Supplementary Information for Nonlinearity Encoding to Improve Extrapolation Capabilities for Unobserved Physical States

Gyoung S. Na*, Seunghun Jang, and Hyunju Chang
Korea Research Institute of Chemical Technology, Republic of Korea
ngs0@krict.re.kr

November 25, 2021

1 Implementation Details of n-Body Problem

In the implementations of DNNs in the n-body problem, we used FCNNs with five hidden layers. For each layer, batch normalization [5] was applied to accelerate the parameter optimization. For AUNE, the margin α and the balancing coefficient λ for the angular distances were set as 2e-1 and 1e-3, respectively. For each iteration of the training of AUNE, two samples were selected ($K = 2$) to calculate the loss function of the distance matching problem. All DNNs were trained by Adam optimizer [6] with an initial learning rate of 1e-3 and a batch size of 16. The model parameters of DNNs were optimized during 500 iterations. To simulate the trajectories of n particles, we used a python code at `https://github.com/pmocz/nbody-python`.

2 Implementation Details of Materials Property Prediction

We used CGCNN with three graph convolution and two dense layers as the competitor and the projection network of AUNE. In this experiment, AUNE trained with the margin $\alpha = 2e - 1$, the balancing coefficient $\lambda = 1e - 2$, and the number of samples $K = 2$. Similarly, all DNNs were trained by Adam optimizer during 300 iterations. The competitor CGCNN was trained with an initial learning rate of 5e-4 and a batch size of 128. AUNE was trained with the initial learning rate of 1e-3 and the batch size of 64. We used PyTorch [7] and PyTorch Geometric Library [2] to implement CGCNN.

The crystal structures of the hybrid perovskites were converted into the mathematical graphs $G = (\mathcal{V}, \mathcal{E}, X, Q)$, where $\mathcal{V}$ is a set of atoms, $\mathcal{E}$ is a set of chemical bonds between the atoms, X is an atom-feature matrix, and Q is a bond-feature matrix. Nine elemental attributes shown in Table 1 were assigned for each atom in $\mathcal{V}$. For each pair of atoms within 5 Å, an edge between the atoms was generated with 128 bond features of the bond length discretized based on radial basis function kernel (RBF). We used Pymatgen [1] to convert the crystal structures into the mathematical graphs.

Table 1: Selected elemental attributes and their characteristics.

Category	Variable name (Unit)	Comment
Size	atomic_volume (cm^3/mol)	Atomic volume
	atomic_weight (-)	Atomic weight
	covalent_radius_bragg (pm)	Covalent radius by Bragg.
Heat	fusion_heat (kJ/mol)	Fusion heat
Electronic	atomic_number (-)	Atomic number
	en_pauling (-)	Pauling's scale of electronegativity
	electron_affinity (eV)	Electron affinity
	period (-)	Period in periodic table
	first_ion_eng (eV)	First ionization energy

3 Algorithmic Description of AUNE-Based Machine Learning

For the reproducibility of this work, we provide an algorithmic description of AUNE-based ML and its training process. Algorithm 1 presents a formal process of the training process of AUNE and the AUNE-based ML. Note that the training arguments of the gradient descent methods are omitted. For a training dataset $\mathcal{D}$ and hyperparameters, the nonlinearity encoder $f(\mathbf{x}; \boldsymbol{\theta})$ is trained to minimize the training loss $L = L_k(\mathcal{D}; \boldsymbol{\theta}) + \lambda\Omega_\alpha(\mathcal{D}; \boldsymbol{\theta})$ by gradient descent method as shown in line 9.

After the training of the nonlineairty encoder, the original data $\mathbf{x}_i$ are projected by the encoder into the latent space where the input features and the target values have a linear relation. The original data $\mathcal{D}$ is transformed into $\mathcal{Z} = f(\mathcal{D}; \boldsymbol{\theta}^*)$ as shown in line 14. For the latent data $\mathcal{Z}$, the prediction

Algorithm 1: Training process of AUNE and AUNE-based ML

Input : Training dataset $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$,
Margin $\alpha > 0$,
Balancing coefficient $\lambda > 0$,
The number of samples $K > 0$

Output: Prediction results $\mathcal{Y} = \{\mathbf{y}_1^{'}, \mathbf{y}_2^{'}, ..., \mathbf{y}_N^{'}\}$

1 // Train nonlinearity encoder
2 **repeat**
3 // Calculate the training loss function the given α, λ, and K.
4 $L_k = \frac{1}{4NK} \sum_{i=1}^{N} \sum_{k=1}^{K} \left(d_p(\mathbf{z}_i, \mathbf{z}_{r_k}; \boldsymbol{\theta}) - d_p(\mathbf{y}_i, \mathbf{y}_{r_k})\right)^2$
5 $\Omega_\alpha = \frac{1}{N} \sum_{i=1}^{N} \sum_{t_{i,j,q} \in \mathcal{T}_i} \max(\Omega(t_{i,j,q}; \boldsymbol{\theta}) - \alpha, 0)$
6 $L = L_k(\mathcal{D}; \boldsymbol{\theta}) + \lambda \Omega_\alpha(\mathcal{D}; \boldsymbol{\theta})$
7 // Optimize model parameters of the encoder for a given learning rate η_f.
8 $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta_f \frac{\partial L}{\partial \boldsymbol{\theta}}$
9 **until** $\boldsymbol{\theta}$ converged;
10 // Train prediction model based on the trained encoder.
11 **repeat**
12 // Project original data into the latent space.
13 $\mathcal{Z} = f(\mathcal{D}; \boldsymbol{\theta}^*)$
14 // Calculate surrogate loss function (e.g., MAE) for the projected inputs.
15 $L_s = \frac{1}{N} \sum_{i=1}^{N} |\mathbf{y}_i - g(\mathbf{z}_i; \boldsymbol{\mu}^*)|$
16 // Optimize model parameters of the prediction model for a given learning rate η_g.
17 $\boldsymbol{\mu} = \boldsymbol{\mu} - \eta_g \frac{\partial L_s}{\partial \boldsymbol{\mu}}$
18 **until** $\boldsymbol{\mu}$ converged;
19 **return** $f(\mathbf{x}; \boldsymbol{\theta}^*)$ and $g(\mathbf{z}; \boldsymbol{\mu}^*)$

model $g(\mathbf{z}; \boldsymbol{\mu})$ is trained to minimize the surrogate loss (e.g., MAE) based on the gradient descent methods as shown in line 18. For the optimized encoder and the trained prediction model, we can predict a physical state $\mathbf{x}$ via $\mathbf{y} = g(f(\mathbf{x}; \boldsymbol{\theta}^*); \boldsymbol{\mu}^*)$. All implementation details and source codes of AUNE are publicly available at *open after the review process*.

4 Approximation Error

For the distance matching problem, we can theoretically derive an approximation error of the sample-based encoding by applying Hoeffding's inequality to the error bound [4]. An important characteristic of the projected relative distances L_i is that $\frac{(d_p(\mathbf{z}_i, \mathbf{z}; \boldsymbol{\theta}) - d_p(\mathbf{y}_i, \mathbf{y}_j))^2}{4}$ is bounded in [0, 1] because $d_p(\mathbf{z}_i, \mathbf{z}; \boldsymbol{\theta})$ and $d_p(\mathbf{y}_i, \mathbf{y})$ are in $[0, 2]$ by the triangle inequality. Furthermore, we assume that each data is independent and identically distributed (i.i.d.). Hence, L_i can be regarded as the empirical mean of the i.i.d. random variable in [0, 1]. For independent random variables $X \in [0, 1]$, Hoeffding's inequality derives the following probability for error bounds as:

$$\mathrm{P}(|\overline{X} - \mathrm{E}[\overline{X}]| \geq \epsilon) \leq 2e^{-2M\epsilon^2}, \tag{1}$$

where $\epsilon \geq 0$ is an expected approximation error, and M is the number of samples to calculate $\overline{X}$. Because the objective function of the approximated problem can be interpreted as an empirical loss of the original objective function of the original problem, this probability can be reformulated for our distance matching problem as:

$$\mathrm{P}\left(|L_k(\mathcal{D}; \boldsymbol{\theta}) - L_d(\mathcal{D}; \boldsymbol{\theta})| \geq \epsilon\right) \leq 2e^{-2NK\epsilon^2}, \tag{2}$$

where $L_d(\mathcal{D}; \boldsymbol{\theta})$ is the relative distances for all data, and $L_k(\mathcal{D}; \boldsymbol{\theta})$ is the relative distances in the stochastic environments. Note that both $L_k(\mathcal{D}; \boldsymbol{\theta})$ and $L_d(\mathcal{D}; \boldsymbol{\theta})$ are bounded within $[0, 1]$ because the relative distances were normalized. Therefore, with a probability at least $1 - 2e^{-2NK\epsilon^2}$, we have the following bound for a given projection function $f(\mathbf{x}; \boldsymbol{\theta})$:

$$|L_k(\mathcal{D}; \boldsymbol{\theta}) - L_d(\mathcal{D}; \boldsymbol{\theta})| \leq \epsilon. \tag{3}$$

Practically, the approximation error is less than 10% with a probability of 96.34% for a given dataset $\mathcal{D}$ with $N = 100$ and the number of samples $K = 2$.

From the probability of the approximation error in Eq. (2), we can derive the minimum number of samples for accurate approximation. For a given expected error ϵ and a probability β, we can formulate the probability that the approximation error is less than ϵ with a probability at least β as:

$$1 - 2e^{-2NK\epsilon^2} \geq \beta. \tag{4}$$

This inequality can be rewritten with respect to the number of samples $K \in \mathbb{N}^+$ as:

$$K \geq \frac{1}{2N\epsilon^2} \log\left(\frac{2}{1-\beta}\right) \tag{5}$$

Thus, K should be a natural number greater than $\frac{1}{2N\epsilon^2}\log\left(\frac{2}{1-\beta}\right)$ to ensure that the approximation error is less than ϵ with the probability being at least β.

5 Prediction Error of n-Body Problem in Future Time

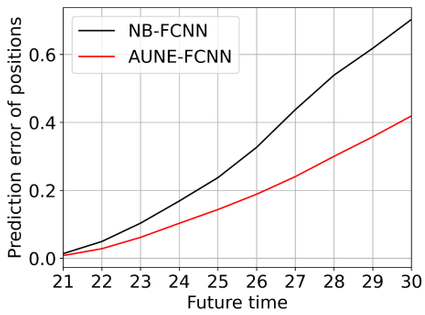

Figure 1: Extrapolation errors in the future time (20, 30]. The extrapolation errors were measured by the mean of the prediction errors of the 10 times repetition of the n-body problem with the random initialization. The black and red solid lines indicate the extrapolation errors of NB-FCNN and AUNE-FCNN in predicting the future positions of the particles, respectively.

Fig. 1 shows the extrapolation errors of NB-FCNN and AUNE-FCNN in the n-body problem. The extrapolation errors are presented for each future time in the figure, and the errors were calculated by the mean of the 10 times repetition of the randomly initialized n-body problems. The black and red solid lines indicate the extrapolation errors of NB-FCNN and AUNE-FCNN in predicting the future positions of the particles, respectively. As shown in the figure, AUNE-FCNN (red line) always showed lower errors in predicting the future positions of the particles.

6 Predicted Trajectories for Different Initial States of n-Body Problem

In the experiments, we measured the prediction errors using the mean of Frobenius norm [3] from 10 times repetitions for different initial states of the particles. Fig. 2-11 present the simulated and predicted trajectories of three particles for each different initial state of the n-body problem.

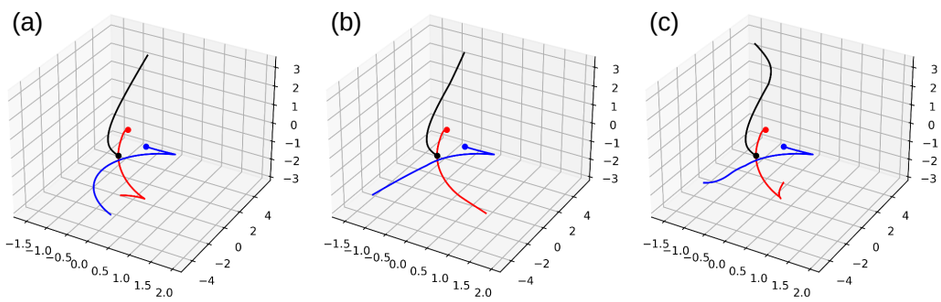

Figure 2: Simulated and predicted trajectories of three particles in case 1. (a): True trajectories simulated by a computer software. (b): Predicted trajectories of AUNE-FCNN. (c): Predicted trajectories of NB-FCNN.

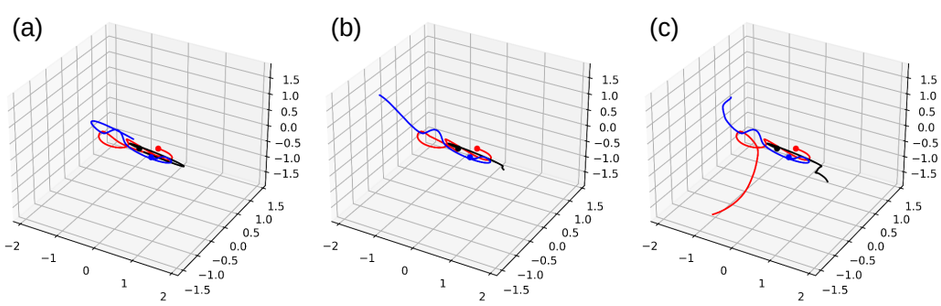

Figure 3: Simulated and predicted trajectories of three particles in case 2.

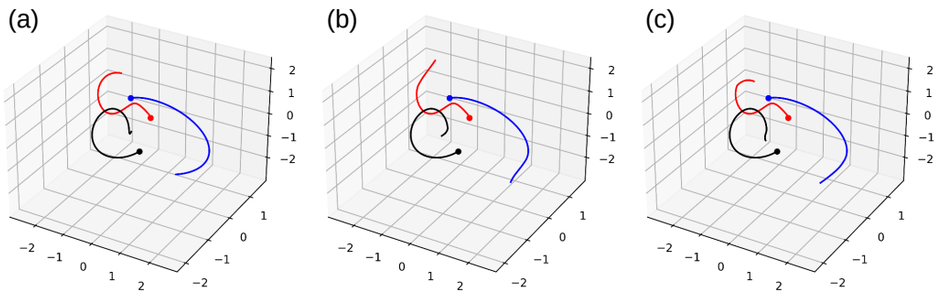

Figure 4: Simulated and predicted trajectories of three particles in case 3.

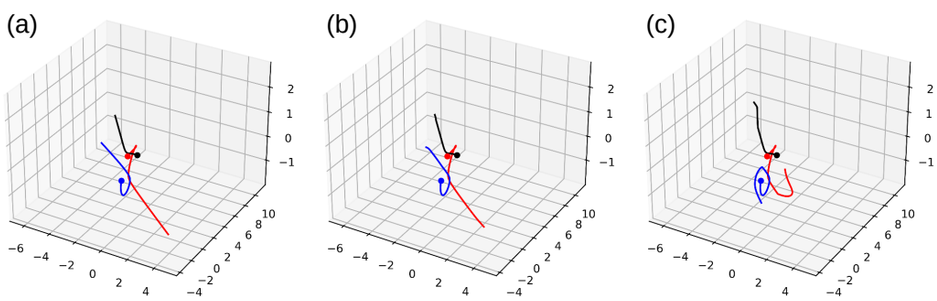

Figure 5: Simulated and predicted trajectories of three particles in case 4.

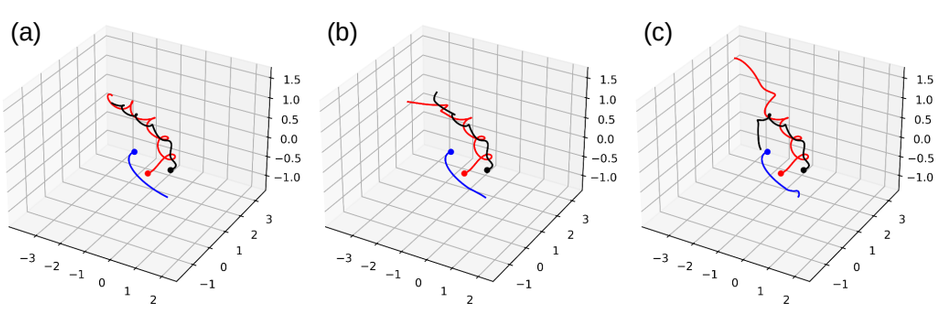

Figure 6: Simulated and predicted trajectories of three particles in case 5.

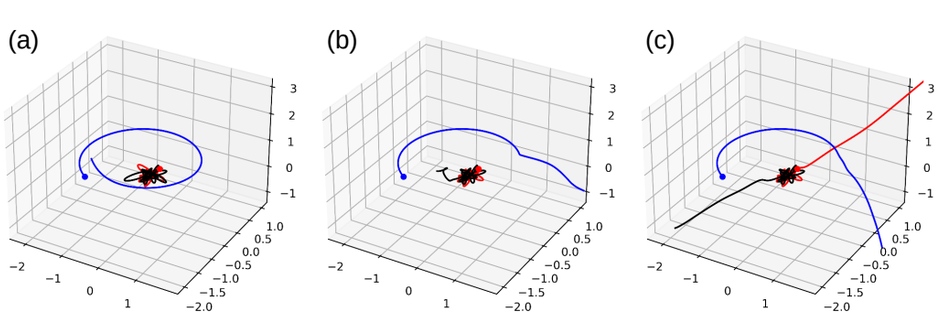

Figure 7: Simulated and predicted trajectories of three particles in case 6.

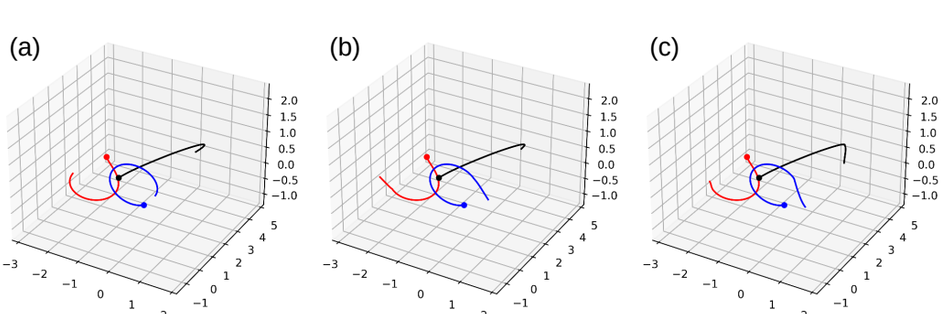

Figure 8: Simulated and predicted trajectories of three particles in case 7.

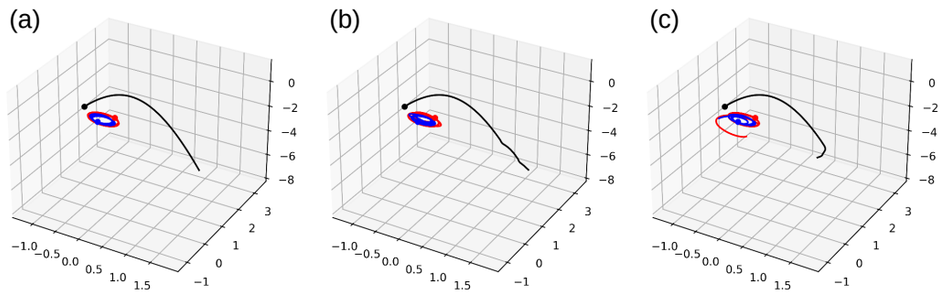

Figure 9: Simulated and predicted trajectories of three particles in case 8.

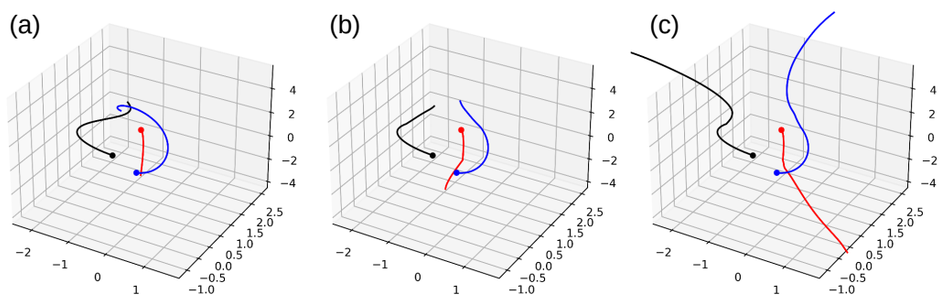

Figure 10: Simulated and predicted trajectories of three particles in case 9.

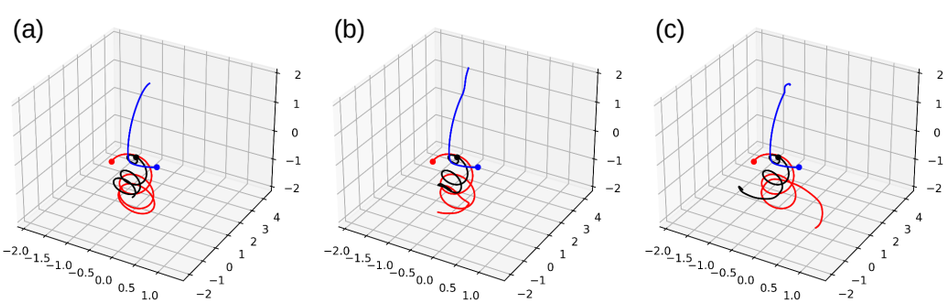

Figure 11: Simulated and predicted trajectories of three particles in case 10.

References

[1] pymatgen. `https://pymatgen.org/`, 2021 (accessed April 21, 2021).

[2] Pytorch-geometric. `https://pytorch-geometric.readthedocs.io`, 2021 (accessed April 21, 2021).

[3] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, USA, 1996. ISBN 0801854148.

[4] Wassily Hoeffding. *Probability Inequalities for sums of Bounded Random Variables*, pages 409–426. Springer New York, 1994. ISBN 978-1-4612-0865-5. doi: 10.1007/978-1-4612-0865-5_26.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. PMLR, 2015.

[6] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017.