Nonlinearity Encoding for Extrapolation of Neural Networks

Gyoung S. Na[1] and Chanyoung Park[2]

[1]Korea Research Institute of Chemical Technology (KRICT), Republic of Korea

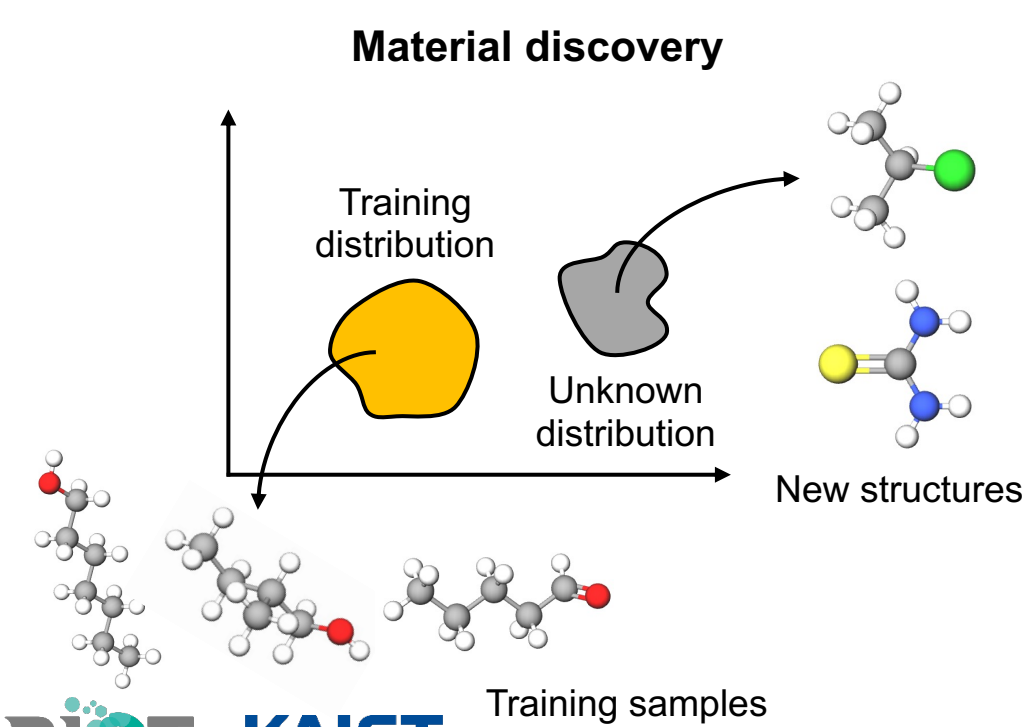
[2]Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea

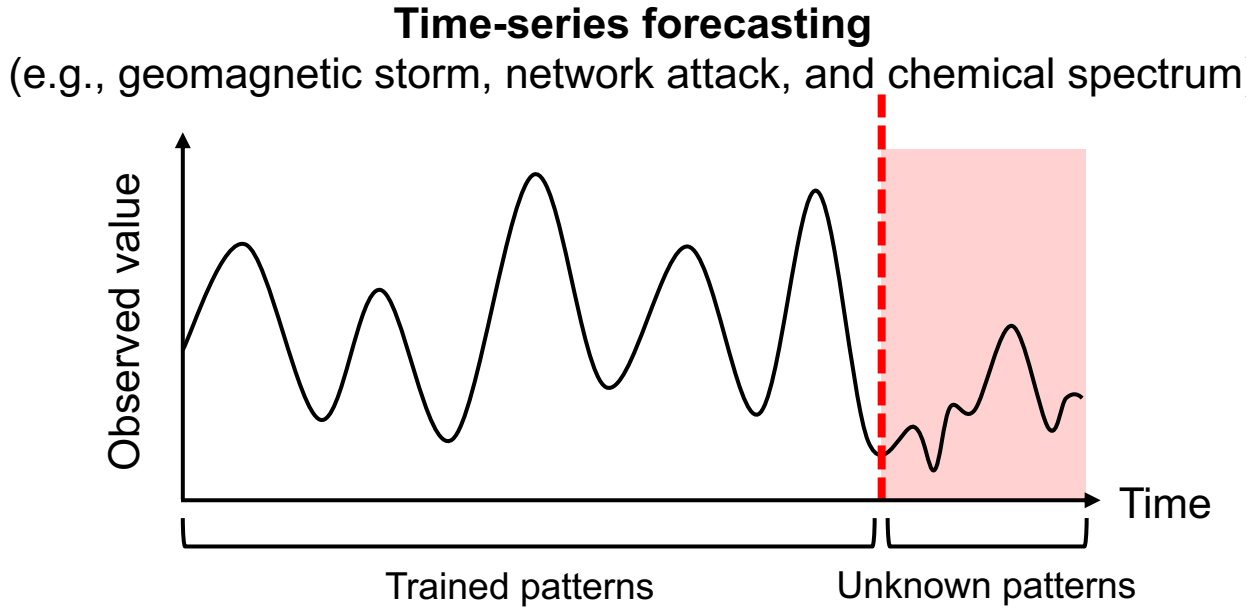
ngs0@krict.re.kr, cy.park@kaist.ac.kr

KRICT KAIST DSAIL Data Science & Artificial Intelligence

Extrapolation

- **Goal:** Predict unseen data **outside the training distribution**
- Extrapolation is challenging because the input data usually follows an unknown distribution
- However, **extrapolation is common in scientific applications** in which discovering unobserved scientific knowledge is crucial





Formal Definition of Extrapolation in Machine Learning

- **Given**: Prediction model $f: \mathcal{X} \rightarrow \mathbb{R}$ trained on a training distribution $\mathcal{D}$
- **Goal**: Minimize the following extrapolation error L_e

Extrapolation Error

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

Training distribution

Prediction model

Data distribution

Loss function (Cross entropy, MSE)

- (x, y): A sample from out of training distribution $\mathcal{X}\setminus\mathcal{D}$

Input data

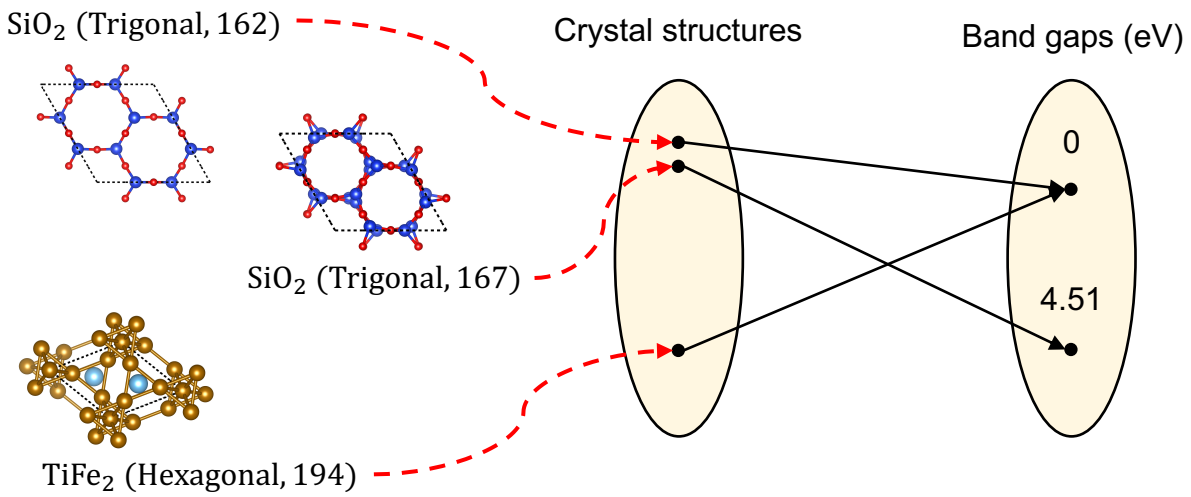
Target response

- Machine learning achieved remarkable extrapolation performance in computer vision [1, 2]
- However, **extrapolation in scientific applications is still far from satisfactory [3, 4]**

KRICT KAIST

Why is Extrapolation Difficult in Scientific Data?

- **Nonlinear input-to-target relationship**
  - Physical and chemical systems have severe **nonlinear relationships with their properties**.

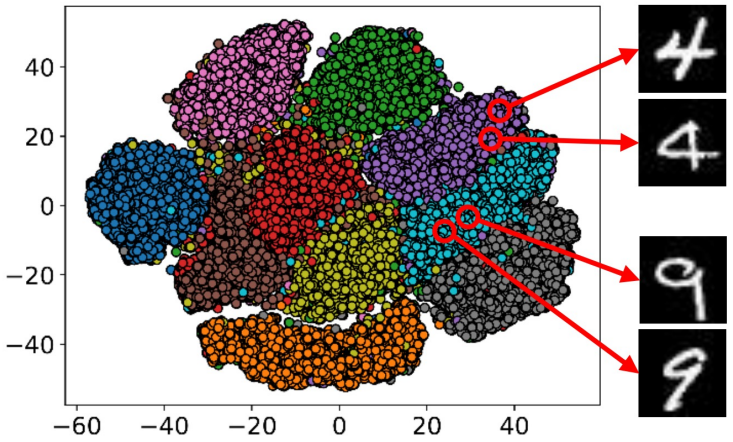


Two **similar structures** have completely **different physical properties**,
whereas two completely **different structures** have **the same physical property**

Image Dataset vs. Scientific Dataset

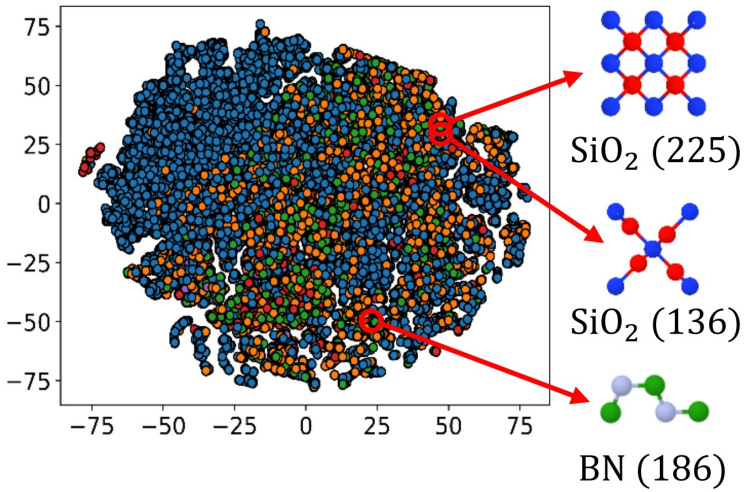
- T-SNE plots of MNIST and Material Project (MP) datasets
- Each point indicates an **image** or a **material** with **target response (label) denoted by colors**.
  - MNIST: class label
  - MP dataset: band gap

(a) MNIST dataset



Similar images share similar labels

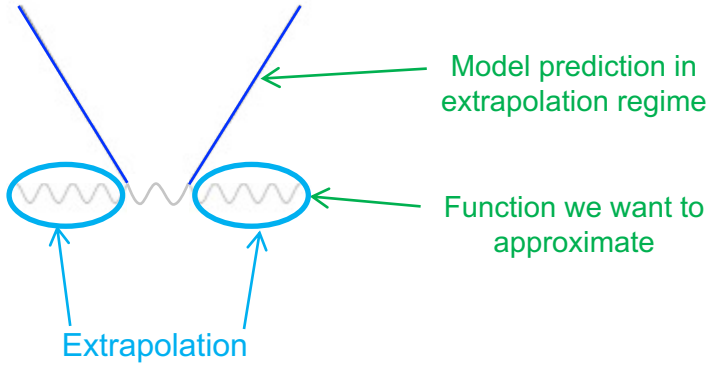
(b) MP dataset

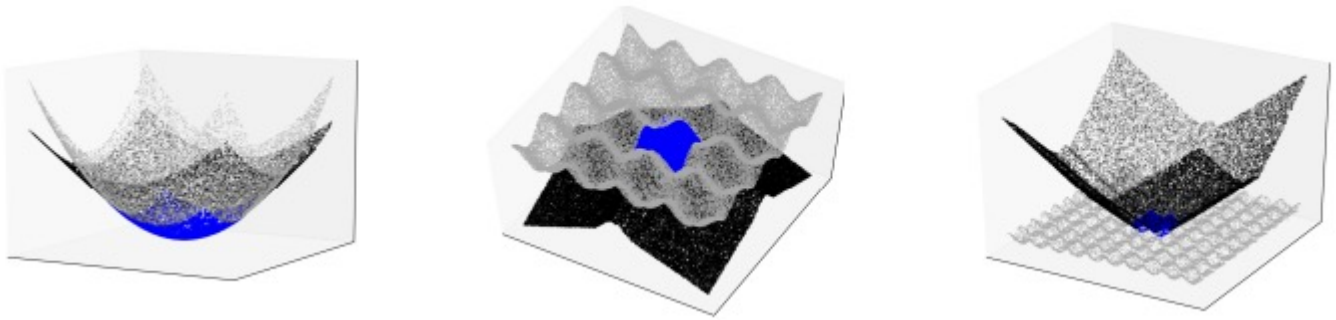


Similar materials do not necessarily share similar labels

How Neural Networks Extrapolate (Xu et al, ICLR21)

- **Theoretical findings in extrapolation:** Neural networks with ReLU → simple linear regression in the extrapolation regime [7]





MLPs converge to linear functions outside the training data range

- **Proposed solution:** Remove nonlinearity from the data itself to linearize the problem
- **Limitation:** Requires domain knowledge to remove nonlinearity, and task-specific / data-specific

Related Work on Extrapolation

- **Representation learning [5]**
  - Pros: Universally applicable method
  - Cons: Constraints on data distributions
- **Transfer learning [6]**
  - Pros: Problem-specific methods, goal-directed learning
  - Cons: Source datasets, similar data distributions, re-training
- **Graph reformulation [7]**
  - Pros: Easy to implement, theoretical backgrounds
  - Cons: Manual reformulation, white-box systems

Most existing studies mainly focus on **supporting extrapolation** rather than learning extrapolation models

Can we learn extrapolation models?

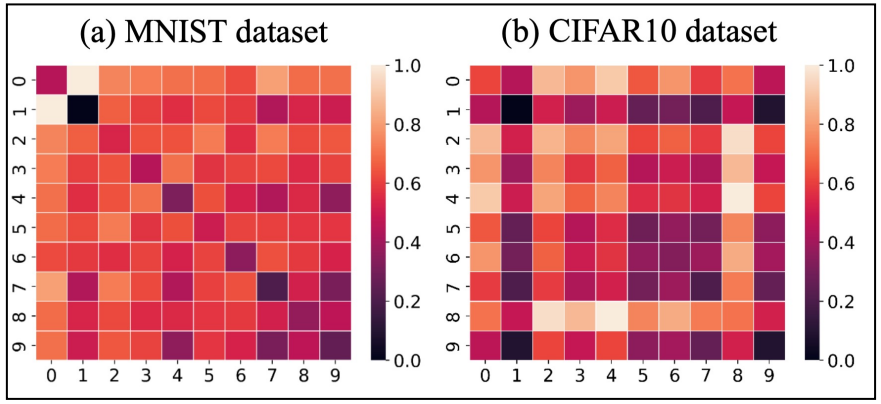
KRICT KAIST

Can we learn extrapolation models?

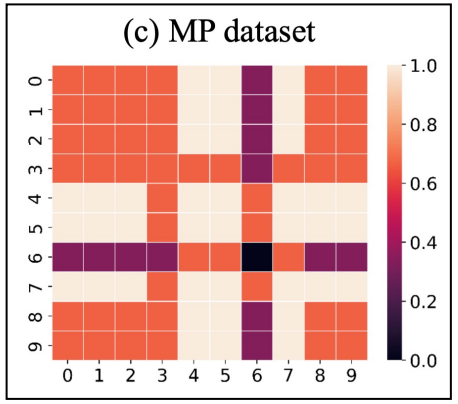
: Image Dataset vs. Scientific Dataset

- Heatmap visualization of **within-** and **between-class distances** on benchmark image and materials datasets

Image



Scientific dataset



Small — Within-class distance (Diagonal) — Large

Easy — Prediction task (Classification) — Difficult

Prediction tasks can be made easier when,
Two inputs with same label → Small input distance

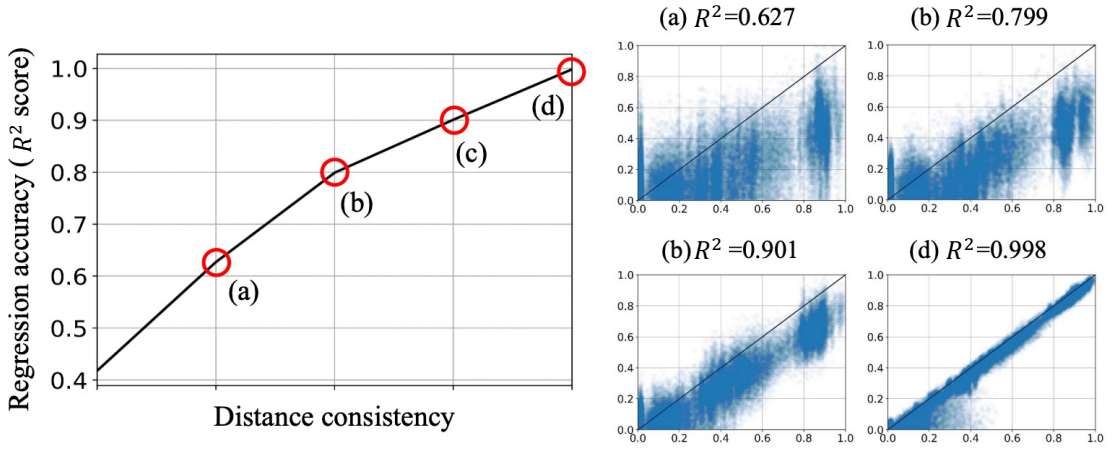
Distance Consistency!

KRICT KAIST

Distance Consistency (DC)

- Consistency w.r.t. the distance between the inputs and their target responses
  - e.g., images > materials
- Extend our argument from classification to **regression**
  - Assume: Classification with infinite number of classes ≈ regression

Linear regression on synthetic datasets



High distance consistency → High accuracy (R^2 score) → **Input-to-target relationship is made simple**

Problem Reformulation of Extrapolation

- We reformulate the extrapolation problem as a **representation learning** problem aiming to **linearize the input-to-target relationships**

- **Our goal:** Increase the **distance consistency** aiming at **simplifying the input-to-target relationships**
  - **Given:** Two pairs of data samples $(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)$
  - **Define**: The distance between them

Dist. btw. targets

$$d(d(\boldsymbol{x}_i, \boldsymbol{x}_j) - d(y_i, y_j))$$

Dist. btw. inputs

Consider all N^2 pairs

$$\sum_{i=1}^{N} \sum_{j=1}^{N} d(d(\boldsymbol{x}_i, \boldsymbol{x}_j) - d(y_i, y_j))$$

We adopt **Wasserstein distance** to measure the distance consistency between input and target

Nonlinearity Encoding based on Wasserstein Distance

- For a set of probability measures Π on $\Omega \times \Omega$, Wasserstein distance is defined by an optimization problem as:

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

Why Wasserstein distance?

Many scientific data has unknown and arbitrary shaped distributions

- However, there is a problem in applying Wasserstein distance in our task
  - Wasserstein distance is defined only for the **data distributions of the same dimensionality**.
- **Our task**: Regression
  - Input: Vector ($\in \mathbb{R}^d$)
  - Target: Scalar ($\in \mathbb{R}$)

Dimension mismatch!

Nonlinearity Encoding based on Wasserstein Distance

- Instead, we define **distance distribution** to apply Wasserstein distance between **two distributions of different dimensions**

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

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

$$(p = 1)$$

Distance consistency btw input and target!

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

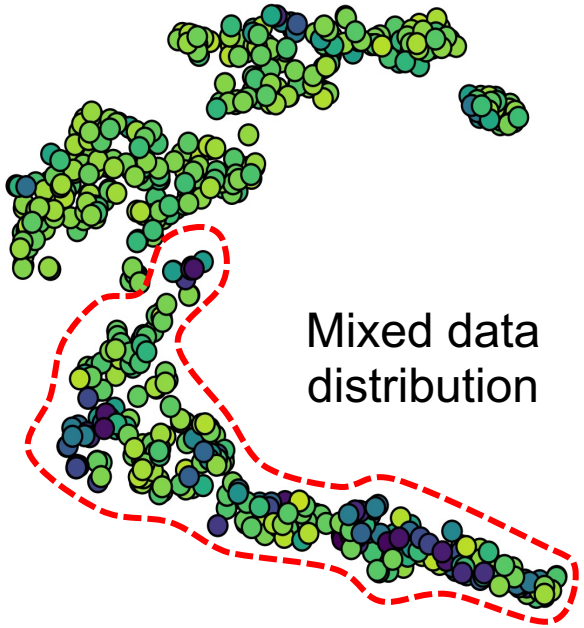
- $r = d\big(\phi(\mathbf{x}; \theta), \phi(\mathbf{x}'; \theta)\big)$: **Dist. btw input data** in embedding space
- $u = d(y, y')$: **Dist. btw target data**

Our goal: **Maximize the distance consistency** between input and target
→ The distance between two inputs should be determined based on the distance between their targets

Problem Definition of Nonlinearity Encoding

- **Our method**: Automatic Nonlinearity Encoding (ANE)

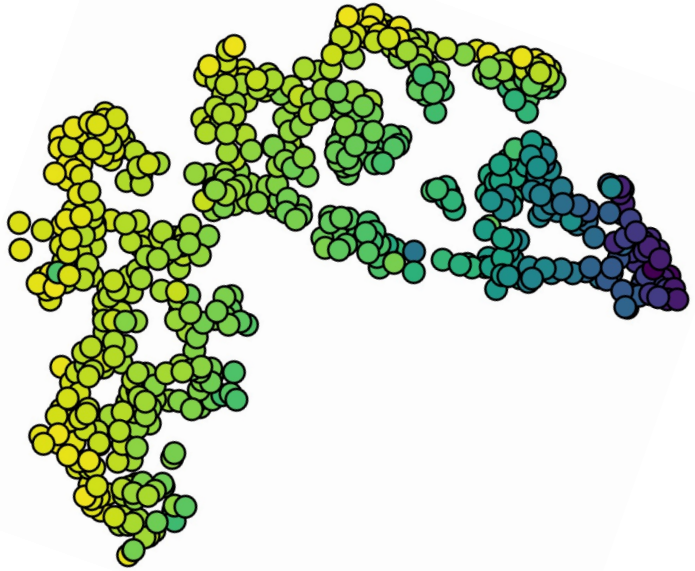
Data distribution in the **original feature space**



Hard

Nonlinearity Encoding

Data distribution in the **embedding space of ANE**



Easy

KRICT KAIST

Optimization: Decomposition of Lagrangian

- Our problem can be defined as follows:

training data

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{N} \inf_{\pi \in \Pi} \int_{\mathcal{M} \times \mathcal{M}} \|r_{ij} - u_{ij}\|_p \pi(r_{ij}, u_{ij}) \mathrm{d}r\mathrm{d}u$$

Joint optimization w.r.t. θ and π

- $r_{ij} = d\left(\phi(\mathbf{x}_i; \theta), \phi(\mathbf{x}_j; \theta)\right)$: **Dist. btw input data** in embedding space
- $u_{ij} = d(y_i, y_j)$: **Dist. btw target data**

- We can define a **Lagrangian of the objective function** as (refer **Kantorovich-Rubinstein duality** [6]):

$$L_W = \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \left(\|r_{ij} - u_{kq}\| - f(r_{ij}) - g(u_{kq})\right) \pi(r_{ij}, u_{kq}) + \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \|r_{ij} - u_{kq}\| \pi(r_{ij}, u_{kq})$$

$$+ \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in I_{ij}} \pi(r_{ij}, u_{kq})\right) f(r_{ij}) + \sum_{(i,j)\in\mathcal{N}} \left(p(u_{ij}) - \sum_{(k,q)\in\mathcal{N}} \pi(r_{kq}, u_{ij})\right) g(u_{ij}) + \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \pi(r_{kq}, u_{ij})\, g(u_{ij}),$$

where $\mathcal{N} = \{(i,j) \mid \text{for all } i,j \in \{1,2,\ldots,N\}\}$, and $I_{ij} = \{(k,q) \mid u_{ij} = u_{kq} \text{ for } (k,q) \in \mathcal{N}\}$.

Pairs with the same target distance

Optimization: Model Parameter Optimization

- In the end, the representation learning problem to encode the nonlinearity is given by:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{N} \inf_{\pi \in \Pi} \int_{\mathcal{M} \times \mathcal{M}} \left\| r_{ij} - u_{ij} \right\|_p \pi(r_{ij}, u_{ij}) \mathrm{d}r\mathrm{d}u$$

- $r_{ij} = d\left(\phi(\mathbf{x}_i; \theta), \phi(\mathbf{x}_j; \theta)\right)$: **Dist. btw input data** in embedding space
- $u_{ij} = d(y_i, y_j)$: **Dist. btw target data**

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{N} \left\| r_{ij} - u_{ij} \right\|$$

Enforce distance consistency between data pairs!

KRICT KAIST

Optimization: Model Parameter Optimization

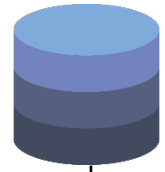
ANE

Prediction model

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 $f(\phi(\mathbf{x}; \boldsymbol{\theta}^*); \boldsymbol{\mu}^*)$



Training dataset
$\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$

Data-agnostic!

ANE

Training dataset with nonlinearity encoding
$\mathcal{Z} = \{(\phi(\mathbf{x}_1; \theta^*), \mathbf{y}_1), \ldots, (\phi(\mathbf{x}_N; \theta^*), \mathbf{y}_N\}$

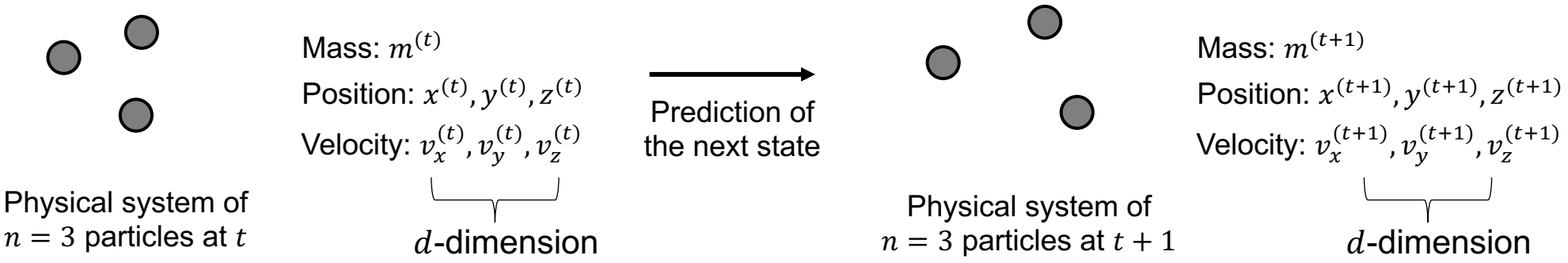
Prediction model

Experiments

- Matrix-shaped data
- Graph-structured data
- Time-series data

Extrapolation on Matrix-Shaped Data: n-Body Problem (1/3)

- **Task:** Given mass, position, and velocity of n particles, estimate future velocities of n particles



- **Data preprocessing:** 3-dimensional 3-body problem. $\boldsymbol{x}_t \in \mathbb{R}^{3\times 7}$ and $\boldsymbol{y}_t \in \mathbb{R}^{3\times 3}$ ← Matrix-shaped data
  - Simulated 10 datasets
  - **Train**: Observations in time [0, 80]
  - **Test**: Predict velocity in future time (80, 100]

Extrapolation on Matrix-Shaped Data: n-Body Problem (2/3)

- **Metric:** Distance correlation (Corr) between the simulated (ground-truth) and predicted velocities
  - To measure how accurately the models predict future **trends** of the velocities

Direct prediction method | **GNN-based methods** | **Metric learning-based method**

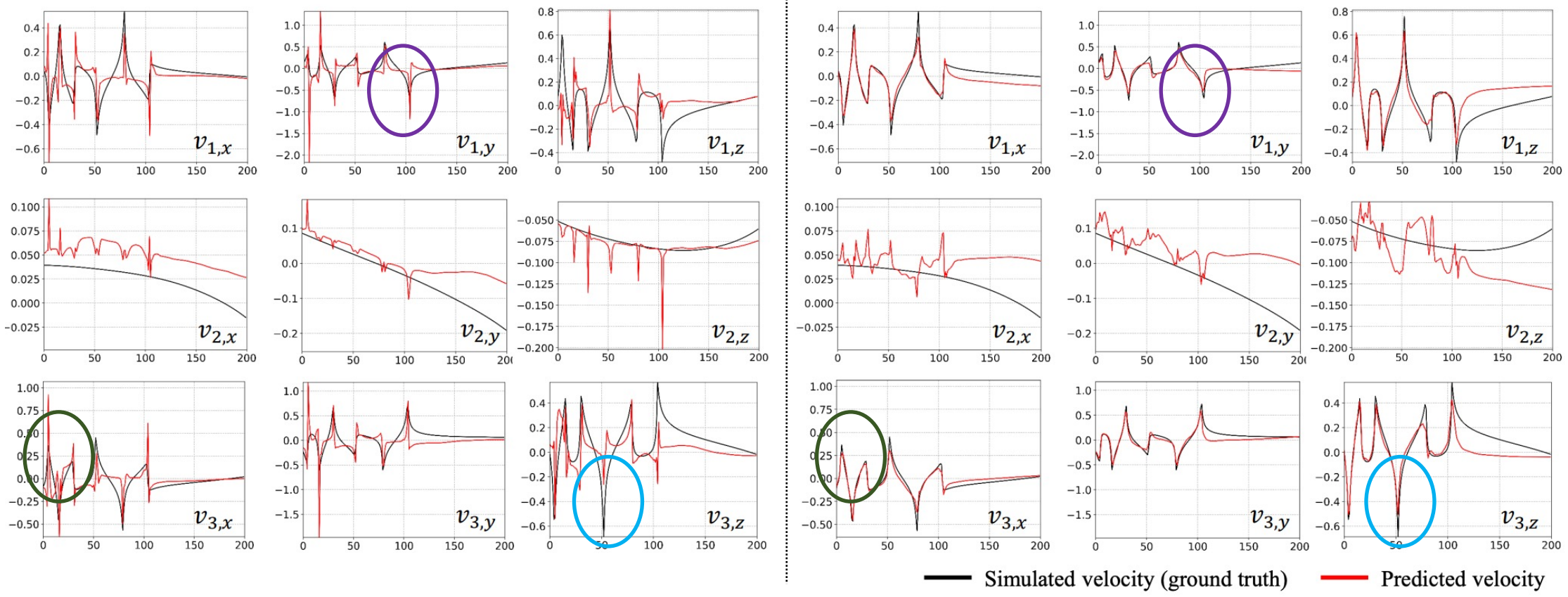
Idx.	NBNet	GIN	MPNN	UMP	LRL-F	SLRL-F	ANE-F
1	0.32	0.54	0.35	0.25	0.43	0.53	**0.18**
2	0.49	0.54	0.53	**0.36**	0.52	0.49	0.45
3	0.57	0.54	0.53	0.46	0.52	0.59	**0.29**
4	0.25	0.68	0.26	0.26	0.09	0.07	**0.03**
5	0.66	0.93	0.71	0.69	0.85	0.65	**0.49**
6	**0.11**	0.22	0.17	0.16	0.12	0.12	0.02
7	0.75	0.94	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 ±std.	0.46 ±0.19	0.62 ±0.24	0.41 ±0.20	0.44 ±0.19	0.41 ±0.23	0.40 ±0.18	**0.23 ±0.17**

ANE generates input representations that are the most effective to reducing the extrapolation errors

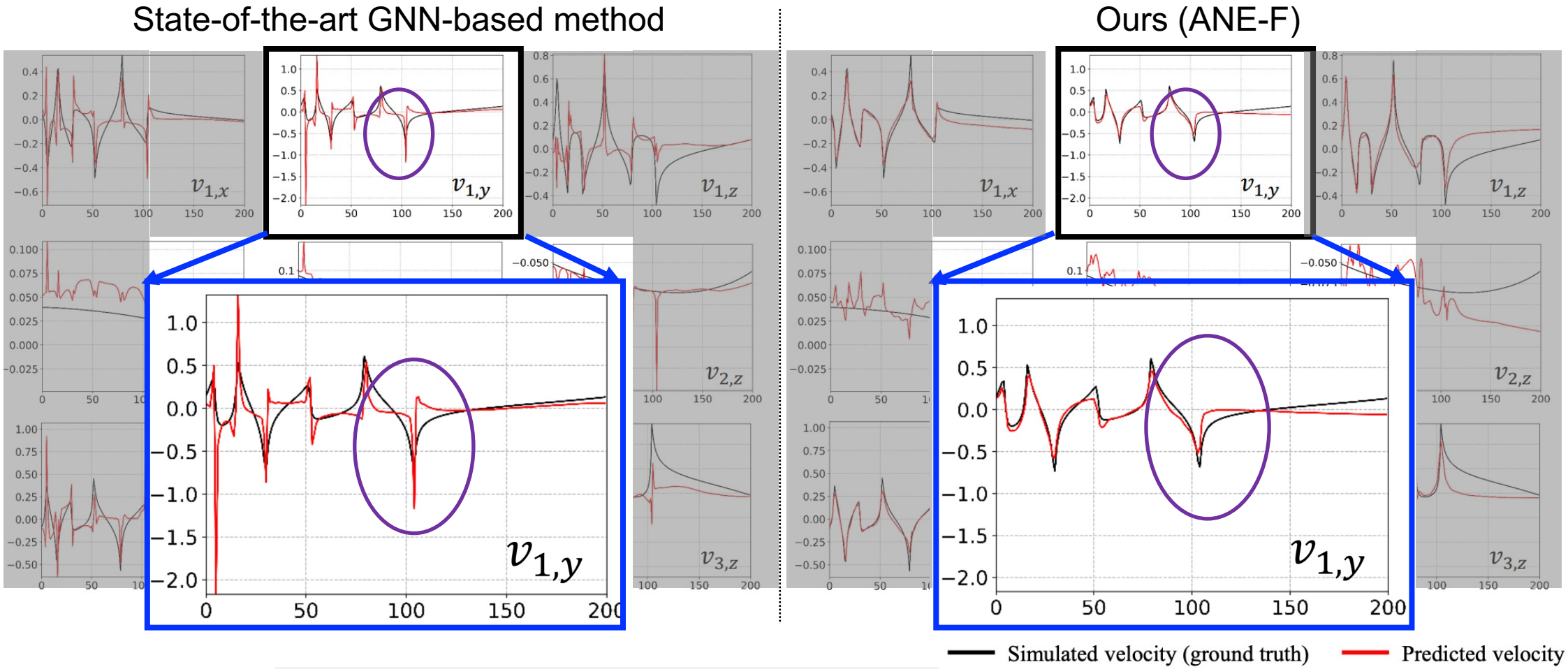
Extrapolation on Matrix-Shaped Data: n-Body Problem (3/3)

State-of-the-art GNN-based method

Ours (ANE-F)



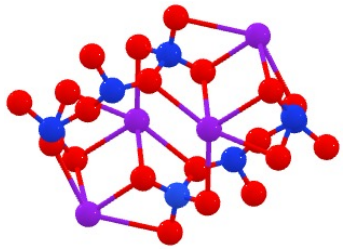
Extrapolation on Matrix-Shaped Data: n-Body Problem (3/3)



ANE is better at predicting **sudden explosions** of velocity

Extrapolation on Graph-Structured Data: Materials Property Prediction

- **Task**: Predict four material properties (Formation energy, Band gap, Shear modulus, Bulk modulus)
  - Discovering novel materials is a fundamental task in various fields (e.g., semiconductor and renewable energy)



$\mathcal{V}$: A set of nodes (atoms)
$\mathcal{U}$: A set of edges (bondings)
$\mathbf{X}$: Node feature matrix
$\mathbf{E}$: Edge feature matrix

→ Prediction model →

Physical and chemical properties of materials

A material can be represented as an attributed graph $G = (\mathcal{V}, \mathcal{U}, \mathbf{X}, \mathbf{E})$.

- **Data preprocessing**
  - MPS dataset: Benchmark materials dataset containing 3,162 materials
  - **Train**: Materials that contain **only two types of elements** (i.e., Binary materials)
  - **Test**: Materials that contain **three/four types of elements** (i.e., Ternary and quaternary materials)

Extrapolation on Graph-Structured Data: Materials Property Prediction

- **Metric:** R^2 score

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)**

ANE-MPNN outperforms state-of-the-art GNNs and metric learning methods

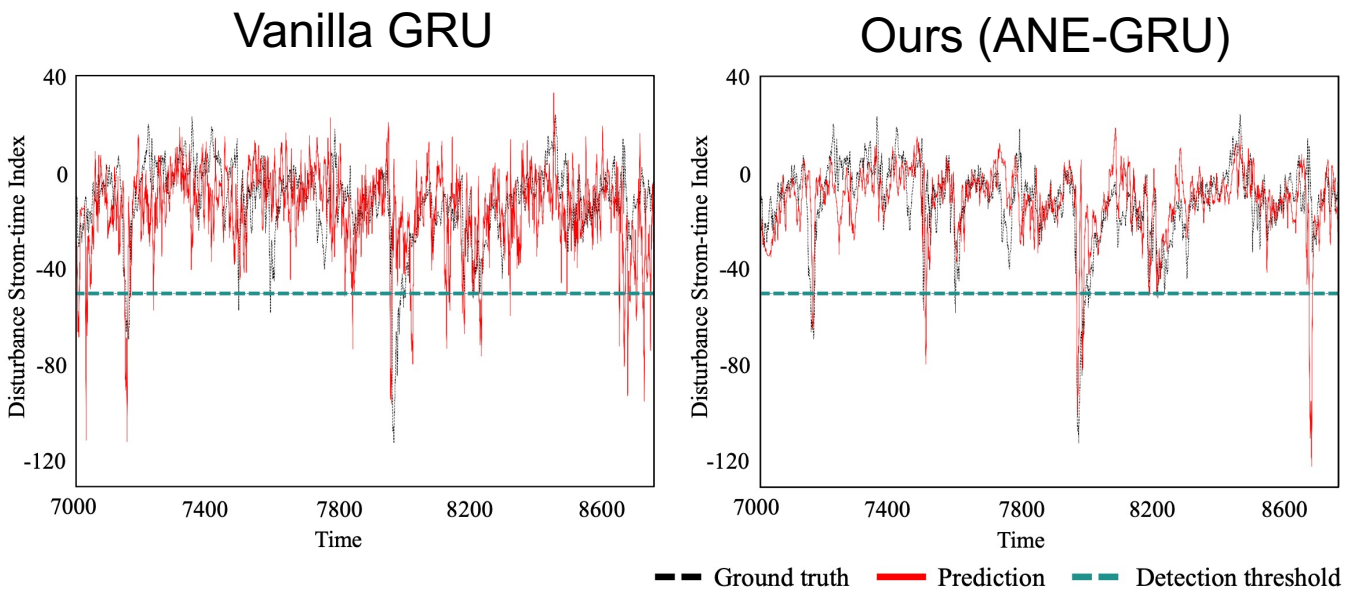
KRICT KAIST

Extrapolation on Time-Series Data: Geomagnetic Storm Forecasting

- **Task**: 1) Predict geomagnetic storm, 2) Detect geomagnetic storm
- **Data preprocessing**
  - Dataset: MagNet NASA dataset
  - 1-year geomagnetic storm data is divided into 4 sequential periods (¾ used for training, ¼ used for test)

Task 1 Task 2

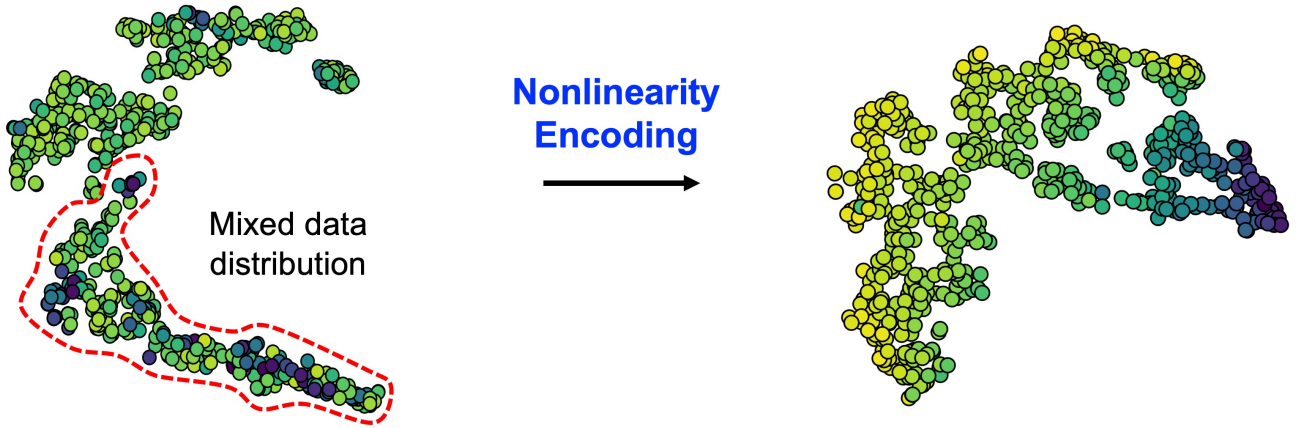
Method	Extrapolation Error		Detection Accuracy		
	MAE	Corr	Precision	Recall	F1-score
RNN	16.089 (±0.806)	0.710 (±0.025)	0.133 (±0.013)	0.281 (±0.065)	0.178 (±0.015)
LSTM	14.721 (±0.702)	0.696 (±0.065)	0.164 (±0.048)	0.260 (±0.087)	0.201 (±0.062)
GRU	14.613 (±0.368)	0.687 (±0.027)	0.145 (±0.027)	0.230 (±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)**



ANE-GRU outperforms GRU, and ANE achieved further improvement over metric learning-based approaches

Conclusion

- Proposed a **data-agnostic** embedding method for improving the **extrapolation capabilities** of ML



Data distribution in the **original feature space**

Data distribution in the **embedding space of ANE**

- Maximized **distance consistency** between the inputs and their targets (Based on **Wasserstein distance**)
  - The **distance between two inputs** should be **determined based on the distance between their targets**
- Demonstrated the effectiveness in **various scientific applications of various data formats**

Thank you!

- Contact: ngs0@krict.re.kr / cy.park@kaist.ac.kr
- Source code: https://github.com/ngs00/ane
- Lab homepage: https://dsail.kaist.ac.kr/

Reference

[1] Richard et al. Oconet: Image extrapolation by object completion. CVPR, 2021.

[2] Yi et al. Wide-context semantic image extrapolation. In CVPR, 2019.

[3] Yuxin et al. Mlatticeabc: generic lattice constant prediction of crystal materials using machine learning. ACS omega, 2021.

[4] Haotong et al. Cryspnet: Crystal structure predictions via neural networks. Phys. Rev. Materials, 2020.

[5] Taylor et al. Learning representations that support extrapolation. ICML, 2020.

[6] Stephanie et al. Finding users who act alike: Transfer learning for expanding advertiser audiences. KDD, 2019.

[7] Xu et al. How neural networks extrapolate: from feedforward to graph neural networks. ICLR, 2021.

[8] Edwards et al. On the Kantorovich-Rubinstein Theorem. Expo. Math., 2011.

[9] NASA and NOAA satellites solar-wind dataset. https://www.kaggle.com/arashnic/soalr-wind.

Appendix

Optimization: Decomposition of Lagrangian

$$L_W = \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \left(\|r_{ij} - u_{kq}\|_2 - f(r_{ij}) - g(u_{kq}) \right) \pi(r_{ij}, u_{kq})$$

Part 1

$$+ \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in I_{ij}} \|r_{ij} - u_{kq}\|_2 \, \pi(r_{ij}, u_{kq})$$

Part 2

$$+ \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in I_{ij}} \pi(r_{ij}, u_{kq}) \right) f(r_{ij}) \quad + \sum_{(k,q)\in\mathcal{N}} \left(p(u_{kq}) - \sum_{(i,j)\in\mathcal{N}} \pi(r_{ij}, u_{kq}) \right) g(u_{kq})$$

$$+ \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \pi(r_{ij}, u_{kq}) g(u_{kq})$$

Part 3

Optimization: Decomposition of Lagrangian

$$L_W = \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \left(\|r_{ij} - u_{kq}\|_2 - f(r_{ij}) - g(u_{kq}) \right) \pi(r_{ij}, u_{kq})$$

Part 1 → **0**

$$+ \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in I_{ij}} \|r_{ij} - u_{kq}\|_2 \, \pi(r_{ij}, u_{kq})$$

$$+ \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in I_{ij}} \pi(r_{ij}, u_{kq}) \right) f(r_{ij}) \quad + \sum_{(k,q)\in\mathcal{N}} \left(p(u_{kq}) - \sum_{(i,j)\in\mathcal{N}} \pi(r_{ij}, u_{kq}) \right) g(u_{kq})$$

$$+ \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \pi(r_{ij}, u_{kq}) g(u_{kq})$$

Best choice of the joint probability π?

Set $\pi(r_{ij}, u_{kq}) = 0$ for all $(i,j) \in \mathcal{N}$ and $(k,q) \in \mathcal{N}\setminus I_{ij}$

(i.e., If two pairs of data ((i,j) and (k,q)) and do not have the same target distance, then the joint probability is 0)

$\because \|r_{ij} - u_{kq}\| - f(r_{ij}) - g(u_{kq}) \geq 0$ by the constraint in Lagrangian multipliers (1-Lipschitz constraint)

Optimization: Decomposition of Lagrangian

$$L_W = \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \left(\|r_{ij} - u_{kq}\|_2 - f(r_{ij}) - g(u_{kq}) \right) \pi(r_{ij}, u_{kq})$$

$$+ \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in I_{ij}} \|r_{ij} - u_{kq}\|_2 \, \pi(r_{ij}, u_{kq})$$

Part 2 → **0**

$$+ \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in I_{ij}} \pi(r_{ij}, u_{kq}) \right) f(r_{ij}) \quad + \sum_{(k,q)\in\mathcal{N}} \left(p(u_{kq}) - \sum_{(i,j)\in\mathcal{N}} \pi(r_{ij}, u_{kq}) \right) g(u_{kq})$$

$$+ \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \pi(r_{ij}, u_{kq}) g(u_{kq})$$

$\pi(r_{ij}, u_{kq})$ is always zero under the **optimized embedding function $\boldsymbol{\phi}(\cdot; \boldsymbol{\theta}^*)$.**

Optimization: Decomposition of Lagrangian

$$L_W = \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \left(\|r_{ij} - u_{kq}\|_2 - f(r_{ij}) - g(u_{kq}) \right) \pi(r_{ij}, u_{kq})$$

$$+ \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in I_{ij}} \|r_{ij} - u_{kq}\|_2 \, \pi(r_{ij}, u_{kq})$$

$$+ \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in I_{ij}} \pi(r_{ij}, u_{kq}) \right) f(r_{ij}) \quad + \sum_{(k,q)\in\mathcal{N}} \left(p(u_{kq}) - \sum_{(i,j)\in\mathcal{N}} \pi(r_{ij}, u_{kq}) \right) g(u_{kq})$$

$$+ \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \pi(r_{ij}, u_{kq}) g(u_{kq})$$

Part 3

↓

0

Always zero by

$$p(r_{ij}) = \sum_{(k,q)\in I_{ij}} \pi(r_{ij}, u_{kq}),$$

$$p(u_{kq}) = \sum_{(i,j)\in I_{kq}} \pi(r_{ij}, u_{kq}),$$

and $\pi(r_{ij}, u_{kq}) = 0$ for all $(i,j) \in \mathcal{N}$ and $(k,q) \in \mathcal{N}\setminus I_{ij}$. ← **From Part 1**

KRICT KAIST

Optimization: Decomposition of Lagrangian

$$L_W = \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \left(\left\| r_{ij} - u_{kq} \right\|_2 - f(r_{ij}) - g(u_{kq}) \right) \pi(r_{ij}, u_{kq})$$

Part 1

$$+ \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in I_{ij}} \left\| r_{ij} - u_{kq} \right\|_2 \pi(r_{ij}, u_{kq})$$

Part 2

$$+ \sum_{(i,j)\in\mathcal{N}} \left(p(r_{ij}) - \sum_{(k,q)\in I_{ij}} \pi(r_{ij}, u_{kq}) \right) f(r_{ij}) \quad + \sum_{(k,q)\in\mathcal{N}} \left(p(u_{kq}) - \sum_{(i,j)\in\mathcal{N}} \pi(r_{ij}, u_{kq}) \right) g(u_{kq})$$

$$+ \sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in\mathcal{N}\setminus I_{ij}} \pi(r_{ij}, u_{kq}) g(u_{kq})$$

Part 3

Hence, one possible optimal joint probability π^* is given as:

$\pi(r_{ij}, u_{kq}) = 0$ for all $(i,j) \in \mathcal{N}$ and $(k,q) \in \mathcal{N}\setminus I_{ij}$ but $\sum_{(i,j)\in\mathcal{N}} \sum_{(k,q)\in I_{ij}} \pi(r_{ij}, u_{kq}) = 1$

Given data pairs that do not have the same target distance,
setting their joint probability to zero is one possible solution
(π should be a valid probability distribution)

Optimization: Model Parameter Optimization

- For the optimal joint probability π^*, the training problem of ANE is simplified as:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{J=1}^{N} |I_{ij}| \left\| r_{ij} - u_{ij} \right\| \pi_{ij}^*.$$

$|I_{ij}|$: # data pairs that share the same target distance with (i, j)

- The joint probability π_{ij} can be empirically estimated by the i.i.d. condition as:

$$\pi_{ij} = \frac{1}{\sum_{l=1}^{N} \sum_{m=1}^{N} |I_{lm}|}, \text{ and } |I_{ij}| \ll \sum_{l=1}^{N} \sum_{m=1}^{N} |I_{lm}|.$$

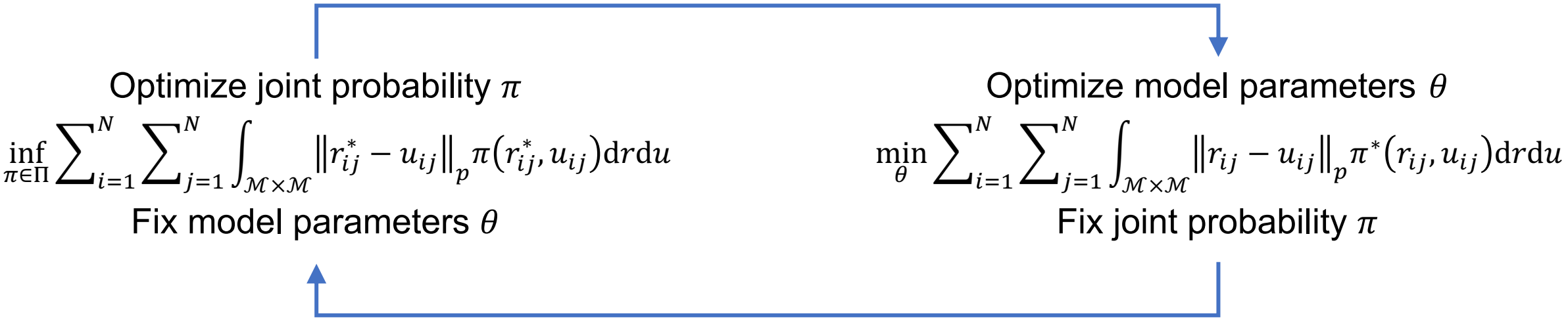
- Therefore, the representation learning problem to encode the nonlinearity is given by:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{N} \left\| r_{ij} - u_{ij} \right\|_2$$

- $r_{ij} = d\left(\phi(\mathbf{x}_i; \theta), \phi\left(\mathbf{x}_j; \theta\right)\right)$: **Dist. btw input data** in embedding space
- $u_{ij} = d(y_i, y_j)$: **Dist. btw target data**

Enforce distance consistency between data pairs

Optimization: Decomposition of Lagrangian



Alternating optimization

Decomposition of Lagrangian: Full derivation

$$
\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}
$$

KRICT KAIST

ANE for Discovering Solar Cell Materials

- **Task:** Predict band gaps of perovskites
  - c.f.) Perovskite has received significant attention as solar cell materials for renewable energy
  - Infer materials properties of crystal structures containing **unseen elemental combinations**
- **Data preprocessing**
  - Divided HOIP dataset by eliminating the materials that contain specific elements
    - **HOIP-HIGH**: HOIP – (Germanium (Ge) and Fluorine (F))
    - **HOIP-LOW**: HOIP – (Lead (Pb) and Iodine (I))
  - Range of band gaps between training and test data is completely different

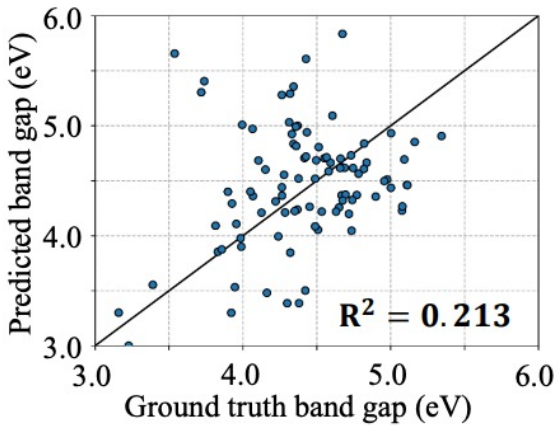
ANE for Discovering Solar Cell Materials

- **Metric:** R^2 score

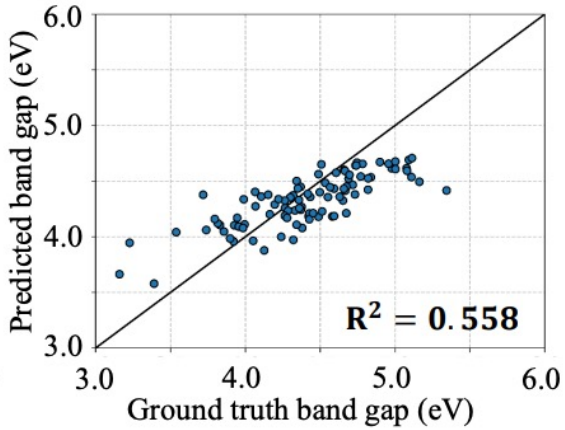
N/A: negative R^2

	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)**

Vanilla GCN



Ours (ANE-MPNN)

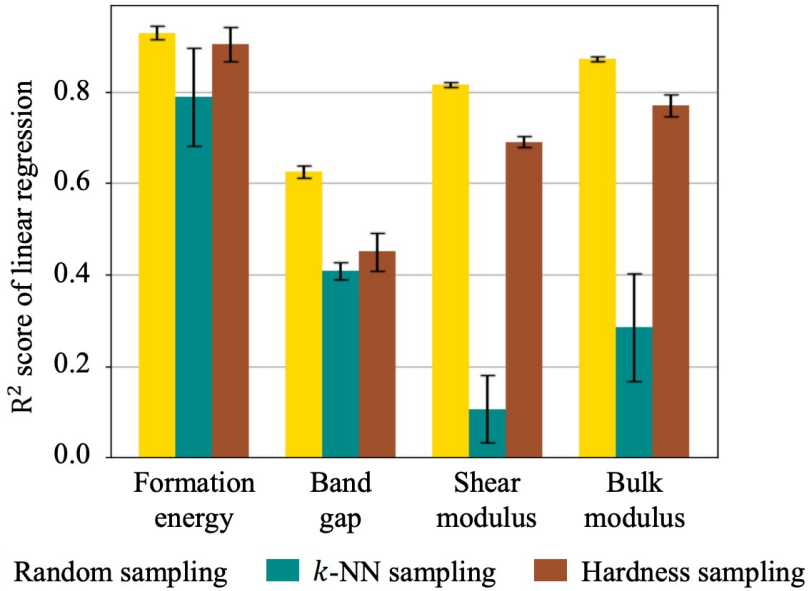


ANE-MPNN roughly captured the relationships, while GCN fails to do so

KRICT KAIST

Sampling Strategies and Extrapolation

- Time complexity of the training process of ANE: $\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{J=1}^{N} \left\| r_{ij} - u_{ij} \right\| \rightarrow \mathbf{O}(\boldsymbol{N^2})$
- Three sampling strategies to reduce the time complexity:
  - **Random sampling:** selecting a data point randomly at each iteration
  - ***k*-NN sampling:** selecting k nearest data points for an anchor data
  - **Hardness sampling:** selecting k data points based on the training errors (top-k largest errors)



Random sampling performs the best despite its simplicity
(∵ Random sampling = Density-based sampling)