ISOMETRIC TRANSFORMATION INVARIANT AND EQUIVARIANT GRAPH CONVOLUTIONAL NETWORKS

Masanobu Horie
University of Tsukuba,
Research Institute for Computational Science Co. Ltd.
horie@ricos.co.jp

Naoki Morita & Yu Ihara
Research Institute for Computational Science Co. Ltd.
{morita,ihara}@ricos.co.jp

Naoto Mitsume
University of Tsukuba
mitsume@kz.tsukuba.ac.jp

ABSTRACT

Graphs correspond to one of the most important data structures used to represent pairwise relations between objects. Specifically, using the graphs embedded in the Euclidean space is essential to solve real problems, such as object detection, structural chemistry analysis, and physical simulation. A crucial requirement to employ the graphs in the Euclidean space is to learn the isometric transformation invariant and equivariant features. In the present paper, we propose a set of the transformation invariant and equivariant models called IsoGCNs that are based on graph convolutional networks. We discuss an example of IsoGCNs that corresponds to differential equations. We also demonstrate that the proposed model achieves high prediction performance on the considered finite element analysis dataset and can scale up to the graphs with 1M vertices.

1 INTRODUCTION

The graph-structured data embedded in the Euclidean space can be utilized in many fields such as object detection, structural chemistry analysis, and physical simulation. To deal with such data, graph neural networks (GNNs) have been introduced. A crucial property of GNNs lies in permutation invariance and equivariance. In addition to permutations, isometric transformation invariance and equivariance have to be addressed considering the graphs in the Euclidean space, as many properties of objects in the Euclidean space do not change under translation and rotation. Therefore, formulating the isometric transformation invariant and equivariant models is critically important, because other approaches have particular issues. For instance, instead of formulating such models, one can: 1) isometrically transform inputs to obtain a canonical position and direction; 2) prepare a large dataset through isometric transformation. The former approach may provide unstable results in the case of noise or small modifications of shapes. The latter approach has lack of efficiency, as models have to learn many geometrically identical shapes. For instance, preprocessing to have canonical position and direction is not stable with noise and small modification of shapes, and preparing large dataset using isometric transformation is not efficient because models have to learn many geometrically identical shapes. In addition, the computationally efficiency is another crucial factor, especially for domains such as physical simulations that imply constructing large-sized graphs.

In the present paper, we propose IsoGCNs, a set of simple yet powerful models providing isometric transformation invariance and equivariance based on graph convolutional networks (GCNs) \cite{Kipf2017}. Specifically, the proposed model is sufficiently simple to realize isometric transformation invariant only by tweaking the definition of an adjacency matrix (equation \ref{eq:iso}). As the proposed approach relies on graphs, it can deal with complex shapes, which are usually presented in mesh data structures. The proposed approach is also computationally efficient in terms of processing the graphs with up to 1M vertices, which are often presented in physical simulations corresponding to realistic problems. The main contributions of the present paper can be summarized as follows:
• We construct general nabla adjacency matrices (GeNAMs), that serve as the foundation to construct the isometric transformation invariant and equivariant GCNs (IsoGCNs).
• We formulate a design rule to implement IsoGCNs for the specified input and output tensor types aiming to realize IsoGCNs.
• We demonstrate that an example of GeNAM has close relations with differential operators.
• We confirm that IsoGCN outperforms conventional GCNs on the considered finite element analysis dataset.

2 RELATED WORK

Graph neural networks. The studies presented by Baskin et al. (1997); Sperduti & Starita (1997); Gori et al. (2005); Scarselli et al. (2008) were the first research works that discussed the concept of GNNs. Although many variants of GNNs were proposed, including those presented by Hamilton et al. (2017), Veličković et al. (2017), these models are then unified under the concept of message passing neural networks (Gilmer et al., 2017). Among these variants, GCNs developed by Kipf & Welling (2017) implying the considerable simplification of GNNs are essential in the present study, as the proposed model is based on GCNs.

Invariant and equivariant neural networks. Group equivariant convolutional neural networks were first proposed by Cohen & Welling (2016) for discrete groups. Subsequent research works generalized it to continuous groups (Cohen et al., 2018), three-dimensional data (Weiler et al., 2018), and general manifolds (Cohen et al., 2019). Concerning point clouds, Thomas et al. (2018); Kondor (2018) discussed the way to provide rotation equivariance. Klicpera et al. (2020) proposed GNNs with rotation invariance by using scalar values, such as distances and angles between vertices, to represent the spatial structure of graphs. Their method achieved the high expressibility based on message passing; however, due to this reason, it required the considerable amount of computational resources. In contrast, the approach proposed in the present study allows reducing computational costs considerably as it is based on GCNs.

Physical simulations with GNNs. Several related studies, including those by Sanchez-Gonzalez et al. (2018; 2019); Alet et al. (2019) were focused on applying GNNs to learn physical simulations. Sanchez-Gonzalez et al. (2019) proposed a general framework for learning simulations based on GNNs. These approaches allowed introducing the physical information into GNNs; however, addressing isometric transformation invariance and equivariance was out of their scope. In the present study, we incorporate isometric transformation invariance and equivariance into GCNs, thereby ensuring the stability of training and inference under isometric transformation. Moreover, the proposed approach is efficient in processing the large graphs with up to 1M vertices that have a sufficient number of degrees of freedom to express complex shapes.

3 ISOMETRIC TRANSFORMATION INVARIANT AND EQUIVARIANT GRAPH CONVOLUTIONAL LAYERS

In this section, we discuss the way of constructing the IsoGCN layers, which correspond to the isometric invariant and equivariant GCN layers. To formulate a model, we consider the following assumptions: 1) there are only attributes associated with vertices but edges; 2) graphs do not contain self-loops. \( G = (\mathcal{V}, \mathcal{E}) \) and \( d \) denote a graph and the dimension of the Euclidean space, respectively.

3.1 CONSTRUCTION OF A GENERAL NABLA ADJACENCY MATRIX

Before constructing IsoGCN, we need to define the general nabla adjacency matrix (GeNAM), which is the core of the IsoGCN concept. The proof of each proposition can be found in Appendix B.

Let \( |\mathcal{V}| \) denote the number of vertices. GeNAM \( \mathbf{G} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times d} \) is defined as follows:

\[
\mathbb{R}^d \ni \mathbf{G}_{ij} := \sum_{k, l \in \mathcal{V}, k \neq l} c_{ijkl}(\mathbf{x}_k - \mathbf{x}_l).
\]

(1)

where \( \mathbf{G}_{ij}, \mathbf{x}_i \in \mathbb{R}^d \), and \( c_{ijkl} \) denote 1-D slice of \( \mathbf{G} \), the position of the \( i \)th vertex, and the untrainable coefficient, respectively.
Proposition 3.1. GeNAM defined in equation (9) is translation invariant and orthogonal transformation equivariant, i.e. for any isometric transformation \( \forall t \in \mathbb{R}^3, U \in O(d), T : x \mapsto Ux + t \):

\[
T : G_{ijk} \mapsto \sum_l U_{kl} G_{ijl},
\]

(2)

where \( G_{ijk} \) denotes element \((i, j, k)\) of \( G \). Based on that observation, one can regard \( G \) as a collection of rank-1 tensors.

Let \( H^1 \in \mathbb{R}^{n \times m \times d} \) and \( H^0 \in \mathbb{R}^{m \times f} \) denote collections of rank-1 and rank-0 tensors, respectively \((n, m, f \in \mathbb{Z}^+, \text{ where } \mathbb{Z}^+ \text{ denotes the positive integers})\). Here, \( H^1 \otimes H^0 \in \mathbb{R}^{n \times m \times d} \) denotes the tensor product of \( H^1 \) and \( H^0 \) defined as follows:

\[
(H^1 \otimes H^0)_{ijk} := \sum_j H_{ijk}^1 H_{jl}^0.
\]

(3)

As the GeNAM is permutated accordingly with a permutation of vertices, the subsequent proposition follows.

Proposition 3.2. Let \( H^0 \in \mathbb{R}^{\mathbb{V} \times f} \) denote a collection of rank-0 tensors. The convolution of the GeNAM and that collection \((G \ast H^0)_{ijk} = G_{ijk} H_{jl}^0 \) is permutation equivariant, i.e. for the permutation matrix \( P_\pi \) corresponding to any permutation \( \pi \) and \( \forall k \in \{1, \ldots, d\} \),

\[
\pi : (G \ast H^0)_{ijk} \mapsto P_\pi G_{ijk} \ast H_\pi^0.
\]

(4)

Let \( H^1 \in \mathbb{R}^{n \times f} \) and \( H^0 \in \mathbb{R}^{m \times f} \) denote the contraction of \( H^1 \in \mathbb{R}^{n \times m \times d} \) and \( H^0 \in \mathbb{R}^{m \times f} \) defined as follows:

\[
(H^1 \otimes H^0)_{il} := \sum_{j,k} H_{ijk}^1 H_{jl}^0.
\]

(5)

As the contraction of GeNAMs can be interpreted as the inner product of each component in GeNAMs, the subsequent proposition follows.

Proposition 3.3. The contraction of GeNAMs \( G \odot G \) is isometric transformation invariant, i.e. for any isometric transformation \( \forall t \in \mathbb{R}^3, U \in O(d), T : x \mapsto Ux + t \):

\[
T : G \odot G \mapsto G \odot G.
\]

(6)

Let \( H^p \in \mathbb{R}^{n \times m \times d^p} \) and \( H^q \in \mathbb{R}^{m \times f \times d^q} \) denote the tensor product of the collections of rank-\( p \) tensors \( H^p \) and rank-\( q \) tensors \( H^q \) defined as follows:

\[
(H^p \otimes H^q)_{ilk_1 k_2 \ldots k_p m_1 m_2 \ldots m_q} := \sum_j H_{ijk_1 k_2 \ldots k_p}^p H_{jl m_1 m_2 \ldots m_q}^q.
\]

(7)

As the tensor product of GeNAMs can be interpreted as the tensor product of each component in GeNAMs, the subsequent proposition follows.

Proposition 3.4. The tensor product of GeNAMs \( G \otimes G \) is isometric transformation equivariant as a collection of rank-2 tensors, i.e. for any isometric transformation \( \forall t \in \mathbb{R}^3, U \in O(d), T : x \mapsto Ux + t \) and \( \forall i, j \in \{1, \ldots, |\mathbb{V}|\} \):

\[
T : (G \otimes G)_{ij} \mapsto U(G \otimes G)_{ij} U^T.
\]

(8)

Is possible to easily generalize this proposition to the tensors of higher ranks by defining the \( p \)th tensor power of \( G \) as follows

\[
\otimes^p G = G
\]

(9)

\[
\bigotimes^k G = \bigotimes^k G \otimes G.
\]

(10)

Namely, \( \bigotimes^k G \) is isometric transformation equivariant as a collection of rank-\( p \) tensors.
Moreover, the convolution can be generalized for the collections of rank-$p$ tensors $\mathbf{pH} \in \mathbb{R}^{n \times m \times d}$ and rank-$0$ tensors $\mathbf{qH}' \in \mathbb{R}^{m \times f \times d}$ as follows:

$$\left[ \mathbf{pH} \ast \mathbf{qH}' \right]_{lk_1k_2\ldots k_p} = \sum_j p_{lk_1k_2\ldots k_p} q_{lj_1j_2\ldots j_{p-1}} H'_{lj}. \quad (11)$$

The contraction can be generalized for the collections of rank-$p$ tensors $\mathbf{pH} \in \mathbb{R}^{n \times m \times d}$ and rank-$0$ tensors $\mathbf{qH}' \in \mathbb{R}^{m \times f \times d}$ as follows:

$$\left[ \mathbf{pH} \odot \mathbf{qH}' \right]_{lf} = \sum_{j,k_1,k_2\ldots k_p} p_{lj} q_{jk_1k_2\ldots k_p} H'_{jk_1k_2\ldots k_p}. \quad (12)$$

### 3.2 Construction of ISOGCN

ISOGCN utilizes GeNAM to realize the isometric transformation invariant and equivariant layers. Using the operations defined above, such as convolution, contraction, and tensor product, we can construct ISOGCN layers, which take the tensors of any rank as input and output with tensors of any rank, which can differ from that of the input.

#### 3.2.1 Isometric Transformation Invariant Layer

As it can be seen in Proposition 3.1, the contraction of GeNAMs is isometric transformation invariant. Therefore, the $l$th isometric transformation invariant layer with rank-$0$ input tensors $f^{(l)} : \mathbb{R}^{|V| \times f_{in}^{(l)}} \ni \mathbf{H}^{(l)} \mapsto \mathbf{H}^{(l+1)} \in \mathbb{R}^{|V| \times f_{out}^{(l)}}$, the activation function $\sigma^{(l)}$, and the trainable parameter matrix $W^{(l)} \in \mathbb{R}^{f_{in}^{(l)} \times f_{out}^{(l)}}$ can be constructed as

$$\mathbf{H}^{(l+1)} = \sigma^{(l)} \left( (\mathbf{G} \odot \mathbf{G}) \mathbf{H}^{(l)} W^{(l)} \right). \quad (13)$$

By defining $\mathbf{L} := \mathbf{G} \odot \mathbf{G} \in \mathbb{R}^{|V| \times |V|}$, it can be simplified as follows:

$$\mathbf{H}^{(l+1)} = \sigma^{(l)} \left( \mathbf{LH}^{(l)} W^{(l)} \right), \quad (14)$$

which has the same form as GCN, except the fact that the renormalized adjacency matrix is replaced with $\mathbf{L}$.

The isometric transformation invariant layer with the rank-$1$ input tensors $\mathbf{1H}^{(l)}$ can be formulated as follows:

$$\mathbf{H}^{(l+1)} = \sigma^{(l)} \left( \mathbf{G} \odot \mathbf{1H}^{(l)} W^{(l)} \right). \quad (15)$$

As these approaches utilize the inner products of vectors in $\mathbb{R}^d$, these operations correspond to the extractions of a relative distance and an angle of each pair of vertices, which are employed explicitly in [Klicpera et al. (2020)](https://www.jmlr.org/papers/v21/19-1244.html).

In general, the $l$th isometric transformation invariant layer having the input features with the rank-$0$ tensors $\mathbf{0H}^{(l)}$, rank-$1$ tensors $\mathbf{1H}^{(l)}$, rank-$2$ tensors $\mathbf{2H}^{(l)}$, ..., rank-$k$ tensors $\mathbf{kH}^{(l)}$ can be constructed as follows:

$$\mathbf{H}^{(l)} = \left[ (\mathbf{G} \odot \mathbf{G}) \mathbf{0H}^{(l)} \mathbf{G} \odot \mathbf{1H}^{(l)} (\mathbf{G} \odot \mathbf{G}) \odot \mathbf{2H}^{(l)} \ldots (\mathbf{G} \odot \mathbf{G}) \odot \ldots \odot \mathbf{kH}^{(l)} \right] \quad (16)$$

$$\mathbf{H}^{(l+1)} = \sigma^{(l)} \left( \mathbf{H}^{(l)} W^{(l)} \right) \quad (17)$$

where $\|$ denotes concatenation in the feature direction.

#### 3.2.2 Isometric Transformation Equivariant Layer

To construct the isometric transformation equivariant layer, we convert the input feature isometric transformation invariant. This is performed as the present research is to develop computationally feasible GNN layers, and therefore, we seek to avoid computational overhead, such as in the case of Fourier space modeling. If an input feature remain isometric transformation equivariant, the layer
Horie, Morita, Ihara, and Mitsume would represent an inappropriate distortion of isometry, as in general, isometric transformation does not commute with the nonlinear activation function. As a result, the orthogonal transformation equivariant layer as a collection of rank-$k$ tensors can be defined as follows:

$$kH^{(l+1)} = \bigotimes_{i=1}^{k} G \ast \sigma^{(l)}(\hat{H}^{(l)}W^{(l)}),$$  \hspace{1cm} (18)

where $\hat{H}^{(l)}$ is an invariant feature converted using GeNAM as in equation 16. To achieve translation equivariance, one can add a constant tensor to equation 18, i.e. $kH^{(l+1)} = \bigotimes_{i=1}^{k} G \ast \sigma^{(l)}(\hat{H}^{(l)}W^{(l)}) + kT$, where $kT$ is a rank-$k$ constant tensor.

3.2.3 DESIGN RULE OF ISOGCN

The minimum design rule of IsoGCN can be formulated as follows:

1. Check the ranks of tensors for the input and output features.
2. If the input feature is a rank-$k \geq 1$ tensor, contract the input feature with the $k$th tensor power of GeNAM $\bigotimes^{k} G \ast H^{(l)}$. Otherwise, perform matrix multiplication $(G \otimes G)H^{(l)}$.
3. If the output feature is a rank-$k \geq 1$ tensor, convolve with the activation of the layer with $k$ production of GeNAM i.e., $\bigotimes^{k} G \ast \sigma^{(l)}(\hat{H}^{(l)}W^{(l)})$. Otherwise, output the activation $\sigma^{(l)}(\hat{H}^{(l)}W^{(l)})$.
4. If one needs establishing translation equivariant in addition to orthogonal transformation equivariant, add a constant tensor with the same rank as the output.

Although that design rule can be used to construct the IsoGCN layers that considers the tensors of any rank as an output and an output, this is the minimum required rule. In other words, the forms of IsoGCN that do not comply with this rule are also possible, for example, such as $G \otimes G \ast L\sigma^{(l)}(L(G \otimes G) \otimes (G \otimes G)G \otimes H^{(l)})$ for a rank-1 tensor input and a rank-2 tensor output. It can be considered as the IsoGCN version of simple graph convolution (Wu et al., 2019).

As implied by the discussion presented in the last paragraph, only the ranks corresponding to the input and the output of the entire network are important. It means that one can add any layer mapping to tensors of any rank as long as the computation is valid, and the ranks corresponding to the input and the output of the entire network are the same as the required ranks. For instance, to realize a neural network mapping to a rank-2 tensor to a rank-1 tensor, one can construct an IsoGCN with the following structure: (rank-2 tensor) $\rightarrow$ (rank-3 tensor) $\rightarrow$ (rank-0 tensor) $\rightarrow$ (rank-1 tensor). Moreover, it is possible to incorporate residual connection (He et al., 2016) into the IsoGCN layers if deemed necessary, as performed in Section 4.1.

4 EXPERIMENT

To test the applicability of the proposed model, we composed a dataset using the linear heat equation and then, applied the proposed approach to the obtained dataset. We considered the task of predicting temperature field based on initial temperature field and mesh geometry information as inputs. We selected 82 CAD shapes randomly considering the first 200 shapes of the ABC dataset (Koch et al., 2019), generate first-order tetrahedral meshes with the mesher program Gmsh (Geuzaine & Remacle, 2009), and finally, performed finite element analysis (FEA) with the FEA program FrontISTR (Morita et al., 2016; Ihara et al., 2017). For more details about the used dataset, see Appendix D.1.

4.1 NEURAL NETWORK MODEL

To transform the FEA data structure into graphs, we denoted nodes and edges in meshes as vertices and edges in graphs, respectively. Then, to stabilize computation, we considered vertices up to five hops as the adjacent ones.
Table 1: Correspondence between the differential operators and the expressions using the nabla adjacency matrix $\tilde{D}$.

| Differential operator | Expression using $\tilde{D}$ |
|-----------------------|-----------------------------|
| Gradient              | $\tilde{D} \ast 0$H         |
| Divergence            | $\tilde{D} \odot 1$H        |
| Laplacian             | $\tilde{D} \odot \tilde{D}^0$H |
| Jacobian              | $\tilde{D} \odot 1$H        |
| Hessian               | $\tilde{D} \odot \tilde{D} \ast 0$H |

As a concrete instance of GeNAM, we defined a nabla adjacency matrix $\tilde{D} \in \mathbb{R}^{V \times V \times d}$ formulated as follows:

$$\tilde{D}_{ijk} = D_{ijk} - \delta_{ij} \sum_l D_{ilk},$$

(19)

$$D_{ijk} = d \frac{x_{jk} - x_{ik}}{\|x_j - x_i\|^2} \sum_l V_{\text{effective}} A_{ij} V_{\text{effective}} A_{il},$$

(20)

where $A$ and $V_{\text{effective}}$ were the adjacency matrix and the effective volume of the $i$th vertex (equation 52), respectively. Here, $d = 3$ because the meshes in the dataset are graphs embedded in the 3D Euclidean space. As both $D$ and $\tilde{D}$ were qualified as GeNAMs, they provided translation invariance and orthogonal transformation equivariance.

Here, $\tilde{D}$ was designed so that $\tilde{D} \odot \tilde{D}$ corresponded to a Laplacian operator (see Appendix C). As presented in Table 1 and Appendix C, $\tilde{D}$ was closely related to many differential operators such as gradient, divergence, Laplacian, Jacobian, and Hessian ones. Therefore, the considered nabla adjacency matrix played an essential role in constructing the neural network models that are capable of learning differential equations.

We constructed a neural network model considering the encode-process-decode configuration (Battaglia et al., 2018). The encoder and decoder comprised component-wise MLPs $\sigma(l) \left( \sum_j H^{(l)}_{ij} W^{(l)}_{jl} \right)$, and the processor included two IsoGCN layers employing $\tilde{D} : D$ as $L$ in equation 13. The IsoGCN layers also had residual connections, so that an IsoGCN layer with residual connection could be described as follows: $\sigma(l) \left( H^{(l)} + \tilde{D} \odot \tilde{D} \right) W^{(l)}$, which corresponds to the first-order Taylor expansion $H^{(l+1)} \approx H^{(l)} + \nabla^2 H^{(l)} \Delta t$. By stacking this layer $p$ times, one could approximate the $p$th order Taylor expansion of the heat equation.

We implemented the proposed models using PyTorch (Paszke et al., 2019). The corresponding implementation and the dataset will be available online.

4.2 Evaluation

As the baseline models, we selected the component-wise MLP and the GCN ones (Kipf & Welling, 2017), as the proposed approach was based on the GCN model. To enable a fair comparison, we implemented the GCN model using the same encode-process-decode configuration except for the processor part that was replaced with the GCN layers. Considering that the proposed model considered the vertices up to 10 hops away in one layer and 20 hops in total, we utilized the GCN layers up to 20 stacks for testing, with and without residual connections. We applied the Adam optimizer (Kingma & Ba, 2014) and Optuna (Akiba et al., 2019) to optimize the learning rate.

1By ignoring all edges in the graph, one can easily understand that the component-wise MLP is also one variant of GCN, meaning that it also has permutation equivariance. In fact, by ignoring all edges, the adjacency matrix after the renormalization trick (Kipf & Welling, 2017) is $I$, which makes $\sigma(l) (I H^{(l)} W^{(l)}) = \sigma(l) (H^{(l)} W^{(l)})$. 

6
Although we used MSE loss of temperatures over the vertices during training, to evaluate the performance, we used the $L^2$ squared error metric defined below:

$$E^2_{L^2} = \frac{\sum_{e \in \text{Elements}} V_e (T_{\text{inference}}^e - T_{\text{target}}^e)^2}{\sum_{e \in \text{Elements}} V_e},$$  \hspace{1cm} (21)$$

where $\text{Elements}$, $V_e$, $T_{\text{inference}}^e$, and $T_{\text{target}}^e$ were the set of elements in the meshes, volume of element $e$, the inferred temperature at the center of mass of element $e$, and the target temperature at the center of mass of element $e$, respectively.

Figure 1 and Table 2 present the results of the qualitative and quantitative comparisons for the test dataset. It can be seen that IsoGCN demonstrates better prediction performance compared with GCN models, which also has high prediction performance.

Table 2: Summary of the training results with the loss (mean squared error) and $E^2_{L^2}$ of the test dataset ($\pm$ the standard error of the mean). Here, “$n \times m$” in the “# hops” column means $m$ residual blocks with $n$ layers for each. For more details, see 3.
Horie, Morita, Ihara, and Mitsume

Figure 2: Comparison between (left) a mesh in the training dataset, (center) ground truth computed through FEA, (right) IsoGCN inference result. One can see that IsoGCN can predict the temperature field for a mesh, which is much larger than those in the training dataset. The $L^2$ squared error for this FEA results is $E^2_{L^2} = 4.62 \times 10^{-3}$.

Moreover, due to the isometric transformation invariant nature of IsoGCN, it also achieved the high prediction performance for the meshes that were significantly larger than those considered in the training dataset (Figure 2). IsoGCN can scales up to 1M vertices, which is a practical scale and considerably greater than previous methods. Therefore, we conclude that the IsoGCN models could be trained on a relatively smaller meshes to save the training time and then, use it to perform inference of larger meshes without observing significant performance deterioration.

As NT & Maehara (2019) has outlined that GNNs could be considered as low-pass filters, it was deemed reasonable that the IsoGCN model could learn the heat equation appropriately. To demonstrate the predictive power of IsoGCNs, additional experiments in other phenomena are also planned.

5 CONCLUSION

In the present work, we proposed the GCN-based isometric transformation invariant and equivariant models called IsoGCN. We discussed an example of a general nabla adjacency matrix (GeNAM) that was closely related to essential differential operators. The experiment results confirmed that the proposed model leveraged spatial structures and could deal with large-scale graphs. The experiment was conducted on the mesh structured dataset based on FEA results. However, we expect IsoGCN to be applicable to various domains, such as object detection, molecular properties prediction, and physical simulations using particles.

ACKNOWLEDGMENTS

The authors gratefully acknowledge hardware donations by NVIDIA.
REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

Ferran Alet, Adarsh Keshav Jeewajee, Maria Bauza Villalonga, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Kaelbling. Graph element networks: adaptive, structured computation and memory. In *ICML*, 2019.

Igor I Baskin, Vladimir A Palyulin, and Nikolai S Zefirov. A neural device for searching direct correlations between structures and properties of chemical compounds. *Journal of chemical information and computer sciences*, 37(4):715–721, 1997.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Matteo Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999, 2016.

Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. In *ICLR*, 2018.

Taco S Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. *ICML*, 2019.

Christophe Geuzaine and Jean-François Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yu Ihara, Gaku Hashimoto, and Hiroshi Okuda. Web-based integrated cloud cae platform for large-scale finite element analysis. *Mechanical Engineering Letters*, 3:17–00520, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *ICLR*, 2020.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
Risi Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv preprint arXiv:1803.01588*, 2018.

Seiichi Koshizuka and Yoshiaki Oka. Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nuclear science and engineering*, 123(3):421–434, 1996.

Naoki Morita, Kazuo Yonekura, Ichiro Yasuzumi, Mitsuyoshi Tsunori, Gaku Hashimoto, and Hirosi Okuda. Development of $3 \times 3$ dof blocking structural elements to enhance the computational intensity of iterative linear solver. *Mechanical Engineering Letters*, 2:16–00082, 2016.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alch´e-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242*, 2018.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, 2019.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W Battaglia. Learning to simulate complex physics with graph networks. *arXiv preprint arXiv:2002.09405*, 2020.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *NeurIPS*, pp. 10381–10392, 2018.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, pp. 6861–6871. PMLR, 2019.
A  NOTATION

\[ G \]  A graph
\[ V \]  A vertex set
\[ |V| \] The number of vertices
\[ \mathcal{E} \] An edge set
\[ \mathbb{Z}^+ \] The positive integers
\[ d \] The dimension of the Euclidean space
\[ x_i \] The position of the \( i \)th vertex
\[ x_{ik} \] Element \( i \) of \( x_i \)
\[ G \in \mathbb{R}^{|V| \times |V| \times d} \] The general nabla adjacency matrix (GeNAM) (equation 1)
\[ G_{ij} : \in \mathbb{R}^d \] 1-D slice of \( G \) (equation 1)
\[ G_{ijk} \in \mathbb{R} \] Element \((i,j,k)\) of \( G \)
\[ pH \in \mathbb{R}^{n \times f \times d^p} \] A collection of rank-\( p \) tensors \((n, f \in \mathbb{Z}^+)\)
\[ pH \circ 0H' \] Convolution of collections of rank-\( p \) tensors \( pH \) and rank-0 tensors \( 0H' \) (equation 3, equation 11)
\[ pH \otimes pH' \] Contraction of collections of rank-\( p \) tensors \( pH \) and \( pH' \) (equation 5, equation 12)
\[ pH^{(l)} \] The input feature of the \( l \)th layer with rank-\( p \) tensors
\[ \sigma^{(l)} \] The activation function of the \( l \)th layer
\[ W^{(l)} \] The trainable parameter matrix of the \( l \)th layer
\[ A \] An adjacency matrix
\[ \delta_{ij} \] The Kronecker delta
\[ V_{i,\text{effective}} \] The effective volume of the \( i \)th vertex (equation 52)
\[ V_{i,\text{mean}} \] The mean volume of the \( i \)th vertex (equation 53)
\[ \tilde{D} \] A nabla adjacency matrix (equation 19)
\[ E_{L^2}^2 \] The \( L^2 \) squared error metric (equation 21)

B  PROOFS OF PROPOSITIONS

In this section, we present the proofs of propositions in Section 3. Let \( \mathbb{R}^3 \ni g(x_i, x_k) = (x_k - x_i) \).

B.1 PROOF OF PROPOSITION 3.1

Proof. First, we demonstrate invariance with respect to translation with \( \forall t \in \mathbb{R}^d \). Each component of GeNAM is transformed as follows under translation:

\[
g(x_i + t, x_j + t) = [x_j + t - (x_i + t)]
= (x_j - x_i)
= g(x_i, x_j)
\]  (22)
Thus,
\[
\sum_{k,l \in V, k \neq l} c_{ijkl}(x_l + t, x_k + t) = \sum_{k,l \in V, k \neq l} c_{ijkl}(x_l, x_k)
= G_{ij}.
\] (23)

Then, we show equivariance with respect to orthogonal transformation with \( \forall U \in O(d) \). Each component of GeNAM is transformed as follows under orthogonal transformation:
\[
g(Ux_i, Ux_j) = Ux_j - Ux_i = U(x_j - x_i) = Ug(x_i, x_j)
\] (24)
Thus,
\[
\sum_{k,l \in V, k \neq l} c_{ijkl}(Ux_l, Ux_k) = \sum_{k,l \in V, k \neq l} c_{ijkl}Ug(x_l, x_k)
= U_{mn}G_{ijn}.
\] (25)
Therefore, \( G \) is translation invariant and orthogonal transformation equivariant.

B.2 Proof of Proposition 3.2

Proof. Let \( P_\pi \) be the permutation matrix corresponding to \( \pi \), and define \( ^0H = H \). The convolution of vertex features with GeNAM \( G * H \) is transformed as follows under the permutation \( \pi \):
\[
(\pi(G) * \pi(0H))_{:k} = P_\pi G_{:k} P_\pi^T P_\pi H
= P_\pi G_{:k} H.
\] (26)

It should be noted that \( G_{:k} \) can be regarded as a matrix. Therefore, \( G * H \) is permutation equivariant.

B.3 Proof of Proposition 3.3

Proof. \( G \odot G \) is translation invariant because \( G \) is translation invariant. Here, we prove rotation invariance under an orthogonal transformation \( \forall U \in O(n) \). \( G \odot G \) is transformed under the \( U \) as follows:
\[
\sum_{j,k} G_{ijk} G_{jlk} \mapsto \sum_{k,l,m,n} U_{km} G_{ijk} U_{kn} G_{jln}
= \sum_{k,l,m,n} U_{km} U_{kn} G_{ijk} G_{jln}
= \sum_{k,l,m,n} U^T_{mk} U_{kn} G_{ijk} G_{jln}
= \sum_{l,m,n} \delta_{mn} G_{ijk} G_{jln} \quad (\because \text{property of the orthogonal matrix})
= \sum_{l,m,n} G_{ijk} G_{jln}
\]
\[
= \sum_k G_{ijk} G_{jlk}. \quad (\because \text{Change the dummy index } m \to k).
\] (27)
Therefore, \( G \odot G \) is isometric transformation invariant.
B.4 Proof of Proposition 3.4

Proof. $G \otimes G$ is transformed under $\forall U \in O(n)$ as follows:

$$
\sum_j G_{ijk} G_{jlm} \mapsto \sum_{n,o} U_{kn} G_{ijn} U_{mo} G_{jlo}
$$

By regarding $G_{ijn} G_{jlo}$ as one matrix $H_{no}$, it follows the coordinate transformation of rank-2 tensor $U H U^T$.

C Physical Intuition of $\tilde{D}$

In this section, we discuss the connection between the nabla adjacency matrix $\tilde{D}$ and the differential operators such as the gradient, the divergence, the Laplacian, the Jacobian, and the Hessian.

Let $pH \in \mathbb{R}^{V \times f \times d^p}$ denote a collection of rank-$p$ tensors ($p \in \mathbb{Z}_0^+$, $f \in \mathbb{Z}^+$). Let us assume a partial derivative model $\partial pH/\partial x_k \in \mathbb{R}^{V \times f \times d^p}$ ($k \in \{1, \ldots, d\}$), that is based on the gradient model in the moving particle semi-implicit method (Koshizuka & Oka, 1996).

$$
\left( \frac{\partial pH}{\partial x_k} \right)_{igk_1k_2...k_p} := d \sum_{j} pH_{jgk_1k_2...k_p} x_j - x_i \sum_{l} w_{il} w_{ij}
$$

$$
= \sum_j D_{ijk} (pH_{jgk_1k_2...k_p} - pH_{ligk_1k_2...k_p})
$$

where $w_{ij}$ is an untrainable function. In the case of the proposed model defined in equation 19 and equation 20, $w_{ij} = (V_j/V_i) A_{ij}$ and $d = 3$. Although one could define $w_{ij}$ as a function of the distance $\|x_j - x_i\|$, we kept $w_{ij}$ constant with respect to the distance to keep the model simple with fewer hyperparameters.

C.1 Gradient

$\tilde{D}$ can be viewed as a Laplacian matrix based on $D$; however, $\tilde{D} \ast 0H$ can be interpreted as the gradient in the Euclidean space. Let $\nabla 0H \in \mathbb{R}^{V \times f \times d}$ the approximation of the gradient of $0H$.

Using equation 30 the gradient model can be expressed as

$$
(\nabla 0H)_{igk} = \frac{\partial 0H_{ig}}{\partial x_k}
$$

$$
= D_{ijk} (0H_{jg} - 0H_{ig}).
$$

Using this gradient model, we can confirm that $(\tilde{D} \ast 0H)_{igk} = (\nabla 0H)_{igk}$ because

$$
\left( \tilde{D} \ast 0H \right)_{igk} = \sum_j \tilde{D}_{ijk} 0H_{jg}
$$

$$
= \sum_j (D_{ijk} - \delta_{ij} \sum_l D_{ilk}) 0H_{jg}
$$

$$
= \sum_j D_{ijk} 0H_{jg} - \delta_{ij} \sum_l D_{ilk} 0H_{jg}
$$

$$
= \sum_j D_{ijk} 0H_{jg} - \delta_{ij} D_{ilk} 0H_{jg}
$$

$$
= \sum_j D_{ijk} 0H_{jg} - \delta_{ij} D_{ilk} 0H_{jg}
$$

(\because \text{Change the dummy index } l \rightarrow j)

$$
= \sum_j D_{ijk} (0H_{jg} - 0H_{ig})
$$

$$
= (\nabla 0H)_{igk}
$$
Therefore, $\tilde{D}^*$ can be interpreted as the gradient operator in the Euclidean space.

C.2 DIVERGENCE

We show that $\tilde{D} \odot 1^H$ corresponds to the divergence. Using $D$, the divergence model $\nabla \cdot 1^H \in \mathbb{R}^{|V| \times f}$ is expressed as follows:

$$
(\nabla \cdot 1^H)_{ig} = \left( \sum_k \frac{\partial 1^H}{\partial x_k} \right)_{ig} = \sum_{j,k} D_{ijk} (1^H_{jgk} - 1^H_{igk}) \tag{35}
$$

Then, $\tilde{D} \odot 1^H$ is

$$
(\tilde{D} \odot 1^H)_{ig} = \sum_{j,k} \tilde{D}_{ijk} 1^H_{igk}
$$

$$
= \sum_{j,k} \left( D_{ijk} - \delta_{ij} \sum_l D_{jlk} \right) 1^H_{igk}
$$

$$
= \sum_{j,k} D_{ijk} 1^H_{jgk} - \sum_{l,k} D_{ilk} 1^H_{igk}
$$

$$
= \sum_{j,k} D_{ijk} (1^H_{jgk} - 1^H_{igk}) \quad (\because \text{Change the dummy index } l \to j)
$$

$$
= (\nabla \cdot 1^H)_{ig}. \tag{37}
$$

C.3 LAPLACIAN OPERATOR

We prove that $\tilde{D} \odot \tilde{D}$ corresponds to the Laplacian operator in the Euclidean space.

Using equation 30 the Laplacian model $\nabla^2 0^H \in \mathbb{R}^{|V| \times f}$ can be expressed as follows:

$$
(\nabla^2 0^H)_{ig} := \sum_k \left[ \frac{\partial}{\partial x_k} \left( \frac{\partial H}{\partial x_k} \right)_{ig} \right]_{\partial x_k}
$$

$$
= \sum_{j,k} D_{ijk} \left[ \left( \frac{\partial H}{\partial x_k} \right)_{jg} - \left( \frac{\partial H}{\partial x_k} \right)_{ig} \right]
$$

$$
= \sum_{j,k} D_{ijk} \left[ \sum_l D_{jlk} 0^H_{ig} - 0^H_{jg} - \sum_l D_{ilk} 0^H_{ig} - 0^H_{ig} \right]
$$

$$
= \sum_{j,k,l} D_{ijk} (D_{jlk} - D_{ilk}) (0^H_{ig} - 0^H_{jg}). \tag{38}
$$
Then, \((\bar{D} \otimes \bar{D})^0H\) is

\[
((\bar{D} \otimes \bar{D})^0H)_{ig} = \sum_{j,k,l} D_{ijk} D_{jlk}^0 H_{ig}
\]

\[
= \sum_{j,k,l} \left( D_{ijk} - \delta_{ij} \sum_m D_{mkl} \right) \left( D_{jlk} - \delta_{jl} \sum_n D_{ijn} \right)^0 H_{ig}
\]

\[
= \sum_{j,k,l} D_{ijk} D_{jlk}^0 H_{ig} - \sum_{j,k,n} D_{ijk} D_{jnk}^0 H_{ig}
\]

\[
- \sum_{k,l,m} D_{imk} D_{ilk}^0 H_{lg} + \sum_{k,j,m} D_{imk} D_{ink}^0 H_{lg}
\]

\[
\quad (\because \text{Change the dummy index } m \rightarrow j \text{ for the third and fourth terms})
\]

\[
= \sum_{j,k,l} D_{ijk} (D_{jlk} - D_{ilk})^0 H_{ig} - D_{ijl}^0 H_{ig} - D_{lijk}^0 H_{lg}
\]

\[
\quad (\because \text{Change the dummy index } n \rightarrow l \text{ for the second and fourth terms})
\]

\[
= (\nabla^2 0H)_{ig} \quad (39)
\]

C.4 JACOBIAN AND HESSIAN OPERATORS

Considering a similar discussion, we can show the following dependences. For the Jacobian model: \(J^1[H] \in \mathbb{R}^{|V| \times f \times d \times d}\),

\[
(J^1[H])_{igkl} = \left( \frac{\partial 1H}{\partial x_l} \right)_{igk} \quad (40)
\]

\[
= \sum_j D_{ijkl}^1 (H_{jgk} - H_{igk}) \quad (41)
\]

\[
= (\tilde{D} \otimes 1H)_{igkl} \quad (42)
\]

For the Hessian model: \(\text{Hess}^0[H] \in \mathbb{R}^{|V| \times f \times d \times d}\),

\[
(\text{Hess}^0[H])_{igkl} = \left( \frac{\partial}{\partial x_l} \frac{\partial}{\partial x_k} 0H \right)_{ig} \quad (43)
\]

\[
= \sum_{j,m} D_{ijkl}^m [D_{jml}^1 (H_{mg} - H_{ig}) - D_{imk} (H_{mg} - H_{ig})] \quad (44)
\]

\[
= [(\tilde{D} \otimes \tilde{D}) \ast \ 0H]_{igkl} \quad (45)
\]
D EXPERIMENTS DETAILS

D.1 DATASET

The purpose of the conducted experiment is to solve the linear heat diffusion under an adiabatic boundary condition. The governing equation is defined as follows:

$$\Omega \subset \mathbb{R}^3,$$

$$\frac{\partial T(x, t)}{\partial t} = D \nabla^2 T(x, t), \text{ in } \Omega,$$

$$T(x, t = 0) = T_{\text{init}}(x), \text{ in } \Omega,$$

$$\nabla T(x, t) \big|_{x = x_b} \cdot n(x_b) = 0, \text{ on } \partial \Omega,$$

where $T$, $T_{\text{init}}$, $D$, and $n(x_b)$ are the temperature field, initial temperature field, diffusion coefficient, and the normal vector at $x_b \in \partial \Omega$, respectively. We set $D = 10^{-2}$ and predict temperature fields at $t = 1.0$. Accordingly, the diffusion number of this problem is $D \Delta t / (\Delta x)^2 = 10.0^4$ with assuming $\Delta x = 10.0^{-3}$.

Figure 3 represents the process of generating the dataset. We generated up to 27 FEA results for each CAD shape. To avoid data leakage in terms of CAD shapes, we first split them into train, validation, and test datasets and then, performed the following process.

Using one CAD shape, we generated up to three meshes with clscale (control parameter of the mesh characteristic lengths) = 0.20, 0.25, and 0.30. To facilitate the training process, we scaled the meshes to fit in the cube with the edge length equal to 1.

Using one mesh, we generated nine initial conditions as through following steps:

1. Generate smooth temperature field $T^{\text{Fourier}}$ with Fourier series from 2nd to 10th orders.
2. To control the “steepness” of the temperature field, apply transformation defined as follows:

$$\tilde{T} = \frac{T_{\text{Fourier}} - \min(T_{\text{Fourier}})}{\max(T_{\text{Fourier}}) - \min(T_{\text{Fourier}})}$$

$$f_{\text{filter}}(\tilde{T}; \alpha) = \frac{\tanh(\alpha \tilde{T})}{\tanh(\alpha)}$$

with $\alpha = 1.0, 5.0$, and 10.0

We generated three initial conditions for each $\alpha$ value with the setting coefficients of the Fourier series randomly. As we obtained three values of $\alpha$, we had nine initial conditions for each mesh. Then, we performed FEA for each initial condition. We applied the implicit method to solve time evolutions and the direct method to solve linear equations. The time step of FEA was set equal to 0.01.

In this process, some meshes or FEA results might not be available due to excessive computation time or non-convergence. Therefore, the size of the dataset was not precisely equal to the number multiplied by 27. Finally, we obtained 1290 FEA results for the training dataset, 411 FEA results for the validation dataset, and 423 FEA results for the test dataset.

D.2 INPUT FEATURES

To express the geometry information, we extracted the effective volume of the $i$th vertex $V_{i, \text{effective}}$ and the mean volume of the $i$th vertex $V_{i, \text{mean}}$ defined as follows:

$$V_{i, \text{effective}} = \sum_{e \in \mathcal{N}_i^e} \frac{1}{4} V_e,$$

$$V_{i, \text{mean}} = \frac{\sum_{e \in \mathcal{N}_i^e} V_e}{|\mathcal{N}_i^e|},$$

where $\mathcal{N}_i^e$ was the set of elements including the $i$th vertex. In the case of GCN models, we used the renormalized adjacency matrix $\hat{A}$ in Kipf & Welling (2017).

For MLP and GCN models, we tested several combinations of input features $T_{\text{init}}$, $V_{\text{effective}}$, $V_{\text{mean}}$, and $x$ (Table 3). For the IsoGCN model, inputs were $T_{\text{init}}$, $V_{\text{effective}}$, and $V_{\text{mean}}$. 
Figure 3: The process of generating the dataset. Smaller clscale parameter generates smaller meshes, and larger $\alpha$ generates a steeper temperature field.

D.3 MODEL ARCHITECTURES

Concerning all models, the encoder and the decoder were the component-wise MLPs with nodes $(f_{\text{input}}, 128, 512)$ and $(512, 512, 128, 1)$, respectively (where $f_{\text{input}}$ is the dimension of input features.) For all models, the number of nodes in each layer within the processor was 512.

For MLP and IsoGCN models, the number of layers in the processor was two. Concerning GCN models, the number of layers in the processor corresponds to the number of hops, as outlined in Table 3.

We used the $\tanh$ activation function except for the output layer because we expect the target temperature field is smooth. Therefore, we avoid using non-differentiable activation functions such as the rectified linear unit (ReLU) [Nair & Hinton, 2010]. The activation function of the output layer was identity.

D.4 RESULT DETAILS

Table 3 represents the detailed comparison of training results. Including $x$ in the input features of GCN models did not improve the performance. In addition, if $x$ is included in the input features, it might lead to the loss of the generalization capacity for larger shapes, as it extrapolates. The proposed model achieved the best performance compared with the considered baseline models. Therefore, we concluded that the essential features regarding mesh shapes are included in $\tilde{D}$. 

17
Table 3: Summary of the training results with the loss (mean squared error) and $E^2_{L_2}$ of the test dataset (± the standard error of the mean). Here, “$n \times m$” in the “# hops” column means $m$ residual blocks with $n$ layers for each.

| Method      | # hops | Residual | Inputs                                      | Loss $\times 10^{-2}$ | $E^2_{L_2} \times 10^{-3}$ |
|-------------|--------|----------|---------------------------------------------|------------------------|-----------------------------|
| MLP         | 0      | No       | $T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}$ | 4.927 ± 0.005          | 50.50 ± 0.37                |
| GCN         | 2      | No       | $\hat{A}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}$ | 3.897 ± 0.004          | 16.93 ± 0.21                |
| GCN         | $1 \times 2$ | Yes     | $\hat{A}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}$ | 3.779 ± 0.004          | 15.34 ± 0.20                |
| GCN         | 10     | No       | $\hat{A}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}$ | 4.009 ± 0.004          | 10.40 ± 0.17                |
| GCN         | 10     | No       | $\hat{A}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}, x$ | 3.653 ± 0.004          | 9.85 ± 0.16                 |
| GCN         | $5 \times 2$ | Yes     | $\hat{A}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}$ | 3.340 ± 0.004          | 6.50 ± 0.13                 |
| GCN         | $5 \times 2$ | Yes     | $\hat{A}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}, x$ | 3.432 ± 0.004          | 7.36 ± 0.14                 |
| GCN         | 20     | No       | $\hat{A}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}$ | 3.557 ± 0.004          | 6.04 ± 0.13                 |
| GCN         | 20     | No       | $\hat{A}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}, x$ | 4.368 ± 0.004          | 8.36 ± 0.15                 |
| GCN         | $10 \times 2$ | Yes    | $\hat{A}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}, x$ | 3.144 ± 0.004          | 3.99 ± 0.10                 |
| GCN         | $10 \times 2$ | Yes    | $\hat{A}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}, x$ | 3.118 ± 0.004          | 4.48 ± 0.11                 |
| IsoGCN (Ours) | $10 \times 2$ | Yes    | $\hat{D}, T_{\text{init}}, V_{\text{effective}}, V_{\text{mean}}$ | $1.617 \pm 0.002$     | $3.52 \pm 0.10$            |