Neural network – based analysis of multimodal bond distributions using their EXAFS spectra

Nicholas Marcella[1*], Stephen Lam,[2] Vyacheslav S. Bryantsev,[3] Santanu Roy[3] and Anatoly I. Frenkel[4,5*]

[1] Department of Chemistry, University of Illinois, Urbana, Illinois 61801, United States

[2] Department of Chemical Engineering, University of Massachusetts Lowell, Lowell, Massachusetts 01854, United States

[3] Chemical Sciences Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830, United States

[4] Department of Materials Science and Chemical Engineering, Stony Brook University, Stony Brook, New York 11794, United States

[5] Chemistry Division, Brookhaven National Laboratory, Upton, New York 11973, United States

ABSTRACT.

Knowledge of the local coordination environment around atomic species in functional materials is critical for understanding their mechanisms of operation. Heterogeneous mixtures of metal complexes are ubiquitous in catalysts, ionic liquids, molten salts, biological enzymes, and geochemical systems, among many others. Extracting information from ensemble-average measurements about the structural and compositional descriptors of each type of coordination complex comprising the mixture is not generally possible, especially when they possess multimodal bond length distributions. We developed a method that enables the mapping of an X-ray absorption spectrum on the radial distribution function describing the average environment of the metal ions. The supervised neural network-based method utilizes an objective training set, for which the choice of the local structural motifs is completely agnostic to the theoretically expected structure and dynamics of the modeled system. The method was validated using first-principles modeling of structural dynamics of nickel complexation in molten salts, and it applies to a large class of heterogeneous systems, including those studied under in-situ and operando conditions.

I. Introduction

Investigating the local environment around atomic or ionic species in functional materials helps understand mechanisms of their operation in diverse applications such as nanocatalysts [1], quantum dots [2,3], molten salt systems for nuclear reactors [4], chemical warfare filtration [5], and micro/nano-actuators [6]. Characterizing atomic distributions in these and other multi-component and strongly disordered systems poses a major challenge due to the need for elemental specificity, which allows for distinguishing between different neighboring atoms in the same

coordination sphere of a given atomic species, and spatiotemporal resolution, which must be sufficient for quantitative analysis of atomic displacements from ensemble-average positions due to static and/or dynamic disorder. Extended X-ray absorption fine structure (EXAFS) is an excellent tool for such purposes due to its excellent elemental specificity and spatiotemporal resolution.

In EXAFS theory, the absorption coefficient of X-rays is given by real-space multiple scattering (RSMS) theory, focusing on the interference between the outgoing and backscattered (by nearest neighbors to the absorber) photoelectron waves [7]. The spectrum for an absorber-neighbor pair is given by:

$$\chi(k) = S_0^2 \int_a^b A(k,r) \frac{1}{r^2} \sin[2kr + \delta(k,r)] g(R) dR, \qquad (1)$$

where S_0^2 is the passive electron reduction factor, $A(k,r)$ is the photoelectron scattering function that also includes inelastic losses, $\delta(k,r)$ is the photoelectron phase shift, and $g(r)$ is the unknown pair distribution function of nearest neighbors to X-ray absorbing atoms, which is normalized by the condition: $\int_a^b g(r)dr = N$, the number of neighbors to the absorber in the distance range $a < r < b$, corresponding to the given shell. Equation 1 can be modified to include multiple scattering contributions by replacing $g(R)$ with corresponding many-atom distribution functions [8]. Provided a 3D structure, the photoelectron scattering parameters are readily calculated via the Rehr-Albers approach under the muffin-tin approximation of the atomic potentials [9] with the FEFF software package [10]. For well-defined structures, such as those of most bulk materials, EXAFS calculations agree well with experimental spectra [11]. However, in most functional

nanomaterials with unknown structure, various methods are used to approximate the needed structural information, in particular the pair distribution function, $g(r)$.

A typical EXAFS research project often approximates the $g(r)$ as quasi-Gaussian, using a finite (usually, first 3-4 terms) number of terms in the cumulant expansion [12] through non-linear least square fitting. This approach notably breaks down in materials with anomalously large, asymmetric bonding disorder [13,14]. "Inverse" methods, such as the "regularization" [15-24] and "splice" methods [25,26] are very useful techniques for reconstructing a model-independent $g(r)$ in different types of ordered and disordered materials. In the "forward" modeling, $g(r)$is calculated via Molecular Dynamics (MD) simulations and validated by matching the EXAFS calculated on MD trajectories (MD-EXAFS) to experimental data [27-30]. Alternative methods like Reverse Monte Carlo (RMC) simulations [31,32] rely on assumed initial structures, which are often not known in low-dimensional and/or strongly disordered materials.

Another "inverse" modeling approach in which the unknown $g(r)$ is obtained from the experimental EXAFS data using neural networks (NN-EXAFS) [13]. In the original work developing this method for studying disordered states of Fe during its high-temperature transition from the *bcc* to the *fcc* phase, and in subsequent applications of this method to the studies of mono- and bimetallic nanoparticles [33,34], Timoshenko, Frenkel, et al., trained the feedforward neural network (NN) to recognize the relationship between the input (EXAFS spectrum) and the output: $g(r)$. The EXAFS signal, including multiple scattering contributions, is mapped onto one the pair distribution functions in this approach. The general concept of the "inversion" of Eq. (1) and mapping EXAFS data on the $g(r)$, is shown in **Fig 1**. In the existing approach for training the NN (**Fig. 1.ii**), classical MD is used as a practical way to create an array of geometries from which

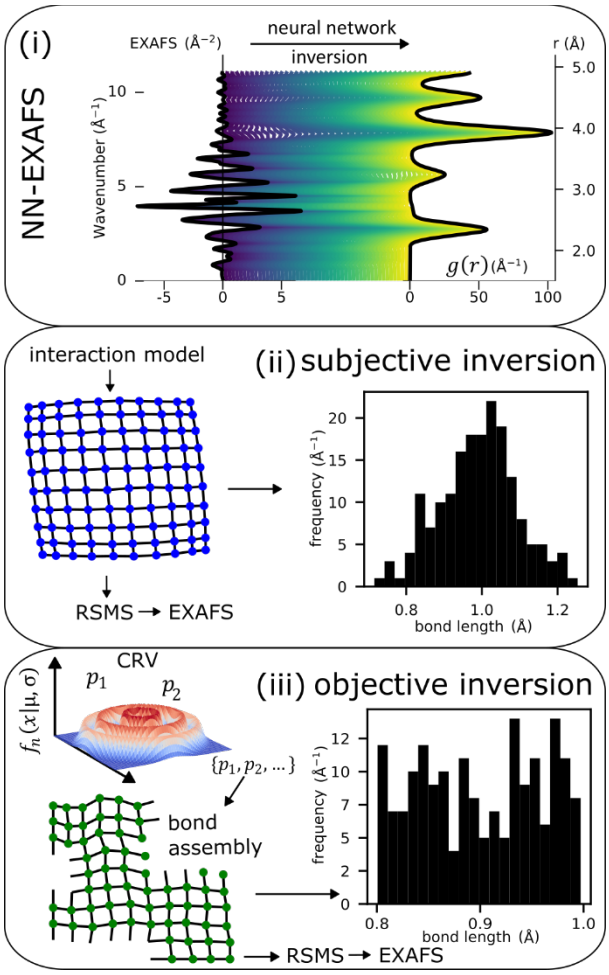

Figure 1. Schematic of the "inversion" of an EXAFS spectrum using the subjective (based on MD simulations) and objective (CRV) methods for structure generation. (i) The neural network function maps $\chi(k)$ (left) to $g(r)$ (right), as shown using an example of bulk Ni. (ii) Existing training data construction scheme relies on an interaction model embedded in molecular dynamics simulations. Real space multiple scattering (RSMS) is used to calculate the EXAFS from the structure model. (iii) The new, objective, approach uses continuous random variables (CRVs) for generating pair distances via the probability density function f(x|μ,σ). The macrostate is the user-defined set of CRVs for the overall pair distance distribution. The microstate, or list of pair distances p, is sampled from the macrostate using random variate sampling (RVS). Ultimately, the microstate, or pair distances, will be assembled into a 3D structure model for RSMS calculations.

EXAFS spectra are calculated per time step and labeled with the corresponding histogram for the $g(r)$. This approach to structure generation is "subjective" in the sense that the structure is determined by an assumed or ab initio interaction potential. MD provides instantaneous 3D structures and $g(r)$, while FEFF calculates EXAFS on those structures. Finally, the NN inverts the input (experimental) spectrum onto $g(r)$, determined by the bond length histogram. Provided the MD simulations adequately represent the system of interest, the FEFF-calculated scattering parameters are accurate, and the background subtraction is handled appropriately, the trained NN can be used to invert the experimental data onto $g(r)$.

For many types of material systems, however, the use of MD for generating training sets is impractical. While effective for bulk and nanoscale metals and alloys [35-37], classical MD faces limitations in reactive environments where bonds frequently form or break, such as, e.g., in metal catalysts, transition metals, and actinides in ionic liquids or molten salts [4, 38-41]. In these cases, *ab initio* Molecular Dynamics (AIMD) can model EXAFS spectra in disordered environments, making it useful for constructing NN-EXAFS training sets [4,29,42,43]. However, AIMD is computationally expensive and subject to sampling biases, limiting the generality of NN-EXAFS. The sensitivity of EXAFS to factors like temperature and composition necessitates re-running AIMD simulations to match experimental conditions. Although machine-learned force fields could reduce costs [44-48], they may still suffer from biases in configurational sampling due to the underlying DFT functionals, which have their own limitations for predicting structural and thermodynamic properties [39].

Ultimately, any of these techniques can be modified on an *ad hoc* basis to provide structures for NN-EXAFS training, such as scaling trajectories or varying thermodynamic conditions. However,

these modifications increase complexity and limit accessibility for typical EXAFS users. In this work, we acknowledge the fact that the NN doesn't need to learn the EXAFS- $g(r)$ relationship solely from MD. Since EXAFS depends on the atomic coordinates, any method generating these coordinates realistically can create a training set. This motivates exploring alternatives for training NN-EXAFS models, aiming for high configurational sampling and low computational costs.

In this Article, we present a new method for the "objective" training the NNs used to "invert" the EXAFS spectra onto $g(r)$, resulting in objectively trained NN-EXAFS (ONNE). Rather than employing MD to generate training data, we use an algorithm to create local structures based on a combination of the simple Gaussian functions describing pair distributions. We call this "objective" structure generation (**Fig. 1.iii**). Our method can generate training structures from any combination of bond length distributions, including partial bond length distributions from coordination shells beyond the first nearest neighbors, and thus be used to interpolate the configurational space to an arbitrary degree. Moreover, we can incorporate multimodal training distributions via objective training. Moreover, we can incorporate multimodal bond distributions via objective training, where “multimodal” refers to the different types of bond distributions (unimodal, bimodal etc.) within a given (e.g. cation-anion) coordination shell. Using this approach, and similar to other “inversion” methods described above, such as regularization method, we obtain the overall distribution “objectively” and forgo the need to learn (or assume) what are the individual species that contribute to the heterogeneous mixture, which is a limitation of forward-modeling methods. We verify the key idea using a toy example of an oscillating dimer and a more complex case of bulk Ni. Finally, we demonstrate this method's utility and limitations by applying it to reconstructing the $g(r)$ from the a priori known (calculated with AIMD) EXAFS spectrum in a heterogeneous environment of Ni^{2+} ions in molten salt.

II. Methods

We present an algorithm to create diverse 3D structures with user-controlled tolerances for coordination numbers, compositions, and radial distances. Essentially, this method inverts a given pair distribution into a 3D structure. Although the problem is ill-posed, leading to multiple structures with equivalent radial distances but random angles, this randomness benefits our aim to generate unique scattering paths over a highly interpolated configurational space.

Bond length distributions are defined by absorber-neighbor pair lengths per coordination shell and per scatterer type if multiple elements are present, utilizing continuous random variables (CRVs) (**Fig. 1.iii**). A CRV for each shell, denoted $(v_{n,e})$, is defined where (n) represents the neighbor shell index and (e) the scattering element. The CRVs, collectively referred to as the "macrostate", each follow a probability density function given by $f_n(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, and are organized into sets $V_e = \{v_{1,e}, v_{2,e}, \dots\}$, where the set V_e is the "macrostate", with adjustable mean (μ_n) and standard deviation (σ_n). To generate the structures, N_n pair-lengths, corresponding to the number of neighbors per shell *n*, are sampled from each CRV of the macrostate to create one list of absorber-neighbor pairs referred to as the "microstate". Finally, the pairs from the list are assembled into a 3D structural configuration for scattering calculations with FEFF.

Assembling the microstates into configurations presents a challenge due to the many configurations possible for one microstate. Given that the microstate is the list of radial distances from the center (absorber) to neighboring atoms, we define a "reasonable" structure as one in which the atoms are placed on spherical shells without getting too close to each other. We utilize a genetic algorithm (GA) to create these 3D structures, employing a customizable fitness function that

considers constraints on bond lengths and coordination numbers, and optimizes bond angles. The fitness of 3D configurations is determined by maximizing the minimum pair distance between all atoms. Because we hold the distance to the central atom constant and only modify the angles this has the effect of keeping atoms separated enough so that they do not overlap in space. The details are in the **Supplemental Methods** [51].The primary assumptions underpinning the ONNE philosophy are that 1) a NN trained on combinations of microstates generated from Gaussian-distributed macrostates can generalize to more complex macrostates, i.e., a network trained on Gaussian distributions can generalize to multimodal distributions, and 2) that a NN trained on combinations of multimodally distributed microstates can generalize to EXAFS calculated on "physically distributed" structures, i.e., the ONNE-trained NN can make predictions on EXAFS that is calculated on structures that are distributed according to either experimental crystal structure refinement or a potential model. We test these assumptions using two cases, first - the harmonically oscillating dimer (see **Appendix**) and, second, bulk Ni foil. While the dimer case illustrates the ability to reconstruct a multimodal distribution from the EXAFS-like spectra, the bulk Ni example confirms that NN trained on combinations of multimodally-distributed microstates can generalize to EXAFS calculated on realistically distributed structures, i.e., those consistent with either reported crystal structures or a plausible models.

III. Results and Discussion

ONNE with Ni foil. --- We utilize the RMC-EXAFS fitting of Ni foil at 298 K from Ref. [13] (inset (iii) **Fig. 2a**) to provide a realistically distributed example. This structure was derived from the iterative perturbation of a periodic FCC Ni crystal structure until the calculated, ensemble-average EXAFS matched the experimental EXAFS. From this structure we calculate EXAFS and $g_{rmc}(r)$.

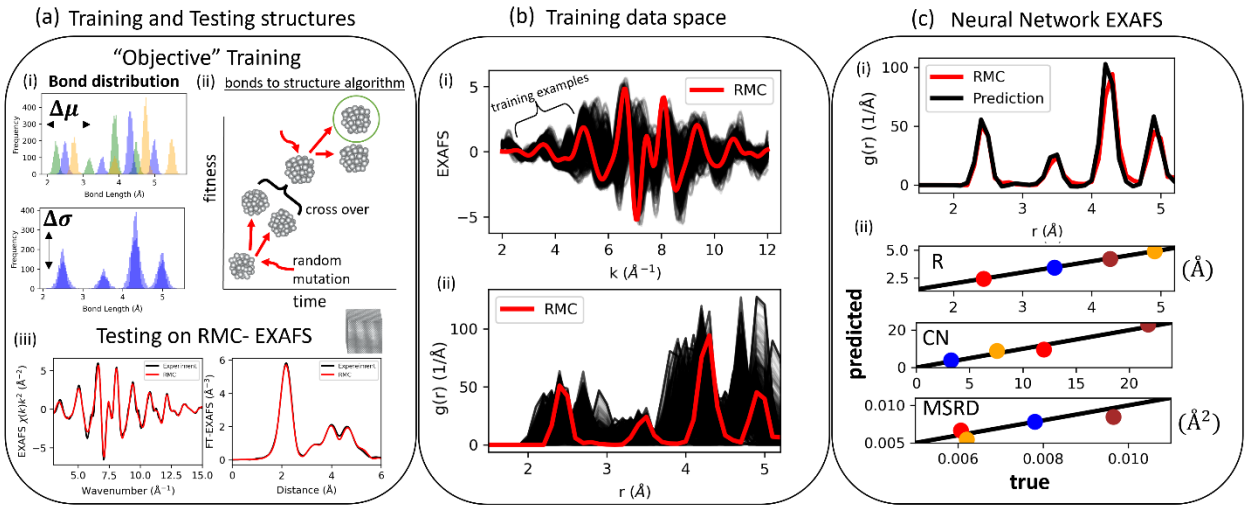

Figure 2. Deploying the objective training framework for the inversion of the Ni K-edge EXAFS of bulk Ni. (a) Bulk training and testing structures. (i) 46 macrostates with variations in distribution means (μ) and standard deviations (σ) are created. (ii) 100 microstates are created and assembled into 3D configurations with a genetic algorithm. (iii) existing RMC-EXAFS is used to test the objectively trained neural network. (b) Examples of the training EXAFS (top) and $g(r)$ (bottom) compared to the testing data (red). (c) (i) The results of prediction (black) vs. true (red) $g(r)$ and (ii) agreement between the structural parameters extracted from the predicted and true $g(r)$ for the first (red), second (blue), third (brown), and fourth (yellow) neighbor shells.

The training data is created by considering a set of four CRVs, $V_{Ni} = \{v_{1,Ni}, v_{2,Ni}, v_{3,Ni}, v_{4,Ni}\}$, where μ is varied within 10% around the underlying lattice's pair length and σ is varied to simulate the temperature-dependent Debye-Waller factor (shown in inset (i) of **Fig 2a**). Microstates are formed using RVS to sample pairs per CRV corresponding to FCC-packed bulk Ni. Finally, for each microstate, one iteration of the GA results in one configuration with an absorber in the middle of four shells of 12, 6, 24, and 12 atoms (inset (ii) of **Fig 2a**). Running the GA multiple times will result in multiple structures with equivalent radial distances and unique angular placements. Full details of data set construction, NN training, and code repository are provided in the **Supplemental Information, Notes, Methods** [51]).

A precursory analysis of the RMC-EXAFS and $g_{RMC}(r)$ ensures that both fall within the training data distribution (**Fig. 2b**). The results are shown in **Fig. 2c**, where the predicted and true $g_{RMC}(r)$ are compared. The agreement is excellent in both the visual examination of $g_{RMC}(r)$ (**Fig. 2b** inset (i)) and the quantitative comparison of interatomic distances (R), coordination numbers (CN), and mean squared relative displacements (MSRD) per neighbor shell. Therefore, we can conclude that the objectively-trained NN can invert "physically"-distributed data despite being trained on nonphysical configurations. The success here is likely due to the appropriateness of the muffin tin approximation for bulk metals and our ability to highly interpolate this relatively constrained configuration space, i.e., the probability is high that we randomly generated scattering paths that exist in the physically distributed model.

ONNE with Ni^{2+} ions in a molten mixture. --- We examine the system we denote as NiClZnK, consisting of dilute (1 wt.%) $NiCl_2$ in molten KCl-$ZnCl_2$ salt mixtures, with two compositions, Comp 1 (45.3 mol% KCl) and Comp 2 (52.5 mol% KCl), previously studied in Ref. [39]. Roy et

al. used EXAFS and AIMD simulations to capture Ni^{2+} local environments for both compositions. They found that, after segmenting the AIMD trajectories into representative Ni complexes by Ni-Cl coordination number and then through linear combination fitting of the EXAFS of those complexes to the experimental spectrum, a match was achieved, resulting in the trust in this physical model and FEFF's ability to simulate EXAFS in this system. Thus, for our testing, their AIMD-EXAFS data will be used, as we know both the spectra and the corresponding $g(r)$.

We define the macrostates using three sets of continuous variables: $V_{Cl} = \{v_{1,Cl}, v_{3,Cl}\}$, $V_{Zn} = \{v_{2,Zn}\}$, $V_K = \{v_{3,K}\}$, corresponding to their respective shells (Cl, Zn and Cl/K). The shell composition is varied by changing the coordination of the first and second shells during RVS (from 1 to 6 for Cl^- and 0 to 2 for Zn^{2+}). The GA creates configurations (**Fig. 3a**), and the NN is trained using weighted linear combinations with an on-the-fly synthetic data generator. The convolutional NN, full details of data set construction, NN training, and code repository are provided in the **Supplemental Information, Notes, Methods** [51]).

The ONNE results for both compositions are shown in the left column of **Fig. 3b**, where the partial distributions $g_{cl}(r)$ and $g_{zn}(r)$ from AIMD-EXAFS are compared to the true distributions from the AIMD trajectories. We trained 10 separate NNs for uncertainty assessment, displaying average predictions (solid lines) and standard deviations (green shadow). Qualitative agreement is observed, such as the bimodality and skewness of $g_{zn}(r)$ in Comp 1 and Comp 2, but it only permits coarse conclusions about the material's local structure. Average coordination number comparison reveals 26% and 15% errors for Comp 1, and 18% and 40% for Comp 2, comparable to conventional EXAFS analysis errors (up to 20%). Moreover, consistent systematic changes in CN are noted between Comp 1 (predicted: $CN_{Ni\text{-}Cl} = 3.18 \pm 0.08$ and $CN_{Ni\text{-}Zn} = 2.13 \pm 0.07$, vs

true : $CN_{Ni\text{-}Cl} = 4.01$ and $CN_{Ni\text{-}Zn} = 2.51$) and Comp 2 (predicted: $CN_{Ni\text{-}Cl} = 3.11 \pm 0.07$ and $CN_{Ni\text{-}Zn} = 2.34 \pm 0.09$, vs. true: $CN_{Ni\text{-}Cl} = 3.68$ and $CN_{Ni\text{-}Zn} = 1.66$). Although gaining information about the second shell ($g_{zn}(r)$) is an advantage over the current state of the art; we aim to improve ONNE to enhance the resolution of information from the EXAFS in these systems. Upon further investigation (presented in **Supplemental Information, Fig. S2 [51]**), we determined that the source of the error stems from an incomplete sampling of the configuration space, which is very

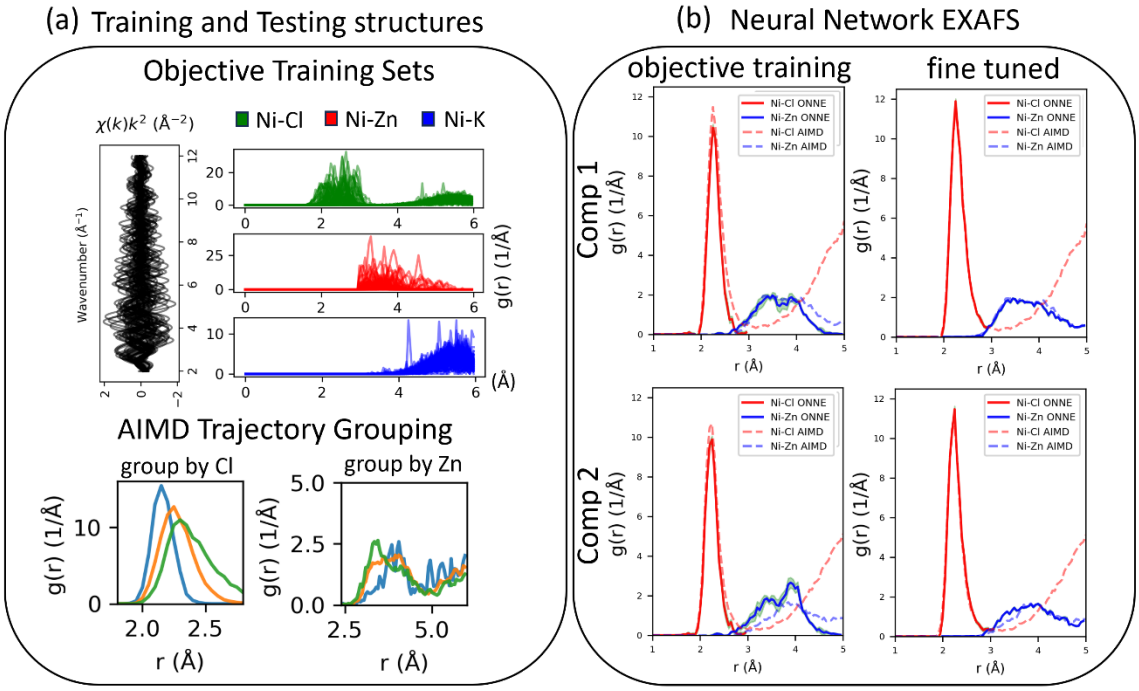

Figure 3. Objective training and fine-tuning NN-EXAFS for NiClZnK Ni K-edge EXAFS spectra inversion. (a) The objective training sets and AIMD test data. (b) Objectively trained Neural Network EXAFS (ONNE) predictions (left column) of the partial Ni-Cl and Ni-Zn $g(r)$ for AIMD simulated NiClZnK of Comp 1) (top row) and Comp 2 (bottom row) at 550 °C. ONNE is compared to fine tuning (right column), where the data from composition 1 was used to fine-tune.

large for a system of this complexity. Thus, in the future development of ONNE, we need to develop a more robust protocol for interpolating the configurational space.

Fine tuning ONNE. --- In the final part, we demonstrate how ONNE can be "fine-tuned" using the available AIMD simulations. Since ONNE is already trained and has learned the general relationship between EXAFS and partial $g_{Cl}(r)$ and $g_{Zn}(r)$, we employ a transfer learning [50] procedure called "fine-tuning" to enhance the model for immediate use in our research. Fine-tuning augments a pre-trained NN with new layers, freezing the previous ones (**Supplemental Information, Fig. S1** [51]). This method is effective if the pre-trained NN's data domain is similar to the new data, which is true here. It's important to note that the use of AIMD data in this context is optional, serving primarily to refine the model's precision for specific scenarios and is not a fundamental aspect of the training process. Care must be taken not to overfit the model to the new, smaller data set. Fine-tuning can be further refined by using enhanced sampling methods beyond AIMD, such as machine learning-based molecular dynamics or temperature-accelerated dynamics. Two fine-tuning data sets were created from AIMD for training and testing to ensure the NN didn't memorize the training data.

The AIMD datasets were constructed by grouping trajectories based on the Ni-Cl and Ni-Zn distribution centers, yielding 86 unique configuration averages. 26 examples are reserved for validation, with the remaining 60 for training. The 10 previously trained NNs are fine-tuned using the on-the-fly synthetic data generator approach, with full details in the **Supplemental Information** [51].

The fine-tuning results for Comp 1 and 2 (**Fig. 3b**, right side) outperform the original ONNE, and the generalization to Comp 2 is promising. Though a more robust ONNE training process may

achieve similar outcomes, this offers a significant improvement "hack" for this system and a novel approach to training NN-EXAFS models when MD or AIMD availability is limited.

Application to experimental data. --- ONNE and fine-tuned ONNE (ONNEF) were used to invert the experimental NiClZnK data from Ref. 39 and the results are presented in Fig. S3 in the **Supplemental Information** [51]. We have obtained the partial Ni-Cl (first shell) and Ni-Zn (second shell) radial distribution functions from the NiClZnK system, which is remarkable when considering that there is no second shell information seen in the Fourier transformed EXAFS (FT-EXAFS) data in S3. The qualitative differences between Comp1 and Comp2 seen in the first shell are in excellent agreement with the FT-EXAFS for both ONNE and ONNEF. The trends in the second shell cannot be distinguished using ONNE, as expected from the validation testing with the AIMD data, but they are distinct using ONNEF.

IV. Summary

We have demonstrated that the training of NN-EXAFS models does not require AIMD (or, indeed, any MD) simulations to generate the training set. Instead, the NN can learn the underlying relationship between photoelectron scattering and the bond length distribution over which it scatters from arbitrary examples generated by our algorithm. The caveat is that the configurational space over which the examples are generated must be fully sampled. This method of training the NN is "objective" because it is model-independent—i.e. agnostic to any interaction model for the purpose of the 3D structure generation for real space scattering calculations. In the simple case of bulk Ni, ONNE performs very well due to the limited configurational space. For the more complex NiClZnK molten salt system, ONNE's predictions are less accurate due to the insufficient sampling. Transfer learning can be employed to fine-tune ONNE, enabling structural analysis of

the next coordination shells via NN-EXAFS for the first time. Future efforts will focus on enhancing ONNE by developing a more robust method for sampling in systems with greater configurational complexity.

AUTHOR INFORMATION

Corresponding Author

*N.M. Email: nmarcella@bnl.gov *A.I.F.: Email: frenkel@bnl.gov

DATA AVAILABILITY STATEMENT

The digital data for all figures, tables, charts, and any other media contained in this publication and its associated supporting information files will be made accessible on the Zenodo repository, under Digital Object Identifier (DOI): 10.5281/zenodo.8279733.

ACKNOWLEDGMENT

This work was primarily supported as part of the Molten Salts in Extreme Environments Energy Frontier Research Center, funded by the U.S. Department of Energy Office of Science. BNL and ORNL are operated under DOE contracts DE-SC0012704 and DE-AC05-00OR22725, respectively. NM acknowledges support by the US Department of Energy, Office of Basic Energy Sciences Award No. DE-SC0022199. SL acknowledges support by the US Department of Energy, Nuclear Energy University Program (NEUP) Award DE-NE0009204. The FEFF simulations used resources of the Center for Functional Nanomaterials, which is a U.S. DOE Office of Science Facility, and the Scientific Data and

Computing Center, a component of the Computational Science Initiative, at BNL under Contract No. DE-SC0012704. We thank Prof. J. J. Rehr for useful discussions.

Appendix on the toy example of an oscillating dimer. --- We begin with the toy example investigating a pair of atoms in a harmonically oscillating dimer with the simplified form of Eq. (1) that we used for creating the "toy" spectrum:

$$\chi_t(k,r) = \int_a^b \frac{1}{r^2} \sin[2kr] g(r) dr \ . \quad (2)$$

Due to the omission of the amplitude factor and photoelectron scattering functions - see the more general Eq. (1) – this demonstration can be easily generalized to any other test $g(r)$ functions using a sample code **(see Supplemental Notes) [51]**. In the toy example, we can work with the macrostates directly since we remove the need for scattering calculations, and thus we use the following implementation (for the full details of data set construction, neural network training, and code repository, see **Supplemental Information [51]**). The macrostates are created by utilizing random variate sampling (RVS) of 1000 numbers (pair lengths) from $f_1(x|\mu,\sigma)$ for a range of different μ and σ. The macrostates are binned according to the *r*-space grid and normalized by Δr, resulting in the training distribution function $g_t(r) = \frac{N}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(r-r_0)^2}{2\sigma^2}\right]$, which is normalized by the condition: $\int_a^b g(r)dr = N$, where N is the number of neighbors to the absorber in the distance range $a < r < b$ (e.g., $N = 1$ for a dimer). Eq. (2) is used to calculate χ_t over a grid of uniform *k*-space from 0 to 12 Å^{-1}. A basic multilayer perceptron (MLP)-NN was trained on 10,000 macrostates. The results are given in **Fig. A1**, where the training (**Fig. A1a**), and validation/testing data (**Fig. A1b**) are compared in terms of true vs. predicted $g_t(r)$. In a test of generality, we find that the NN, which was trained to invert χ_t calculated on Gaussian $g(r)$, does an excellent job of

predicting $g_t(r)$ from χ_t calculated on asymmetric and multimodal distributions. As a final test, we demonstrated its applicability to an extreme case of a non-Gaussian $g_t(r)$, that of a harmonic oscillator, in which the $g_t(r)$ can be calculated analytically (see **Supplemental Methods [51]**). The excellent agreement confirms that an NN trained on combinations of Gaussian-distributed macrostates can generalize to more complex macrostates (**Fig. A1c**). This implies that NN-EXAFS learns a general relationship between the EXAFS signal and the pair distribution function, more general than the underlying shape of the training data.

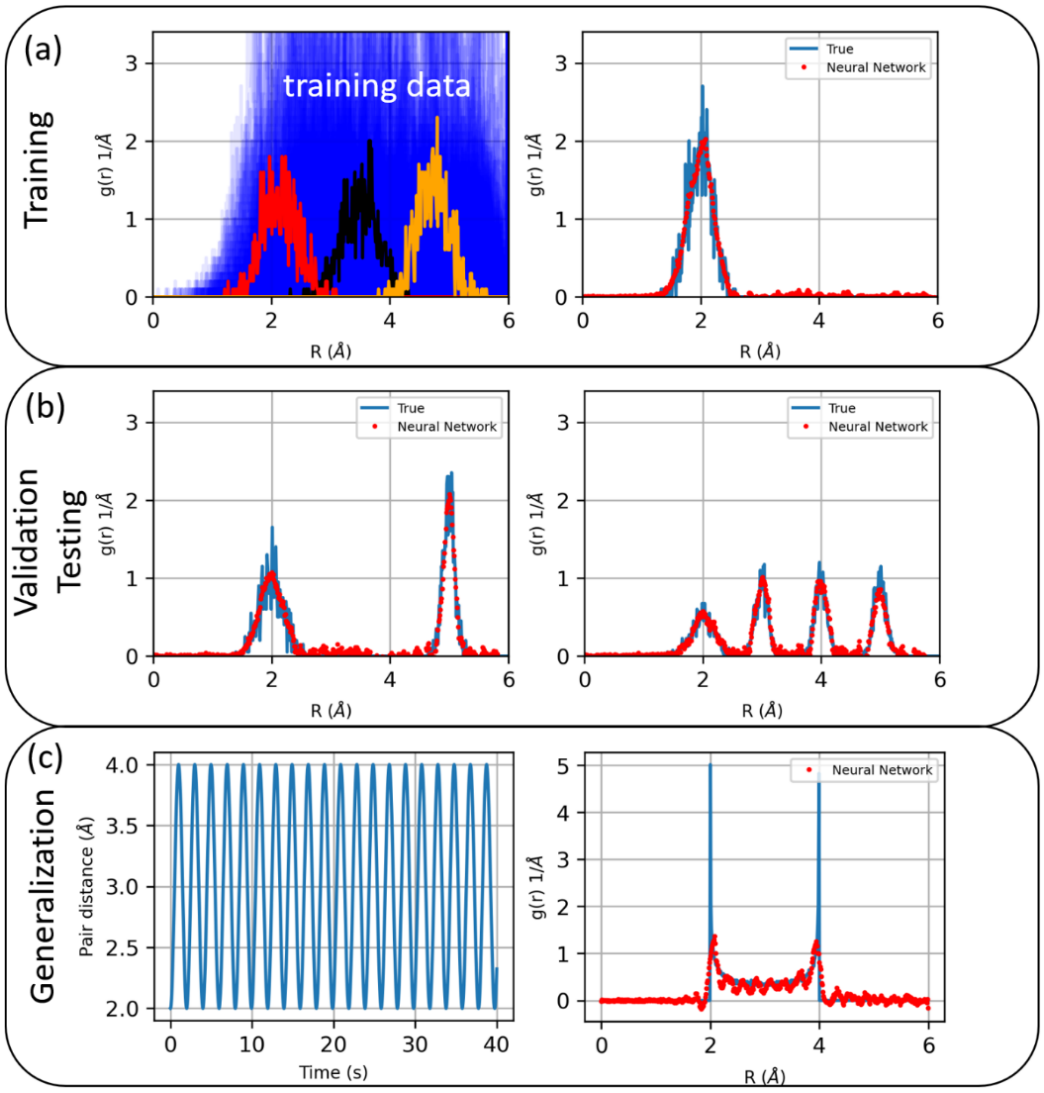

Figure A1. (a) On the left: examples of the training data distribution with three random examples highlighted (red, black, orange). On the right, one instance of the trained NN prediction vs. true $g_t(r)$. (b) Examples of the validation and testing data in which the trained NN predicts on $\chi_t(k)$ created from non-Gaussian $g_t(r)$. (c) A demonstration of the generalization power of the trained NN. (left) a dimer bond length distribution is simulated numerically as a harmonic oscillator. (right) The results of the trained NN prediction on $\chi_t(k)$ created from the harmonic oscillator.

References:

[1] Y. Li and A. I. Frenkel, Deciphering the local environment of single-atom catalysts with X-ray absorption spectroscopy, Acc. Chem. Res. **54**, 2660 (2021).

[2] L. Asor, J. Liu, Y. Ossia, D. C. Tripathi, N. Tessler, A. I. Frenkel, and U. Banin, InAs nanocrystals with robust p-type doping, Adv. Funct. Mater. **31**, 2007456 (2021).

[3] J. Liu, Y. Amit, Y. Li, A. M. Plonka, S. Ghose, L. Zhang, E. A. Stach, U. Banin, and A. I. Frenkel, Reversed nanoscale Kirkendall effect in Au–InAs hybrid nanoparticles, Chem. Mater. **28**, 8032 (2016).

[4] S. K. Gill, J. Huang, J. Mausz, R. Gakharm, S. Roy, F. Vila, M. Topsakal, W.C. Phillips, B. Layne, S. Mahurin *et al.*, Connections between the Speciation and Solubility of Ni(II) and Co(II) in Molten ZnCl2, J. Phys. Chem. B **124**, 1253 (2020).

[5] A. M. Plonka, Q. Wang, W. O. Gordon, A. Balboa, D. Troya, W. Guo, C. H. Sharp, S. D. Senanayake, J. R. Morris, C. L. Hill *et al.*, In Situ Probes of Capture and Decomposition of Chemical Warfare Agent Simulants by Zr-Based Metal Organic Frameworks, J. Am. Chem. Soc. **139**, 599 (2017).

[6] E. Makagon, E. Wachtel, L. Houben, S. R. Cohen, Y. Li, J. Li, A. I. Frenkel, and I. Lubomirsky, All-Solid-State Electro-Chemo-Mechanical Actuator Operating at Room Temperature, Adv. Funct. Mater. **31**, 2006712 (2021).

[7] J. J. Rehr and R. C. Albers, Theoretical approaches to x-ray absorption fine structure, Rev. Mod. Phys. **72**, 621 (2000).

[8] A. Filipponi, A. Di Cicco, and C. R. Natoli, X-ray-absorption spectroscopy and n-body distribution functions in condensed matter. I. Theory, Phys. Rev. B **52**, 15122 (1995).

[9] J. J. Rehr and R. C. Albers, Scattering-matrix formulation of curved-wave multiple-scattering theory: Application to x-ray-absorption fine structure, Phys. Rev. B **41**, 8139 (1990).

[10] J. Kas, F. Vila, C. Pemmaraju, T. Tan, and J. Rehr, J. Synchrotron Radiat. **28**, 1801 (2021).

[11] J. J. Rehr, R. C. Albers, and S. I. Zabinsky, Scattering-matrix formulation of curved-wave multiple-scattering theory: Application to x-ray-absorption fine structure, Phys. Rev. Lett. **69**, 3397 (1992).

[12] G. Bunker, Application of the ratio method of EXAFS analysis to disordered systems, Nucl. Instrum. Methods Phys. Res. **207**, 437 (1983).

[13] J. Timoshenko, A. Anspoks, A. Cintins, A. Kuzmin, J. Purans, and A. I. Frenkel, Neural network approach for characterizing structural transformations by X-ray absorption fine structure spectroscopy, Phys. Rev. Lett. **120**, 225502 (2018).

[14] A. Yevick and A. I. Frenkel, Effects of surface disorder on EXAFS modeling of metallic clusters, Phys. Rev. B **81**, 115451 (2010).

[15] N. V. Ershov, A. L. Ageev, V. V. Vasin, and Y. A. Babanov, A new interpretation of EXAFS spectra in real space: II. A comparison of the regularization technique with the Fourier transformation method, Phys. Status Solidi (b) **108**, 103 (1981).

[16] D.S. Yang and G. Bunker, Improved R-space resolution of EXAFS spectra using combined regularization methods and nonlinear least-squares fitting, Phys. Rev. B **54**, 3169-3172 (1996).

[17] G. Khelashvili and G. Bunker, Practical regularization methods for analysis of EXAFS spectra, J. Synchrotron Radiat. 6, 271-273 (1999).
[18] A. Kuzmin and J. Purans, Dehydration of the molybdenum trioxide hydrates MoO3· nH2O: in situ x-ray absorption spectroscopy study at the Mo K edge, J. Phys.: Condens. Matter 12, 1959-1970 (2000).
[19] A. Kuzmin and J. Purans, Local atomic and electronic structure of tungsten ions in AWO4 crystals of scheelite and wolframite types, Radiat. Meas. 33, 583-586 (2001).
[20] M. Kunicke, I. Yu. Kamensky, Yu. A. Babanov, and H. Funke, Phys. Scr. T115, 237-239 (2005).
[21] Yu. A. Babanov, Yu. A. Salamatov, I. Yu. Kamensky, A. V. Ryazhkin, and V. V. Ustinov, Efficient determination of optimal regularization parameter for inverse problem in EXAFS spectroscopy, J. Electron Spectrosc. Relat. Phenom. 175, 27-30 (2009).
[22] Yu. A. Babanov, D. A. Ponomarev, V. V. Ustinov, A. N. Baranov, and Ya. V. Zubavichus, Local atomic structure of solid solutions with overlapping shells by EXAFS: The regularization method, J. Electron Spectrosc. Relat. Phenom. 211, 1-11 (2016).
[23] I. Jonane, A. Cintins, A. Kalinko, R. Chernikov, and A. Kuzmin, Probing the Thermochromic Phase Transition in CuMoO4 by EXAFS Spectroscopy, Phys. Status Solidi B 255, 1800074 (2018).
[24] Yu. A. Babanov, V. V. Vasin, D. A. Ponomarev, D. I. Devyaterikov, L. N. Romashev, and V. V. Ustinov, Atomic Structure of Multilayered Low-Contrast Fe/Cr Thin Films: Mathematical Formalism and Numerical Experiments, Phys. Met. Metallogr. 120, 756-762 (2019).
[25] E. A. Stern, Y. Ma, O. Hanske-Petitpierre, and C. E. Bouldin, Radial distribution function in X-ray-absorption fine structure, Phys. Rev. B **46**, 687 (1992).
[26] A. Frenkel, E. A. Stern, A. Voronel, A. Rubshtein, Y. Ben-Ezra, and V. Fleurov, Redistribution of La-Al nearest-neighbor distances in the metallic glass Al0.91La0.09, Phys. Rev. B **54**, 884 (1996).
[27] A. Kuzmin and R. A. Evarestov, Quantum mechanics–molecular dynamics approach to the interpretation of X-ray absorption spectra, J. Phys. Condens. Matter. **21**, 055401 (2009).
[28] F. Vila, J. J. Rehr, J. Kas, R. G. Nuzzo, and A. I. Frenkel, Dynamic structure in supported Pt nanoclusters: Real-time density functional theory and X-ray spectroscopy simulations, Phys. Rev. B **78**, 121404 (2008).
[29] B. J. Palmer, D. M. Pfund, and J. L. Fulton, Direct modeling of EXAFS spectra from molecular dynamics simulations, J. Phys. Chem. **100**, 13393 (1996).
[30] S. W. T. Price, N. Zonias, C.-K. Skylaris, T. I. Hyde, B. Ravel, and A. E. Russell, Fitting EXAFS data using molecular dynamics outputs and a histogram approach,Phys. Rev. B **85**, 075439 (2012).
[31] J. Timoshenko and A. I. Frenkel, Probing structural relaxation in nanosized catalysts by combining EXAFS and reverse Monte Carlo methods, Catal. Today **280**, 274 (2017).
[32] R. L. McGreevy and L. Pusztai, Reverse Monte Carlo Simulation: A New Technique for the Determination of Disordered Structures, Mol. Simul. **1**, 359 (1988).
[33] J. Timoshenko, C. J. Wrasman, M. Luneau, T. Shirman, M. Cargnello, S. R. Bare, J. Aizenberg, C. M. Friend, and A. I. Frenkel, Probing atomic distributions in mono-and bimetallic nanoparticles by supervised machine learning, Nano Lett. **19**, 520 (2019).

[34] M. Rüscher, A. Herzog, J. Timoshenko, H. S. Jeon, W. Frandsen, S. Kühl, and B. Roldan Cuenya, Tracking heterogeneous structural motifs and the redox behaviour of copper–zinc nanocatalysts for the electrocatalytic CO2 reduction using operando time resolved spectroscopy and machine learning, Catal. Sci. Technol. **12**, 3028 (2022).
[35] P. D'Angelo, A. Di Nola, M. Mangoni, and N. V. Pavel, An extended x-ray absorption fine structure study by employing molecular dynamics simulations: Bromide ion in methanolic solution, J. Chem. Phys. **104**, 1779 (1996).
[36] A. Anspoks and A. Kuzmin, Interpretation of the Ni K-edge EXAFS in nanocrystalline nickel oxide using molecular dynamics simulations, Journal of Non-Cryts. Solids **357**, 2604 (2011).
[37] J. Timoshenko, Z. Duan, G. Henkelman, R. M. Crooks, and A. I. Frenkel, Solving the structure and dynamics of metal nanoparticles by combining X-ray absorption fine structure spectroscopy and atomistic structure simulations, Annual Rev. Anal. Chem. **12**, 501-522 (2019).
[38] A. Yevick and A. I. Frenkel, Effects of surface disorder on EXAFS modeling of metallic clusters, Phys. Rev. B **81**, 115451 (2010).
[39] S. Roy, Y. Liu, M. Topsakal, E. Dias, R. Gakhar, W.C. Phillips, J. F. Wishart, D. Leshchev, P. Halstenberg, S. Dai *et al.*, A holistic approach for elucidating local structure, dynamics, and speciation in molten salts with high structural disorder, J. Am. Chem. Soc. **143**, 15298 (2021).
[40] E. T. Dias, S. K. Gill, Y. Liu, P. Halstenberg, S. Dai, J. Huang, J. Mausz, R. Gakhar, W. C. Phillips, S. Mahurin *et al.*, Radiation-Assisted Formation of Metal Nanoparticles in Molten Salts, J. Phys. Chem. Lett. **12**, 157 (2021).
[41] X. Liu, A. Ronne, L. Yu, Y. Liu, M. Ge, C. Lin, B. Layne, P. Halstenberg, D. S. Maltsev, A. S. Ivanov *et al.*, Formation of three-dimensional bicontinuous structures via molten salt dealloying studied in real-time by in situ synchrotron X-ray nano-tomography,Nat. Commun. **12**, 3441 (2021).
[42] P. Liu, V. Johansson, H. Trilaksana, J. Rosdahl, G. G. Andersson, and L. Kloo, ACS Appl. Mater. Interfaces **9**, 19773 (2017).
[43] R. Atta-Fynn, E. J. Bylaska, G. K. Schenter, and W. A. de Jong, EXAFS, ab Initio Molecular Dynamics, and NICIS Spectroscopy Studies on an Organic Dye Model at the Dye-Sensitized Solar Cell Photoelectrode Interface, J. Phys. Chem. A **115**, 4665 (2011).
[44] D. P. Kovács, C. v. d. Oord, J. Kucera, A. E. Allen, D. J. Cole, C. Ortner, and G. Csányi, J. Chem. Theory Comput. **17**, 7696 (2021).
[45] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, Linear atomic cluster expansion force fields for organic molecules: beyond rmse, Nat. Commun. **13**, 2453 (2022).
[46] A. Musaelian, S. Batzner, A. Johansson, L. Sun, C. J. Owen, M. Kornbluth, and B. Kozinsky, Learning local equivariant representations for large-scale atomistic dynamics, Nat. Commun. **14**, 579 (2023).
[47] C. J. Owen, S. B. Torrisi, Y. Xie, S. Batzner, J. Coulter, A. Musaelian, L. Sun, and B. Kozinsky, Complexity of Many-Body Interactions in Transition Metals via Machine-Learned Force Fields from the TM23 Data Set, arXiv preprint arXiv:2302.12993 (2023).
[48] J. Vandermause, Y. Xie, J. S. Lim, C. J. Owen, and B. Kozinsky, Active learning of reactive Bayesian force fields applied to heterogeneous catalysis dynamics of H/Pt, Nat. Commun. **13**, 5183 (2022).

[49] C. J. Owen, N. Marcella, Y. Xie, J. Vandermause, A. I. Frenkel, R. G. Nuzzo, and B. Kozinsky, Unraveling the Catalytic Effect of Hydrogen Adsorption on Pt Nanoparticle Shape-Change, arXiv preprint arXiv:2306.00901 (2023).

[50] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, A survey on deep transfer learning, in *Artificial Neural Networks and Machine Learning – ICANN 2018*, edited by V. Kůrková *et al.* (Springer International Publishing, Cham, 2018), pp. 270.

[51] See Supplemental Material at _____ for the link to our code repository, technical details of GA and NN implementation, and figure with further analysis of the training data with respect to Ni-Zn radius, which includes Refs. [52-54]

[52] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, DEAP: Evolutionary algorithms made easy, J. Mach. Learn. Res. 13, 2171 (2012).

[53] R. Swinbank and R. James Purser, Fibonacci grids: A novel approach to global modelling, Q. J. R. Meteorol. Soc. 132, 1769 (2006)

[54] P. Gavrikov, visualkeras, GitHub repository (2020).

Neural network – based analysis of multimodal bond distributions using their EXAFS spectra

Nicholas Marcella[1*] , Stephen Lam,[2] Vyacheslav S. Bryantsev,[3] Santanu Roy[3] and Anatoly I. Frenkel[4,5*]

[1] Department of Chemistry, University of Illinois, Urbana, Illinois 61801, United States

[2] Department of Chemical Engineering, University of Massachusetts Lowell, Lowell, Massachusetts 01854, United States

[3] Chemical Sciences Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830, United States

[4] Department of Materials Science and Chemical Engineering, Stony Brook University, Stony Brook, New York 11794, United States

[5] Chemistry Division, Brookhaven National Laboratory, Upton, New York 11973, United States

This document contains:

Supplemental Notes

Supplemental Methods

Supplemental Figures

Supplemental References

Supplemental Notes

Note 1. Code Repository

We have created a package of Python scripts for performing all calculations described in the article. The full code can be accessed at https://github.com/nmarcella/Xron/tree/main/Objective_training.

Functions referenced in this document are in bold font.

Supplemental Methods

1. Genetic algorithms

1.1 A genetic algorithm (GA) is used to create 3D structures with the Python-based DEAP [52] evolutionary computation package. The GA is a population-based, metaheuristic optimization algorithm based on the principles of Darwinian evolution. Solutions are scored according to the fitness function. The fitness score dictates the construction of new generations using crossover and mutation operations. This process repeats iteratively until the optimal solution is found.

In our case, we use the GA to optimize the θ and ϕ values of atoms located at pair distances (ρ) from the central absorbing atom. To implement this, we start with a list of pairs of $V_\rho = \{p_1, p_2, \dots\}_N$ of length N. A vector of length 2N is created that will contain N θ values followed by N ϕ values $V_{\theta\phi} = \{\theta_1, \dots, \theta_N, \phi_1, \dots, \phi_N\}_{2N}$. This vector is the "chromosome" that will be iterated upon, where the angles are the "genes". The initial values are set with the Fibonacci sphere algorithm, which is inspired by "Fibonacci grids" [53].

During each iteration, each gene has a 20% probability of being mutated, or perturbed by a random number between -0.02 and 0.02 radians. The chromosome mutation probability is 50% and the crossover probability is 50%. To find the best fit structures, the fitness function is designed to find and return the minimum pair distance.

2.2 The GA is implemented using a Python script that is available at the link in Note 1. The goal of this algorithm is to find a perturbation vector V_r, that when added to $V_{\theta\phi}$, results in an increase in fitness. After finding the best V_r, $V_{\theta\phi}$ is updated and the loop is restarted. The entire process is built into the **optimize_structure** function.

We quantify the fitness of the configuration by the minimum pair distance between all atoms. By maximizing the minimum distance, while maintaining the radial distances from the center, we ensure the pair distribution is retained while the atoms do not overlap. We accomplish this by employing the helper function called **reassemble_points** to assemble the radial and angle vectors into a matrix of spherical coordinates $V_{\rho\theta\phi_{Nx3}}$. Next, we find the distance matrix D, determined by finding the Euclidian distance $d_{i,j}$ between n points in spherical coordinates:

$$d_{i,j} = \sqrt{r_i^2 + r_j^2 - 2r_i r_j \cos(\theta_i - \theta_j) - 2_{r_i r_j} \sin(\theta_i) \sin(\theta_j)(\cos(\phi_i - \phi_j) - 1)}$$

$$D = \begin{pmatrix} 0 & \cdots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & 0 \end{pmatrix}$$

The function **find_min(*D*)**, which is also the fitness function, finds the minimum non-zero value in the upper triangular part of the distance matrix D, i.e., min $\{D_{i,j}: i < j, D_{i,j} > 0\}$.

2. Creating RDF

2.1 In general, we approximate $g(r)$ (RDF) numerically as a histogram of absorber-neighbor pair distances over an r-space grid (**rmesh**). The RDF per absorbing atom per time step is defined below. Given an atomic configuration with N_{abs} absorbing atoms, the RDF $g_i(r)$ for each absorbing atom i is calculated as follows:

$$g_i(r) = \frac{dN_i}{dr} = \frac{N_{i,bin}}{\Delta r}$$

Here, $N_{i,bin}$ is the number of nearest neighbors within a bin of radius r and thickness Δr surround ding the i^{th} absorbing atom.

To compute the average RDF for the entire atomic configuration, we use:

$$g(r) = \frac{1}{N_{abs}} \sum_{i=1}^{N_{abs}} g_i(r)$$

In this equation, we sum the $g_i(r)$ values per absorbing atom and divide by N_{abs}.

2.2 The process of calculating the RDF is simplified when the dedicated computer program is available. We have developed a set of Python notebooks for performing these calculations. They are described as follows.

The implementation is different for the toy example and "real" examples, i.e., calculated from real space pair distances. The function **make_rdf_feff** is used to calculate the RDF on real space pair distances (from feff.inp files). It takes a single input argument, **distances**, a **NumPy** array of distance values between particles. The **rmesh** is defined with **np.arange**, where the argument is (0, 6, 0.01) for the toy example, (0, 6, 0.1) for the bulk Ni, and (0, 6, 0.05) for the NiClZnK example. It uses **np.digitize** to assign each distance to a bin in **rmesh**. Then, it applies **np.unique** to these bin assignments to find the unique bin indices and how many times each index occurs (i.e., how many distances fall within each bin). It iterates over the unique bin indices after initializing a counter list of zeros with the same length as **rmesh**. For each unique index, it assigns the corresponding count normalized by the width of the bin (**rmesh** [41]-**rmesh** [0]) to the counter list (i.e., it normalizes by Δr). Finally, it returns the counter list as a **NumPy** array. This array represents the radial distribution function derived from the input distances, with each element being the density of distances falling into a specific bin.

The second Python function, **make_rdf_toy**, much like the first one, **make_rdf_feff**, generates an RDF from a given set of distances. However, the most significant difference lies in the normalization of the resulting RDF. In **make_rdf_toy**, the RDF is normalized by the area under the curve with **np.trapz** applied to the counter array over the range from the minimum to the maximum distance. This step ensures that the RDF integrates to 1 over this range (a dimer has a coordination number of 1).

3. Random variate sampling (RVS)

3.1 RVS is used to generate random variates from the a Gaussian distribution $N(\mu, \sigma^2)$, defined by the following probability density function:

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Here, $f(x|\mu, \sigma^2)$ represents the PDF of the Gaussian distribution with mean μ and variance σ^2.

The RVS method is then used to generate n random variates $X_1, X_2, \ldots, X_n$ each following the distribution $N(\mu, \sigma^2)$:

$$X_i \sim N(\mu, \sigma^2), i = 1, 2, \ldots, n$$

These X_i represent the pair distances that are to be converted into 3D structures.

3.2 The implementation of the RVS sampling in the dedicated computer package available in the Note 1 is described here. The random variate sampling (**rvs)** method is part of the **scipy.stats** module and is used to generate random variates from the specified statistical distribution. First, the distribution is defined with the **stats.norm** function from the **SciPy** package, which represents a normal continuous random variable. The **stats.norm** function is parametrized by **loc** and **scale**, which correspond to the mean and standard deviation of the distribution, respectively. The **stats.norm(loc=mu, scale=sigma)** method takes an optional size argument, which determines the shape of the returned array of random variates, i.e., when the size argument is provided as an integer number, say n, the method will return n random variates from the specified distribution. For example, **stats.norm(loc=mu, scale=sigma).rvs(1000)** generates 1000 random variates drawn from a normal (Gaussian) distribution with mean **mu** and standard deviation **sigma**. These variates are the pair distances that we would like to convert into 3D structures.

4. Toy Example Implementation

4.1. In this Section and in SI Section 5 we describe the implementation of the method to invert the EXAFS-like spectrum for an oscillating dimer (see main text, ***Appendix on the toy example of an oscillating dimer***).

4.2. 10,000 training examples, each consisting of fEXAFS (fake-EXAFS, f meaning "fake", because we omit the backscattering parameters) and RDF are calculated from randomly generated distributions of distances. The code defines a function **make_distro_norm** that creates

a normal distribution given parameters for lower and upper bounds, mean (**mu**), and standard deviation (**sigma**).

The loop iterates 10,000 times. In each iteration, it generates parameters for the normal distribution: a random minimum radius (**min_r**) between 1.5 and 5, a maximum radius (**max_r**) that's 1 unit larger than **min_r**, and a mean radius (**mean_r**) that's a random value between **min_r** and **max_r**. It uses these parameters to generate a set of 1,000 pair distances. It calculates fEXAFS and RDF from these distances using the **make_Fexafs** function, which uses the **chi_r** function. **chi_r** is a function of each distance over k-space (with the user-defined **kspace** grid variable), and **make_rdf_toy** (see Methods 2) digitizes the distances into bins, calculates the count of distances in each bin, normalizes these counts, and integrates them over the range from minimum to maximum distance. The EXAFS and RDF for each set of distances are stored in the **all_exafs** and **all_rdf** lists, respectively.

The validation data is created in a similar way to the training data but using two normal distributions instead of one. This is implemented with the **generate_exafs_rdf** function. Thus, we note that the training data is composed of normal distributions, while the validation data comprises asymmetric and bimodal distributions; hence, the NN hyperparameters were tuned so that the NN learns relevant features for both distributions.

The NN model is constructed using Keras. The function **baseline_model()** builds a feed-forward NN. It has an input layer that reshapes data into a one-dimensional vector with a size of 100. This is followed by three blocks, each composed of a densely connected layer with 516 nodes and SELU activation function, and a dropout layer (where dropout is .2, .1, and 0 per block). The final output layer is a dense layer with 600 nodes. The model is compiled using mean squared error (MSE) as the loss function and the Adam optimization algorithm with learning rate 0.0001, beta1=0.9, beta2=0.999, and a clipnorm of 1.0.

The training process uses a batch size of 64 and is planned for 100 epochs. The training data and targets are reshaped appropriately before being fed into the model. Model training also includes a validation step using a separate validation dataset, and uses a **ModelCheckpoint** callback to save the best model based on validation loss. If training is interrupted by the user, the code stops training and loads the best saved weights.

5. Harmonic RDF Implementation

To simulate an oscillating dimer, we simulate a two-body system connected by a spring (a dimer) using the 4th order Runge-Kutta method for numerical integration. Two balls with masses **m1** and **m2** are initially placed at positions 1.5 and 2.5, respectively, and their initial velocities are both zero. The spring connecting the balls has a spring constant **k** and a natural length of 2.0.

The function **calculate_acceleration** calculates the acceleration of each ball due to the force exerted by the spring, i.e.:

$$a_i = \frac{F_{spring}}{m_i} = \frac{k(x_2 - x_1 - L)}{m_i}$$

Here, F_{spring} is the force exerted by the spring, and x_1 and x_2 are the positions of m_1 and m_2, respectively. The spring force is calculated using Hooke's Law and Newton's third law is used to find the force on the second ball.

The time integration loop runs for 4000 steps, with each step representing a time increment dt of 0.01. The positions and velocities of the balls are updated at each step using the 4th order Runge-Kutta method, which involves calculating four increments (k1, k2, k3, and k4 for positions; l1, l2, l3, and l4 for velocities) at each step.

For positions x_i:

$$k_1 = f(x_i, t)$$

$$k_2 = f\left(x_i + \frac{\Delta t}{2} k_1, t + \frac{\Delta t}{2}\right)$$

$$k_3 = f\left(x_i + \frac{\Delta t}{2} k_2, t + \frac{\Delta t}{2}\right)$$

$$k_4 = f(x_i + \Delta t k_3, t + \Delta t)$$

$$x_i(t + \Delta \mathrm{t}) = \mathrm{x}_i(t) + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4$$

For velocities v_i:

$$l_1 = g(v_i, t)$$

$$l_2 = g\left(v_i + \frac{\Delta t}{2} l_1, t + \frac{\Delta t}{2}\right)$$

$$l_3 = g\left(v_i + \frac{\Delta t}{2} l_2, t + \frac{\Delta t}{2}\right)$$

$$l_4 = g(v_i + \Delta t l_3, t + \Delta t)$$

$$v_i(t + \Delta \mathrm{t}) = \mathrm{v}_i(t) + \frac{\Delta t}{6}(l_1 + 2l_2 + 2l_3 + l_4$$

Finally, the code calculates the distance between the two balls at each time step by subtracting the first ball's position from the second ball's position $d = |x_2 - x_1|$ and stores these distances in the **distances** array.

The distances array is input to **make_Fexafs** (see methods 4).

6. Bulk Ni Example Implementation

6.1. In this section, we describe the process of constructing the neural network for analyzing EXAFS spectra of bulk Ni. The key steps are data generation, and neural network construction. In data generation, a function is used to create a weighted linear combination.

In summary, the bulk Ni model is a transformer-based NN that is trained on synthetic data. The synthetic data is created by the random weighted linear combination of up to 10 base examples (out of 46 total base examples) where the weights are randomly selected, ensuring they add to 1**.** The training data is created on the fly with a data generator. Noise of random intensity is added to the training data. The validation data is created in the same manner but without the noise. During training, the MSE loss is minimized. Testing is performed on the RMC-EXAFS example from Ref. [13].

6.2 Data generation.

Here, we describe how the 46 base examples are created. First, 46 macrostates are created by considering 4 four continuous variables, $V_{Ni} = \{v_{1,Ni}, v_{2,Ni}, v_{3,Ni}, v_{4,Ni}\}$. The variables are set with the **stats.norm** function around the lattice constant **alpha**=3.524. We loop over a range of scaling factors (r) and standard deviations (s) and use RVS to sample 12, 6, 24, and 12 pairs from each continuous variable per shell, respectively, generating 50 microstates (**n_frames**) per macrostate.

Next, we utilize the GA to turn the microstates into configurations. The initial atomic positions in each shell are determined by implementing a Fibonacci grid (**fibonacci_sphere_with_rescaling**), which generates evenly distributed points on a sphere. The resulting list of initial structures (**a_structure_list**) is input into the optimization function **optimize_structure** with the minimum distance set to 2.23. The **optimize_structure** function is agnostic to atom type and number of atoms; it only maximizes the minimum distance.

FEFF10 is used to calculate the K-edge EXAFS for each example, where the amplitude reduction factor was set to 1, EXAFS set to 20, COREHOLE set to FSR, RPATH set to 6, CRITERIA set to '0.0 0.1', and SCF set to '4.0 0 100 0.2 1'.

The data is organized in directories according to macrostate, thus, the function **specific_data** was created to process examples within the macrostate directories and return the configuration average EXAFS and RDF pair. The returned EXAFS is interpolated onto a uniform **kspace** grid (here, we use **np.arange**(2, 12.05, .05)) and it is k^2 weighted. The returned RDF is constructed according to **Methods 2**.

The synthetic training data is generated as follows. The linear combination function **combinator_s**, first selects a random number **ran_i** between 1 and 10. Next, **ran_i** number of base examples are randomly selected for linear combination. Before the linear combination, weights are assigned randomly. To select a number of weights, and ensure they add to 1, we sample from the Dirichlet distribution, which is the multivariate generalization of the Beta distribution. For K categories, the PDF of the Dirichlet distribution is given by:

$$p(w|\alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} w_i^{\alpha_i - 1}$$

Where $w = (w_1, w_2, \ldots w_K)$ are the weights and $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_K)$ are the concentration parameters. We set all α to 1 to sample uniformly. The term $B(\alpha)$ is the multivariate Beta function, defined as:

$$B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)}$$

Here, $\Gamma(\alpha_i)$ is the Gamma function.

This is implemented by using **np.random.dirichlet.** Finally, the RDF is sliced between index 15 and 55 in order to crop out some of the zero intensity range, and the weighted average EXAFS and RDF are returned. The **generate_examples** function calls **combinator_s** for the number of examples desired and can be used to generate the validation data. The **data_generator** function is called during training. It generates examples according to the **batch_size** and adds a random amount of noise to the EXAFS.

6.3 Neural Network Architecture.

The model's architecture comprises an initial dense layer with 64 nodes (**d_model**), a transformer encoder layer, a **GlobalAveragePooling1D** layer, and an output layer with 40 nodes. The transformer encoder layer consists of a **MultiHeadSelfAttention** mechanism with 4 heads (**num_heads**), followed by two sequential feed-forward networks, each having a dropout rate of 0.4. The input to the model has a shape of (201,1), and the output shape is (1,40). Each attention head in the **MultiHeadSelfAttention** mechanism divides the input into a depth (**d_model**/**num_heads**) using a dense layer for each of query, key, and value (**wq**, **wk**, **wv**) computations. The outputs from the attention mechanism are then concatenated and passed through a final dense layer. The transformer encoder also includes dropout layers following the attention and pointwise feed-forward networks, and **LayerNormalization** after the sum of inputs and these two respective layers. The feed-forward network is a sequential model comprising two dense layers, the first with 516 nodes (**dff**) and RELU activation, and the second with nodes equal to **d_model**.

The model is trained using the **Adam** optimizer (learning rate = 0.001) and mean squared error (**MSE**) loss function for 100 epochs, with batch sizes of 64. A checkpoint callback saves the model after each epoch.

7. NiClZnK Example Implementation

7.1. In this section and in the Section 8 we describe the implementation of the neural network-based method for extracting the g(r) for NiClZnK complex from its EXAFS spectra. The method works identically for both AIMD-EXAFS and experimental EXAFS spectra.

The NiClZnK model is a convolution-based NN that is trained on synthetic data. The synthetic data is created by the random weighted linear combination of up to 5 base examples (out of 135 total base examples) where the weights are randomly selected, ensuring they add to 1**.** The training data is created on the fly with a data generator. Noise of random intensity is added to the training data. The validation data is created in the same manner but without the noise. During

training, the MSE loss is minimized. Testing is performed on AIMD data of composition 1 (comp 1) and composition 2 (comp 2) from Ref [39].

7.3 Data Generation.

Here, we describe how the 135 base examples are created. First, 135 macrostates are created by considering three sets of three continuous variables, i.e., $V_{Cl} = \{v_{1,Cl}, v_{3,Cl}\}$, $V_{Zn} = \{v_{2,Zn}\}$, $V_K = \{v_{3,K}\}$, which correspond to three shells, where shell 1 contains Cl, shell 2 contains Zn, and shell 3 contains Cl and K. The variables are set with the **stats.norm** function. We loop over a range of scaling factors (r) and standard deviations (s) and use RVS to sample pairs from each continuous variable per shell, respectively, generating 100 microstates (**n_frames**) per macrostate. The number of pairs sampled in the first (**n_cl**) and second shell (**n_zn**) are varied between 1 and 6 and 1 and 2, respectively. The third shell always contains 11 atoms, in which the number of Cl and K depends on the user-defined composition parameter **n_frac**, which is set to 20%.

Next, we utilize the GA to turn the microstates into configurations. The initial atomic positions in each shell are determined by **fibonacci_sphere_with_rescaling**. The resulting list of initial structures (**a_structure_list**) is input into the optimization function **optimize_structure** with the minimum distance set to 2.5. The **optimize_structure** function is agnostic to atom type and number of atoms; it only maximizes the minimum distance.

FEFF10 is used to calculate the K-edge EXAFS for each example, where the amplitude reduction factor was set to 1, EXAFS set to 20, COREHOLE set to FSR, RPATH set to 6, CRITERIA set to '0.0 0.1', and SCF set to '4.0 0 100 0.2 1'.

The synthetic training data is generated as follows. The linear combination function **combinator_s**, first selects a random number **ran_i** between 1 and 5. Next, **ran_i** number of base examples are randomly selected for linear combination. Before the linear combination, weights are assigned randomly using **np.random.dirichlet,** which is described in more detail in the previous section**.** EXAFS and partial Ni-Cl and Ni-Zn RDFs are returned. The **generate_examples** function calls **combinator_s** for the number of examples desired and can be used to generate the validation data. The **data_generator** function is called during training. It generates examples according to the **batch_size** and adds a random amount of noise to the EXAFS. We note that the NN is trained on the concatenated RDFs, i.e., the Ni-Cl (length 40) and Ni-Zn (length 60) RDFs are concatenated into one length 100 vector when using the **generate_examples** and **data_generator** function.

The model is a sequential architecture consisting of two blocks of 1D convolutional layers, each followed by batch normalization, SELU activation, max pooling, and a 0.2 dropout. The first block uses 64 filters with kernel size 3 and an input shape of (201,1). The second block has 128 filters with kernel size 3. After the convolutional blocks, the model employs a Flatten layer followed by a dense layer with 768 nodes, with batch normalization, SELU activation, and a 0.2 dropout. The model concludes with a dense output layer of 100 nodes.

The model is trained using the **Adam** optimizer (learning rate = 0.0001) and mean squared error (**MSE**) loss function for 100 epochs, with batch sizes of 128. A checkpoint callback saves the

model after each epoch so that the training stops if the validation loss does not decrease by at least 0.001 over 5 epochs, and the weights from the epoch with the best validation loss are restored. Training is performed 10 times to create an ensemble of networks for uncertainty estimation.

8. NiClZnK fine-tuning

Two datasets (test and train) are constructed from the comp 1 and comp 2 datasets and used for fine-tuning. To do this, the comp 1 and comp 2 datasets are grouped based on the center of mass of the Ni-Cl and Ni-Zn RDF. We calculate each group's configuration average EXAFS and RDF pair, resulting in 86 examples. The 86 examples are split randomly into the train (60 examples) and test (26 examples) sets and then used for synthetic training data generation. The linear combination function **combinator_s_ft**, example creation function **generate_examples_ft**

The data generator **data_generator_ft** function is the same as the ones described in Note 7, except that it operates on the AIMD training datasets **x_ft_train** (EXAFS) and **y_ft_train** (RDFs). The **combinator_s_ft_test** is used to create the validation data from the AIMD test dataset (**x_ft_test** and **y_ft_test**).

The fine-tuning occurs 10 times, once for each model trained in Note 7. For each model, the algorithm retrieves the corresponding base model from a dictionary of models (model_dict_40k_af). It then expands the base model by adding two dense layers with 256 units and SELU activation functions, each followed by a dropout layer (0.2) for regularization. A final dense layer (100) with a linear activation function is added, which serves as the output layer of the model. The weights of the base model layers are frozen to preserve the learned features, ensuring that only the weights of the newly added layers will be updated during training.

The new model is compiled using mean squared error (MSE) as the loss function and the Adam optimizer with a learning rate of 0.0001. An early stopping callback is used to monitor the validation loss. The training stops if the validation loss does not decrease by at least 0.001 over 5 epochs, and the weights from the epoch with the best validation loss are restored.

The model is trained using a data generator function (**data_generator_ft**) for 100 epochs with a batch size of 128.

Supplemental Figures

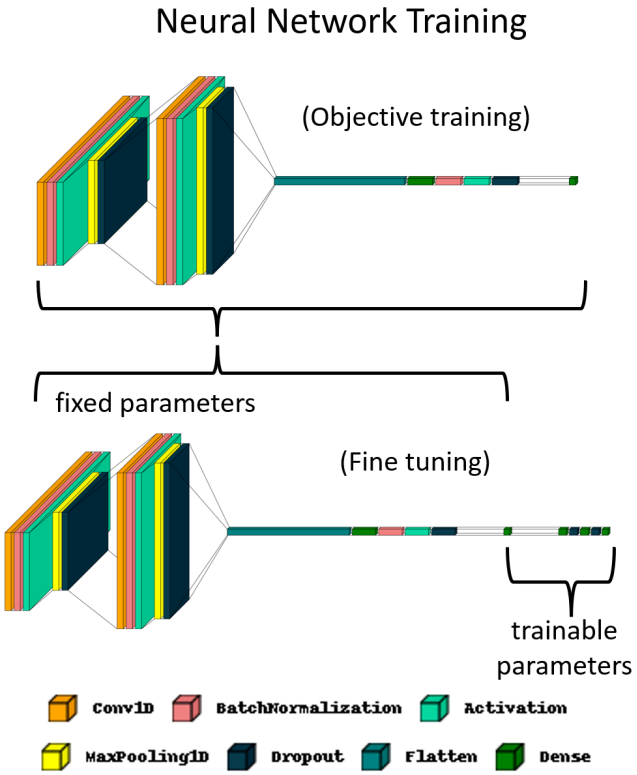

Figure S1. The NN architecture of the objectively trained and fine-tuned NN-EXAFS models. Figure made with the visualkeras repository [54].

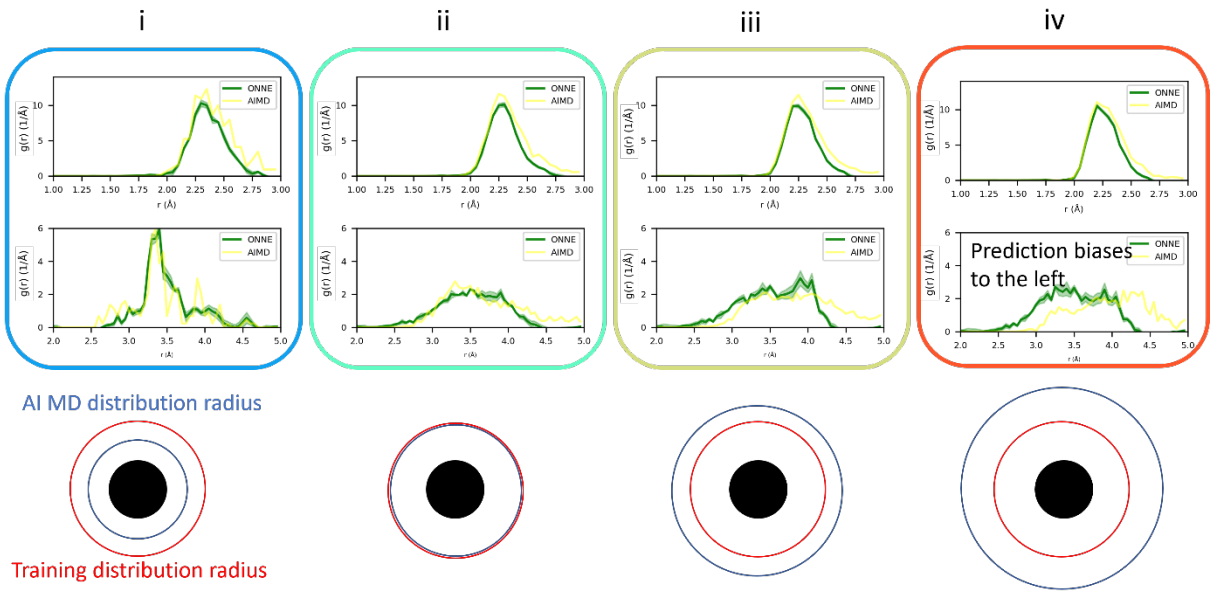

Figure S2. Trained NiClZnK model predictions (ONNE) compared to AIMD data. The Ni-Cl (top) and Ni-Zn (bottom) RDFs are shown in each block i – iv. The difference between the blocks is the mean radius of the Ni-Zn distribution that is being predicted on, where i is the smallest radius and iv is the largest. The cartoon on the bottom illustrates the concept. When the radius is too large, surpassing the radius of the training data, the NN predictions fail, skewing to a lower radius.

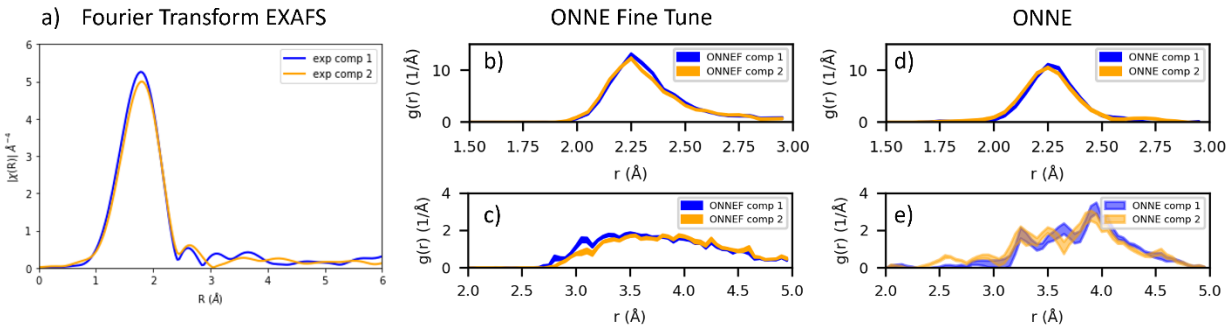

Figure S3. (a) The Fourier Transformed (between 2 and 9 $Å^{-1}$) k^3-weighted Ni K-edge experimental EXAFS data from Ref 29. (b), (c) The result of inversion, using the fine-turned ONNE method (ONNEF), to the Ni-Cl and Ni-Zn radial distribution functions respectively. (d), (e) The result of inversion, using the ONNE method, to the Ni-Cl and Ni-Zn radial distribution functions respectively. Comp 1 is blue, and Comp 2 is orange.

References

[13] J. Timoshenko, A. Anspoks, A. Cintins, A. Kuzmin, J. Purans, and A. I. Frenkel, Neural network approach for characterizing structural transformations by X-ray absorption fine structure spectroscopy, Phys. Rev. Lett. **120**, 225502 (2018).

[39] S. Roy, Y. Liu, M. Topsakal, E. Dias, R. Gakhar, W.C. Phillips, J. F. Wishart, D. Leshchev, P. Halstenberg, S. Dai *et al.*, A holistic approach for elucidating local structure, dynamics, and speciation in molten salts with high structural disorder, J. Am. Chem. Soc. **143**, 15298 (2021).

[52] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, DEAP: Evolutionary algorithms made easy, J. Mach. Learn. Res. 13, 2171 (2012).

[53] R. Swinbank and R. James Purser, Fibonacci grids: A novel approach to global modelling, Q. J. R. Meteorol. Soc. 132, 1769 (2006)

[54] P. Gavrikov, visualkeras, GitHub repository (2020).