Nonlinearity Encoding for Extrapolation of Neural Networks

Gyoung S. Na*
Korea Research Institute of Chemical Technology
Republic of Korea
ngs0@krict.re.kr

Chanyoung Park*
Korea Advanced Institute of Science and Technology
Republic of Korea
cy.park@kaist.ac.kr

ABSTRACT

Extrapolation to predict unseen data outside the training distribution is a common challenge in real-world scientific applications of physics and chemistry. However, the extrapolation capabilities of neural networks have not been extensively studied in machine learning. Although it has been recently revealed that neural networks become linear regression in extrapolation problems, a universally applicable method to support the extrapolation of neural networks in general regression settings has not been investigated. In this paper, we propose automated nonlinearity encoder (ANE) that is a data-agnostic embedding method to improve the extrapolation capabilities of neural networks by conversely linearizing the original input-to-target relationships without architectural modifications of prediction models. ANE achieved state-of-the-art extrapolation accuracies in extensive scientific applications of various data formats. As a real-world application, we applied ANE for high-throughput screening to discover novel solar cell materials, and ANE significantly improved the screening accuracy.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**;
• **Applied computing** → **Physical sciences and engineering**.

KEYWORDS

extrapolation, metric learning, scientific application

ACM Reference Format:
Gyoung S. Na and Chanyoung Park. 2022. Nonlinearity Encoding for Extrapolation of Neural Networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539326

1 INTRODUCTION

Extrapolation to approximate input-to-target relationships of unseen data outside the training distribution is one of the most challenging problems in machine learning [33, 41, 46]. At the same time, extrapolation problems are common in real-world applications, such as classifying noisy images [42] and forecasting new patterns [32]. In particular, being able to extrapolate is at the core of discovering new scientific knowledge and novel chemical compounds in many scientific applications, such as physical dynamics interpretation [13], drug discovery [38], and inverse design of materials [28]. For example, it is crucial to accurately predict toxicities and aqueous solubility of newly-designed molecular structures to develop safe and reliable drugs [38].

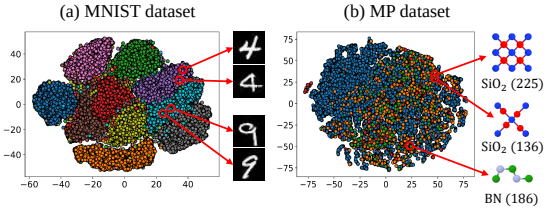

Figure 1: t-SNE plots of MNIST and MP datasets. Each point is an image or a material with class labels denoted by colors.

In this paper, we focus on the extrapolation capabilities of machine learning methods. Formally, for a given prediction model $f : \mathcal{X} \rightarrow \mathbb{R}$ trained on a training distribution $\mathcal{D}$, we consider f to extrapolate well if it has a small extrapolation error L_e as:

$$L_e = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{X}\setminus\mathcal{D}}\left[L_s(\mathbf{y}, f(\mathbf{x}))\right], \tag{1}$$

where $\mathbf{x} \in \mathcal{X}$ is an input data sample, $y \in \mathbb{R}$ is a target response of $\mathbf{x}$, and $L_s : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is a loss function to measure the prediction error. To facilitate the extrapolation, various training strategies and network architectures have been proposed. However, most existing methods are limited to classification problems, and impose apriori assumptions such as white-box systems [33] and symmetric data domain [41]. In particular, despite previous efforts in machine learning extrapolation on image datasets [4, 40], extrapolation in scientific applications is still far from satisfactory [23, 24]. As an example, materials science is one of the largest scientific fields where extrapolation using machine learning is still questionable [24], as shown in the experimental evaluations on the most well-known benchmark Materials Project (MP) dataset [14].

In order to tackle the extrapolation in machine learning, Xu et al. [46] proposed an extrapolation strategy that encodes nonlinearities into the input representation, based on the theoretical analysis that neural networks with ReLU activation become linear regression on $\mathcal{X} \setminus \mathcal{D}$, i.e., outside the training distribution. By doing so, the input-to-target relationships were simplified, which in turn improved the extrapolation accuracy in several scientific applications. This result implies that a simple input-to-target relationship is essential in machine learning extrapolation. However, we argue that such a nonlinearity encoding process lacks generalizability owing to the fact that *the encoding of nonlinearities is task-dependent, i.e., nonlinearities should be carefully encoded into the input representations for each application (or equivalently data) at hand.* Therefore, we need an encoding method to automatically simplify the input-to-target relationships regardless of the prediction problems.

*These authors contributed equally to this work

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539326

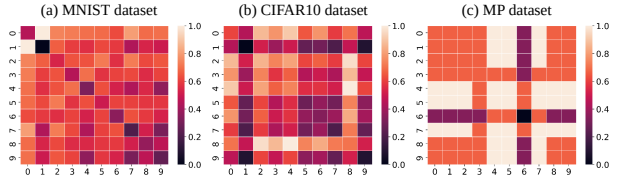

Figure 2: Heatmap visualization of within- and between-class distances on benchmark image and materials datasets. The classification tasks become more challenging from (a) MNIST to (c) MP datasets, and the classification accuracies dropped from 99.36% to 77.12% accordingly.

To conceptually describe the simplicities of the input-to-target relationships, first, we compare t-SNE plots of MNIST dataset [2] and MP dataset [14] in Fig.1. Each point[1] represents an image (Fig.1a) and a crystal structure of the material (Fig.1b), respectively, and the colors indicate their target responses, i.e., label for MNIST and band gap for MP[2]. We observe that images are well-separated according to their labels whereas materials are not. This is expected since it is natural for similar images to share similar labels. e.g., two slightly different digit-4s are located close to each other, whereas in MP dataset, *similarity in the input does not always imply similarity in the target response, and vice versa.* For example, SiO_2 (225) and SiO_2 (136) have completely different band gaps (2.00 eV and 5.50 eV) despite their highly similar crystal structures, while SiO_2 (136) and BN (186) have almost the same band gaps (5.50 eV and 5.36 eV) despite their completely different composition and crystal structures.

To elaborate our observations in Fig. 1, we calculate the average pairwise input distance within a class (i.e., within-class distance) and between different classes (i.e., between-class distance) for three datasets in Fig. 2. We observe that the within-class distances (i.e., diagonal) are generally smaller than the between-class distances (i.e., off-diagonal) in MNIST dataset of Fig. 2a, and such a trend gradually vanishes as we move from Fig. 2b (CIFAR10) to Fig. 2c (MP). Note that it is widely known that classification in CIFAR10 dataset is generally more challenging than in MNIST dataset, which can be explained by the differences between the tendencies of the within- and between-class distances in Fig. 2a and 2b. Besides, the difficulty of classification on MP dataset, which is challenging to achieve 77.12% classification accuracy, can be also explained by Fig. 2c. In this regard, we suppose that prediction tasks can be made easier by enforcing the distance between inputs having the same target smaller.

We can extend our argument to the regression tasks by assuming that the number of classes in the classification tasks is infinite and the classes are ordered. In this case, we argue that the regression problem can be simplified by enforcing the distance between inputs to be proportional to the distance between their corresponding targets, and we refer to such a consistent relationship between the input and target distances as *distance consistency*. To corroborate our argument, we conduct linear regression on four synthetic datasets whose distance consistencies are intentionally controlled, as shown

[1]The input representations are obtained by convolutional neural network (CNN) [21] for MNIST and graph neural network (GNN) [20] for MP.
[2]Since band gap is a numerical value, we convert it to a categorical variable. Further details are described in Appendix 1

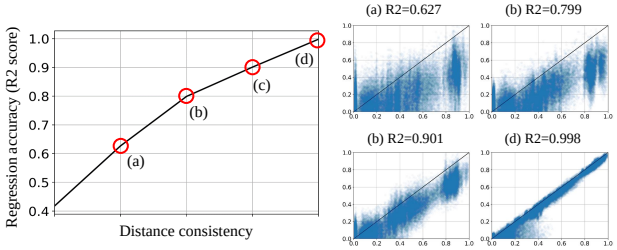

Figure 3: Regression accuracies of linear regression on synthetic datasets for different distance consistency.

in Fig. 3. We observe that high distance consistency yields high R^2 scores [22] of the linear regression, which implies that the input-to-target relationships can also be made simpler by increasing the distance consistency.

In this paper, we propose a novel automated nonlinearity encoder (ANE) for improving the extrapolation capabilities of machine learning extrapolation in general regression tasks. The main idea is to increase the distance consistency aiming at simplifying the input-to-target relationships. More precisely, ANE encodes nonlinearities of regression problems into the input embeddings by minimizing the Wasserstein distance between the pairwise distances of the data samples in the input and the target spaces. That is, ANE can be regarded as deep metric learning on regression settings. By applying the trained ANE on a given dataset, the input-to-target relationships of the regression problems are automatically simplified (i.e., distance consistency is maximized) so that neural networks can extrapolate well. ANE is data-agnostic and generally applicable to various scientific applications because there is no assumption on prediction problems and data distributions. To demonstrate the effectiveness of ANE in the extrapolation problems, we conduct comprehensive experiments on various scientific applications with different input data formats, such as n-body problem [31], crystal graph regression [43], and geomagnetic storm forecasting [1]. As a real-world scientific application, we applied ANE to discover novel perovskite materials [17] that are getting much attention as solar cell materials to overcome the global climate crisis. From our empirical analysis, we demonstrate that ANE significantly reduces the extrapolation errors of neural networks without manual feature engineering and problem reformulation as done in [46].

2 RELATED WORK

2.1 Machine Learning Extrapolation

Generalization refers to the ability of a machine learning model to adapt to noisy and abnormal data drawn from the same distribution as the one used to train the model. On the other hand, extrapolation refers to ability of the trained models to adapt to data generated from a completely unknown distribution [4, 33, 40, 46]. Universal and different set domain adaptations methods were devised to predict target responses of unseen data from unknown domain [47]. Similarly, continual learning with new class emergence aims to train a prediction model to accurately predict target responses of the data from unseen classes [25]. However, these tasks are distinguished from the extrapolation problems in that they require re-training of the model on the newly introduced data, whereas

extrapolation is a prediction capability of the trained model for the data from the data distributions outside the training distribution. In addition, existing studies proposed several training strategies and prediction networks to facilitate and support machine learning extrapolation in various perspectives, such as representation learning [41], transfer learning [9, 15], and out-of-distribution detection [16]. However, they are not applicable to the extrapolation problems on general regression settings because they are designed for classification problems on discrete target values.

Most recently, Xu et al. [46] demonstrated through a theoretical analysis that neural networks with ReLU activation become linear regression in the extrapolation settings, and achieved state-of-the-art extrapolation accuracies in several scientific applications. Despite its success, its general applicability is limited to several applications, because the input-to-target relationships should be manually simplified for each task (or equivalently data) at hand.

2.2 Metric Learning on Regression Settings

Deep metric learning (DML) is a widely studied methodology in computer vision to generate latent embeddings that simplify the input-to-target relationships with supervision [34, 36, 48]. Various loss functions to train embedding networks have been proposed in DML, such as contrastive loss [7] and triplet loss [34]. Moreover, sophisticatedly-designed loss functions are widely studied for multi-class classification problems, such as N-pair loss [36] and circular loss [48]. However, although various metric learning losses have been proposed in computer vision, most existing metric learning methods were limited to discrete target values on image data, i.e., image classification tasks.

To extend the applicability of DML beyond the classification settings, log-ratio loss (LRL) for continuous target values was proposed for human pose detection tasks [18]. LRL evaluates the embedding quality of data based on the log-ratio of the distances between the input and target data. However, LRL cannot be readily applied to scientific data that contain large measurement and calculation noises, because it is numerically unstable to unexpected input values owing to the logarithm in the loss formulation [27]. To overcome the numerical instability of LRL, smoothed log-ratio loss (SLRL) was proposed by smoothing the logarithm with an exponentiation based approximation function [27]. SLRL achieved state-of-the-art prediction accuracies in regression and classification tasks on several molecular datasets. However, the embedding capabilities of LRL and SLRL have not been evaluated on various data formats and embedding networks in scientific applications.

3 PRELIMINARIES: WASSERSTEIN DISTANCE

For a n-dimensional measurable space $\Omega \subseteq \mathbb{R}^n$ and a set of probability measures Π on $\Omega \times \Omega$, Wasserstein distance is defined by an optimization problem as:

$$W_p = \left(\inf_{\pi \in \Pi} \int_{\Omega \times \Omega} \|\mathbf{x} - \mathbf{y}\|_2^p \pi(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} \right)^{\frac{1}{p}}, \tag{2}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ are data instances, and $\pi \in \Pi$ is a joint probability. By solving the above optimization problem with respect to π, we can calculate the distance between the data distributions of $\mathbf{x}$ and $\mathbf{y}$. When $p = 1$, 1-Wasserstein distance is also known as earth mover's distance (EMD) for optimal transportation problem [29]. Besides in machine learning, Wasserstein distance has been widely used to calculate the divergence between two unknown and nonparametric data distributions [3, 44]. In this study, we adopt Wasserstein distance to measure the distance consistency between the input and target spaces in scientific applications because most scientific data is generated from unknown mixture distributions [27, 28].

4 METHOD

4.1 Problem Definition

Our goal is to develop an embedding method that matches data distributions in the embedding and target spaces for improving the extrapolation capabilities of the prediction models. Although Wasserstein distance has been successfully applied to measure the divergences of two data distributions in various machine learning applications, a direct integration of Wasserstein distance with our embedding problem is not feasible because it assumes the same dimensionality of two data spaces. To devise a generally-applicable Wasserstein embedding loss regardless of the data dimensionality, first, we define a distance distribution that always has 1-dimensionality for all regression problems as follows.

Definition 4.1. (informal) For a n-dimensional space $\mathcal{X} \subseteq \mathbb{R}^n$, distance distribution $\mathcal{K}$ is defined as a probability distribution of pairwise distances $d(\mathbf{x}, \mathbf{x}')$ for all $(\mathbf{x}, \mathbf{x}') \in \mathcal{X} \times \mathcal{X}$, where $d : \mathcal{X} \times \mathcal{X} \rightarrow [0, \infty)$ is a distance metric.

For the distance distributions $\mathcal{K}$, Wasserstein distance is always defined in 1-dimensional measurable space $\mathcal{M} \subseteq \mathbb{R}$ regardless of the dimensionalities of the input and target data. For the input data $\mathbf{x} \in \mathcal{X}$ and the target data $\mathbf{y} \in \mathcal{Y}$, Wasserstein distance between the distance distributions is given by:

$$W_p(\mathcal{K}_x, \mathcal{K}_y; \pi) = \left(\inf_{\pi \in \Pi} \int_{\mathcal{M} \times \mathcal{M}} \|v - u\|_2^p \pi(u, v) \mathrm{d}u\mathrm{d}v \right)^{\frac{1}{p}}, \tag{3}$$

where $v = d(\mathbf{x}, \mathbf{x}')$ is the pairwise distance for $(\mathbf{x}, \mathbf{x}') \in \mathcal{X} \times \mathcal{X}$, $u = d(\mathbf{y}, \mathbf{y}')$ is the pairwise distance for $(\mathbf{y}, \mathbf{y}') \in \mathcal{Y} \times \mathcal{Y}$, and $\pi(v, u)$ is a joint probability of u and v. Note that $\mathcal{K}_x$ and $\mathcal{K}_y$ are the distance distributions of the input and target data, respectively.

From the empirical analysis in Fig. 3, the embedding objective of ANE to simplify the input-to-target relationships is defined as the 1-Wasserstein distance between the distance distributions in the embedding and target spaces as:

$$W_1(\mathcal{K}_x, \mathcal{K}_y; \pi, \boldsymbol{\theta}) = \inf_{\pi \in \Pi} \int_{\mathcal{M} \times \mathcal{M}} \|r - u\|_2 \pi(r, u) \mathrm{d}r\mathrm{d}v, \tag{4}$$

where $r = d(\phi(\mathbf{x}; \boldsymbol{\theta}), \phi(\mathbf{x}'; \boldsymbol{\theta}))$ is the pairwise distance of the input data in the embedding space, and $\phi : \mathcal{X} \rightarrow \mathcal{Z}$ is an embedding function parameterized by $\boldsymbol{\theta}$. The experimental results in Appendix 6 will empirically demonstrate that an embedding network to simplify (especially linearize) the input-to-target relationships can be generated by minimizing the above loss function.

4.2 Optimization

To obtain the optimal embedding network $\phi(\cdot; \boldsymbol{\theta}^*)$, we need to optimize $\boldsymbol{\theta}$ and $W_1(\mathcal{K}_x, \mathcal{K}_y; \pi, \boldsymbol{\theta})$ simultaneously. However, since the joint optimization of $\boldsymbol{\theta}$ and $W_1(\mathcal{K}_x, \mathcal{K}_y; \pi, \boldsymbol{\theta})$ requires large computational costs from the backpropagation and joint probability

calculation, we employ alternative optimization methods designed for efficient optimization of multiple primal variables [8, 11] in order to optimize the model parameters of the embedding network in ANE. Based on the alternative optimization technique, we optimize $\boldsymbol{\theta}$ and $W_1(\mathcal{K}_x, \mathcal{K}_y; \pi, \boldsymbol{\theta})$ alternatively as: $W_1(\mathcal{K}_x, \mathcal{K}_y; \pi, \boldsymbol{\theta})$ is minimized over the fixed $\boldsymbol{\theta}$, and then $\boldsymbol{\theta}$ is optimized for a fixed $W_1(\mathcal{K}_x, \mathcal{K}_y; \pi, \boldsymbol{\theta})$. In the training of ANE, we first estimate the optimal joint probability π^* for an optimal embedding network $\phi(\cdot; \boldsymbol{\theta}^*)$ in which case $r = v$ for all data pairs. Then, we optimize $\boldsymbol{\theta}$ for the estimated optimal joint probability π^*.

To estimate the optimal joint probability, we introduce Lagrangian of the Wasserstein distance in Eq. (4). For a given dataset $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$ containing independent and identically distributed (i.i.d.) N samples, we define the embedding distance $r_{ij} = d(\phi(\mathbf{x}_i; \boldsymbol{\theta}), \phi(\mathbf{x}_j; \boldsymbol{\theta}))$ and target distance $u_{ij} = d(\mathbf{y}_i, \mathbf{y}_j)$ for a pair of data $((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \mathbf{y}_j))$. Lagrangian of $W_1(\mathcal{K}_x, \mathcal{K}_y; \boldsymbol{\theta})$ on $\mathcal{D}$ is:

$$
\begin{aligned}
L_W = & \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}} \|r_{ij} - u_{kq}\|_2 \pi(r_{ij}, u_{kq}) \\
& + \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in\mathcal{N}} \pi(r_{ij}, u_{kq}) \right) f(r_{ij}) \\
& + \sum_{(i,j)\in\mathcal{N}} \left(p(u_{ij}) - \sum_{(k,q)\in\mathcal{N}} \pi(r_{kq}, u_{ij}) \right) g(u_{ij}), \qquad (5)
\end{aligned}
$$

where $\mathcal{N} = \{(i, j) \mid \text{for all } i, j \in \{1, 2, ..., N\}\}$ is a set of all possible index pairs in $\mathcal{D}$. To prevent the divergence of L_W to $-\infty$, a constraint $f(r_{ij}) + g(u_{kq}) \leq \|r_{ij} - u_{kq}\|_2$ for all r_{ij} and u_{kq} is introduced for the Lagrangian multipliers f and g. This constraint is referred to as well-known 1-Lipschitz constraint in the dual form of Wasserstein distance [3].

4.2.1 Decomposition of Lagrangian. We define $\mathcal{I}_{ij} = \{(k, q) \mid u_{ij} = u_{kq} \text{ for} (k, q) \in \mathcal{N}\}$, which is a set of index pairs with the same target distances. To estimate the optimal joint probability, we decompose L_W based on the index pairs in $\mathcal{I}_{ij}$ and $\mathcal{N} \setminus \mathcal{I}_{ij}$ as follows. Note that a detailed derivation for the decomposition of the Lagrangian is provided in Appendix 2.

$$
\begin{aligned}
L_W = & \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus\mathcal{I}_{ij}} \left(\|r_{ij} - u_{kq}\|_2 - f(r_{ij}) - g(u_{kq}) \right) \pi(r_{ij}, u_{kq}) \\
& + \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{I}_{ij}} \|r_{ij} - u_{kq}\|_2 \pi(r_{ij}, u_{kq}) \\
& + \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in\mathcal{I}_{ij}} \pi(r_{ij}, u_{kq}) \right) f(r_{ij}) \\
& + \sum_{(i,j)\in\mathcal{N}} \left(p(u_{ij}) - \sum_{(k,q)\in\mathcal{N}} \pi(r_{kq}, u_{ij}) \right) g(u_{ij}) \\
& + \sum_{(k,q)\in\mathcal{N}} \sum_{(i,j)\in\mathcal{N}\setminus\mathcal{I}_{kq}} \pi(r_{kq}, u_{ij}) g(u_{ij}). \qquad (6)
\end{aligned}
$$

4.2.2 Estimation of Optimal Joint Probability. For an optimal embedding function $\phi(\cdot; \boldsymbol{\theta}^*)$ where $r_{ij} = u_{ij}$ for all $(i, j) \in \mathcal{N}$, one possible joint probability that globally minimizes $W_1(\mathcal{K}_x, \mathcal{K}_y; \boldsymbol{\theta})$ is given as follows.

- For the first term, the best choice of the joint probability is to set $\pi(r_{ij}, u_{kq}) = 0$ for all $(i, j) \in \mathcal{N}$ and $(k, q) \in \mathcal{N} \setminus \mathcal{I}_{ij}$ because $\|r_{ij} - u_{kq}\|_2 - f(r_i j) - g(u_{kq}) \geq 0$ for all r_{ij} and u_{kq} from the constraint in Lagrangian multipliers.
- For the second term, although $\pi(r_{ij}, u_{kq})$ is not explicitly determined for $(i, j) \in \mathcal{N}$ and $(k, q) \in \mathcal{I}_{ij}$, it always zero by $r_{ij} = u_{kq}$ under $\phi(\cdot; \boldsymbol{\theta}^*)$.
- The values of the remaining terms are always zero by $p(r_{ij}) = \sum_{(k,q)\in\mathcal{I}_{ij}} \pi(r_{ij}, u_{kq})$, $p(u_{ij}) = \sum_{(k,q)\in\mathcal{I}_{ij}} \pi(r_{kq}, u_{ij})$, and $\pi(r_{kq}, u_{ij}) = 0$ for all $(k, q) \in \mathcal{N}$ and $(i, j) \in \mathcal{N} \setminus \mathcal{I}_{kq}$.

Therefore, one possible optimal joint probability π^* to minimize $W_1(\mathcal{K}_x, \mathcal{K}_y; \boldsymbol{\theta})$ is given as $\pi(r_{ij}, u_{kq}) = 0$ for all $(i, j) \in \mathcal{N}$ and $(k, q) \in \mathcal{N} \setminus \mathcal{I}_{ij}$ but $\sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{I}_{ij}} \pi(r_{ij}, u_{kq}) = 1$. As presented above, π^* leads $W_1(\mathcal{K}_x, \mathcal{K}_y; \boldsymbol{\theta})$ to zero, which is the global minimum of Wasserstein distance. In the training of the embedding network in ANE, we will optimize the model parameters $\boldsymbol{\theta}$ under the optimal joint probability π^*.

4.2.3 Model Parameter Optimization. In the decomposed Lagrangian in Eq. (6), we showed that our optimal joint probability $\pi^*(r_{ij}, u_{kq})$ globally minimizes $W_1(\mathcal{K}_x, \mathcal{K}_y; \boldsymbol{\theta})$ under the optimal embedding network $\phi(\cdot; \boldsymbol{\theta}^*)$ where $r_{ij} = u_{ij}$ for all $(i, j) \in \mathcal{N}$. From the i.i.d condition on $\mathcal{D}$, $\pi(r_{ij}, u_{ij}) = \pi(r_{kq}, u_{kq})$ for all $\{(i, j), (k, q)\} \in \mathcal{N} \times \mathcal{N}$. Thus, the embedding problem of ANE can be reduced as:

$$
\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sum_{j=1}^{N} |\mathcal{I}_{ij}| \|r_{ij} - u_{ij}\|_2 \pi^*_{ij}, \qquad (7)
$$

where the simplified notation $\pi_{ij} = \pi(r_{ij}, u_{ij})$ indicates the joint probability of the same index pairs.

In the objective function of Eq. (7), the joint probability π_{ij} can be empirically estimated by the i.i.d. condition as:

$$
\pi^*(r_{ij}, u_{kq}) = \frac{1}{\sum_{l=1}^{N} \sum_{m=1}^{N} |\mathcal{I}_{lm}|}. \qquad (8)
$$

From the empirical joint probability, the coefficient $|\mathcal{I}_{ij}|\pi^*_{ij}$ are reduced to constant by $|\mathcal{I}_{ij}| \ll \sum_{l=1}^{N} \sum_{m=1}^{N} |\mathcal{I}_{lm}|$ for all $(i, j) \in \mathcal{N}$ in the regression setting. Finally, the embedding problem of ANE under the optimal joint probability is given by:

$$
\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sum_{j=1}^{N} \|r_{ij} - u_{ij}\|_2. \qquad (9)
$$

The goal of ANE is to build an embedding network $\phi(\cdot; \boldsymbol{\theta})$ by solving the optimization problem in Eq. (9) to improve the distance consistency of the given dataset $\mathcal{D}$ for supporting machine learning extrapolation over $\mathcal{D}$.

4.2.4 Sampling Strategies in Optimization. The objective function in Eq. (9) requires $O(N^2)$ time complexity over $\mathcal{D}$ for each epoch in the training. To remove the quadratic time complexity of ANE, we can employ the random sampling in practical implementation of ANE. In Section 5.3.2, we will evaluate the embedding performances of ANE for three different sampling methods to validate that the efficient random sampling is sufficient for ANE.

Algorithm 1 describes the overall training process of ANE. After the training of the embedding network $\phi(\cdot; \boldsymbol{\theta})$ with Algorithm 1, a prediction model $f(\phi(\cdot; \boldsymbol{\theta}^*); \boldsymbol{\mu})$ is trained to predict the target values $\mathbf{y}$ for the input embeddings from the trained embedding network, where $\boldsymbol{\mu}$ is model parameters of the prediction model.

Algorithm 1: Training of ANE-based prediction model

Input : Training dataset $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$; Embedding network $\phi(\mathbf{x}; \boldsymbol{\theta})$; Prediction model $f(\phi(\mathbf{x}; \boldsymbol{\theta}); \boldsymbol{\mu})$; Sampling method $\psi(\mathbf{x}; \mathcal{D})$; Distance metric d

repeat
  for $i = 1;\ i < N;\ i++$ **do**
    $s = \psi(\mathbf{x}_i; \mathcal{D})$ // List of indices of the samples.
    for $j = 1;\ j < |s|;\ j++$ **do**
      $r_{ij} = d(\phi(\mathbf{x}_i; \boldsymbol{\theta}), \phi(\mathbf{x}_{s_j}; \boldsymbol{\theta}))$ and $u_{ij} = d(\mathbf{y}_i, \mathbf{y}_{s_j})$
      $L_W += ||r_{ij} - u_{ij}||_2$
    end
  end
  Optimize $\boldsymbol{\theta}$ with respect to L_W.
until $\boldsymbol{\theta}$ converged;
Optimize $\boldsymbol{\mu}$ on $\mathcal{Z} = \{(\phi(\mathbf{x}_1; \boldsymbol{\theta}^*), \mathbf{y}_1), ..., (\phi(\mathbf{x}_N; \boldsymbol{\theta}^*), \mathbf{y}_N)\}$.
Return $\phi(\mathbf{x}; \boldsymbol{\theta}^*)$ and $f(\phi(\mathbf{x}; \boldsymbol{\theta}^*); \boldsymbol{\mu}^*)$

4.3 Time Complexity

The time complexity of ANE basically follows the time complexity of DML. As shown in Algorithm 1, the time complexity of the training of ANE is given by $O(IN|s|\kappa)$, where I is the number of epochs, N is the number of data instances in $\mathcal{D}$, $|s|$ is the number of samples for each data instance, and κ is the time complexities of forward and backward steps of the embedding networks. However, we can claim that ANE is sufficiently efficient because widely used DML methods have quadratic or cubic time complexities for the number of data [7, 34, 36]. Besides, the time complexity $O(IN|s|\kappa)$ of ANE is comparable to the time complexity $O(INB\kappa)$ of LRL [18] and SLRL [27] because $|s|$ is usually less than the batch size B.

5 EXPERIMENT

To evaluate ANE-based models in a comprehensive manner, we measured their extrapolation errors on regression problems in various domains with various data formats: 1) Matrix data for N-body problem Sec. 5.1, 2) Graph-structured data for materials property prediction Sec. 5.2, 3) Time-series data for geomagnetic storm forecasting Sec. 5.3. All neural networks were trained by Adam optimizer [19], and the training hyperparameters were determined by a grid search. The initial learning rate and the batch size were selected in {1e-4, 5e-4, 1e-3, 5e-3, 1e-2, 5e-2} and {16, 24, 32, 48, 64}, respectively. The L_2 regularization coefficient is also determined by the grid search in {1e-7, 5e-7, 1e-6, 5e-6, 1e-5}. The selected hyperparameters for each method are provided in Appendix 3.

For the DML methods, we selected appropriate embedding network $\phi(\cdot; \boldsymbol{\theta})$ for each data format. However, we used FCNNs as the prediction models $f(\cdot; \boldsymbol{\mu})$ of the DML methods for all experiments because the input data is always converted into the vector-shaped embeddings by the embedding networks. We repeated all the experiments five times and reported the averages of the extrapolation performances with standard deviations. The extrapolation performances were measured with appropriate metrics for each scientific application, such as correlation distance (Corr), R^2 score (R^2), and mean absolute error (MAE).

Table 1: Extrapolation errors for simulated n-body problem datasets. Idx. denotes the dataset index. The prediction error is measured by distance correlation (Corr) between the simulated and predicted velocities of the particles

Idx.	NBNet	GIN	MPNN	UMP	LRL-F	SLRL-F	ANE-F
1	0.32	0.54	0.35	0.25	0.43	0.53	**0.18**
2	0.49	0.54	0.53	**0.36**	0.52	0.49	0.45
3	0.57	0.54	0.53	0.46	0.52	0.59	**0.29**
4	0.25	0.68	0.26	0.26	0.09	0.07	**0.03**
5	0.66	0.93	0.71	0.69	0.85	0.65	**0.49**
6	**0.11**	0.22	0.17	0.16	0.12	0.12	0.02
7	0.75	0.94	0.63	0.67	0.61	0.44	**0.40**
8	0.44	0.85	0.26	0.29	0.27	0.38	**0.15**
9	0.39	0.26	0.10	0.70	0.18	0.40	**0.03**
10	0.64	0.72	0.55	0.54	0.53	0.37	**0.27**
mean	0.46	0.62	0.41	0.44	0.41	0.40	**0.23**
±std.	±0.19	±0.24	±0.20	±0.19	±0.23	±0.18	±**0.17**

5.1 Extrapolation on Matrix-Shaped Data: n-Body Problem

The purpose of the n-body problem [31] is to estimate future trajectories or velocities of n particles in a physical system. Each particle has mass, initial position, and initial velocities. The n-body is common and essential in many scientific applications in physics [31] and astronomy [30]. We predict the velocities of the n particles at time $t+1$ for given positions and velocities of the particles at time t, Hence, the input data at time t is a matrix $\mathbf{X}_t \in \mathbb{R}^{n\times(2d+1)}$ containing the positions and the velocities in d-dimensional space, and the mass of the particle. The target data is also a matrix $\mathbf{y}_t \in \mathbb{R}^{n\times d}$ containing the velocities of the particles at time $t+1$. In this experiment, we measured the extrapolation errors of ANE in a common 3-dimensional 3-body problem. Thus, the input and target data are the matrices $\mathbf{x}_t \in \mathbb{R}^{3\times 7}$ and $\mathbf{y}_t \in \mathbb{R}^{3\times 3}$, respectively.

We followed the computational n-body simulation of the state-of-the-art method in machine learning extrapolation [46] to generate the n-body problem datasets. In the simulation, the gravitational constant and the time step were fixed to 1 and 0.01, respectively. Total 10 datasets were generated with randomly initialized positions and velocities of the particles.To evaluate the extrapolation accuracies, we trained prediction models for the observations at the time [0, 80] and predicted the velocities using the trained models in the future time (80, 100]. In the experiment, we implemented ANE-F by employing FCNNs as the embedding networks $\phi(\cdot; \boldsymbol{\theta})$ and the prediction model $f(\phi(\cdot; \boldsymbol{\theta}^*); \boldsymbol{\mu})$ of ANE. To evaluate the extrapolation capabilities of ANE-F, we compared with various machine learning methods from three approaches as follows.

- **Direct prediction method:** As a baseline method, the extrapolation errors of NBNet [31] were compared in the experiment. NBNet is a FCNN-based network to predict the velocities of the particles from their current positions and velocities.
- **Graph neural networks (GNNs):** GNNs have achieved state-of-the-art extrapolation performances in the n-body problem by representing the particles and their interactions as the nodes and the edges in the mathematical graphs, respectively [46]. We compared the extrapolation errors of four GNNs: graph convolutional network (GCN) [20], graph isomorphism network (GIN) [45], message passing neural

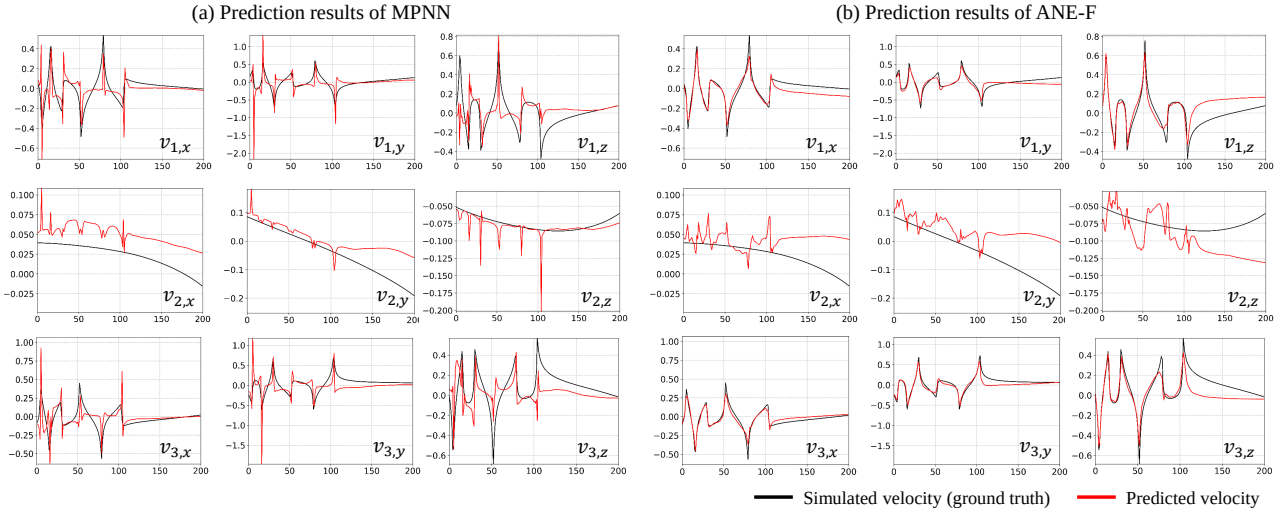

Figure 4: Prediction results of the n-body problem where the velocities of the particles were most dynamically changed. In this case, the best and the second-best models were ANE-F and MPNN. (a): Prediction results of MPNN for each coordinate of the particle. (b): Prediction results of ANE-F. In the figures, the velocity of the i^{th} particle is denoted by $v_{i,r}$ where r indicates x, y, and z coordinates. X and Y axes indicate future time and velocity, respectively.

network (MPNN) [10], and transformer-based graph neural network (UMP) [35].

- **Metric learning methods:** Although DML is hardly applied to machine learning extrapolation, LRL [18] and SLRL [27], both of which are metric learning methods on continuous target values, can be used to simplify the input-to-target relationships. We generated two prediction models, LRL-F and SLRL-F, by employing FCNNs as the embedding networks and the prediction models.

The extrapolation errors were calculated by distance correlation (Corr) [37] between the simulated and predicted velocities to measure how accurately the machine learning methods predicted future trends of the velocities. Note that we did not use MAE in this experiment because a few outliers in the prediction dominates the entire prediction errors in MAE. Table 1 shows the extrapolation errors of the machine learning methods for each n-body dataset. By comparing NBNet and GNNs, we observe that the extrapolation errors were generally reduced by employing graph-based representation rather than the primitive matrix representation of the n-body problem. Similarly, the metric learning methods were as effective as GNNs in the extrapolation of the n-body problem because they automatically simplify the input-to-target relationships for a given dataset. In particular, the arithmetic mean of Corr in ANE-F was the smallest among the errors of all the competitors including GNNs and metric learning methods, i.e., ANE generated the input representations that were the most effective to reduce the extrapolation errors in the n-body problem.

Fig. 4 shows the predicted velocities of the particles for each axis. We presented the predicted velocities for the n-body dataset of index 8 in which the trajectories of the particles changed most dynamically in the future time (80, 100]. In this case, the best method was ANE-F, and the extrapolation error measured by Corr was 0.15. The second best method was MPNN, and its Corr was 0.26. One of the most remarkable improvements by ANE-F is that sudden explosions in the predicted target values were significantly removed when compared to the extrapolation results of MPNN.

Table 2: Extrapolation accuracies for each target property on MPS datasets. N/A means a negative R^2 score indicating a failure of extrapolation.

Method	Formation Energy	Band Gap	Shear Modulus	Bulk Modulus
GCN	0.662 (±0.019)	0.254 (±0.071)	0.526 (±0.025)	0.574 (±0.037)
MPNN	0.072 (±0.052)	N/A	0.352 (±0.344)	0.714 (±0.007)
CGCNN	N/A	0.163 (±0.424)	0.405 (±0.441)	0.732 (±0.011)
UMP	0.763 (±0.042)	0.351 (±0.069)	0.552 (±0.003)	0.707 (±0.022)
LRL-MPNN	0.819 (±0.024)	0.259 (±0.034)	0.704 (±0.009)	0.769 (±0.021)
SLRL-MPNN	0.841 (±0.018)	0.396 (±0.052)	0.693 (±0.013)	0.767 (±0.007)
ANE-MPNN	**0.879 (±0.017)**	**0.447 (±0.055)**	**0.716 (±0.015)**	**0.790 (±0.011)**

5.2 Extrapolation on Graph-Structured Data: Materials Property Prediction

The physical properties of the materials essentially determine the performances and efficiencies of various industrial applications, such as semiconductors and electronic sensors. For this reason, discovering novel materials of desired properties has a significant impact in various engineering fields. The materials properties are determined by the crystal structures that is a unique identifier of the materials [14, 43]. Hence, an accurate extrapolation of unseen crystal structures is crucial in discovering novel materials [24].

In this experiment, we evaluated the extrapolation accuracies of ANE on MPS dataset [43] that contains 3,162 crystal structures from

Table 3: Extrapolation errors and detection accuracies of geomagnetic storms on the MNN-A dataset.

Method	Extrapolation Error		Detection Accuracy		
	MAE	Corr	Precision	Recall	F1-score
RNN	16.089 (±0.806)	0.710 (±0.025)	0.133 (±0.013)	0.281 (±0.065)	0.178 (±0.015)
LSTM	14.721 (±0.702)	0.696 (±0.065)	0.164 (±0.048)	0.260 (±0.087)	0.201 (±0.062)
GRU	14.613 (±0.368)	0.687 (±0.027)	0.145 (±0.027)	0.230 (±0.055)	0.177 (±0.034)
TF	13.106 (±0.717)	0.670 (±0.031)	0.185 (±0.115)	0.145 (±0.074)	0.159 (±0.084)
LRL-GRU	13.700 (±0.581)	0.499 (±0.031)	0.189 (±0.035)	0.519 (±0.186)	0.272 (±0.054)
SLRL-GRU	10.986 (±0.332)	0.455 (±0.040)	0.260 (±0.065)	0.336 (±0.111)	0.291 (±0.077)
ANE-GRU	**10.534 (±0.407)**	**0.428 (±0.041)**	**0.513 (±0.044)**	**0.495 (±0.071)**	**0.502 (±0.042)**

general applications. Implementation details to convert the crystal structures into the attributed graphs are provided in Appendix 4. To make a complete extrapolation problem, we trained machine learning models on the crystal structures containing only two types of elements (binary materials). Then, we predict the materials properties of the crystal structures containing three or four types of elements (ternary and quaternary materials, respectively). That is, we evaluated the machine learning models in an extrapolation problem of how the models well infer the complex structures (ternary and quaternary materials) from relatively simple structures (binary materials). In this experiment, we used MPNNs and FCNNs as the embedding networks and prediction models of the DML methods, respectively. Table 2 presents R^2 scores of the machine learning methods in the extrapolaiton to predict four materials properties. For all target materials properties, ANE-MPNN achieved the highest R^2 score. From the theoretical analysis of machine learning extrapolation [46], the improvements by ANE in the extrapolation can be justified by the fact that ANE generates more linear input-to-target relationships. Its empirical demonstrations are in Appendix 6.

5.3 Extrapolation on Time-Series Data: Geomagnetic Storm Forecasting

We predicted the magnetic field of Earth to evaluate the extrapolation performances of ANE on time-series data. For the evaluation, we used MagNet NASA (MNN) dataset [1] containing input multivariate time-series and target disturbance storm time (Dst) observation. The data was sampled for each observation for an hour from the original dataset. To avoid including seasonal patterns that appear in the training dataset, we sampled only one year data to make both training and test datasets. In this way, we can be ascertain that the test datsets contain unseen time-series patterns, which makes the prediction an extrapolation problem. According to the provided categories in the original dataset, we generated three sub-datasets MNN-A, -B, and -C each containing 8,762 data instances with 14 input features. 80% of data instances were used to train the prediction models, and 20% of the data instances were used to evaluate the extrapolation performances. That is, the data instances at the time [1, 7009] were used for the training, and the data instances in the future time [7010, 8762] were used for extrapolation.

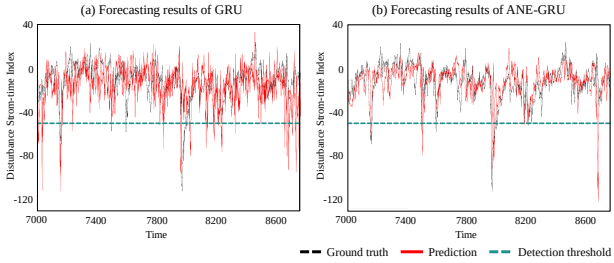

Figure 5: Forecasting results on the MNN-A dataset. (a): Forecasting results of GRU. (b): Forecasting results of ANE-GRU.

We predicted Dst from the time-series data using naive recurrent neural network (RNN) [12], long short-term memory (LSTM) [12], gated recurrent unit network (GRU) [6], and transformer network (TF) [39]. For the evaluation, we generated prediction models LRL-GRU, SLRL-GRU, and ANE-GRU, by employing GRUs and FCNNs as embedding networks and prediction models, respectively. Also, we used FCNNs as the prediction models of LRL-GRU, SLRL-GRU, and ANE-GRU. After the Dst prediction, we forecasted the geomagnetic storms based on the predicted Dst values to evaluate the extrapolation performances of ANE in real-world scientific applications. The experiment settings and the evaluation results on the MNN-B and MNN-C datasets are provided in Appendix 5.

5.3.1 Dst Prediction. The extrapolation errors were measured by MAE and Corr between the ground truth and predicted sequences of Dst. Table 3 shows the extrapolation errors on MNN-A dataset. In the experiment, the predicted sequence of Dst from ANE-GRU showed the smallest MAE and Corr for the ground truth. Fig. 5-(a) and -(b) show the Dst values predicted by GRU and ANE-GRU, respectively. In predicting future Dst, the predicted values of GRU severely fluctuated and the predicted Dst values were not reliable. On the other hand, the fluctuations in the predicted values were significantly removed in ANE-GRU, and the predicted sequence roughly followed the patterns of the ground truth.

5.3.2 Geomagnetic Storm Detection. From the predicted sequences of Dst, we detected the geomagnetic storm with a threshold of Dst in future time. The detection threshold was set to -50 based on domain knowledge to detect moderate, intense, and super geomagnetic storms. That is, geomagnetic storm detection can be regarded as a challenging task of detecting outliers in the extrapolation problems. Table 3 shows precision, recall, and f1-score of the detection results. In geomagnetic storm detection, ANE-GRU outperformed all competitors, and the F1-score was greatly improved from the previous maximum of 0.291 to 0.502. The detection threshold is also presented in Fig. 5 as a green dotted line, and the false positives were significantly removed in the prediction results of ANE-GRU.

5.4 ANE for Discovering Solar Cell Materials

Perovskite [17] has received significant attention as solar cell materials for renewable energy to overcome the global climate crisis [5]. HOIP dataset [17] provides 1,345 crystal structures of the perovskites with their band gaps, which roughly determine the applications of the perovskite solar cells. One of the most challenging points in the real-world materials discovery is that we should

Table 4: R^2 scores of the GNN and DML methods in predicting band gaps of the perovskite materials.

	Method	Dataset	
		HOIP-HIGH	HOIP-LOW
GNN methods	GCN	0.213(±0.162)	N/A
	MPNN	N/A	N/A
	CGCNN	N/A	N/A
	UMP	N/A	N/A
DML methods	LRL-MPNN	N/A	0.521(±0.131)
	SLRL-MPNN	0.182(±0.160)	0.486(±0.096)
	ANE-MPNN	**0.558(±0.044)**	**0.664±(0.071)**

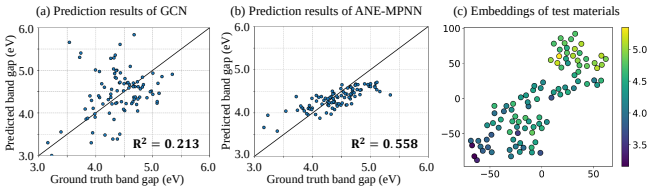

Figure 6: Prediction and embedding results in predicting materials properties on HOIP-HIGH dataset. (a): Prediction results of the best competitor GCN. (b): Prediction results of ANE-MPNN. (c): Embedding results of ANE on the test dataset. Each point is a crystal structure, and the colors indicate the band gaps of the crystal structures.

infer the materials properties of the crystal structures containing unseen elemental combinations. To make an extrapolation problem like the real-world materials discovery, we generated two datasets from HOIP dataset, called HOIP-LOW and HOIP-HIGH, each of which is generated by eliminating crystal structures containing certain elements. Specifically, HOIP-HIGH containing 1,248 and 97 training and test crystal structures is generated by eliminating the crystal structures containing germanium (Ge) and fluorine (F). Similarly, HOIP-LOW containing 1,228 and 117 training and test crystal structures is generated by eliminating the crystal structures containing lead (Pb) and iodine (I) from the original HOIP dataset. Furthermore, the band gaps of the test crystal structures are in completely different ranges compared with the band gaps of the training crystal structures. Therefore, the experiment settings completely cover the real-world applications for discovering novel solar cell materials.

In the experiments on the HOIP-HIGH and HOIP-LOW datasets, ANE-MPNN outperformed all competitors as shown in Table 4. In particular, the best competitor GCN was could not capture the relationships between the crystal structures and their band gaps, whereas ANE-MPNN roughly captured the relationships as shown in Fig. 6-(a) and -(b). Also, ANE-MPNN generated an embedding space where the test crystal structures are roughly arranged according to their band gaps in Fig. 6-(c), which in turn improve the extrapolation capabilities of machine learning methods [46].

With the improvements by ANE in extrapolation, we conducted a high-throughput screening based on ANE as a real-world application to discover novel solar cell materials of desired band gaps. To quantitatively present the screening accuracy, we define a hit rate

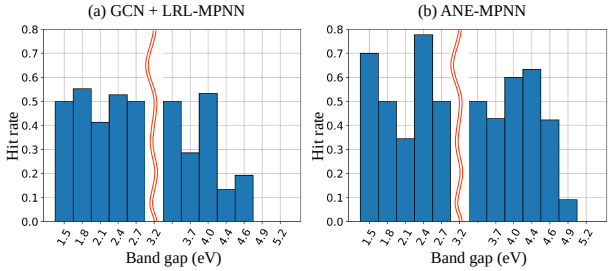

Figure 7: High-throughput screening results to discover novel solar cell materials. (a): Hit rates of the best competitors GCN and LRL-MPNN. (b): Hit rates of ANE-MPNN.

$\mathcal{H}(a,b)$ for a range $[a,b)$ as $\mathcal{H}(a,b) = \frac{\text{\# of predicted data in } [a,b)}{\text{\# of ground truth data in } [a,b)}$. We compared the hit rates of ANE-MPNN with the hit rates of GCN and LRL-MPNN that were the best competitors on HOIP-HIGH and -LOW, respectively. As shown in Fig. 7, the hit rates were generally improved by ANE and higher than 40% for most ranges. These high-throughput screening results show the usefulness of ANE in virtual discovery of novel solar cell materials.

5.5 Sampling Strategies and Extrapolation

To avoid the $O(N^2)$ time complexity over the datasets in the training of ANE, an appropriate sampling strategy is necessary in ANE. In this experiment, we evaluate the extrapolation accuracies of ANE for three efficient sampling methods as follows.

- **Random sampling:** A data sample is randomly selected in a mini-batch. The random sampling is a useful and efficient method to sample the data based on density of the data distribution.
- **k-NN sampling:** k data samples are selected in a mini-batch based on the target distance from the anchor data. The k-NN sampling can be used to sample data for approximating the local densities around the anchor data.
- **Hardness sampling:** k data samples of largest $||r_{ij} - u_{ij}||_2$ for an anchor data $\mathbf{x}_i$ are sampled in a mini-batch, i.e., top k data samples of the largest errors are selected for each anchor data in a mini batch.

For the evaluation, we predicted the materials properties on the MPS dataset using ANE-MPNN with the three different sampling methods. Fig. 8 presents the extrapolation accuracies for each sampling method, and the random sampling was the best sampling strategy despite its simplicity. The superiority of the random sampling can be explained by the objective function of ANE, whose goal is to reduce Wasserstein distance of the embedding and target distance distributions. Therefore, a sampling method to preserve the data distribution is necessary for ANE, and the random sampling that is a well-known density-based sampling [26] satisfies this requirement of ANE for estimating the data distributions.

6 CONCLUSION

This paper proposed a novel and generally applicable nonlinearity encoder (ANE) for improving machine learning extrapolation. ANE minimizes the Wasserstein distance between the pairwise distances of the data samples in the input and the target spaces to remove

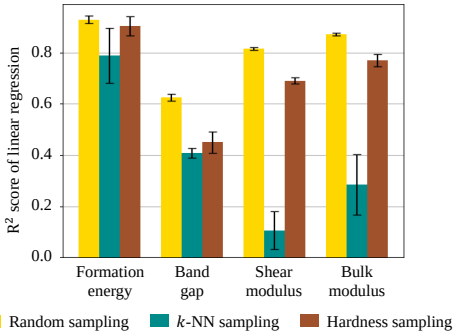

Figure 8: Extrapolation accuracies of ANE with three different sampling methods on MPS dataset.

nonlinearity in the prediction problem by encoding it into the data representation. ANE outperformed state-of-the-art deep neural networks and deep metric learning methods in the experiments on various scientific applications and data formats. ANE paved the way for machine learning extrapolation to discover new scientific knowledge.

ACKNOWLEDGEMENTS

This study was supported by Korea Research Institute of Chemical Technology (No.SI2151-10, 50%) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00157, 50%).

REFERENCES

[1] NASA and NOAA satellites solar-wind dataset. https://www.kaggle.com/arashnic/soalr-wind. Accessed: 202-01-22.
[2] The mnist database. http://yann.lecun.com/exdb/mnist/, 2022 (accessed January 21, 2022).
[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.
[4] Richard Strong Bowen, Huiwen Chang, Charles Herrmann, Piotr Teterwak, Ce Liu, and Ramin Zabih. Oconet: Image extrapolation by object completion. In *CVPR*, 2021.
[5] Karsten Bruening, Benjia Dou, John Simonaitis, Yu-Ying Lin, Maikel FAM van Hest, and Christopher John Tassone. Scalable fabrication of perovskite solar cells to meet climate targets. *Joule*, 2018.
[6] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
[7] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.
[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. Series B Stat. Methodol.*, 1977.
[9] Stephanie deWet and Jiafan Ou. Finding users who act alike: Transfer learning for expanding advertiser audiences. In *KDD*, 2019.
[10] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.
[11] Magnus R. Hestenes. Multiplier and gradient methods. *J. Optim. Theory Appl.*, 1969.
[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 1997.
[13] Raban Iten, Tony Metger, Henrik Wilming, Lídia del Rio, and Renato Renner. Discovering physical concepts with neural networks. *Phys. Rev. Lett.*, 2020.
[14] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Mater.*, 2013.
[15] Dipendra Jha, Kamal Choudhary, Francesca Tavazza, Wei-keng Liao, Alok Choudhary, Carelyn Campbell, and Ankit Agrawal. Enhancing materials property prediction by leveraging computational and experimental data using deep transfer learning. *Nat. Commun.*, 2019.
[16] Mohammad Mahdi Kamani, Sadegh Farhang, Mehrdad Mahdavi, and James Z. Wang. Targeted data-driven regularization for out-of-distribution generalization. In *KDD*, 2020.
[17] Chiho Kim, Tran Doan Huan, Sridevi Krishnan, and Rampi Ramprasad. A hybrid organic-inorganic perovskite dataset. *Sci. Data*, May 2017.
[18] Sungyeon Kim, Minkyo Seo, Ivan Laptev, Minsu Cho, and Suha Kwak. Deep metric learning beyond binary supervision. In *CVPR*, June 2019.
[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
[20] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.
[21] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 1998.
[22] Colin Lewis-Beck and Michael Lewis-Beck. *Applied regression: An introduction*, volume 22. Sage publications, 2015.
[23] Yuxin Li, Wenhui Yang, Rongzhi Dong, and Jianjun Hu. Mlatticeabc: generic lattice constant prediction of crystal materials using machine learning. *ACS omega*, 6(17):11585–11594, 2021.
[24] Haotong Liang, Valentin Stanev, A. Gilad Kusne, and Ichiro Takeuchi. Cryspnet: Crystal structure predictions via neural networks. *Phys. Rev. Materials*, 2020.
[25] Sudhanshu Mittal, Silvio Galesso, and Thomas Brox. Essentials for class incremental learning. In *CVPR*, 2021.
[26] Gyoung S Na, Donghyun Kim, and Hwanjo Yu. Dilof: Effective and memory efficient local outlier detection in data streams. In *KDD*, 2018.
[27] Gyoung S Na, Hyunju Chang, and Hyun Woo Kim. Machine-guided representation for accurate graph-based molecular machine learning. *Phys. Chem. Chem Phys.*, 2020.
[28] Gyoung S. Na, Seunghun Jang, and Hyunju Chang. Predicting thermoelectric properties from chemical formula with explicitly identifying dopant effects. *Npj Comput. Mater.*, 7(1), 2021.
[29] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Found. Trends Mach. Learn.*, 2019.
[30] Wang Qiu-Dong. The global solution of the n-body problem. *Celest. Mech. Dyn. Astron.*, 1990.
[31] Marcelino Quito, Christopher Monterola, and Caesar Saloma. Solving *N*-body problems with neural networks. *Phys. Rev. Lett.*, 2001.
[32] Ibai Roman, Roberto Santana, Alexander Mendiburu, and Jose A. Lozano. Evolving gaussian process kernels from elementary mathematical expressions for time series extrapolation. *Neurocomputing*, 2021.
[33] Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *ICML*, 2018.
[34] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.
[35] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *IJCAI*, 2021.
[36] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*, 2016.
[37] Gábor J Székely, Maria L Rizzo, and Nail K Bakirov. Measuring and testing dependence by correlation of distances. *Ann. Stat.*, 2007.
[38] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, and Shanrong Zhao. Applications of machine learning in drug discovery and development. *Nat. Rev. Drug Discov.*, 2019.
[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
[40] Yi Wang, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Wide-context semantic image extrapolation. In *CVPR*, 2019.
[41] Taylor Webb, Zachary Dulberg, Steven Frankland, Alexander Petrov, Randall O'Reilly, and Jonathan Cohen. Learning representations that support extrapolation. In *ICML*, 2020.
[42] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015.
[43] Tian Xie and Jeffrey C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.*, 2018.
[44] Jie Xu, Lei Luo, Cheng Deng, and Heng Huang. Multi-level metric learning via smoothed wasserstein distance. In *IJCAI*, 2018.
[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *ICLR*, 2019.
[46] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon Shaolei Du, Ken-Ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *ICLR*, 2021.
[47] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Universal domain adaptation. In *CVPR*, June 2019.
[48] Wenzhao Zheng, Chengkun Wang, Jiwen Lu, and Jie Zhou. Deep compositional metric learning. In *CVPR*, 2021.

APPENDIX 1. CLASSIFICATION ON MP DATASET

MP dataset contains 45,941 crystal structures with their real-valued target properties called band gaps [43]. We made a classification problem on MP dataset by quantizing the range of the band gaps in [0, 16.586] into 10 classes. For example, the first and second classes contain the crystal structures of the band gaps in [0, 1.659) and [1.659, 3.317), respectively.

APPENDIX 2. DECOMPOSITION OF LAGRANGIAN

In Section 4.2.1, we introduced a decomposed Lagrangian L_W in Eq. (6) to estimate optimal joint probability of the loss function of ANE. A full derivation of the composed Lagrangian L_W is given by:

$$
\begin{aligned}
L_W &= \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}} ||r_{ij}-u_{kq}||_2 \pi(r_{ij},u_{kq}) + \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in\mathcal{N}} \pi(r_{ij},u_{kq}) \right) f(r_{ij}) + \sum_{(i,j)\in\mathcal{N}} \left(p(u_{ij}) - \sum_{(k,q)\in\mathcal{N}} \pi(r_{kq},u_{ij}) \right) g(u_{ij}) \\
&= \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus\mathcal{I}_{ij}} ||r_{ij}-u_{kq}||_2 \pi(r_{ij},u_{kq}) + \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{I}_{ij}} ||r_{ij}-u_{kq}||_2 \pi(r_{ij},u_{kq}) \\
&+ \sum_{(i,j)\in\mathcal{N}} p(r_{ij}) f(r_{ij}) - \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus\mathcal{I}_{ij}} \pi(r_{ij},u_{kq}) f(r_{ij}) - \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{I}_{ij}} \pi(r_{ij},u_{kq}) f(r_{ij}) \\
&+ \sum_{(i,j)\in\mathcal{N}} p(u_{ij}) g(u_{ij}) - \sum_{(k,q)\in\mathcal{N}} \sum_{(i,j)\in\mathcal{N}\setminus\mathcal{I}_{kq}} \pi(r_{kq},u_{ij}) g(u_{ij}) - \sum_{(k,q)\in\mathcal{N}} \sum_{(i,j)\in\mathcal{I}_{kq}} \pi(r_{kq},u_{ij}) g(u_{ij}) \\
&= \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus\mathcal{I}_{ij}} \left(||r_{ij}-u_{kq}||_2 - f(r_{ij}) - g(u_{kq}) \right) \pi(r_{ij},u_{kq}) + \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{I}_{ij}} ||r_{ij}-u_{kq}||_2 \pi(r_{ij},u_{kq}) + \sum_{(i,j)\in\mathcal{N}} p(u_{ij}) g(u_{ij}) \\
&+ \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in\mathcal{I}_{ij}} \pi(r_{ij},u_{kq}) \right) f(r_{ij}) - \sum_{(k,q)\in\mathcal{N}} \sum_{(i,j)\in\mathcal{N}} \pi(r_{kq},u_{ij}) g(u_{ij}) + \sum_{(k,q)\in\mathcal{N}} \sum_{(i,j)\in\mathcal{N}\setminus\mathcal{I}_{kq}} \pi(r_{kq},u_{ij}) g(u_{ij}) \\
&= \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus\mathcal{I}_{ij}} \left(||r_{ij}-u_{kq}||_2 - f(r_{ij}) - g(u_{kq}) \right) \pi(r_{ij},u_{kq}) + \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{I}_{ij}} ||r_{ij}-u_{kq}||_2 \pi(r_{ij},u_{kq}) \\
&+ \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in\mathcal{I}_{ij}} \pi(r_{ij},u_{kq}) \right) f(r_{ij}) + \sum_{(i,j)\in\mathcal{N}} \left(p(u_{ij}) - \sum_{(k,q)\in\mathcal{N}} \pi(r_{kq},u_{ij}) \right) g(u_{ij}) + \sum_{(k,q)\in\mathcal{N}} \sum_{(i,j)\in\mathcal{N}\setminus\mathcal{I}_{kq}} \pi(r_{kq},u_{ij}) g(u_{ij}).
\end{aligned}
$$

APPENDIX 3. IMPLEMENTATION DETAILS

Table 5 presents implementation details of ANE for each experiment. In the table, $\eta^{(0}$ and L_2 mean the initial learning rate and the L_2 regularization coefficient, respectively. Note that the length of the sequences was fixed to 8 in geomagnetic storm forecasting. PyTorch and Torch Geometric libraries were used in the implementation. All experiments were performed in a machine of Intel i9-12900K CPU, 64G memory, and GeForce RTX 3090. The source code of ANE and the experiment scripts of this paper are available at https://github.com/ngs00/ane.

Table 5: Implementation details of ANE for each experiment. $\eta^{(0)}$, L_2, and m mean the initial learning rate, L_2 regularization coefficient, and dimensionality of embeddings, respectively.

Experiment		Embedding Network $\phi(\cdot;\boldsymbol{\theta})$					Prediction Model $f(\cdot;\boldsymbol{\mu})$			
		Method	$\eta^{(0)}$	L_2	Batch size	m	Method	$\eta^{(0}$	L_2	Batch size
N-body problem (Section 5.1)		FCNN	5e-4	5e-6	32	32	FCNN	5e-4	5e-6	32
Materials property prediction (Section 5.2)	MPS-FE	MPNN	5e-4	0	32	32	FCNN	5e-4	1e-6	64
	MPS-BG	MPNN	1e-3	0	32	32	FCNN	5e-4	1e-6	64
	MPS-SM	MPNN	5e-4	0	32	48	FCNN	5e-4	1e-6	64
	MPS-BM	MPNN	5e-4	1e-7	32	48	FCNN	5e-4	1e-6	64
Geomagnetic storm forecasting (Section 5.3)	MNN-A	GRU	5e-4	1e-6	32	24	FCNN	1e-3	1e-6	32
	MNN-B	GRU	5e-4	0	32	24	FCNN	1e-3	1e-6	32
	MNN-C	GRU	5e-4	0	32	64	FCNN	1e-3	1e-6	32

APPENDIX 4. CRYSTAL GRAPH GENERATION

To feed the crystal structures into GNNs, the crystal structures should be converted into the attributed graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$, where $\mathcal{V}$ is a set of nodes (atoms), $\mathcal{E}$ is a set of edges (chemical bondings), $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$ is a d-dimensional node-feature matrix, and $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}|\times l}$ is a

l-dimensional edge-feature matrix. We used well-known Pymatgen[3] package to extract a unit cell of the lattice crystal structures of the materials. For the node features, we used 8 pre-defined chemical and physical properties of the elements. For the edge features, we used the physical distance between two atoms, which is quantized by radial basis function [43]. The atomic cutoff to define neighborhood atom was fixed to 5 Å and 4 Å on MPS and HOIP datasets, respectively.

APPENDIX 5. EXTRAPOLATION ERRORS AND DETECTION ACCURACIES ON MNN DATASETS

Tables 6 and 7 shows the extrapolation errors and the dectection accuracies of the machine learning methods on MNN-B and MNN-C datasets. Like the results on MNN-A dataset, ANE-GRU achieved the best performance for all metrics on MNN-B dataset. Although ANE-GRU was the third best method in Corr on MNN-C datasets, it achieved the best performances for all other metrics.

Table 6: Extrapolation errors and detection accuracies on MNN-B dataset.

Metric		RNN Methods				DML Methods		
		RNN	LSTM	GRU	TF	LRL-GRU	SLRL-GRU	ANE-GRU
Extrapolation Error	MAE	13.291±0.408	11.894±0.384	12.118±0.537	11.444±0.244	9.800±0.492	9.306±0.459	**8.993±0.393**
	Corr	0.718±0.072	0.672±0.046	0.677±0.049	0.657±0.058	0.442±0.075	0.410±0.039	**0.381±0.055**
Detection Accuracy	Precision	0.253±0.070	0.341±0.065	0.280±0.080	0.364±0.072	0.728±0.188	0.648±0.121	**0.741±0.088**
	Recall	0.310±0.120	0.276±0.126	0.248±0.091	0.319±0.103	0.448±0.243	0.519±0.084	**0.581±0.123**
	F1-score	0.276±0.093	0.296±0.094	0.261±0.087	0.537±0.122	0.471±0.195	0.570±0.084	**0.638±0.070**

Table 7: Extrapolation errors and detection accuracies on MNN-C dataset.

Metric		RNN Methods				DML Methods		
		RNN	LSTM	GRU	TF	LRL-GRU	SLRL-GRU	ANE-GRU
Extrapolation Error	MAE	15.464±0.782	14.586±0.742	14.486±0.890	11.907±0.762	11.967±0.713	11.280±0.727	**10.925±0.504**
	Corr	0.721±0.032	0.698±0.044	0.694±0.041	0.564±0.041	0.440±0.018	**0.418±0.029**	0.466±0.054
Detection Accuracy	Precision	0.036±0.026	0.048±0.042	0.032±0.033	0.037±0.039	0.132±0.059	0.336±0.191	**0.891±0.095**
	Recall	0.076±0.049	0.086±0.070	0.057±0.056	0.048±0.052	0.295±0.171	0.238±0.074	**0.299±0.106**
	F1-score	N/A	N/A	N/A	N/A	0.172±0.087	0.263±0.106	**0.347±0.142**

APPENDIX 6. LINEARITY VALIDATION

The goal of ANE is to simplify the input-to-target relationships for machine learning extrapolation. In this experiment, we quantitatively evaluate how well the DML methods simplified the input-to-target relationships on the materials datasets by measuring MAE and R^2 score of a simple linear regression because low MAE and high R^2 score of the linear regression imply the simplest linearly-approximatable function. We measured the linearity on the entire dataset to evaluate how well each DML method simplifies the relationships between the crystal structures and their materials properties over the entire materials space. In the linearity evaluation, ANE achieved highest R^2 scores for all datasets, and its R^2 scores were greater than 0.9 for most datasets as shown in Table 8. The highest linearity of ANE reveals the reasons for the highest extrapolation accuracies in predicting materials properties of Sections 5.2 and 5.4.

Table 8: Linearity measured by MAE and R^2 score of linear regression in predicting the materials properties from the generated materials embeddings on the benchmark materials datasets.

Method	MPS (Formation Energy)		MPS (Band Gap)		MPS (Shear Modulus)		MPS (Bulk Modulus)		HOIP-HIGH		HOIP-LOW	
	MAE	R^2 score	MAE	R^2 score	MAE	R^2 score	MAE	R^2 score	MAE	R^2 score	MAE	R^2 score
LRL-MPNN	0.165 (±0.015)	0.914 (±0.018)	0.571 (±0.031)	0.425 (±0.032)	0.208 (±0.003)	0.834 (±0.003)	0.155 (±0.006)	0.879 (±0.008)	0.112 (±0.011)	0.967 (±0.005)	0.101 (±0.009)	0.975 (±0.003)
SLRL-MPNN	0.160 (±0.004)	0.916 (±0.005)	0.372 (±0.003)	0.693 (±0.008)	0.190 (±0.003)	0.869 (±0.003)	0.143 (±0.003)	0.884 (±0.003)	0.106 (±0.007)	0.971 (±0.003)	0.104 (±0.008)	0.976 (±0.003)
ANE-MPNN	**0.152 (±0.003)**	**0.938 (±0.001)**	**0.336 (±0.009)**	**0.753 (±0.006)**	**0.185 (±0.008)**	**0.877 (±0.005)**	**0.135 (±0.004)**	**0.909 (±0.004)**	**0.078 (±0.007)**	**0.984 (±0.002)**	**0.095 (±0.007)**	**0.981 (±0.003)**

[3]https://pymatgen.org