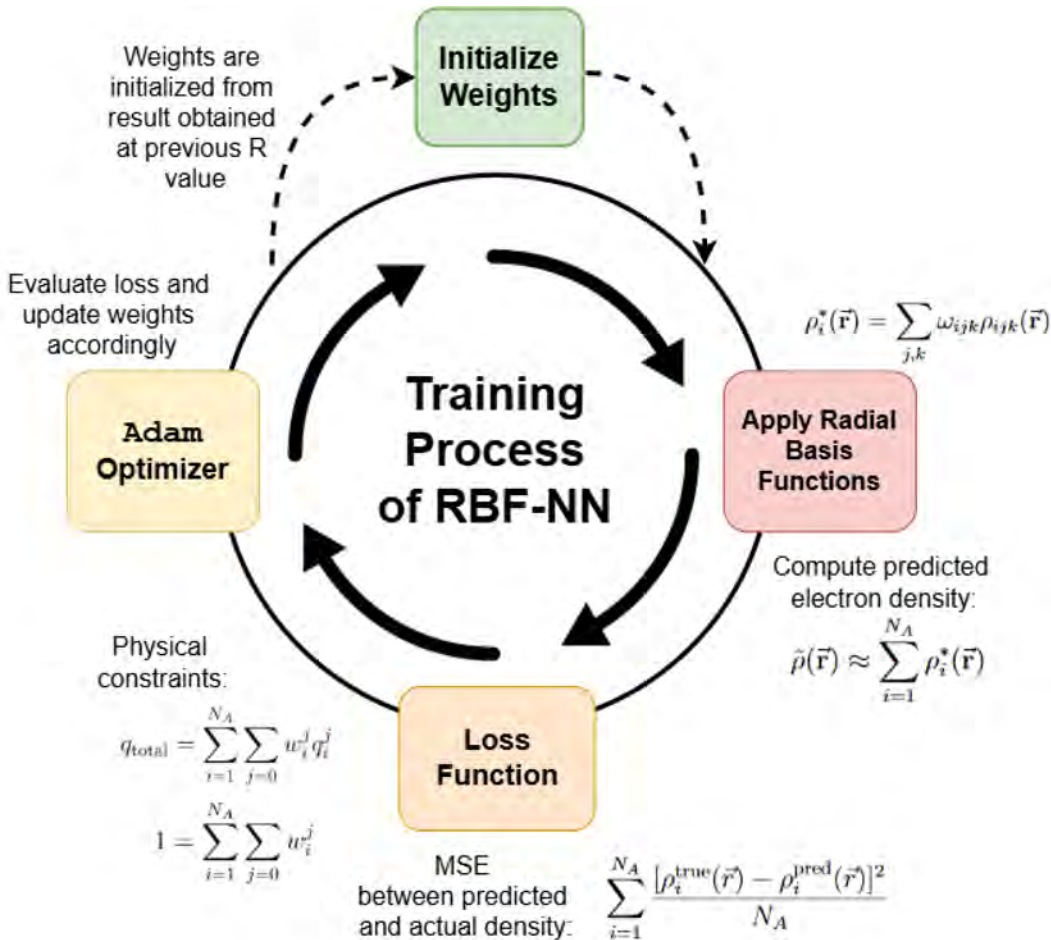


Figure 9: Training process of the RBF-NN per epoch. After initializing the weights, training follows the cycle of applying the radial basis functions to construct a predicted electron density. This predicted density is then evaluated through the loss function, and the ADAM optimizer updates the weights accordingly.

After the weights are initialized, the radial basis functions in the neural network are applied on the training grid to give the predicted molecular electron density distribution. To improve computational efficiency and stabilize optimization, the grid points are divided into smaller groups called *batches*, with each batch containing a subset of the total data points. The result for each batch is compared to the corresponding true density values to compute the loss, which is then fed into the ADAM optimizer [59]. The optimizer updates the weights, and the cycle continues. Each full pass over the entire training set, consisting of all batches, is referred to as an *epoch*. Multiple epochs are typically required for the model to converge.

As discussed previously (Section 2.2), the network’s radial basis functions serve as a compact representation of atomic electron densities, reducing the complexity of the training process. The predicted electron density distribution can be determined by applying the radial basis functions with the current weights as (Eqs. (20) and (21)). The loss function described in Section 2.3 is used to evaluate the success of this prediction through both comparison to the actual density distribution and its obedience to the physical constraints (charge conservation and weight normalization).

The ADAM optimizer takes the loss as input, and uses it to inform an adjustment of the weights. This process continues until the model converges to an optimized representation of the molecular electron density. The resulting weight distributions provide insight into the contributions of individual atomic charge states, enabling a physically-informed AIM decomposition.

It is important to clarify that the RBF-NN is not performing *classification* in the conventional machine learning sense. Instead, the RBF-NN is used here purely for optimization. The objective is to minimize the mean-squared error (MSE) between the predicted electron density and the reference (input) quantum-mechanical density. Training proceeds until the loss fails to improve, which serves as the stopping criterion (this will be described in detail in Section 3.3.4). Once this criterion is met, the optimization halts

and the weights determining the superposition are no longer adjusted. At that point, the model is considered “trained,” and the final superposition of the radial basis functions determined by the converged weights constitutes the “prediction.” This prediction is not speculative; it is a direct output of the optimized physical model, yielding both the ensemble density and the effective atomic charges determined by the learned weights.

Although the RBF-NN is trained on electron densities computed by `Gaussian’16`, its outputs do not simply reproduce those densities. The ensemble representation provided by the RBF-NN offers a compact and chemically interpretable description of the system. From this representation, we can extract changes in effective atomic charges as a function of internuclear separation, track the onset of ionic bonding character, and recover neutral atom limits at large internuclear separations. These features are not directly available from the input `Gaussian’16` data. The value of the model lies in how it reorganizes that information: instead of a grid of density values, we obtain a structured decomposition in terms of physically meaningful atomic components. This enables both visualization and analysis of bonding character in a manner that extends beyond the original electronic structure calculations.

3.3.1 Loss Function Design

The loss function is implemented using Keras’ custom loss functionality, ensuring that both the accuracy of the predicted molecular density and physical constraints are incorporated into the training process. The loss function is computed dynamically for each training batch and is passed to the `ADAM` optimizer, which updates the network’s weights ac-

cordingly. For a simple diatomic molecule, AB, the loss function is structured as follows:

$$\begin{aligned}
 \text{Loss} = & \Omega_1 \cdot \sum_{k=1}^N \frac{[\rho_k^{\text{true}}(\vec{r}) - \rho_k^{\text{pred}}(\vec{r})]^2}{N} \\
 & + \Omega_2 \cdot |(w_0^A \cdot 0) + (w_1^A \cdot 1) + (w_2^A \cdot -1) + (w_0^B \cdot 0) + (w_1^B \cdot 1) + (w_2^B \cdot -1)| \\
 & + \Omega_3 \cdot |w_0^A + w_1^A + w_2^A - 1| \\
 & + \Omega_4 \cdot |w_0^B + w_1^B + w_2^B - 1|. \tag{27}
 \end{aligned}$$

This loss function is informed by the true and predicted electron density distributions, $\rho_k^{\text{true}}(\vec{r})$ and $\rho_k^{\text{pred}}(\vec{r})$, as well as the weights associated with each state.

In Keras, the loss function is defined as a custom function that takes the predicted and true electron densities as inputs and returns a scalar loss value. The first term in Eq. (27) represents the mean squared error (MSE), which ensures the predicted electron density aligns with the reference data (Eq. (22)). The remaining terms implement weighted penalties for deviations from charge neutrality and normalization constraints, as seen in Eqs. (23) and (24). These terms are scaled by adjustable *hyperparameters*, denoted by $\Omega_1, \Omega_2, \Omega_3$, and Ω_4 , which determine the relative importance of each constraint during optimization.

The weighting factors Ω_i allow for flexibility in training, as they can be fine-tuned to balance numerical accuracy with physical consistency. For instance, if charge neutrality deviates significantly during training, increasing Ω_2 reinforces this constraint, ensuring the model prioritizes charge conservation. Similarly, Ω_3 and Ω_4 control the strength of weight normalization constraints, preventing the network from assigning unphysical weight distributions. These parameters are adjusted manually by the user, if the RBF-NN is observed as not prioritizing a constraint or quality of fit.

3.3.2 Implementing the ADAM Optimizer

As noted above, the ADAM optimizer [59] is used to minimize the loss function during the training of the RBF-NN by adaptively adjusting the weights. The optimizer combines the advantages of two other popular optimization algorithms: Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). It leverages both first and second moment estimates to adapt the learning rate of each parameter [61].

One of the most important hyperparameters in the ADAM optimizer is the *learning rate*. The learning rate determines the step size at which the model's weights are updated during training. While ADAM adaptively adjusts the learning rate for each parameter based on the moving averages of past gradients, the base learning rate still plays a crucial role in scaling these updates. If the learning rate is set too high, the optimizer may overshoot the optimal point, leading to oscillations or divergence in the loss function. Conversely, if the learning rate is too low, convergence will be slow, and the model may get stuck in local minima. Selecting an appropriate learning rate is crucial for efficient and accurate training.

The optimizer also employs a technique called *weight decay*. Weight decay is a regularization technique used to prevent overfitting by penalizing large weights during training [38]. The fundamental idea is to add a penalty term to the loss function that is proportional to the squared magnitude of the weights. The weight decay term helps maintain generalization performance, especially when the network is prone to overfitting [63, 64].

An additional feature of the ADAM optimizer that can be enabled is `amsgrad`. Setting `amsgrad = True` modifies the optimizer to use the AMSGrad variant of ADAM, which is designed to improve convergence stability. AMSGrad addresses a known issue with the standard ADAM optimizer where the learning rate can become excessively small due to exponentially decaying average gradients. By maintaining a long-term maximum of past squared gradients, AMSGrad ensures that the adaptive learning rate does not decrease too rapidly.

This has been shown to result in more stable and reliable convergence [65].

3.3.3 Batching in RBF-NN Training

Batching refers to the process of splitting the training dataset into smaller subsets and training the model on these smaller sets rather than the entire dataset at once. Instead of updating the network’s weights after every individual data point or waiting to process all data points in a single pass, batching allows for more efficient use of memory and computational resources. The training process works by applying the model to each batch, calculating the loss, and updating the weights before moving to the next batch. This process is repeated for several epochs until the model converges to an optimized solution [38].

In the context of the RBF-NN, rather than using the entire grid of molecular density data at once, the data is randomly divided into smaller batches. Each batch consists of a subset of grid points where the network applies the radial basis functions and computes the predicted molecular density. The loss is then computed for that batch by comparing the predicted density with the actual density. The optimizer (in this case, **ADAM**) uses this loss to adjust the network’s weights. This cycle repeats for each batch until all batches have been processed for one epoch.

Using batches has several advantages in the training process. It allows for faster and more memory-efficient computations, especially when working with large datasets, like the 3D density grids used in this work [55]. Batching also introduces some stochasticity into the training. Instead of learning from the entire dataset at once, the network updates its weights based on the smaller batches, which helps prevent overfitting and improves generalization [38].

In the training code, batching is implemented using the `batch_size` parameter in the `model.fit()` function in TensorFlow [57]. The batch size determines how many data points are processed together before the weights are updated.

The batch size can be adjusted as needed. Smaller batch sizes typically lead to more frequent updates in the weights and can help escape local minima in the loss, but can introduce more noise into the training process [38]. Larger batch sizes are more computationally efficient and result in smoother convergence but can be prone to overfitting if too large.

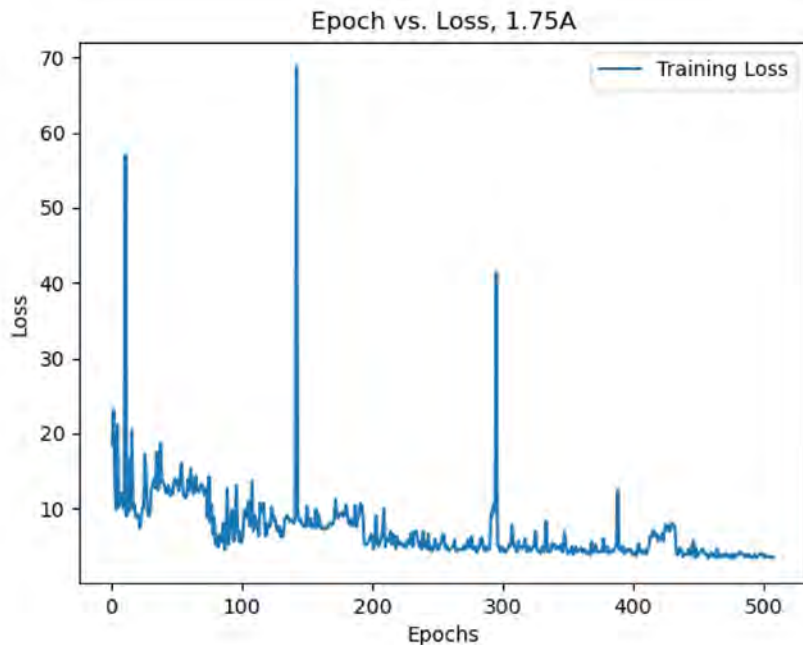


Figure 10: Loss as a function of epoch for an RBF-NN run of hydrogen fluoride at internuclear separation 1.75 Å, tracked with early stopping routine in order to determine convergence.

3.3.4 Early Stopping Routine

The implementation of the RBF-NN also includes an *early stopping* routine, which works to identify when the neural network has properly found a solution and therefore end the training. It acts as a form of regularization [38], preventing overfitting in the final result. The early stopping routine used in the RBF-NN functions by monitoring the loss over all epochs. If the loss does not decrease beyond a specific tolerance over a spec-

ified *patience*, or a set range of epochs, the training is ended with the best weights (with smallest loss). For instance, if the patience parameter was set to 50 epochs, then if the loss does not decrease by a significant amount over the 50 epochs, the training is ended.

As seen in Fig. 10, as the RBF-NN trains the loss decreases and converges to a minimum. Some oscillations in the loss occur as the weights are altered using the ADAM optimizer; however, the loss still decays steadily as a function of epoch. Although the maximum number of epochs for this run was set to 1500 epochs, the early stopping routine acted just after 500 epochs, stopping training as convergence was found to have occurred.

3.4 Training as a Function of Internuclear Separation

In order to study how the ensemble AIM formulation is able to represent charge transfer, it is relevant to conduct the AIM partitioning as a function of molecular geometry. For diatomics, this is accomplished by varying the internuclear separation, R (see Fig. 11), bringing the two atoms together or pulling them apart.

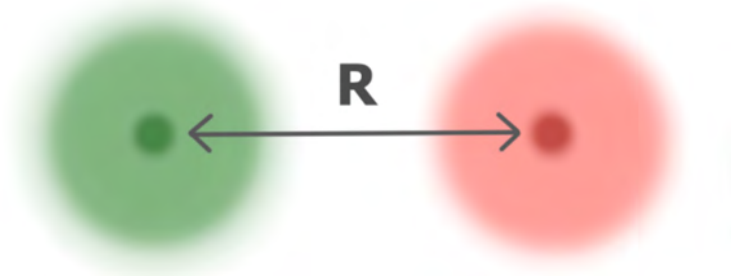


Figure 11: Geometry of a heteronuclear diatomic molecule, as a function of internuclear separation R .

Thus, for a range of selected values of R , the RBF-NN can be applied with training molecular density data calculated at a given internuclear separation in order to determine the weights for that molecular geometry. The weights can be tracked as the value of R changes, to measure how the AIM formulation changes as a function of molecular geometry.

At each R value, the effective atomic charge per atom i , q_i , can be calculated according to:

$$q_i = \sum_{j=0} w_i^j q_i^j, \tag{28}$$

where the q_i^j are the effective charges of each individual state of the atom with their corresponding weights, w_i^j . Thus, the effective charges on each atom can be compared as a function of R in order to evaluate how charge transfer occurs as atoms in the molecule are brought together or pulled apart.

The initialization of weights at the start of RBF-NN training requires careful consideration. The many adjustable weights of the system represent a high-dimensional space in which the function approximation occurs. Initializing the weights places the beginning of the learning at some location in this space. Nonphysical initializations may lead to non-physical optimized weights, even after training. Thus, the initial weights must be chosen carefully. At large internuclear separations, the atoms in the molecule are effectively isolated, with the neutral atomic state dominating the charge distribution. Initializing the weights with neutral states at these large separations ensures a physically consistent starting point.

To maintain continuity in the training process, the optimized weights from one value of R are used as the initial weights for the subsequent, closer internuclear separations. This approach of utilizing the final optimized weights from one configuration to initialize the next ensures that the training remains consistent and physically meaningful throughout the entire range of R . As the atoms are brought together, these continuous weights are recorded in order to capture the process of charge transfer.

4 Application to Various Molecular Systems

4.1 LiF

LiF was selected to be the first molecule studied using the novel RBF-NN architecture due to its status as a simple heteronuclear diatomic dominated by ionic interactions: in a naïve picture of the chemical bond, Li donates an electron to F, so that both atoms “look like” closed-shell atoms (He and Ne, respectively), with nominal charges Li^+F^- . This molecule was also previously studied by Amo-Kwao [4], using a very different nested-grid optimization procedure that – for numerical reasons – could be only extended out to $R = 2.2 \text{ \AA} = 4.16 \text{ a.u.}$, well within the ionic region. Thus, this previous work was limited in its ability to show the dissociation to neutrals as the atoms are pulled far apart, which occurs at $R \approx 13 \text{ a.u.}$ [66]. The RBF-NN successfully reproduces these previous results and extends them to internuclear separations where dissociation to isolated atoms can be observed.

As described in Section 3.4, we can train the RBF-NN as a function of the internuclear separation, R . At each value of R , the predicted electron density can be found with the superposition of states using the optimized weights. This predicted electron density distribution can be compared to the actual electron density distribution, calculated for each value of R using the **Gaussian’16** electronic structure code at the MP4 level of theory.

The electron density distributions can be visualized using contour plots of the natural log of the density, $\ln(\rho(\vec{r}))$, over a yz -plane ($x = 0$) slice of the 3D grid including both atomic nuclei. The contour plots for the ‘true’, **Gaussian’16** generated, electron density distributions and the RBF-NN predicted electron density distributions are shown in Figures 12 and 13 respectively.

Using the RBF-NN, we are able to successfully capture LiF’s neutral-to-ionic transition as a function of R , as shown in Fig. 14. To monitor the transition, we compute the

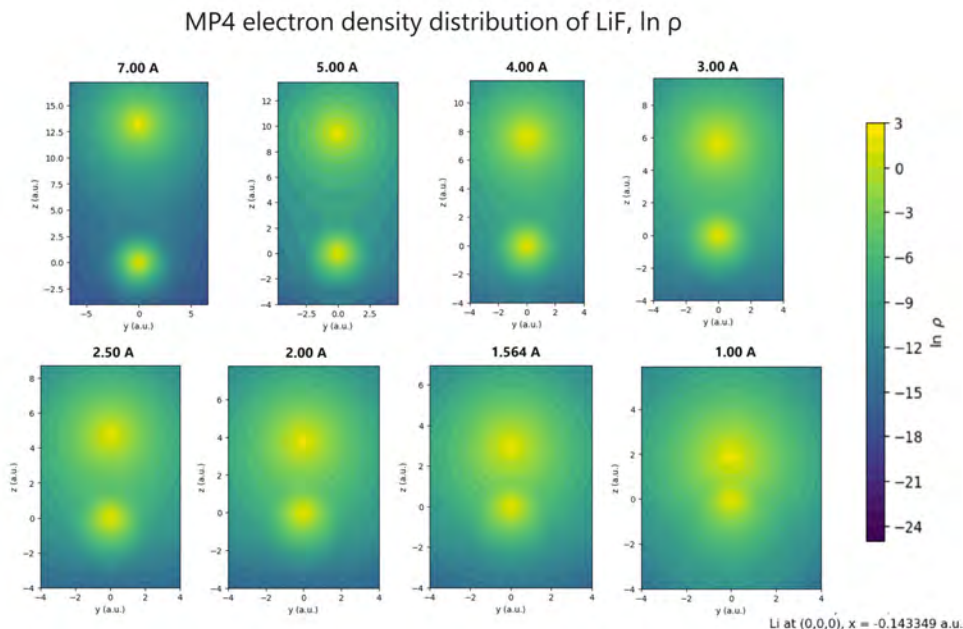


Figure 12: Contour plot of natural log of the true, QM-calculated density of LiF, $\ln(\rho(\vec{r}))$ at various values of internuclear separations, R , shown within the yz -plane. Density computed at MP4 level of theory using Gaussian'16 [62].

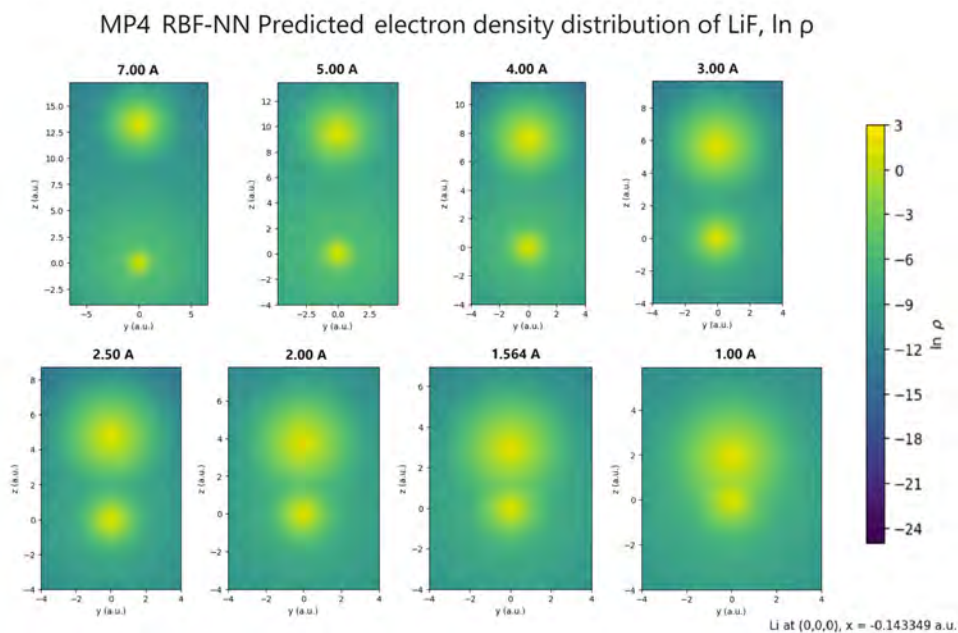


Figure 13: Contour plot of natural log of RBF-NN predicted density of LiF, $\ln(\rho^{pred}(\vec{r}))$ shown within the yz -plane at various values of internuclear separations, R (same values as Fig. 12).

effective charge on each atom as a function of R using Eq. (28). At large separations the atoms are neutral, but as they are brought together the atoms acquire charges close to one. We are also able to quantify the distance at which the transition occurs, as the first point before the increase in charge occurs in the distribution.

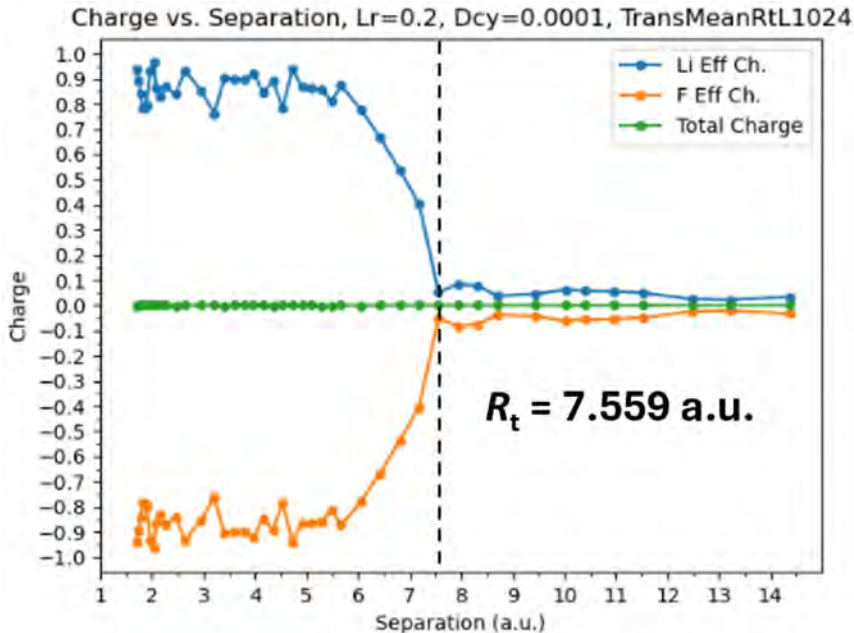


Figure 14: Effective charge of LiF as a function of internuclear separation R based on the ensemble AIM representation of the reference molecular density $\rho^{true}(\vec{r})$ computed at the MP4 level of theory.

We observe that the atoms gain effective charges of approximately ± 0.9 when the ionic bond is formed. This can be compared to the effective charge computed by Bader for LiF, utilizing his topological AIM approach. For LiF, Bader computed an effective atomic charge for Li of 0.938 (equal and opposite for F), at the LiF equilibrium separation of 1.564 Å (2.955 a.u.) [27]. This is indicated as the yellow diamond in Fig. 15. This plot also includes previous work by Amo-Kwao [4] which represented LiF using the ensemble AIM formulation, but with the nested-grid approach as previously described (red curve in Fig. 15). Amo-Kwao’s computed Bader atomic charges as a function of internuclear separation are also shown (black curve).

The magnitudes of charges computed in the present work agree well with both Bader

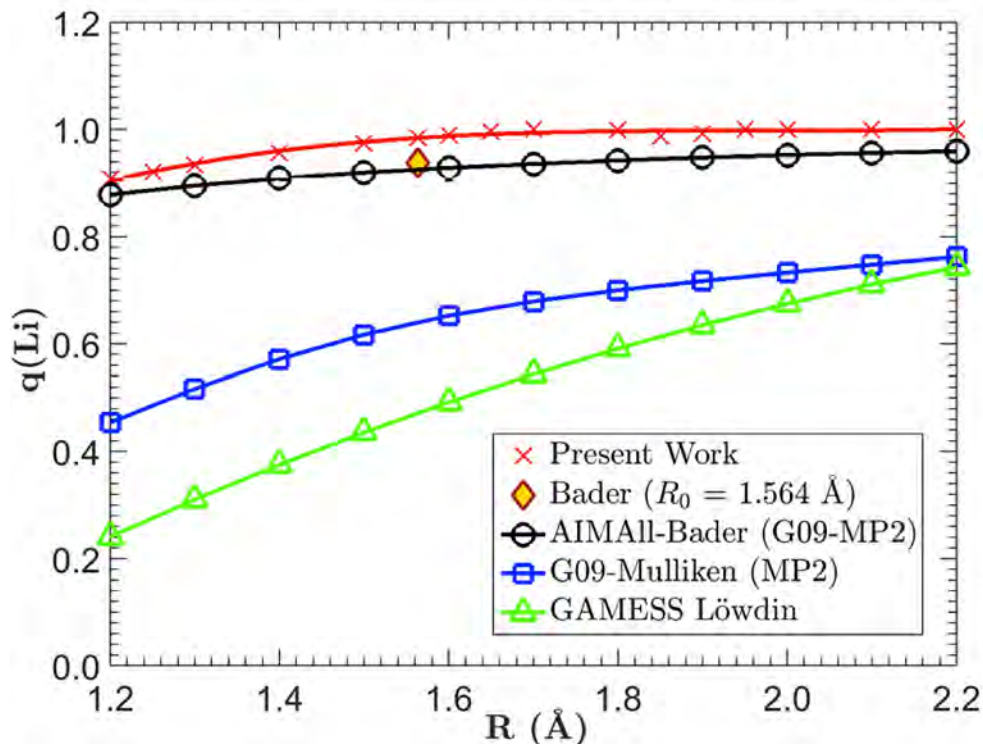


Figure 15: Charge transfer, $q(\text{Li})$, as a function of internuclear separation R in the bonding region of LiF. Data from work done in [4] (red) was determined using the same ensemble AIM formulation as the present work, but utilized nested grids and a standard BFGS method to determine the optimal ensemble weights. Effective charges computed using Bader’s AIMAll code is also shown (black). Figure from [4].

and Amo-Kwao. All three methods find that bonded Li and F have effective charges of $\sim \pm 0.9$ in the ionic bonding range.

As can be seen in Fig. 14, initial results using the MP4-computed molecular densities locate the transition from neutral to ionic at ~ 7.5 a.u.; however, highly accurate quantum chemical calculations [66] place it further out, at ~ 13 a.u. This discrepancy is explored in the next section.

4.1.1 Comparison with SWE Results of Varandas

In a 2009 study by Varandas [66], the ground and excited state potential energy curves and charge transfer behavior of LiF were computed using a series of high-level quantum

chemical methods. The work focused on accurately capturing the transition between neutral and ionic character as a function of internuclear separation. This transition is described as the coupling between the ground and first excited electronic states of the molecule. Varandas’ calculations clearly demonstrate the transition between ionic and neutral character by monitoring the computed dipole moment of the molecule as the constituent atoms are brought together or pulled apart (Fig. 16). The dipole moment as a function of the internuclear distance exhibits a sharp transition at approximately 13 a.u., indicating that a neutral-to-ionic transition occurs for LiF at this separation. These results therefore serve as a useful reference for validating the performance of the RBF-NN.

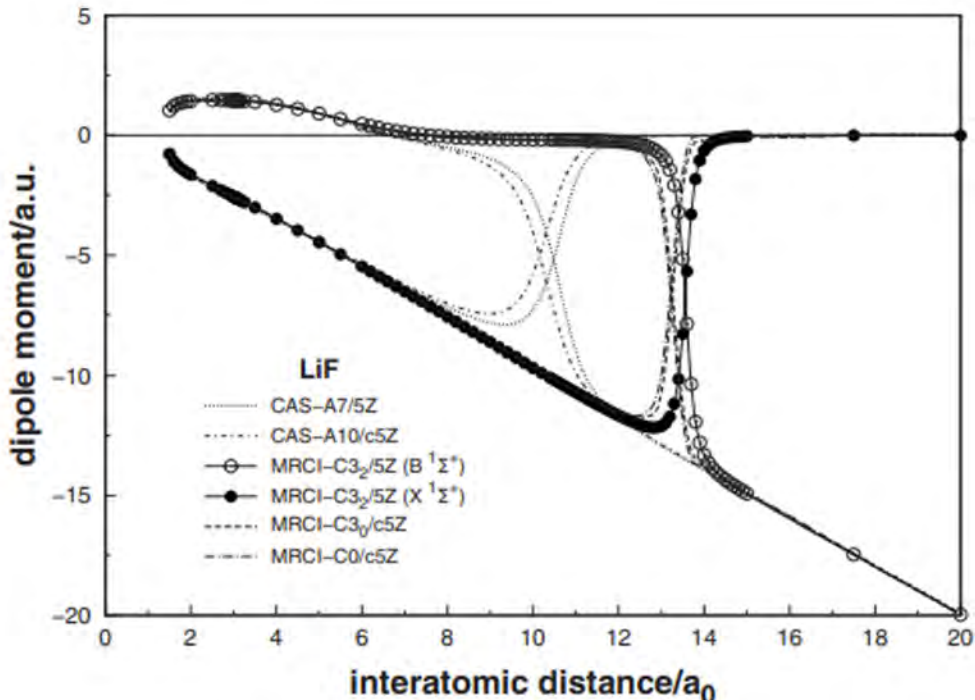


Figure 16: Dipole moment as a function of interatomic distance for the ground and first excited state of LiF, computed at different levels of quantum chemical theory. From [66].

In order to compare with Varandas’ results for the dipole moment as a function of internuclear separation R , calculations using Gaussian’16 were performed at different levels of theory and plotted in Fig. 17. Notably, the MP4 level of theory used in generating the training molecular electron density for the RBF-NN, fails to capture the neutral-to-ionic

transition via the dipole moment as Varandas’ data does.

Nevertheless, the RBF-NN atom-in-molecule representation yields a effective charge distribution as a function of R that clearly indicates a neutral-to-ionic transition, as seen in Fig. 14. This implies that the MP4-generated electron density distribution used to train the RBF-NN still contains underlying bonding information of atoms in the molecule despite the lack of indication in the MP4-computed dipole moment.

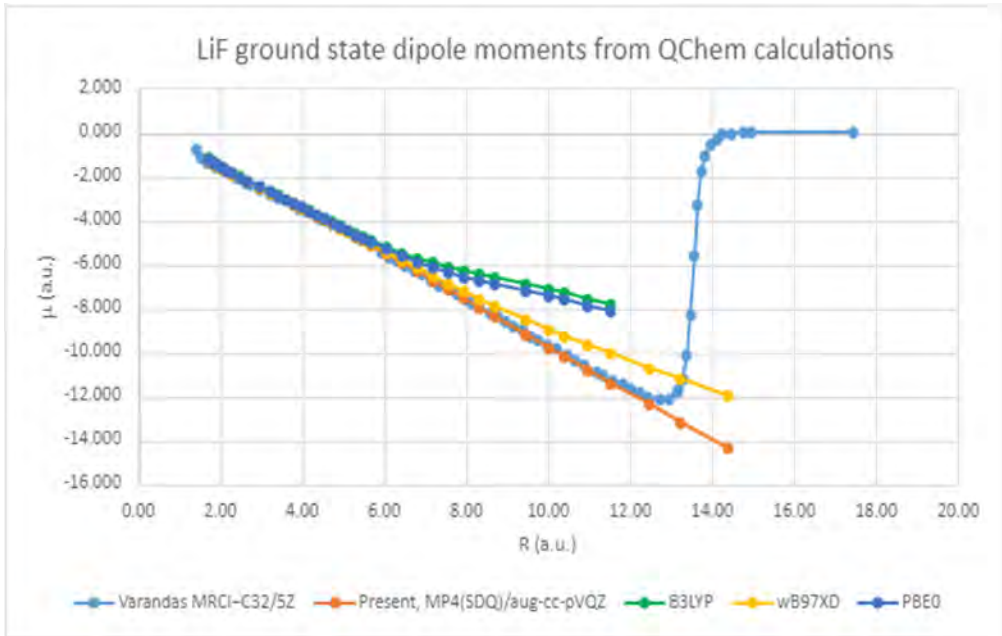


Figure 17: Dipole moment as a function of interatomic distance for various levels theory of LiF, calculated using `Gaussian’16`. Also includes the Varandas [66] dipole moment from Fig. 16.

Although the RBF-NN is able to capture the bonding transition using the MP4 training data, it still needs to be understood why the transition occurs at the wrong location. At this point in the work, it was observed that Varandas had implemented two levels of theory in his the electronic structure calculations: MRCI (Multi-Reference Configuration Interaction) and CAS (Complete Active Space Self-Consistent Field), the results of which are both displayed in Fig. 16. MRCI captures both static and dynamic electron correlation, which is critical for accurately modeling bond breaking and charge transfer, while CAS captures only static correlation. As seen in the figure, the dipole moment generated

at the less exact level of theory, CAS, indicates a bonding transition at a shorter internuclear separation (~ 10 a.u.) than MRCI (~ 13 a.u.).

The difference between the two levels of quantum chemical theory in Varandas’ work suggested that we explore the effect of level of theory on the computed neutral-to-ionic transition using the RBF-NN. This is discussed in the following section.

4.1.2 LiF RBF-NN Analysis at Various Levels of Theory

In order to investigate the possibility that the level of theory of the training data can impact the transition location, we explored how the RBF-NN predicts the bonding behavior of LiF starting from reference molecular densities computed at various levels of theory. The levels of theory selected for the training data include MP4 [3], PBE0 [15], B3LYP [13, 14], and wB97XD [16, 17]. These were chosen to represent a range of treatments of exchange and correlation.

MP4 (Møller–Plesset perturbation theory to 4th order) [3] is a post-Hartree-Fock method that combines exact exchange from Hartree-Fock with a perturbative correction to account for electron correlation. This is the only explicitly wavefunction-based method used in this work.

PBE0 [15] is a hybrid exchange-correlation energy density functional $E_{xc}[\rho]$ that mixes a small percentage of exact Hartree-Fock exchange with exchange and correlation from the Perdew-Burke-Ernzerhof (PBE) density functional [67]. PBE itself is a generalized gradient approximation functional, in which $E_{xc}[\rho]$ is expressed in terms of the local density and gradients of the density [67]. PBE0 is known for its robustness in solid-state systems and has become widely used due to its improved balance between accuracy and computational efficiency.

B3LYP [13] is another hybrid DFT functional that includes empirical parameters. It combines Slater exchange and Becke’s gradient correction [14] with correlation con-

tributions derived from the Colle-Salvetti model of the helium atom [13, 68]. B3LYP includes three empirically optimized parameters and is among the most popular functionals in quantum chemistry, particularly for small molecular systems, having been shown to outperform MP2 in both efficiency and accuracy [69].

ω B97XD [16, 17] is a long-range-corrected hybrid functional that partitions the exchange-correlation energy into short and long range components. It uses Becke’s 1997 exchange functional [70] for short-range exchange, exact Hartree-Fock for long-range exchange, and includes a damped empirical dispersion correction applied post-calculation. This correction is designed to handle non-covalent interactions and long-range charge transfer more accurately. Of all the levels of theory included, ω B97XD is the functional most designed to model charge transfer.

All four of the selected levels of theory described above were used to generate training data for the RBF-NN representation of LiF as a function of internuclear separation R (see Fig. 17). The RBF-NN was then used to determine the weights defining the ensemble AIM superposition, which then could be used to track the effective charge as a function of R . The plots included in Fig. 18 display the effective charge as a function of R for all four levels of theory. The location of the bonding transition, R_t , was determined, as described above, as the value of R at which the sudden change in atom-in-molecule computed effective charges began to increase from zero, to a finite value.

As can be seen in Fig. 18, changes in the level of theory used for the training data cause a clear shift in the location of the bonding transition location. Moreover, ω B97XD, the level of theory specifically designed to provide an improved description of charge transfer in DFT, displays a bonding transition location (10.016 a.u.) closest to the Varandas’ wavefunction based value of ~ 13 a.u.

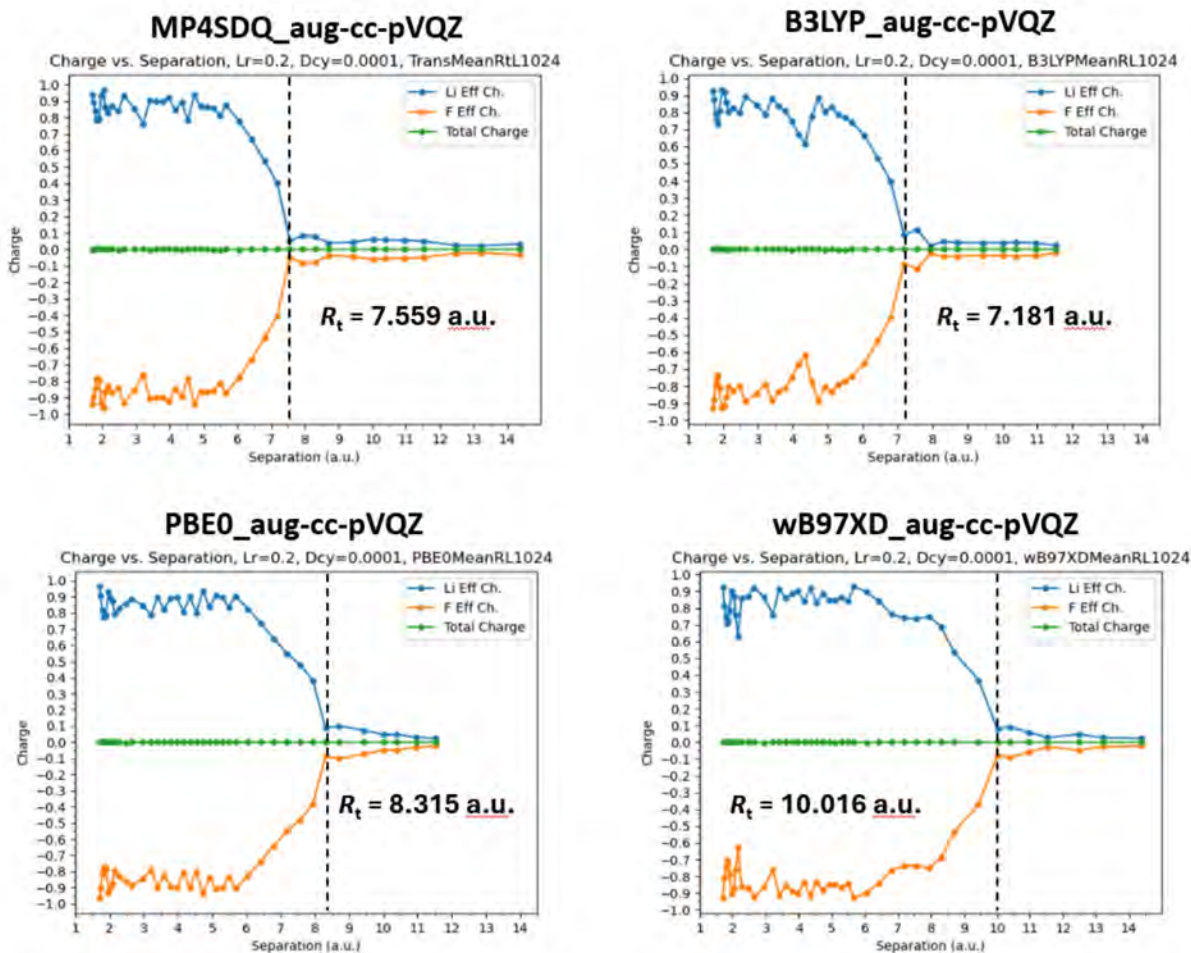


Figure 18: Effective charges of LiF as a function of internuclear separation R at the four levels of theory described in the main text: MP4, PBE0, B3LYP, and ω B97XD.

4.2 HF

Hydrogen fluoride was selected as an additional molecule for study by the RBF-NN in order to test the implementation of the hydrogen cation state within the AIM ensemble representation, as discussed in Section 2.4. The use of a molecule including hydrogen also provides the ability to include excited states within the ensemble representation, since their electron density distributions are known analytically.

For the purpose of analyzing the effects of including excited states of hydrogen in the RBF-NN model, the first five excited states of H, with principal quantum numbers $n = 2$ and $n = 3$, were successively included in the ensemble. These states, as well as the ground

state of hydrogen, are plotted in Fig. 19. Adding excited states in this 'ladder' of states procedure [4], enables us to assess how the RBF-NN responds to new states in a controlled manner.

As more states are included in the RBF-NN representation, weight given to the hydrogen cation state can be analyzed in order to determine the impact of its implementation within the RBF-NN.

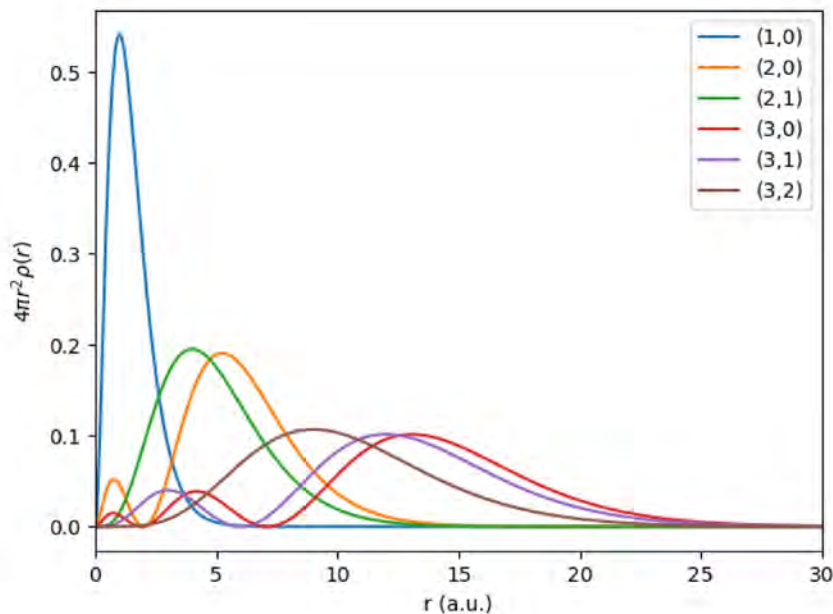


Figure 19: Radial density distributions for the ground and first five excited states of hydrogen. States are labeled by quantum numbers (n, l) .

4.2.1 Ladder Implementation of Hydrogen Excited States

To assess how excited states of hydrogen contribute to the RBF-NN representation, each excited state was added one by one to the ensemble model of HF. The corresponding weight profiles were analyzed as a function of internuclear separation R , as the two atoms, H and F, are brought together. This procedure allows for direct observation of how each

added state affects the overall representation, and whether their presence interferes with or supports the identification of charge transfer through the hydrogen cation weight.

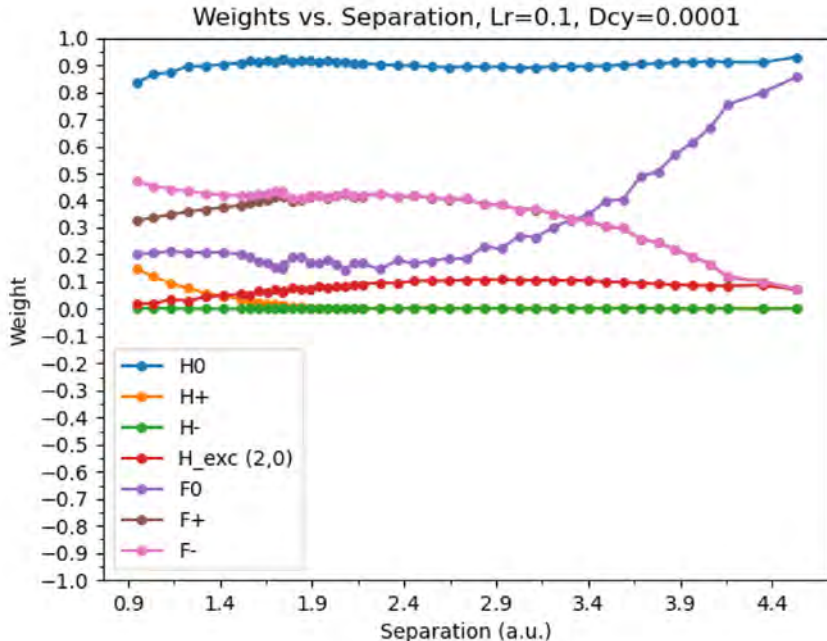


Figure 20: Weights for HF as a function of internuclear separation. States included are fluorine cation (brown), neutral (purple), anion (pink) and hydrogen cation (orange), neutral (blue), anion (green), and the first excited state, ($n = 2, l = 0$) (red).

Fig. 20 displays results from including only the first excited state ($n = 2, l = 0$). The weight associated with this state (red) causes a dip in the hydrogen neutral weight (blue), particularly at intermediate separations. This excited state peaks at larger R values and then falls off as the hydrogen cation (orange) weight begins to grow near the equilibrium separation, (~ 1.7 a.u.) [27]. Notably, the fluorine ion weights (F^+ and F^-) are indistinguishable until this transition, suggesting that their separation becomes relevant only when charge transfer is initiated.

Fig. 21 shows the effect of adding the second excited state of H ($n = 2, l = 1$) alongside the first. This addition leads to a more pronounced dip in the neutral hydrogen weight and the formation of a “band” structure in the fluorine weights, where F^+ , F^0 , and F^- all carry approximately equal weight. Again, the fluorine cation and anion weights remain

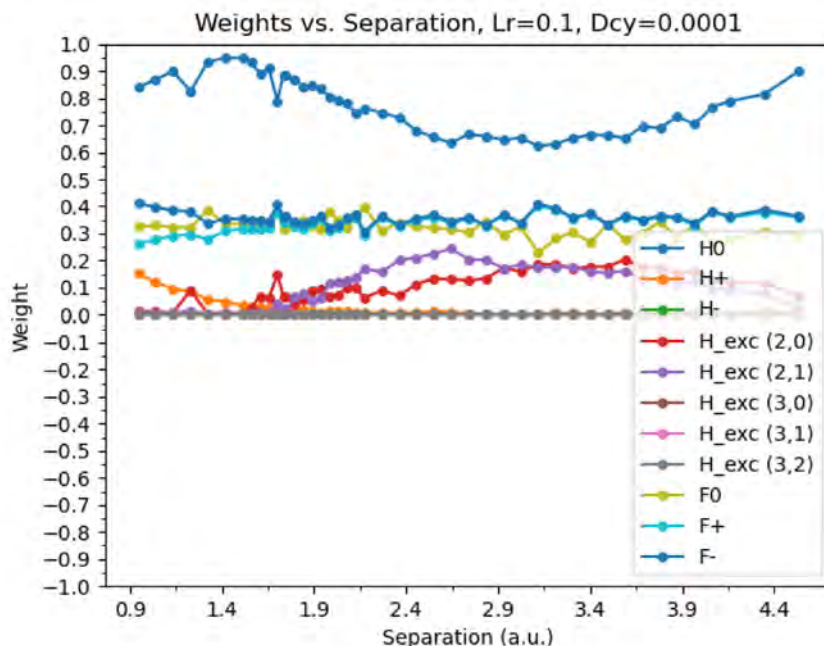


Figure 21: Weights for HF as a function of internuclear separation. States included are fluorine cation (light blue), neutral (light green), anion (dark blue) and hydrogen cation (orange), neutral (blue), anion (green), and the first two excited states, ($n = 2, l = 0$) (red) and ($n = 2, l = 1$) (purple).

overlapped until the hydrogen cation gains weight at smaller internuclear separations.

In Fig. 22, the third excited state ($n = 3, l = 0$) is added. This state dominates over the lower excited states and effectively “steals” weight from them. Interestingly, the lower excited states peak at the farthest separations, followed by the higher excited state, after which all excited weights fall and the hydrogen cation gains weight – clearly illustrating the emergence of ionic character at smaller interatomic distances.

Fig. 23 shows the inclusion of the fourth excited state ($n = 3, l = 1$). This state and the previous one share dominance, and now weight is taken not from just the hydrogen neutral state but among the lower excited states. The redistribution of weight indicates that new excited states can compete more directly with previously added states rather than just pulling from the ground state.

Finally, Fig. 24 introduces the fifth excited state ($n = 3, l = 2$). This state shows

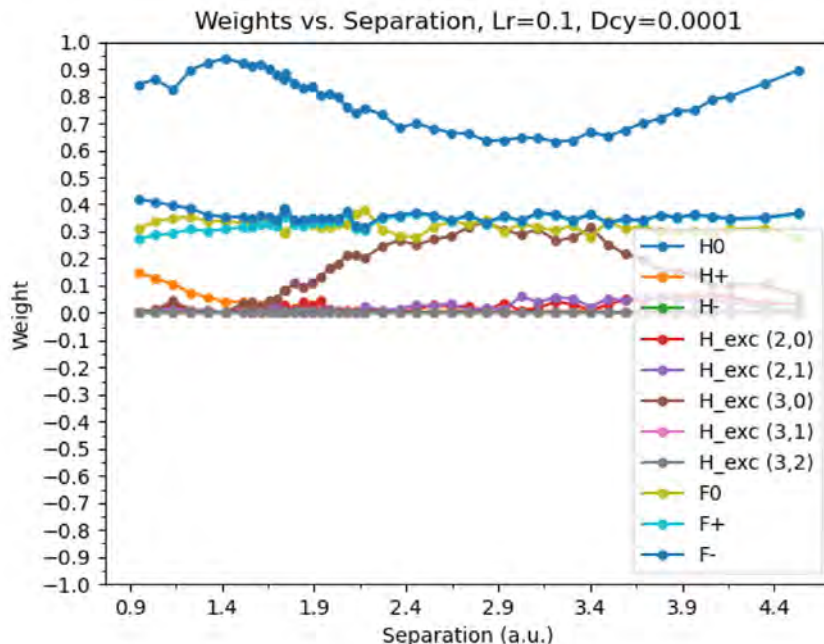


Figure 22: Weights for HF as a function of internuclear separation. States included are fluorine cation (light blue), neutral (light green), anion (dark blue) and hydrogen cation (orange), neutral (blue), anion (green), and the first three excited states, ($n = 2, l = 0$) (red), ($n = 2, l = 1$) (purple), and ($n = 3, l = 0$) (brown).

nearly identical behavior to the fourth, with the weight curves falling directly on top of one another, suggesting redundancy in the information each state contributes to the ensemble. This redundancy implies a saturation point in the excited-state basis for hydrogen within the model. Despite these additions, the fluorine weight band structure remains consistent, and the F^+ and F^- weights continue to overlap until hydrogen’s cation state gains weight at ~ 1.7 a.u.

Note that with each additional excited state added, the RBF-NN will supply weight to it. Excited states associated with the same H principal quantum number n tend to fall directly on top of each other. This seems reasonable since, in H, states with the same values of n are associated with the same energy.

The internuclear separation R at which the excited state weights are given the most weight is also of interest. As the two atoms of HF are brought together, the excited states

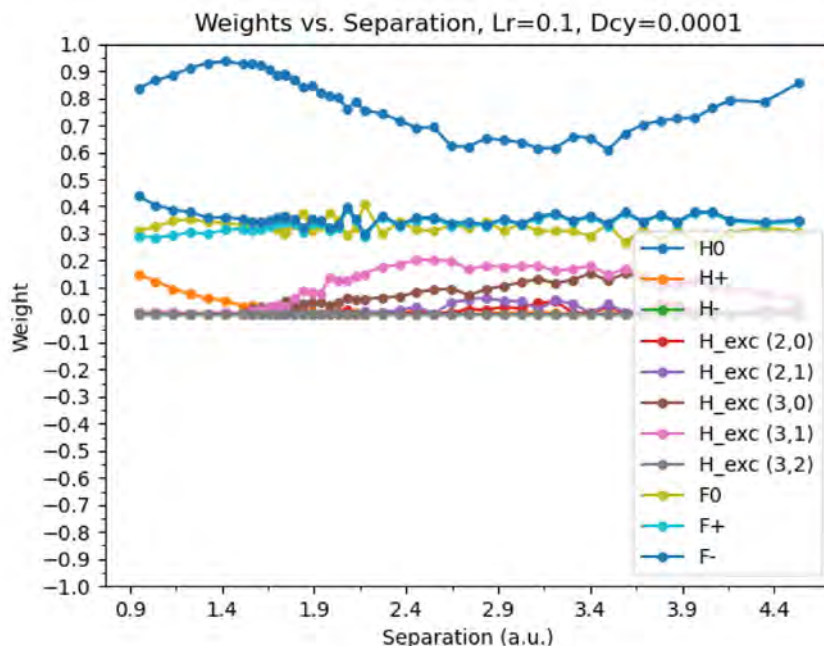


Figure 23: Weights for HF as a function of internuclear separation. States included are fluorine cation (light blue), neutral (light green), anion (dark blue) states and hydrogen cation (orange), neutral (blue), anion (green), and the hydrogen excited states: ($n = 2, l = 0$) (red), ($n = 2, l = 1$) (purple), ($n = 3, l = 0$) (brown), and ($n = 3, l = 1$) (pink).

are initially given weight, but closer to the known equilibrium bond distance (1.7328 a.u. [27]), the excited state weights fall and the hydrogen cation is given increased weight. This makes sense as well as one would expect that the single electron of hydrogen would become increasingly excited as the atom is brought closer to the demanding fluorine until it is pulled away from the hydrogen entirely, converting the hydrogen into a cation state.

Some elements of the observed weight vs. separation distributions are of concern. For instance, the fluorine ‘band’ consisting of all three charge states of fluorine having the same weight is hard to explain, but is likely due to fluorine’s lack of ensemble states in comparison with hydrogen: In the present calculations, fluorine represented only by its three charge states: F^0 , F^{+1} , and F^{-1} . No excited states of fluorine were included in the superposition. This imbalance could explain the fluorine band, as weight that would optimally put in a fluorine excited state had to instead be distributed among the charge state

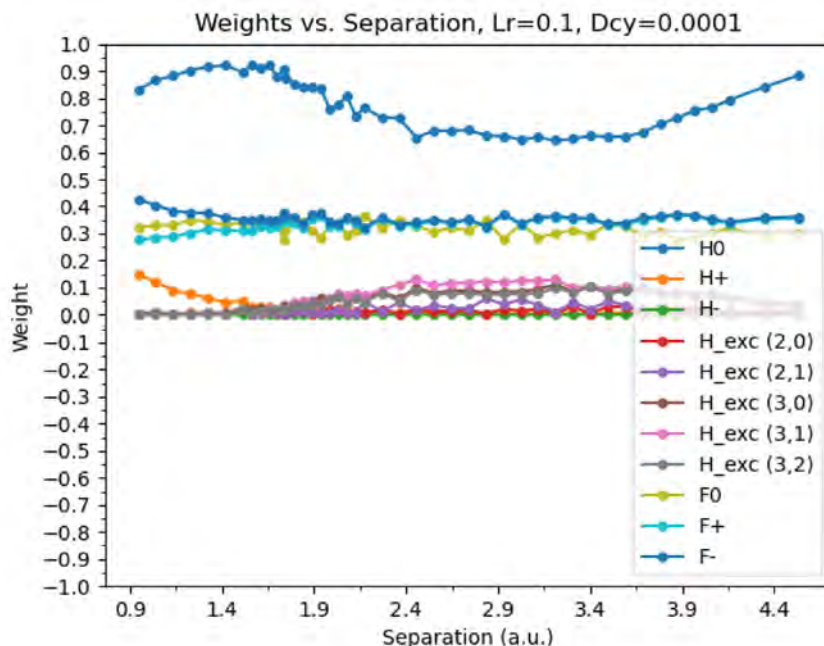


Figure 24: Weights for HF as a function of internuclear separation. States included are fluorine cation (light blue), neutral (light green), anion (dark blue) and hydrogen cation (orange), neutral (blue), anion (green), and all $n = 2$ and $n = 3$ hydrogen excited states.

weights.

4.2.2 Evaluation of the H^+ State Implementation

In order to evaluate the success of the RBF-NN hydrogen cation implementation for the AIM ensemble formulation, as detailed in Section 2.4, the effective charges of hydrogen and fluorine as a function of internuclear separation R were computed using Eq. (28). The results are plotted in Fig. 25.

As can be seen in the figure, at large internuclear separations the effective charges of both atoms are zero, and the atoms are neutral. This is as expected for two well-separated atoms. As the atoms are brought together, their effective charges increases due to charge transfer. Hydrogen gains a positive effective charge as fluorine gains a compensating negative charge. This is consistent with what would be expected in a HF bond, in which hy-

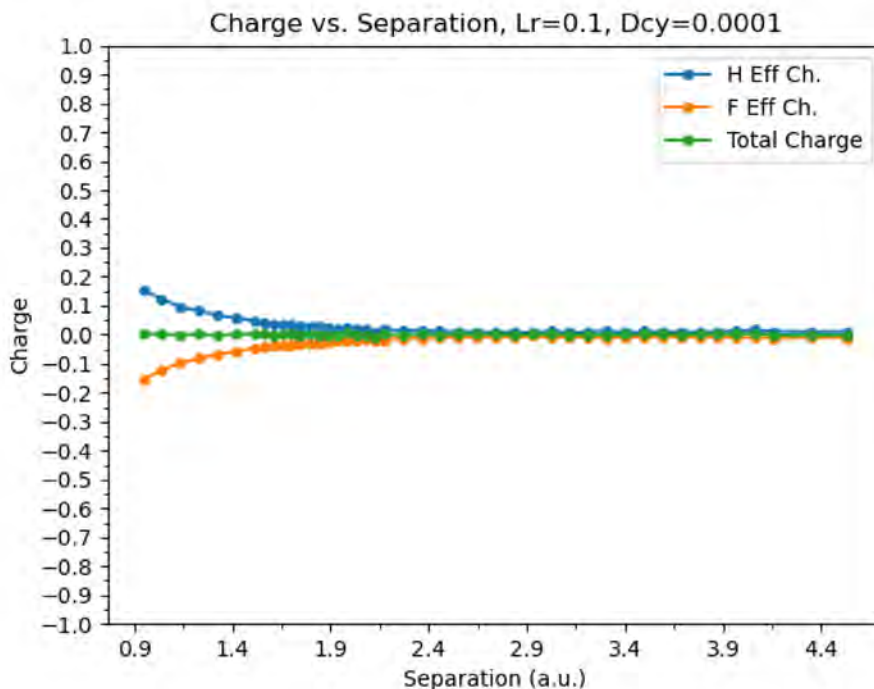


Figure 25: Effective charge of atoms of HF: hydrogen (blue) and fluorine (orange), as a function of internuclear separation. The total charge (green) is also included, constrained to be neutral by the RBF-NN loss function.

hydrogen acts as the cation (electron donor) and fluorine the anion (electron acceptor). Notably, the atoms of HF gain charge at ~ 1.7 a.u., the equilibrium distance of HF. This shows that the implemented hydrogen cation state acts consistently with what is expected chemically.

It must be noted that the effective charge given to the atoms of HF is quite small, with charges of approximately 0.2 in magnitude. Unlike the case of LiF reported above and in [4], this value disagrees with the results of Bader’s topological formulation, which gives an expected charge of 0.761 at the HF equilibrium separation of 1.7328 a.u. [27]. Thus, the effective charges of HF, although exhibiting the expected trend as a function of R , are smaller in magnitude than expected.

A likely culprit for the lower than expected charges is the lack of excited charge states of fluorine in the ensemble atom-in-molecule representation, in contrast to hydrogen, for

which up to five excited states were included. This was due to the fact that exact radial functions for excited states of hydrogen are known exactly, while for other larger elements, like fluorine, a 3D quantum mechanically generated electron density distribution for the isolated, excited states for fluorine would need to be generated. This calculated distribution could then be sphericalized and fit to the analytical functions detailed in Eq. (19) as was done for the radial basis functions for other the charge states. The generation of excited state density distributions for many-electron atoms is more involved than the straightforward computations of ground state neutral and ionic densities, and is presently underway [71]. Preliminary results for the first three excited states of Li^0 are shown in Fig. 26.

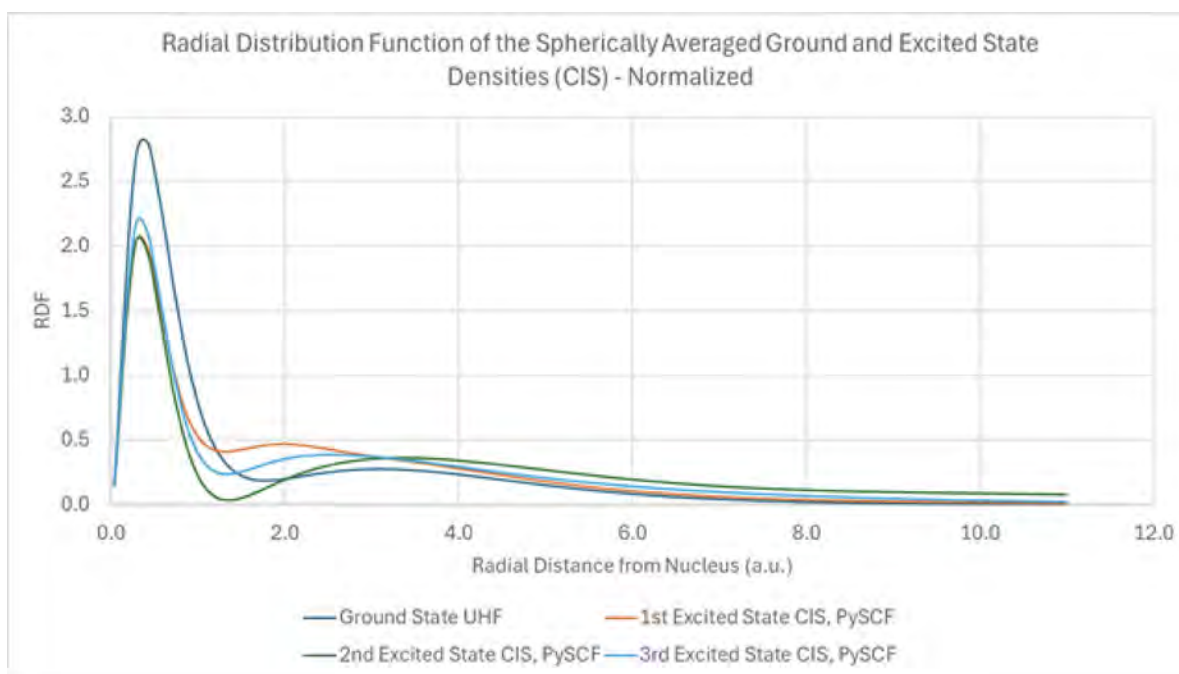


Figure 26: Radial distribution functions of the first three isolated excited states of Li [71].

The lack of fluorine excited states could have significant impact on the effectiveness of the hydrogen cation state. As noted earlier, the hydrogen cation state is informed only by the loss function: it has no density distribution to be included in the ensemble of states, but its weight is included in the loss function physical constraints. Thus, the hydrogen cation state responds to any increase in weight assigned to the fluorine anion state in order to meet the physical constraint that the total molecular charge is kept neutral. As

observed in Fig. 24, the fluorine states are distinguished from each other; instead, their weights form a single band of equal weight. This may be due to a failure of the RBF-NN formulation to be ‘strong’ enough to represent the fluorine capture of hydrogen’s single electron without inclusion of fluorine excited states. Adding these states may resolve this issue and lead to a more complete and physical hydrogen cation representation. Inclusion of such excited states will inspire further work in this project, enabling the exploration of how excited states affect charge transfer as a function of internuclear separation for HF, LiF, and other more complex molecules.

5 Spherical DFT and Relation to an AIM formulation

The work described in this section is a summary of a separate ms. to be submitted for publication [72].

Spherical density functional theory (DFT) [73] is a reformulation of the classic theorems of DFT, in which the role of the total density of a many-electron system, $\rho(\mathbf{r})$, is replaced by a *set* of sphericalized densities, $\{\bar{\rho}_i(r_i; \mathbf{R}_i)\}$, each centered at a nuclear position \mathbf{R}_i . This formulation was introduced by Theophilou [73] and subsequently extended by Nagy [74] to encompass constrained-search DFT approaches [75]. In spherical DFT, each $\bar{\rho}_i$ is obtained by averaging the total electron density over angular coordinates about the *i*th nucleus:

$$\bar{\rho}_i(r_i; \mathbf{R}_i) \equiv \int \rho(\mathbf{r}_i) d\Omega_i, \tag{29}$$

where $\mathbf{r}_i = \mathbf{r} - \mathbf{R}_i$, r_i is the radial distance from atom *i*, and Ω_i is the angular component of the differential volume in spherical polar coordinates centered about atom *i*.

In both Theophilou’s and Nagy’s original arguments, the nuclear positions $\{\mathbf{R}_i\}$ are treated as known inputs: Theophilou assumes them explicitly in defining local potentials $v_i(r_i; \mathbf{R}_i)$, while Nagy assumes them in order to apply Kato’s theorem [76, 77] to extract nuclear charges from known nuclear cusp locations in the density. Kato’s cusp condition

provides a differential relationship between the radial derivative of the electron density at a nucleus and the nuclear charge. The condition,

$$Z_i = -\frac{1}{2\rho_i(\mathbf{r} = \mathbf{R}_i)} \left. \frac{\partial \rho_i(\mathbf{r})}{\partial r} \right|_{\mathbf{r}=\mathbf{R}_i}, \quad (30)$$

implies that the electron density of an atom i decays exponentially very close to its nucleus, with rate of decay proportional to the nuclear charge. This allows the extraction of the nuclear charge, Z_i , from observed density behavior at known cusp locations.

Under these assumptions, both proofs establish that the set $\{\bar{\rho}_i\}$ suffices to recover the total electron density $\rho(\mathbf{r})$, and hence the external potential $v(\mathbf{r})$, in direct analogy with the original Hohenberg-Kohn theorem [78]. This established the equivalence of spherical DFT and traditional DFT.

In response to these works, we were motivated to address a specific assumption underlying the original proofs: the requirement that each sphericalized density be “tagged” with its nuclear origin. In recent work [72], we have shown that this assumption is unnecessary. The nuclear location information is already encoded within the set $\{\bar{\rho}_i\}$, through the preservation of nuclear cusp features. This result follows from methodology based in distance geometry [79–81], which enables the reconstruction of three-dimensional structures (i.e. all atomic 3D coordinate information) from knowledge of the pairwise distances between all atoms alone.

This process is illustrated numerically by plotting examples of sphericalized densities for component atoms of two exemplar systems glycine (Fig. 27) and LiF (Fig. 28). Fig. 27 depicts the sphericalized electron density for a hydrogen atom in glycine. Each sphericalized density retains its central cusp, but also exhibits smaller peaks corresponding to the nuclear cusps of other atoms in the system. These features arise from the fact that spherical averaging preserves cusp behavior, even at non-central nuclei (see Lemma 2 in [72]). By collecting the locations of these peaks across all $\bar{\rho}_i$, we can recover the full set of pair-

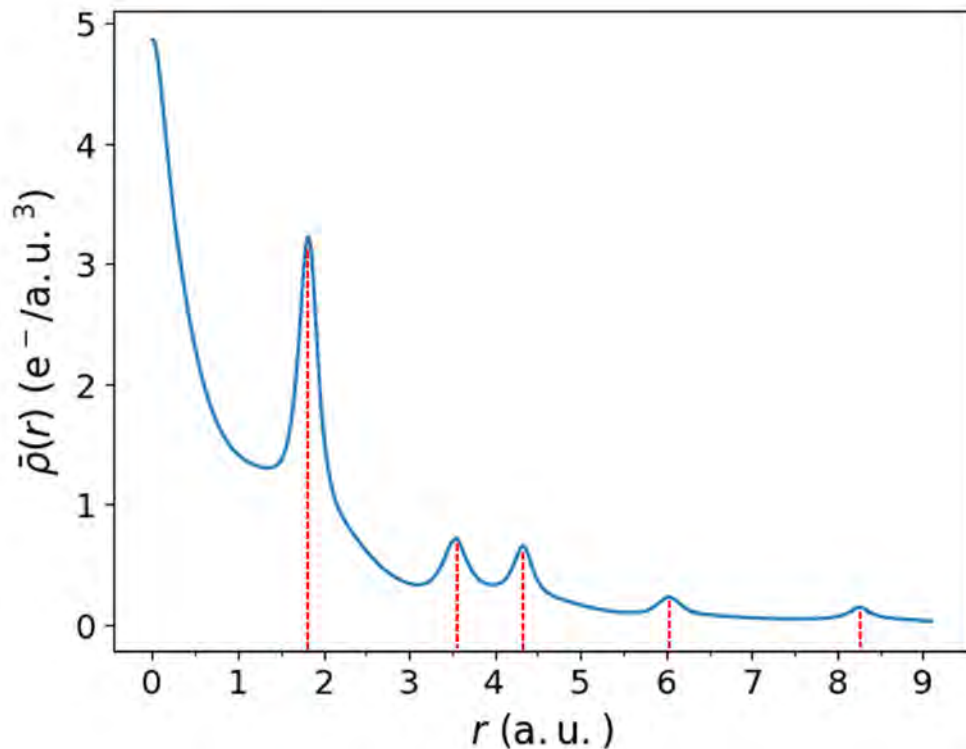


Figure 27: Spherical density distribution as a function of radial distance from the nucleus of a hydrogen atom of glycine. Although the electron density at the nucleus dominates the distribution, additional peaks are easily identified numerically as local maxima (indicated via red vertical lines), corresponding to the radial distances of other atoms from the center. As the distance from the origin increases, the magnitude of the peaks decreases, but they remain present. From [72].

wise distances between atoms in the molecule. Constructing a Euclidean distance matrix from this data, we then determine the nuclear positions via classical multidimensional scaling (MDS), without requiring any *a priori* knowledge of the 3D atomic coordinates.

5.1 Nuclear Location Reconstruction From Sphericalized Densities

We illustrate the encoding of atomic coordinate information within the spherical DFT electron density distributions using the LiF heteronuclear diatomic and the simplest amino acid, glycine ($\text{C}_2\text{H}_5\text{NO}_2$), whose structure is depicted in Fig. 29.

For both molecules, the total molecular electron density of the molecule at its opti-

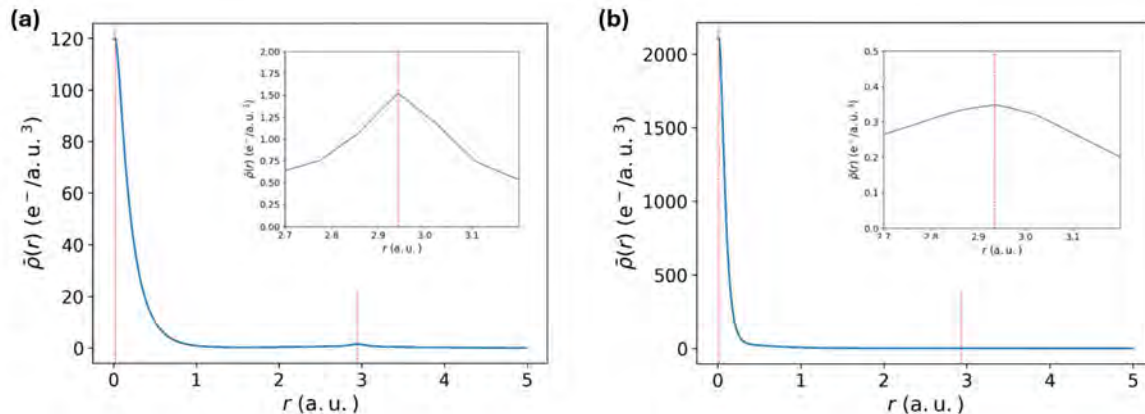


Figure 28: Sphericalized densities for the heteronuclear diatomic, LiF, at the equilibrium separation 1.564 Å (2.956 a.u.) (a) Origin of sphericalized density at Li. The location of the F atom is discernible as a peak in the distribution at the correct internuclear separation (see inset). (b) The second sphericalized density distribution, with F located at the origin. Although the peak for Li is smaller (due to its smaller size) it is still discernible in the inset, again at the correct internuclear separation. From [72].

mized ground state geometry was first generated using Gaussian'16 on a cubic grid with 0.0833 a.u. spacing between points. Each sphericalized electron density, found for each atom in the molecule, was then computed as a function of the radial distance r from its center (nucleus) by angle-averaging about the center using a custom Python code implementing Eq. (29). The sphericalized sets of electron densities for each molecule were then used to reconstruct the molecular geometry.

At the MP2/cc-aug-pvQZ level of theory, the LiF electron density distribution was calculated at a 1.564 Å separation, its accepted equilibrium separation [27]. For both atoms of LiF, the sphericalized electron density was computed as a function of the radial distance, r , resulting in Fig. 28. As can be seen in this figure, the sphericalization process for LiF results in two distributions: one with Li located at the origin and the other with F at the origin. For both distributions, a discernable peak occurs at a radius of 1.564 Å from the origin.

Since LiF is a diatomic, the locations of the sphericalized peaks trivially determine the geometry of the molecule. Each atom's sphericalized density distribution locates the other

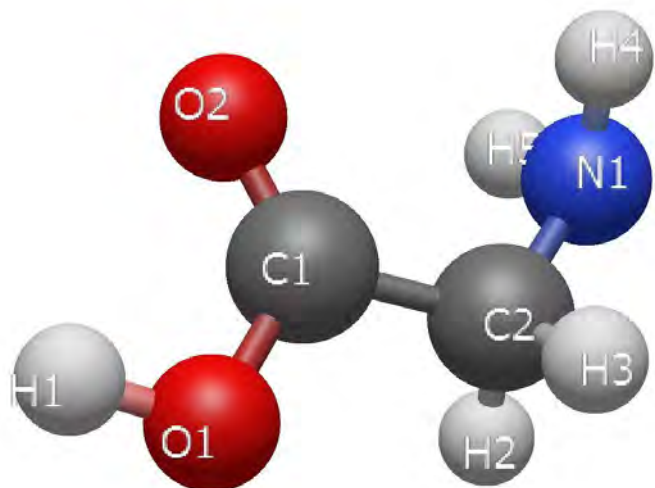


Figure 29: Optimized geometry of glycine from quantum mechanical calculations, calculated using Gaussian'16. From [72].

atom 1.564 Å away. Complexity arises when more than two atoms are involved.

This was seen after performing the same process of sphericalization for the atoms of glycine. The total molecular electron density of glycine at its optimized ground state geometry was used to generate the sphericalized densities about each of the ten glycine atoms using the aforementioned custom Python code. The sphericalized density for atom 6 of glycine (an H atom, labeled H1 in Fig. 29) is illustrated in Fig. 27, with six nuclear cusp peaks indicated by vertical red dashed lines. These correspond to the nuclear cusps at the central atom, a hydrogen, and five additional “heavy atom” (N,C,O) peaks of the glycine molecule (N1, C1, C2, O1, O2 in Fig. 29). These Kato peaks indicate the distances of other atoms from the center hydrogen. For each atom’s sphericalized density, these peaks can be identified, giving information on the distances between atoms.

As illustrated in Fig. 27, the set of atomic peak distances detected within each sphericalized density contribute a distinct atomic “fingerprint” of information from the given atom. Note that each calculated spherical density distribution exists as its own entity in isolation: once computed, it retains no knowledge of the atomic location about which its sphericalization was performed. In order to reconstruct this 3D atomic coordinate infor-

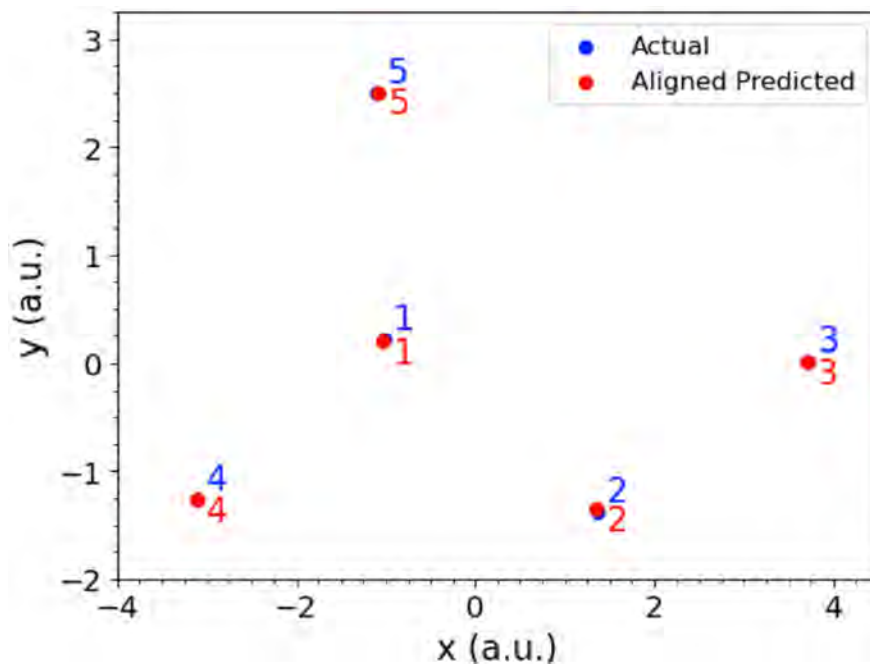


Figure 30: Two-dimension MDS projection comparing actual coordinates (blue) to the procrustes-aligned predicted coordinates (red). Atoms are labeled numerically.

mation, the distance information from all atoms in the molecule must be analyzed in combination. This is done by constructing a distance matrix which categorizes the distances from all atoms in the molecule to all other atoms. This distance matrix can then be used to determine the atomic locations of the atoms in the molecule through the use of a classical *multidimensional scaling* (MDS) algorithm, as detailed in [81].

The actual coordinates for glycine, calculated via geometry optimization of glycine at the MP2 level of theory using `Gaussian'16`, were used to evaluate the accuracy of the atomic coordinate reconstruction using the MDS algorithm and distance matrix computed using Kato cusp distances from the sphericalized densities (as illustrated in Fig. 27). The predicted and actual coordinates of glycine’s five heavy atoms, which lie in a 2D plane, are compared in Fig. 30. The agreement is excellent.

These numerical results highlight the main theoretical result of [72], namely, that the set of sphericalized densities contains more information than previously recognized. In particular, the nuclear positions assumed necessary for the prior proofs by Nagy and Theophilou,

are in fact already encoded within the sphericalized densities. Numerical results for LiF and glycine confirm that this encoding is recoverable in practice, at least for heavy atoms, using standard electronic structure data. The loss of angular information during sphericalization is compensated by the redundancy of the full spherical density set, which encodes relative distances between all atomic centers.

The process of sphericalization in spherical DFT mirrors the radial basis representation used in the RBF neural network framework, where the density is expressed as a sum of sphericalized atomistic electron densities centered at each nucleus. In both cases, angular information is discarded, and the result is a set of rotationally invariant functions. The sphericalized densities $\{\bar{\rho}_i\}$ in spherical DFT behave similarly to the RBF components in the RBF-NN model. This parallel raises a natural question. While spherical DFT does not define an explicit atom-in-molecule partitioning of the total molecular density, the spherical density ensemble $\{\bar{\rho}_i\}$ shares several properties expected of an atom-in-molecule (AIM): localization and a recoverable relation to the full electron density. Therefore, it is natural to ask whether a sphericalized set as used in spherical DFT can be regarded as an AIM representation of the molecule.

5.2 A Sphericalized Set of Densities as an AIM

In her 2019 paper on this subject [82], Nagy addresses the question of whether a set of sphericalized densities, as introduced in spherical DFT, resembles the pseudoatoms of an atom-in-molecule representation. She acknowledges the resemblance between each $\bar{\rho}_i$ and an atomic electron density, particularly in the near-nucleus region, where the density obeys a Schrödinger-like differential equation and displays the expected nuclear cusp behavior. However, Nagy argues that despite these similarities, the sphericalized densities do not constitute an AIM in the traditional sense. In particular, they lack *transferability*, a desirable property of many AIM models, in which the pseudoatom components retain simi-

lar forms across different molecular environments.

This can be seen when analyzing the sphericalized density distribution shown in Fig. 27. The sphericalized density distribution belongs to a hydrogen atom, but not just any hydrogen atom, the sphericalized density of a particular hydrogen in glycine. Such a sphericalized density could not be used to represent a hydrogen in another molecular system, e.g. H₂O, as remnants of other atoms in glycine exist in the density distribution for the particular hydrogen (seen as the Kato cusp peaks). This specific electron density distribution for hydrogen could not be applied even to a different hydrogen atom in glycine, as every hydrogen is located in a different spatial position within the molecule and has peaks in the distribution at different locations. Transferability is lost as each sphericalized density is unique to a particular atom in a particular molecular environment.

Nagy also notes that while the short-range portion of a sphericalized density resembles that of an isolated atom, the long-range behavior reflects the presence of a molecule: each $\bar{\rho}_i$ integrates to the total number of electrons, not the atomic number, and decays according to the ionization potential of the whole molecular system. This hybrid character – locally atomic but globally molecular – arguably negates the ability of spherical set of densities to be parts of an AIM construct, as each individual sphericalized density contains information from the entire molecule.

Several key points must be brought up in address to Nagy’s arguments that a sphericalized set of densities as used in spherical DFT do not qualify as an AIM formulation. The argument that pseudoatoms of molecular system must be transferable between other molecular systems excludes other, well-known AIM formulations including Bader’s topological approach [25, 27]. Bader’s formulation, discussed earlier in Section 1.2.1, is characterized by a partitioning of the landscape of a molecule’s electron density along zero-flux surfaces. This also results in pseudoatoms dependent on the global, molecular environment. Since the hard spatial partitioning that defines individual pseudoatoms is dependent on an atom’s relation to other atoms, each pseudoatom is dependent on its specific molec-

ular environment. By Nagy’s argument that an atom-in-molecule must be transferable between molecular systems, this definition of a pseudoatom would not qualify.

Instead, it can be argued that Bader’s topological approach qualifies as an AIM formulation due to its consistent application of a defined methodology. Transferability exists in the sense that the process of dividing atoms along zero-flux surfaces can be applied to any molecular system, if computational cost is disregarded. This sense of transferability can also be associated with the definition of a sphericalized set of densities in spherical DFT. The process of sphericalizing a molecular density about each nucleus in a molecule is a procedural method that can be applied to any molecular system. In that sense it is just as transferable as Bader’s formulation.

The more definitive requirement that prevents a set of sphericalized densities from qualifying as an AIM is that the sum of all pseudoatoms does not sum to the total molecular density at every point in space, violating the condition of Eq. (8). That is,

$$\rho_{\text{mol}}(\vec{\mathbf{r}}) \neq \sum_{i=1}^{N_{\text{atoms}}} \bar{\rho}_i(\vec{\mathbf{r}}), \quad (31)$$

where $\bar{\rho}_i(\vec{\mathbf{r}})$ designates the sphericalized density for an atom i in the molecule. This is especially clear due to the fact that each sphericalized density distribution integrated over all space results in the total number of electrons. Summing all sphericalized densities together would therefore over-count the number of electrons, and the amount of density, in a molecular system.

Thus, in agreement with Nagy, a set of sphericalized densities as used in spherical DFT stand as a different conceptual device than an AIM, as it is most commonly defined.

5.3 Sphericalization Used in the Ensemble AIM Formulation

The sphericalized densities used as the basis for the ensemble AIM discussed in this work are distinct from a sphericalized set as used in spherical DFT most obviously due to their representation in terms of isolated atom and ion densities, rather than those of a molecular system. In this sense, they correspond to the transferable atom-in-molecule partitions referred to by Nagy. Each sphericalized, ensemble state density distribution exists independently from any molecular system and can therefore be placed into any environment without any change in its form. It is only the weights defining the superposition of the states that change with incorporation into a new chemical environment. This offers insight into the scalability of the ensemble AIM. The ensemble AIM can be applied to any molecular system without the individual partitioned ensemble densities being affected by the increased complexity of the molecular environment.

An exploration of information encoded by a sphericalized set of densities also displays the preservation of information despite the sphericalization process. A sphericalization routine, by definition, removes all angular information. The dimensionality of the electron density is reduced from three-dimensions, r, θ, ϕ , to only one, r . Despite this seeming loss of information, we have demonstrated that information still exists in a sphericalized density about the surrounding of a nucleus. Taken together, we have demonstrated that the set of sphericalized densities contains sufficient encoded information so as to enable the reconstruction the locations of all nuclei in a molecule.

The sphericalization of the ensemble electron densities for use in our AIM formulation was done so in order to reduce complexity. Including an angular component to the basis densities would have required careful consideration of the orientation of every atom in a molecule, thus limiting the scope of potential chemical interactions and bonding characters that can be modeled. This represents a current limitation of so-called *equivariant neural networks* currently used to model potential energy functions for molecular dynamics simu-

lations [83]. Notably, the 3rd generation of the AlphaFold neural network model of protein structure utilizes distance, rather than angular information [48]. Using sphericalized ensemble electron densities also enabled direct use of a radial basis function neural network. As exemplified by our analysis of the information encoded by the sphericalization routine in spherical DFT, the sacrifice of angular components does not signify a complete loss of information. Complex patterns inherent within in the electron density distributions of individual atomic states survive being angularly averaged over, and contribute through their relative spatial center locations to yield an accurate description of the total electron density of a molecule.

6 Conclusions and Future Work

This work introduced a physically-motivated, radially symmetric neural network architecture that encodes molecular electron densities using atom-centered basis functions. By structuring the network around ensemble representations of isolated atomic states, we have shown that we can capture meaningful quantum chemical behavior, such as charge transfer, bonding, and dissociation. The radial basis function neural network (RBF-NN) is not only compact, but interpretable: each weight in the network corresponds directly to a chemically relevant atomic state. Despite its architectural simplicity, the model is able to track bonding transitions in diatomics and detect electron redistribution even when other observables, like the dipole moment, fail to do so.

This work also contributes a viable scheme for modeling the hydrogen cation, H^+ , within the RBF-NN framework. Additionally, the inclusion of exact hydrogen excited states within the ensemble formulation enables a more nuanced picture of state occupations and total electron density of molecules containing hydrogen, like HF.

The RBF-NN approach demonstrates that spherical representations can still preserve the quantum mechanical structure inherent in the electron density. In parallel work, we have explored how this is accomplished in the formal application of spherical DFT. Quan-

tum mechanical information is encoded in both of these representations: as spherical atomic densities in the RBF-NN framework and as sphericalized total densities in spherical DFT. While distinct in formulation, both approaches encode molecular information through reduced radial forms that remove angular dependence without discarding chemically-relevant structure. In both cases, spherical symmetry enables a basis for representing complex molecular behavior. The radial decomposition used in the RBF-NN reflects an ensemble atom-in-molecule perspective, while the spherical DFT framework involves a system-level electronic structure viewpoint, formally leading to atom-centric sphericalized densities. Taken together, these methods demonstrate that sphericalization – often viewed as a simplification – can serve as a powerful encoding tool, capturing the essential features of molecular electronic structure and chemical environments in a scalable and physically interpretable way.

Future directions include extending this framework to larger polyatomic systems, where the advantage of radial density scaling becomes even more critical. Different molecules can be selected to represent more diverse bonding patterns, including covalent bonding (CO) and charge polarization in homonuclear diatomics (Li_2). Larger molecules can be modeled in the future, such as amino acids, the building blocks of proteins. This is now made possible by the ability to represent hydrogen’s cation state in the RBF-NN formulation, another important contribution of this work.

In order to model more complex bonding patterns, excited states will need to be included in the ensemble representation for atoms other than hydrogen. The inclusion of excited states completes the formal ensemble representation and will enable planned extensions to increasingly complex bonding patterns.

Finally, this work has contributed to establishing a conceptual and practical validation for the ensemble density concept as a key component of the ensemble charge-transfer embedded atom method (ECT-EAM) [18, 35]. ECT-EAM is a generalized force field for describing the potential energy surfaces of molecules and materials. In ECT-EAM, the co-

hesive energy is described using statistical ensembles of energetic contributions, in addition to constituent pseudoatoms, with a single set of ensemble weights $\{\omega_i\}$ applied consistently in both energy and density ensembles. That is, the same weights, used to express the total density as

$$\rho(\mathbf{r}; \{\omega_i\}) = \sum_{i=1}^{N_{\text{ens}}} \omega_i \rho_i(\mathbf{r}), \quad (32)$$

also define the ensemble energy,

$$E_v[\{\omega_i\}; \rho(\mathbf{r})] = \sum_{i=1}^{N_{\text{ens}}} \omega_i E_i. \quad (33)$$

By demonstrating that these weights can be learned from molecular total density data and are sufficient to reconstruct molecular densities and represent bond formation and breaking, the present work provides numerical validation of the ensemble density construction in preparation for integration with the ensemble energy formalism of ECT-EAM [18].

References

- [1] Robert O Jones. Density functional theory: Its origins, rise to prominence, and future. *Reviews of Modern Physics*, 87(3):897–923, 2015.
- [2] Douglas L Strout and Gustavo E Scuseria. A quantitative study of the scaling properties of the Hartree–Fock method. *The Journal of Chemical Physics*, 102(21):8448–8452, 1995.
- [3] Trygve Helgaker, Poul Jorgensen, and Jeppe Olsen. *Molecular Electronic-Structure Theory*. John Wiley & Sons, New York, 2013.
- [4] Godwin Amo-Kwao. *Radial basis densities and the density functional-based atom-in-molecule: Designing charge-transfer potentials*. Ph.D. dissertation, University of New Mexico, 2020.
- [5] Godwin Amo-Kwao and Susan R. Atlas. Radial basis function electron densities with asymptotic constraints. *arXiv preprint*, [chem-phys], 2025.
- [6] Felix Musil, Andrea Grisafi, Albert P Bartók, Christoph Ortner, Gábor Csányi, and Michele Ceriotti. Physics-inspired structural representations for molecules and materials. *Chemical Reviews*, 121(16):9759–9815, 2021.
- [7] Norbert Schuch and Frank Verstraete. Computational complexity of interacting electrons and fundamental limitations of density functional theory. *Nature Physics*, 5(10):732–735, 2009.
- [8] Pierre Hohenberg and Walter Kohn. Inhomogeneous electron gas. *Physical Review*, 136(3B):B864, 1964.
- [9] Walter Kohn and Lu Jeu Sham. Self-consistent equations including exchange and correlation effects. *Physical Review*, 140(4A):A1133, 1965.

- [10] John P Perdew and Karla Schmidt. Jacob’s ladder of density functional approximations for the exchange-correlation energy. *AIP Conference Proceedings*, 577:1–20, 2001.
- [11] Axel D Becke. A new mixing of Hartree-Fock and local density-functional theories. *Journal of Chemical Physics*, 98(2):1372–1377, 1993.
- [12] Michael G Medvedev, Ivan S Bushmarinov, Jianwei Sun, John P Perdew, and Konstantin A Lyssenko. Density functional theory is straying from the path toward the exact functional. *Science*, 355(6320):49–52, 2017.
- [13] Chengteh Lee, Weitao Yang, and Robert G Parr. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Physical Review B*, 37(2):785, 1988.
- [14] Axel D Becke. Density-functional thermochemistry III. The role of exact exchange. *The Journal of Chemical Physics*, 98(7):5648–5652, 1993.
- [15] Carlo Adamo and Vincenzo Barone. Toward reliable density functional methods without adjustable parameters: The PBE0 model. *The Journal of Chemical Physics*, 110(13):6158–6170, 1999.
- [16] Jeng-Da Chai and Martin Head-Gordon. Systematic optimization of long-range corrected hybrid density functionals. *The Journal of Chemical Physics*, 128(8):84106, 2008.
- [17] Jeng-Da Chai and Martin Head-Gordon. Long-range corrected hybrid density functionals with damped atom–atom dispersion corrections. *Physical Chemistry Chemical Physics*, 10(44):6615–6620, 2008.
- [18] Susan R Atlas. Embedding quantum statistical excitations in a classical force field. *The Journal of Physical Chemistry A*, 125(17):3760–3775, 2021.

- [19] Robert G Parr, Paul W Ayers, and Roman F Nalewajski. What is an atom in a molecule? *Journal of Physical Chemistry A*, 109(17):3957–3959, 2005.
- [20] Farnaz Heidar-Zadeh, Paul W Ayers, Toon Verstraelen, Ivan Vinogradov, Esteban Vöhringer-Martinez, and Patrick Bultinck. Information-theoretic approaches to atoms-in-molecules: Hirshfeld family of partitioning schemes. *The Journal of Physical Chemistry A*, 122(17):4219–4245, 2017.
- [21] Chérif F Matta and Richard FW Bader. An experimentalist’s reply to “What is an atom in a molecule?”. *The Journal of Physical Chemistry A*, 110(19):6365–6371, 2006.
- [22] Robert S Mulliken. Electronic population analysis on LCAO–MO molecular wave functions. I. *The Journal of Chemical Physics*, 23(10):1833–1840, 1955.
- [23] Per-Olov Löwdin. On the non-orthogonality problem connected with the use of atomic wave functions in the theory of molecules and crystals. *The Journal of Chemical Physics*, 18(3):365–375, 1950.
- [24] JC Gil Montoro and JLF Abascal. The Voronoi polyhedra as tools for structure determination in simple disordered systems. *The Journal of Physical Chemistry*, 97(16):4211–4215, 1993.
- [25] RFW Bader, TT Nguyen-Dang, and Y Tal. A topological theory of molecular structure. *Reports on Progress in Physics*, 44(8):893, 1981.
- [26] Fred L Hirshfeld. Bonded-atom fragments for describing molecular charge densities. *Theoretica Chimica Acta*, 44:129–138, 1977.
- [27] R F W Bader. *Atoms in Molecules: A Quantum Theory*. Clarendon Press, Oxford, 1990.

- [28] Richard FW Bader and Preston J MacDougall. Toward a theory of chemical reactivity based on the charge density. *Journal of the American Chemical Society*, 107(24):6788–6795, 1985.
- [29] Richard FW Bader and Chérif F Matta. Atoms in molecules as non-overlapping, bounded, space-filling open quantum systems. *Foundations of Chemistry*, 15:253–276, 2013.
- [30] Graeme Henkelman, Andri Arnaldsson, and Hannes Jónsson. A fast and robust algorithm for Bader decomposition of charge density. *Computational Materials Science*, 36(3):354–360, 2006.
- [31] Ernest R Davidson and Subhas Chakravorty. A test of the Hirshfeld definition of atomic charges and moments. *Theoretica Chimica Acta*, 83(5):319–330, 1992.
- [32] Patrick Bultinck, Paul W Ayers, Stijn Fias, Koen Tiels, and Christian Van Alsenoy. Uniqueness and basis set dependence of iterative Hirshfeld charges. *Chemical Physics Letters*, 444(1-3):205–208, 2007.
- [33] Farnaz Heidar-Zadeh, Carlos Castillo-Orellana, Maximilian van Zyl, Leila Pujal, Toon Verstraelen, Patrick Bultinck, Esteban Vohringer-Martinez, and Paul W Ayers. Variational hirshfeld partitioning: General framework and the additive variational hirshfeld partitioning method. *Journal of Chemical Theory and Computation*, 20(22):9939–9953, 2024.
- [34] Roman F Nalewajski and Robert G Parr. Information theory, atoms in molecules, and molecular similarity. *Proceedings of the National Academy of Sciences (USA)*, 97(16):8879–8882, 2000.
- [35] Krishna Muralidharan, Steven M Valone, and Susan R Atlas. Environment dependent charge potential for water. *arXiv:0705.0857* [cond-mat], 2007.

- [36] Max Born. Statistical interpretation of quantum mechanics. *Science*, 122(3172):675–679, 1955.
- [37] Kurt Wüthrich. Nmr studies of structure and function of biological macromolecules. *Bioscience Reports*, 23(4):119–168, 2003.
- [38] Christopher M Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.
- [39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [40] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
- [41] David Lowe and D Broomhead. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2(3):321–355, 1988.
- [42] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- [43] Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.
- [44] 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.
- [45] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, 2021.
- [46] Carlos Lassance, Vincent Gripon, and Antonio Ortega. Representing deep neural networks latent space geometries with graphs. *Algorithms*, 14(2):39, 2021.

- [47] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Židek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.
- [48] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016):493–500, 2024.
- [49] Qichao Que and Mikhail Belkin. Back to the future: Radial basis function networks revisited. In *Artificial Intelligence and Statistics*, volume 51, pages 1375–1383. Proceedings of Machine Learning Research, 2016.
- [50] John McCormick. Radial Basis Function Network (RBFN) Tutorial. mccormickml.com/2013/08/15/radial-basis-function-network-rbf-tutorial/, 2013.
- [51] Yue Wu, Hui Wang, Biaobiao Zhang, and K-L Du. Using radial basis function networks for function approximation and classification. *International Scholarly Research Notices*, 2012(1):324194, 2012.
- [52] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003.
- [53] Michael J Willatt, Félix Musil, and Michele Ceriotti. Atom-density representations for machine learning. *The Journal of Chemical Physics*, 150(15):154110, 2019.
- [54] Michael A Nielsen. *Neural Networks and Deep Learning*, volume 25. Determination Press San Francisco, CA, USA, 2015.
- [55] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks

for machine learning lecture 6a: Overview of mini-batch gradient descent.

www.cs.toronto.edu/hinton/coursera/lecture6/lec6.pdf, 2012.

- [56] Martin Karplus and Richard N Porter. *Atoms and molecules: An introduction for students of physical chemistry*. W.A. Benjamin, New York, 1970.
- [57] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467* [cs.DC], 2016.
- [58] François Chollet et al. Keras. <https://keras.io>, 2015.
- [59] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980* [cs.LG], 2014.
- [60] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.
- [61] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747* [cs.LG], 2016.
- [62] M J Frisch, G W Trucks, H B Schlegel, G E Scuseria, M A Robb, J R Cheeseman, G Scalmani, V Barone, G A Petersson, H Nakatsuji, et al. Gaussian '16 Revision C.01, 2016. Gaussian Inc. Wallingford CT.
- [63] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv:1711.05101* [cs.LG], 2017.
- [64] Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in Adam. *arXiv:1711.05101* [cs.LG], 5:5, 2017.

- [65] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. *arXiv:1808.02941* [cs.LG], 2018.
- [66] AJC Varandas. Accurate ab initio potential energy curves for the classic LiF ionic-covalent interaction by extrapolation to the complete basis set limit and modeling of the radial nonadiabatic coupling. *The Journal of Chemical Physics*, 131(12):124128, 2009.
- [67] John P Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Physical Review Letters*, 77(18):3865, 1996.
- [68] Renato Colle and Oriano Salvetti. Approximate calculation of the correlation energy for the closed shells. *Theoretica Chimica Acta*, 37:329–334, 1975.
- [69] Benny G Johnson, Peter MW Gill, and John A Pople. The performance of a family of density functional methods. *The Journal of Chemical Physics*, 98(7):5612–5626, 1993.
- [70] Axel D Becke. Density-functional thermochemistry. V. Systematic optimization of exchange-correlation functionals. *The Journal of Chemical Physics*, 107(20):8554–8560, 1997.
- [71] Chance M Baxter, Susan R Atlas, and Steven M Valone. Ensemble embedding functions for atomistic dynamics. *arXiv preprint* [cond-mat], 2025.
- [72] Sol Samuels, Chance M Baxter, and Susan R Atlas. Information encoding in spherical DFT. *arXiv preprint* [chem-phys], 2025.
- [73] Andreas K Theophilou. A novel density functional theory for atoms, molecules, and solids. *The Journal of Chemical Physics*, 149(7):074104, 2018.
- [74] A Nagy. Density functional theory from spherically symmetric densities. *The Journal of Chemical Physics*, 149(20):204112, 2018.

- [75] Mel Levy. Universal variational functionals of electron densities, first-order density matrices, and natural spin-orbitals and solution of the v -representability problem. *Proceedings of the National Academy of Sciences (USA)*, 76(12):6062–6065, 1979.
- [76] T. Kato. On the eigenfunctions of many-particle systems in quantum mechanics. *Communications on Pure and Applied Mathematics*, 10(2):151–177, 1957.
- [77] Erich Steiner. Charge densities in atoms. *The Journal of Chemical Physics.*, 39(9):2365–2366, 1963.
- [78] Pierre Hohenberg and Walter Kohn. Inhomogeneous electron gas. *Physical Review*, 136(3B):B864, 1964.
- [79] Gordon M Crippen and Timothy F Havel. Stable calculation of coordinates from distance information. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 34(2):282–284, 1978.
- [80] Timothy F Havel. Distance geometry: Theory, algorithms, and chemical applications. *Encyclopedia of Computational Chemistry*, 120:723–742, 1998.
- [81] Ivan Dokmanic, Reza Parhizkar, Juri Ranieri, and Martin Vetterli. Euclidean distance matrices: Essential theory, algorithms, and applications. *IEEE Signal Processing Magazine*, 32(6):12–30, 2015.
- [82] Á Nagy. Spherical density functional theory and atoms in molecules. *The Journal of Physical Chemistry A*, 124(1):148–151, 2019.
- [83] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1):2453, 2022.

A Python Code for Implementing the RBF-NN

The following Python code implements the Radial Basis Function Neural Network (RBF-NN) for modeling a diatomic molecule as described in the main text. The code is fully general, but is specialized here to the heteronuclear diatomic hydrogen fluoride (HF) for clarity.

A.1 Neural Network Configuration

The RBF-NN is implemented using TensorFlow and Keras, incorporating a custom layer that applies radial basis functions as activation functions. Key hyperparameters, including learning rate, batch size, and loss function weights, are defined to guide the training process.

The core of the network is the `RadialBasisFunction` layer, which computes the activation of each node based on the radial distance from the atomic centers. This layer encapsulates the mathematical formulation of the basis functions and their dependencies on atomic parameters. Each node corresponds to a distinct atomic charge state, with initial weights assigned to each state to facilitate optimization during training.

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.integrate import quad
import keras.backend as K
import sys
from scipy.integrate import trapz
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import Callback
from rbf_functions import M5, M6
from rbf_functions import hydrogen_density as HDen
print('imports complete')
```

```

trial = '7all5exc'

#Define parameters:
lr = 0.1 #ADAM learning rate
batch_s = 1024 #Batch Size
start_dcy = 1e-4 #ADAM weight decay (L2 regularization)
end_dcy = start_dcy
epoch_num = 500 #Max number of epochs
stopping_patience = 50 #patience for training end
epsilon = 1e-2 #Tolerance to end training

#Loss function parameters:
omega1 = 1e3
omega2 = 10
omega3 = 100
omega4 = 100

# Number of Nodes = Number of atomic states
nodes = 11

#Class Defines RBF Layer
class RadialBasisFunction(tf.keras.layers.Layer):
    def __init__(self, nodes, R, N, A0, B0, C0, D0, Z, beta, zeta, m, alpha, gamma, eta,
        weights):
        super(RadialBasisFunction, self).__init__()
        self.units = nodes
        self.R = R
        self.N = N

        self.A0 = A0
        self.B0 = B0
        self.C0 = C0
        self.D0 = D0

        self.Z = Z
        self.m = m

        self.alpha = alpha
        self.beta = beta
        self.zeta = zeta
        self.gamma = gamma

```

```

self.eta = eta

self.epsilon = 1e-8

#Hydrogen Charged
self.IwA0 = weights[0]
self.IwA1 = weights[1]
self.IwA2 = weights[2]

#Fluorine Charged
self.IwB0 = weights[3]
self.IwB1 = weights[4]
self.IwB2 = weights[5]

#Hydrogen Excited
self.IwA3 = weights[6]
self.IwA4 = weights[7]
self.IwA5 = weights[8]
self.IwA6 = weights[9]
self.IwA7 = weights[10]

def build(self, input_shape):
    # Adds weight variables to the layer

    #HYDROGEN ground (0, +, -)
    self.wA0 = self.add_weight(name='wA0',
                               shape=(1,),
                               initializer=tf.keras.initializers.Constant(value=self.IwA0
                                                                              ),
                               trainable=True,
                               constraint = tf.keras.constraints.NonNeg())

    self.wA1 = self.add_weight(name='wA1',
                               shape=(1,),
                               initializer=tf.keras.initializers.Constant(value=self.IwA1
                                                                              ),
                               trainable=True,
                               constraint = tf.keras.constraints.NonNeg())

    self.wA2 = self.add_weight(name='wA2',
                               shape=(1,),
                               initializer=tf.keras.initializers.Constant(value=self.IwA2
                                                                              ),

```

```

        trainable=True,
        constraint = tf.keras.constraints.NonNeg())

#FLUORINE (0, +, -)
self.wB0 = self.add_weight(name='wB0',
                           shape=(1,),
                           initializer=tf.keras.initializers.Constant(value=self.IwB0
                                ),
                           trainable=True,
                           constraint = tf.keras.constraints.NonNeg())
self.wB1 = self.add_weight(name='wB1',
                           shape=(1,),
                           initializer=tf.keras.initializers.Constant(value=0.0),
                           trainable=True,
                           constraint = tf.keras.constraints.NonNeg())
self.wB2 = self.add_weight(name='wB2',
                           shape=(1,),
                           initializer=tf.keras.initializers.Constant(value=self.IwB2
                                ),
                           trainable=True,
                           constraint = tf.keras.constraints.NonNeg())

#Hydrogen Excited
self.wA3 = self.add_weight(name='wA3',
                           shape=(1,),
                           initializer=tf.keras.initializers.Constant(value=self.IwA3
                                ),
                           trainable=True,
                           constraint = tf.keras.constraints.NonNeg())
self.wA4 = self.add_weight(name='wA4',
                           shape=(1,),
                           initializer=tf.keras.initializers.Constant(value=self.IwA4
                                ),
                           trainable=True,
                           constraint = tf.keras.constraints.NonNeg())
self.wA5 = self.add_weight(name='wA5',
                           shape=(1,),
                           initializer=tf.keras.initializers.Constant(value=self.IwA5
                                ),
                           trainable=True,
                           constraint = tf.keras.constraints.NonNeg())

```

```

self.wA6 = self.add_weight(name='wA6',
                           shape=(1,),
                           initializer=tf.keras.initializers.Constant(value=self.IwA6
                                ),
                           trainable=True,
                           constraint = tf.keras.constraints.NonNeg())
self.wA7 = self.add_weight(name='wA7',
                           shape=(1,),
                           initializer=tf.keras.initializers.Constant(value=self.IwA7
                                ),
                           trainable=True,
                           constraint = tf.keras.constraints.NonNeg())

super(RadialBasisFunction, self).build(self.units)

def call(self, inputs):
    coor = inputs

    # H (0, +, -, exc)
    RBF_0 = (
        self.wA0 * HDen(self, coor, 0, n=1, l=0) +
        self.wA1 * M5(self, coor, 0, 1, 1) +
        self.wA2 * M5(self, coor, 0, 2, 1) +
        self.wA3 * HDen(self, coor, 0, n=2, l=0) +
        self.wA4 * HDen(self, coor, 0, n=2, l=1) +
        self.wA5 * HDen(self, coor, 0, n=3, l=0) +
        self.wA6 * HDen(self, coor, 0, n=3, l=1) +
        self.wA7 * HDen(self, coor, 0, n=3, l=2)
    )

    # F (0, +, -)
    RBF_1 = (
        self.wB0 * M5(self, coor, self.R, 0, 2) +
        self.wB1 * M5(self, coor, self.R, 1, 2) +
        self.wB2 * M6(self, coor, self.R, 2, 2)
    )

    RBF_r = RBF_0 + RBF_1

    return RBF_r

```

A.2 Loss and Early Stopping Functions

The following is code used in order to evaluate characteristics of the training output as the RBF-NN learns. The loss function (Section 2.3) is included as the function `loss_fit()`. This function's output is what is given to the ADAM optimizer. The early stopping routine is also included as the class `EarlyStoppingWithBestWeights(Callback)`.

```
def loss_fit(rho_true, rho_pred):
    #squared difference = (rho_true - rho_pred)^2
    try:
        weights = model.layers[0].weights
    except NameError:
        weights = model_val.layers[0].weights

    # Extract the required weights
    wA0 = weights[0]
    wA1 = weights[1]
    wA2 = weights[2]

    wB0 = weights[3]
    wB1 = weights[4]
    wB2 = weights[5]

    wA3 = weights[6]
    wA4 = weights[7]
    wA5 = weights[8]
    wA6 = weights[9]
    wA7 = weights[10]

    reduced_loss = (omega1 * (tf.reduce_mean(tf.square(rho_true - rho_pred))) +
                    omega2 * tf.abs((wA0 * 0) + (wA1 * 1) + (wA2 * -1) + (wB0 * 0) + (wB1 * 1) + (
                        wB2 * -1))) +
                    omega3 * tf.abs((wA0 + wA1 + wA2 + wA3 + wA4 + wA5 + wA6 + wA7) - 1) +
                    omega4 * tf.abs(wB0 + wB1 + wB2 - 1))
    return reduced_loss

#End of training early stopping callback routine
class EarlyStoppingWithBestWeights(Callback):
```

```

def __init__(self, monitor='loss', min_delta=0, patience=5, n_average=20, factor=0.9,
             verbose=0, mode='auto', baseline=None, start_epoch=0):
    super(EarlyStoppingWithBestWeights, self).__init__()

    self.monitor = monitor
    self.min_delta = min_delta
    self.patience = patience
    self.n_average = n_average
    self.factor = factor
    self.verbose = verbose
    self.wait = 0
    self.stopped_epoch = 0
    self.best = None
    self.best_weights = None # New attribute to store the best weights
    self.mode = mode
    self.baseline = baseline
    self.loss_history = []
    self.start_epoch = start_epoch

def on_train_begin(self, logs=None):
    self.best = float('inf')

def on_epoch_end(self, epoch, logs=None):
    if epoch < self.start_epoch:
        return # Do not evaluate until start_epoch is reached

    current = logs.get(self.monitor)

    if current is not None:
        if self.mode == 'auto':
            if 'acc' in self.monitor or 'f1' in self.monitor:
                mode = 'max'
            else:
                mode = 'min'
        else:
            mode = self.mode

    # Update the loss history and check for improvement
    if current is not None:
        self.loss_history.append(current)
        if current < 5:
            # average_loss = np.mean(self.loss_history[-self.n_average:])

```

```

if current < self.best:
    self.best = current
    self.best_weights = self.model.get_weights() # Update best weights
    self.wait = 0
else:
    self.wait += 1

if self.wait >= self.patience:
    self.stopped_epoch = epoch
    self.model.set_weights(self.best_weights) # Set model weights to the
        best weights
    self.model.stop_training = True
    if self.verbose > 0:
        print(f'\nEpoch_{epoch+1}:_Early_stopping_due_to_no_improvement_in
            _{self.monitor}._Best_weights_restored_with_Loss_{self.best}.')

```

A.3 Calling the RBF-NN

The following code calls the RBF-NN and demonstrates how training and final weight calibration data are output from the network. This includes setting up the optimizer, ADAM, with the specific learning rate and weight decay factors. It also includes implementing the architecture of the custom Keras RBF layer.

```

adam = tf.keras.optimizers.legacy.Adam(learning_rate=lr, decay = dcy, amsgrad=True)
# Creating a sequential model with RBF layer
model = tf.keras.Sequential()
model.add(tf.keras.layers.InputLayer(input_shape=(3,)))
model.add(RadialBasisFunction(nodes, R, N, A0, B0, C0, D0, Z, beta, zeta, m, alpha,
    gamma, eta, ini_weights)) #will need to add other parameters

# Compiling the model with the specified optimizer and loss function
model.compile(optimizer=adam, loss=loss_fit, metrics=[total_charge, AtomA_sum,
    AtomB_sum, q, rmse])

# Defining an EarlyStopping callback
#early_stopping = EarlyStopping(monitor="rmse", patience = 100, mode='min',
    restore_best_weights=True, start_from_epoch=500)

```

```

early_stopping = EarlyStoppingWithBestWeights(monitor='loss', patience=
    stopping_patience, verbose=0, mode='min')

# Training the model

model_history = model.fit(coor_train, rho_train, batch_size = batch_s, epochs=
    epoch_num, verbose=0, callbacks=[early_stopping])#, validation_split=0.2,
    callbacks=[early_stopping])

training_loss = model_history.history['loss']
#print(model_history.history.keys())
q_per_epoch = model_history.history['q']
rmse_ep = model_history.history['rmse']

```

B Python Code for Evaluating the Analytic Forms of the Radial Basis Functions

The following Python functions are called by the Radial Basis Function Neural Network (RBF-NN) in order to implement the radial basis functions that describe the atomic state electron density distributions for general neutral and ionic atomic states (B.1) and hydrogen ground and excited states (B.2).

B.1 Analytical Fitted Models

The following code implements the analytical fitted models, Eq. (19), as described in [4, 5]. The analytic models are implemented as two separate functions, M5 and M6. M5 is used for neutral and cation charged states and M6 is used for anion states. They differ in their inclusion of the parameter ζ , which is non-zero for anions in order to capture their longer range decay in density due to the extra bound electron.

```

import tensorflow as tf
import matplotlib.pyplot as plt
from scipy.integrate import trapz

```

```

def M5(self, coor, R, i, j):
    r = tf.sqrt((coor[:,0])**2 + (coor[:,1])**2 + (coor[:,2] - R)**2)

    A_term = self.A0[i,j] * tf.exp(-2 * self.Z[i,j] * tf.abs(r))
    B_term = self.B0[i,j] * tf.abs(r)**(2*self.beta[i,j]) * tf.exp(-2 * self.alpha[i,j] * tf
        .abs(r))
    C_term = self.C0[i,j] * (self.m[i,j] - self.Z[i,j] * tf.abs(r))**2 * tf.exp(-2 * self.
        gamma[i,j] * tf.abs(r))
    D_term = self.D0[i,j] * (self.Z[i,j])**2 * (tf.abs(r))**2 * tf.exp(-2 * self.eta[i,j] *
        tf.abs(r))
    M5 = A_term + B_term + C_term + D_term
    return M5

def M6(self, coor, R, i, j):
    r = tf.sqrt((coor[:,0])**2 + (coor[:,1])**2 + (coor[:,2] - R)**2)

    beta_eff = self.beta[i,j] + self.zeta[i,j] * tf.abs(r)

    A_term = self.A0[i,j] * tf.exp(-2 * self.Z[i,j] * tf.abs(r))
    B_term = self.B0[i,j] * tf.abs(r)**(2 * beta_eff) * tf.exp(-2 * self.alpha[i,j] * tf.abs
        (r))
    C_term = self.C0[i,j] * (self.m[i,j] - self.Z[i,j] * tf.abs(r))**2 * tf.exp(-2 * self.
        gamma[i,j] * tf.abs(r))
    D_term = self.D0[i,j] * (self.Z[i,j])**2 * (tf.abs(r))**2 * tf.exp(-2 * self.eta[i,j] *
        tf.abs(r))
    M6 = A_term + B_term + C_term + D_term
    return M6

```

B.2 Exact Hydrogen Radial Densities

The following code implements the exact radial distributions for the ground and excited states of Hydrogen, as described in Section 2.4. The function inputs coordinate location information and the quantum numbers (n, l) and outputs the radial electron distribution function for the specified state. This code is a Python translation of a Matlab code from [4].

```

def hydrogen_density(self, coor, R, n, l):
    r = tf.sqrt((coor[:,0])**2 + (coor[:,1])**2 + (coor[:,2] - R)**2)

    if n < 1:
        raise ValueError('Error: invalid value of n; Exiting...')
    elif n > 4:
        raise ValueError('Value of n exceeds supported values; Exiting...')

    Z = 1.0 # Nuclear charge (1 for H, but keep in code for generalizability to other
            # hydrogenic wavefunctions.
    rscal = 2 * Z / n # Scale factor for distance (see: http://en.citizendium.org/wiki/Hydrogen-like\_atom,
                    # referencing Pauling and Wilson)
    Rnorm = Z**1.5 # Normalization factor for Rnl

    # Scale the radius
    rmatscal = r * rscal # scaled r values for specified value of n (denoted by rho_n in
                        # most texts).
    expfac = tf.exp(-rmatscal * 0.5) # exponential prefactor common to all Rnl

    # compute radial wavefunction x angle-averaged angular wfn (Anglenorm) for quantum
    # numbers (nl) and store in fMat
    if n == 1 and l == 0: # R10
        fmat = Rnorm * expfac * 2.0
    elif n == 2 and l == 0: # R20
        fmat = Rnorm * expfac * (0.5 / tf.sqrt(2.0)) * (2.0 - rmatscal)
    elif n == 2 and l == 1: # R21
        fmat = Rnorm * expfac * (0.5 / tf.sqrt(6.0)) * rmatscal
    elif n == 3 and l == 0: # R30
        fmat = Rnorm * expfac * (1.0 / (9.0 * tf.sqrt(3.0))) * (6.0 - 6.0 * rmatscal +
            rmatscal**2)
    elif n == 3 and l == 1: # R31
        fmat = Rnorm * expfac * (1.0 / (9.0 * tf.sqrt(6.0))) * (4.0 - rmatscal) * rmatscal
    elif n == 3 and l == 2: # R32
        fmat = Rnorm * expfac * (1.0 / (9.0 * tf.sqrt(30.0))) * rmatscal**2
    elif n == 4 and l == 0: # R40
        rmatscalsq = rmatscal**2
        fmat = Rnorm * expfac * (1.0 / 96.0) * (24.0 - 36.0 * rmatscal + 12.0 * rmatscalsq -
            rmatscalsq * rmatscal)
    elif n == 4 and l == 1: # R41
        rmatscalsq = rmatscal**2

```

```

    fmat = Rnorm * expfac * (1.0 / (32.0 * tf.sqrt(15.0))) * (20.0 - 10.0 * rmatscal +
        rmatscalsq) * rmatscal
elif n == 4 and l == 2: # R42
    rmatscalsq = rmatscal**2
    fmat = Rnorm * expfac * (1.0 / (96.0 * tf.sqrt(5.0))) * (6.0 - rmatscal) *
        rmatscalsq
elif n == 4 and l == 3: # R43
    rmatscalcub = rmatscal**3
    fmat = Rnorm * expfac * (1.0 / (96.0 * tf.sqrt(35.0))) * rmatscalcub
else:
    raise ValueError(f'Unsupported quantum numbers n={n}, l={l}')

pi = 3.141592653589793

# rhoMat (total density) is the radial wfn squared, times 1/(4*pi) for the angle-
    averaged (theta, phi) wfn-sqd component.
rho = (fmat**2) / (4.0 * pi)

return rho

```