ISOMETRIC TRANSFORMATION INVARIANT AND EQUIVARIANT GRAPH CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graphs are one of the most important data structures for representing pairwise relations between objects. Specifically, a graph embedded in a Euclidean space is essential to solving real problems, such as object detection, structural chemistry analyses, and physical simulation. A crucial requirement to applying a graph in a Euclidean space is learning the isometric transformation invariant and equivariant features. In the present paper, we propose a set of transformation invariant and equivariant models based on graph convolutional networks (GCNs), called IsoGCNs. We demonstrate that the proposed model outperforms state-of-the-art methods on tasks related with geometrical and physical data. Moreover, the proposed model can scale up to the graphs with 1M vertices and conduct an inference faster than a conventional finite element analysis.

1 INTRODUCTION

Graph-structured data embedded in a Euclidean space can be utilized in many different fields such as object detection, structural chemistry analysis, and physical simulation. Graph neural networks (GNNs) have been introduced to deal with such data. Crucial properties of a GNN include its permutation invariance and equivariance. In addition to permutations, isometric transformation invariance and equivariance must be addressed when considering the graphs in a Euclidean space because many properties of objects in the Euclidean space do not change under translation and rotation. Therefore, the formulation of the isometric transformation invariant and equivariant models is critical. In addition to the invariance and equivariance, computational efficiency is another crucial factor, particularly 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) (Kipf & Welling, 2017). Specifically, by simply tweaking the definition of an adjacency matrix, the proposed model is sufficiently simple for realizing isometric transformation invariance. Because 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. Moreover, the proposed model showed faster inference compared to a conventional finite element analysis approach at the same level of accuracy.

The main contributions of the present paper can be summarized as follows:

- We construct isometric invariant and equivariant GCNs, called IsoGCNs for the specified input and output tensor ranks.
- We demonstrate that an IsoGCN model outperforms GCN and its variant models on the considered tasks related to physical simulations.
- We confirm that IsoGCN is sufficiently efficient computationally to scale up to graphs with 1M vertices and achieves inference considerably faster than a conventional finite element analysis.

2 RELATED WORK

Graph neural networks. The concept of a GNN was first proposed by Baskin et al. (1997); Sperduti & Starita (1997) and then improved by (Gori et al., 2005; Scarselli et al., 2008). Although many variants of GNNs have been proposed, including those presented by Hamilton et al. (2017); Veličković et al. (2017), such models have been unified under the concept of message passing neural networks (Gilmer et al., 2017). Among these variants, the use of a GCN developed by Kipf & Welling (2017), which is a considerable simplification of a GNN, is essential to the present study, because the proposed model is based on GCNs.

Invariant and equivariant neural networks. A function $f : X \to Y$ is said to be equivariant to a group G when $f(g \cdot x) = g \cdot f(x)$ for all $g \in G$ and $x \in X$, assuming the group G acts on both X and Y. In particular, when $f(g \odot x) = f(x)$, f is said to be invariant to the group G. Group equivariant convolutional neural networks were first proposed by Cohen & Welling (2016) for discrete groups. Subsequent studies have categorized such networks into continuous groups (Cohen et al., 2018), three-dimensional data (Weiler et al., 2018), and general manifolds (Cohen et al., 2019). Thomas et al. (2018); Kondor (2018) discussed the way to provide rotation equivariance to point clouds. In addition, Klicpera et al. (2020) proposed the use of GNNs with rotation invariance by applying scalar values, such as the distances and angles between vertices, to represent the spatial structure of a graph. Their method achieves a high expressibility based on message passing; however, for this reason, a considerable amount of computational resources are required. By contrast, because it is based on a GCN the approach proposed in the present study allows a considerable reduction in the computational costs.

Physical simulations using GNNs. Several related studies, including those by Sanchez-Gonzalez et al. (2018; 2019); Alet et al. (2019); Chang & Cheng (2020) were focused on applying GNNs to learn physical simulations. Sanchez-Gonzalez et al. (2020) 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 the scope of the their research.

In the present study, we incorporate isometric transformation invariance and equivariance into GCNs, thereby ensuring the stability of the training and inference under an isometric transformation. Moreover, the proposed approach is efficient in processing 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 a way to construct IsoGCN layers that correspond to the isometric invariant and equivariant GCN layers. To formulate a model, we consider the following assumptions: 1) Only attributes associated with vertices and not edges our applied, and the 2) graphs do not contain self-loops. Here, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and d denote a graph and the dimensions of a Euclidean space, respectively.

3.1 CONSTRUCTION OF AN ISOMETRIC ADJACENCY MATRIX

Before constructing an IsoGCN, we need to define an **isometric adjacency matrix** (IsoAM), which is at 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. For an IsoAM, $\mathbf{G} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times d}$ is defined as follows:

$$\mathbb{R}^d \ni \mathbf{G}_{ij:} := \sum_{k,l \in \mathcal{V}, k \neq l} \boldsymbol{T}_{ijkl} (\boldsymbol{x}_k - \boldsymbol{x}_l). \tag{1}$$

where $\mathbf{G}_{ij:}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, and $\boldsymbol{T}_{ijkl} \in \mathbb{R}^{d \times d}$ denote a 1-D slice of $\mathbf{G}$, the position of the ith vertex (rank-1 tensor), and the transformation invariant rank-2 tensor, respectively. Here, a rank-k tensor $\mathbf{S} \in \mathbb{R}^{d^k}$ indicates a multidimensional array whose components transform as $\mathsf{S}_{i_1 i_2 \dots i_k} \mapsto \mathsf{U}_{i_1 j_1} \mathsf{U}_{i_2 j_2} \dots \mathsf{U}_{i_k j_k} \mathsf{S}_{j_1 j_2 \dots j_k}$, where $\mathsf{U}_{i_l j_l}$ denotes the element (i_l, j_l) of a transformation matrix. Here, $\mathbf{G}$ is composed of only untrainable parameters, and thus it can be determined prior to training.

Proposition 3.1. *IsoAM defined in equation 1 is both translation invariant and orthogonal transformation equivariant, i.e., for any isometric transformation* $\forall \boldsymbol{t} \in \mathbb{R}^3, \boldsymbol{U} \in \mathrm{O}(d), T : \boldsymbol{x} \mapsto \boldsymbol{U}\boldsymbol{x} + \boldsymbol{t}$*:*

$$T : \mathbf{G}_{ijk} \mapsto \sum_l U_{kl} \mathbf{G}_{ijl}. \tag{2}$$

Based on such an observation, one can regard $\mathbf{G}$ as a collection of rank-1 tensors.

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

$$({}^1\mathbf{H} * {}^0\mathbf{H}')_{ilk} := \sum_j {}^1H_{ijk}\, {}^0H'_{jl}. \tag{3}$$

Because the IsoAM is permutated accordingly to the permutation of the vertices, IsoAM is also permutation equivariant (see Appendix B.1).

Let ${}^1\mathbf{H} \odot {}^1\mathbf{H}' \in \mathbb{R}^{n \times f}$ denote the **contraction** of ${}^1\mathbf{H} \in \mathbb{R}^{n \times m \times d}$ and ${}^1\mathbf{H}' \in \mathbb{R}^{m \times f \times d}$ ($n, m, f \in \mathbb{Z}^+$) which is defined as follows:

$$({}^1\mathbf{H} \odot {}^1\mathbf{H}')_{il} := \sum_{j,k} {}^1H_{ijk}\, {}^1H'_{jlk}. \tag{4}$$

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

Proposition 3.2. *The contraction of IsoAMs* $\mathbf{G} \odot \mathbf{G}$ *is isometric transformation invariant, i.e., for any isometric transformation* $\forall \boldsymbol{t} \in \mathbb{R}^3, \boldsymbol{U} \in \mathrm{O}(d), T : \boldsymbol{x} \mapsto \boldsymbol{U}\boldsymbol{x} + \boldsymbol{t}$*:*

$$T : \mathbf{G} \odot \mathbf{G} \mapsto \mathbf{G} \odot \mathbf{G}. \tag{5}$$

Let ${}^p\mathbf{H} \otimes {}^q\mathbf{H}' \in \mathbb{R}^{n \times r \times d^p \times d^q}$ denote the **tensor product** of the collections of rank-p tensors ${}^p\mathbf{H} \in \mathbb{R}^{n \times m \times d^p}$ and rank-q tensors ${}^q\mathbf{H}' \in \mathbb{R}^{m \times f \times d^q}$ ($n, m, p, q, f \in \mathbb{Z}^+$) defined as follows:

$$({}^p\mathbf{H} \otimes {}^q\mathbf{H}')_{ilk_1k_2\ldots k_p m_1 m_2 \ldots m_q} := \sum_j {}^pH_{ijk_1k_2\ldots k_p}\, {}^qH'_{jlm_1m_2\ldots m^q}. \tag{6}$$

Because the tensor product of an IsoAM can be interpreted as the tensor product of each component in the IsoAMs, the subsequent proposition follows:

Proposition 3.3. *The tensor product of IsoAMs* $\mathbf{G} \otimes \mathbf{G}$ *is isometric transformation equivariant as a collection of rank-2 tensors, i.e., for any isometric transformation* $\forall \boldsymbol{t} \in \mathbb{R}^3, \boldsymbol{U} \in \mathrm{O}(d), T : \boldsymbol{x} \mapsto \boldsymbol{U}\boldsymbol{x} + \boldsymbol{t}$ *and* $\forall i, j \in 1, \ldots, |\mathcal{V}|$*:*

$$T : (\mathbf{G} \otimes \mathbf{G})_{ij::} \mapsto \boldsymbol{U}(\mathbf{G} \times \mathbf{G})_{ij::}\boldsymbol{U}^T. \tag{7}$$

It is possible to easily generalize this proposition to the tensors of higher ranks by defining the pth tensor power of $\mathbf{G}$ as follows: $\bigotimes^1 \mathbf{G} = \mathbf{G}$ and $\bigotimes^p \mathbf{G} = \bigotimes^{p-1} \mathbf{G} \otimes \mathbf{G}$. Namely, as a collection of rank-p tensors, $\bigotimes^k \mathbf{G}$ is isometric transformation equivariant.

Moreover, the convolution can be generalized for the collections of rank-p tensors ${}^p\mathbf{H} \in \mathbb{R}^{n \times m \times d^p}$ and rank-0 tensors ${}^0\mathbf{H}' \in \mathbb{R}^{m \times f}$ ($n, m, p, f \in \mathbb{Z}^+$) as follows:

$$\left[{}^p\mathbf{H} * {}^0\mathbf{H}'\right]_{ilk_1k_2\ldots k_p} = \sum_j {}^pH_{ijk_1k_2\ldots k_p}\, {}^0H'_{jl}. \tag{8}$$

The contraction can be generalized for the collections of rank-p tensors ${}^p\mathbf{H} \in \mathbb{R}^{n \times m \times d^p}$ and rank-q tensors ${}^p\mathbf{H} \in \mathbb{R}^{m \times f \times d^q}$ ($n, m, f, p, q \in \mathbb{Z}^+, p > q$) as specified below:

$$[{}^p\mathbf{H} \odot {}^q\mathbf{H}]_{ilk_1k_2\ldots k_{p-q}} = \sum_{j, m_1, m_2, \ldots, m_q} {}^pH_{ijk_1k_2\ldots k_{p-q}m_1m_2\ldots m_q}\, {}^qH'_{jlm_1m_2\ldots m_q}. \tag{9}$$

3.2 CONSTRUCTION OF ISOGCN

An IsoGCN utilizes an IsoAM to realize the isometric transformation invariant and equivariant layers. Using the operations defined above, such as a convolution, contraction, and tensor product, we can construct IsoGCN layers, which take the tensors of any rank as input, and output the tensors of any rank, which can differ from those of the input.

3.2.1 ISOMETRIC TRANSFORMATION INVARIANT LAYER

As it can be seen in Proposition 3.1, the contraction of IsoAMs is isometric transformation invariant. Therefore, for the lth isometric transformation invariant layer with rank-0 input tensors $f^{(l)} : \mathbb{R}^{|\mathcal{V}| \times f_{\mathrm{in}}^{(l)}} \ni \boldsymbol{H}^{(l)} \mapsto \boldsymbol{H}^{(l+1)} \in \mathbb{R}^{|\mathcal{V}| \times f_{\mathrm{out}}^{(l)}}$ ($f_{\mathrm{in}}^{(l)}, f_{\mathrm{out}}^{(l)} \in \mathbb{Z}^+$), the activation function $\sigma^{(l)}$, and the trainable parameter matrix $W^{(l)} \in \mathbb{R}^{f_{\mathrm{in}}^{(l)} \times f_{\mathrm{out}}^{(l)}}$ can be constructed as $\boldsymbol{H}^{(l+1)} = \sigma^{(l)}\left((\mathbf{G} \odot \mathbf{G})\, \boldsymbol{H}^{(l)} \boldsymbol{W}^{(l)}\right)$. By defining $\boldsymbol{L} := \mathbf{G} \odot \mathbf{G} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, it can be simplified as $\boldsymbol{H}^{(l+1)} = \sigma^{(l)}\left(\boldsymbol{L} \boldsymbol{H}^{(l)} \boldsymbol{W}^{(l)}\right)$, which has the same form as a GCN, with the exception that the renormalized adjacency matrix is replaced with $\boldsymbol{L}$.

An isometric transformation invariant layer with the rank-1 input tensors ${}^1\mathbf{H}^{(l)}$ can be formulated as $\boldsymbol{H}^{(l+1)} = \sigma^{(l)}\left(\left[\mathbf{G} \odot {}^1\mathbf{H}^{(l)}\right] \boldsymbol{W}^{(l)}\right)$. Because such approaches utilize the inner products of the 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).

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

$$\tilde{\boldsymbol{H}}^{(l)} = f_{\mathrm{gather}}\left((\mathbf{G} \odot \mathbf{G})\, {}^0\mathbf{H}^{(l)}, \mathbf{G} \odot {}^1\mathbf{H}^{(l)}, (\mathbf{G} \otimes \mathbf{G}) \odot {}^2\mathbf{H}^{(l)}, \ldots, \left[\bigotimes^{k} \mathbf{G}\right] \odot {}^k\boldsymbol{H}^{(l)}\right), \tag{10}$$

$$\boldsymbol{H}^{(l+1)} = \sigma^{(l)}(\tilde{\boldsymbol{H}}^{(l)} \boldsymbol{W}^{(l)}), \tag{11}$$

where f_{gather} denotes a function such as summation, product and concatenation in the feature direction.

3.2.2 ISOMETRIC TRANSFORMATION EQUIVARIANT LAYER

To construct an isometric transformation equivariant layer, one can use linear transformation, convolution and tensor product to the input tensors. If both the input and the output tensor ranks are greater than 0, one can apply neither nonlinear activation nor bias addition because these operations will cause an inappropriate distortion of the isometry, as in general, isometric transformation does not commute with them. However, conversion using only linear transformation, convolution, and tensor product does not have nonlinearity, which limit the predictive performance of the model. To add nonlinearity to such a conversion, we can first convert input tensors to rank-0 ones, apply nonlinear activations, and then multiply them to the higher rank tensors.

For instance, the nonlinear l isometric transformation equivariant layer with rank-1 input tensors ${}^1\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times f_{\mathrm{in}}^{(l)}}$ and rank-3 output tensors ${}^3\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times f_{\mathrm{out}}^{(l)}}$ ($f_{\mathrm{in}}^{(l)}, f_{\mathrm{out}}^{(l)} \in \mathbb{Z}^+$) can be $\sigma^{(l)}(\mathbf{G} \odot {}^1\mathbf{H}\boldsymbol{W}_0^{(l)})\ (\mathbf{G} \otimes \mathbf{G} \otimes {}^1\mathbf{H}\boldsymbol{W}_1^{(l)})$, where $\boldsymbol{W}_0^{(l)}, \boldsymbol{W}_1^{(l)} \in \mathbb{R}^{f_{\mathrm{in}}^{(l)} \times f_{\mathrm{out}}^{(l)}}$ are trainable weight matrices. To achieve translation equivariance, a constant tensor can be added to the output collection of tensors.

3.2.3 DESIGN RULE OF ISOGCN

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

1. Check the ranks of the tensors for the input and output features.
2. If the rank of the input feature is lower than that of the output feature, use convolution or tensor product of IsoAM to raise the tensor rank. Otherwise, use contraction of IsoAM to reduce the tensor rank.

3. 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 consider the tensors of any rank as an input 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 $\mathbf{G} \otimes \mathbf{G} * \sigma^{(l)}((\mathbf{G} \otimes \mathbf{G}) \odot (\mathbf{G} \otimes \mathbf{G}) \mathbf{G} \odot \boldsymbol{H}^{(l)})$ for a rank-1 tensor input and a rank-2 tensor output, which can be considered as the IsoGCN version of simple graph convolution (Wu et al., 2019).

As implied by the discussion presented above, only the ranks corresponding to the input and the output of the entire network are important for realizing isometric transformation invariance and equivariance. This means that one can add any layer mapping to the 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 from a rank-2 tensor to a rank-1 tensor, an IsoGCN with the following structure can be constructed: (rank-2 tensor) $\rightarrow$ (rank-3 tensor) $\rightarrow$ (rank-0 tensor) $\rightarrow$ (rank-1 tensor). However, because a lower rank tensor generally has less information than a higher rank tensor, our recommendation is to avoid reducing the tensor rank unless needed. For the detailed information of IsoGCN modeling, see Appendix D.

3.3 Example of IsoAM

To apply IsoGCN to mesh structured data, we define a concrete instance of IsoAM $\tilde{\mathbf{D}}, \mathbf{D} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times d}$ as follows:

$$\tilde{D}_{ijk} = D_{ijk} - \delta_{ij} \sum_l D_{ilk}, \tag{12}$$

$$D_{ij:} = \boldsymbol{M}_i^{-1} \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|^2} w_{ij} A_{ij}(m), \tag{13}$$

$$\boldsymbol{M}_i = \sum_l \frac{\boldsymbol{x}_l - \boldsymbol{x}_i}{\|\boldsymbol{x}_l - \boldsymbol{x}_i\|} \otimes \frac{\boldsymbol{x}_l - \boldsymbol{x}_i}{\|\boldsymbol{x}_l - \boldsymbol{x}_i\|} w_{il} \boldsymbol{A}_{ij}(m), \tag{14}$$

Table 1: Correspondence between the differential operators and the expressions using the IsoAM $\tilde{\mathbf{D}}$.

Differential operator	**Expression using $\tilde{\mathbf{D}}$**
Gradient	$\tilde{\mathbf{D}} * {}^0\mathbf{H}$
Divergence	$\tilde{\mathbf{D}} \odot {}^1\mathbf{H}$
Laplacian	$\tilde{\mathbf{D}} \odot \tilde{\mathbf{D}}^0\mathbf{H}$
Jacobian	$\tilde{\mathbf{D}} \otimes {}^1\mathbf{H}$
Hessian	$\tilde{\mathbf{D}} \otimes \tilde{\mathbf{D}} * {}^0\mathbf{H}$

where $\boldsymbol{A}(m) \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ and $w_{ij} \in \mathbb{R}$ are an adjacency matrix up to m hops and an untrainable weight between the ith and jth vertices determined depending on the tasks, respectively[1]. Because both $\mathbf{D}$ and $\tilde{\mathbf{D}}$ are qualified as IsoAMs, they provide both translation invariance and orthogonal transformation equivariance. $\tilde{\mathbf{D}}$ can be obtained only from mesh geometry information, thus can be computed in the preprocessing step of a training.

Here, $\tilde{\mathbf{D}}$ is designed such that it corresponds to the gradient operator model used in physical simulations (Tamai & Koshizuka, 2014; Swartz & Wendroff, 1969) (see Appendix C). As presented in Table 1 and Appendix C, $\tilde{\mathbf{D}}$ is closely related to many differential operators such as gradient, divergence, Laplacian, Jacobian, and Hessian ones. Therefore, the considered IsoAM considered plays an essential role in constructing the neural network models that are capable of learning differential equations.

4 Experiment

To test the applicability of the proposed model, we composed the following two datasets: 1) a differential operator dataset of grid meshes, and 2) an anisotropic nonlinear heat equation dataset of meshes generated from CAD data. In this section, we discuss our machine learning model, the definition of the problem, and the results for each dataset.

Using $\tilde{\mathbf{D}}$ defined in Section 3.3, we constructed a neural network model considering an encode-process-decode configuration (Battaglia et al., 2018). The encoder and decoder comprised

[1] $\boldsymbol{M}_i$ is invertible when the number of independent vectors in $\{\boldsymbol{x}_l - \boldsymbol{x}_i\}_l$ is greater than or equal to the space dimension d, which is true for common meshes, e.g., solid mesh in the 3D Euclidean space.

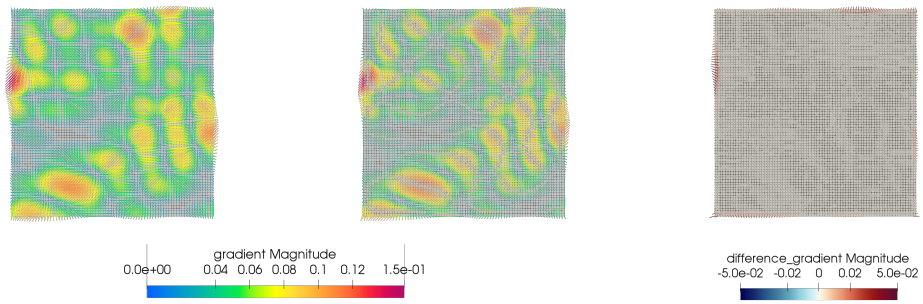

Figure 1: (left) The test data sample of the ground truth gradient field calculated analytically from the given scalar field, (center) the gradient field predicted by the IsoGCN model, and (right) the error vector between the ground truth and the prediction. The error vectors are exaggerated by a factor of 2 for a clear visualization.

component-wise MLPs and tensor operations. For each task, we tested $m = 2, 5$ in equation 12 to investigate the effect of the number of hops considered. As baseline models, we chose GCN (Kipf & Welling, 2017), GIN (Xu et al., 2018), SGCN (Wu et al., 2019), Cluster-GCN (Chiang et al., 2019), and GCNII (Chen et al., 2020). To compare the performance, we simply replaced an IsoGCN layer with a GCN or its variant layers while keeping the number of hops m the same to enable a fair comparison. We implemented these models using PyTorch 1.6.0 (Paszke et al., 2019) and PyTorch Geometric 1.6.1 (Fey & Lenssen, 2019). For each experiment, we minimized the mean squared loss using the Adam optimizer (Kingma & Ba, 2014). The corresponding implementation and the dataset will be made available online. The details of the experiments can be found in Appendix F.

4.1 Differential operator dataset

To demonstrate the expressive power of IsoGCN, we created a dataset to learn the differential operators. We first generate a pseudo-2D grid mesh randomly, which has only one cell in the Z direction and 10 to 100 cells in the X and Y directions. We then generated scalar fields on the grid meshes, and then calculated the gradient, Laplacian, and Hessian fields analytically. We generated 100 samples for each train, validation, and test dataset. For simplicity, we set $w_{ij} = 1$ in equation 12 for all $i, j \in \mathcal{V}$. For more details regarding the model architecture, see Appendix E. We conducted the experiments using the following settings: 1) inputting the scalar field and predicting the gradient field (rank-0 $\rightarrow$ rank-1 tensor); 2) inputting the scalar field and predicting the Hessian field (rank-0 $\rightarrow$ rank-2 tensor); 3) inputting the gradient field and predicting the Laplacian field (rank-1 $\rightarrow$ rank-0 tensor); and 4) inputting the gradient field and predicting the Hessian field (rank-1 $\rightarrow$ rank-2 tensor).

Figure 1 and Table 2 present a visualization of the task and comparison predictive performance, respectively. The results show that an IsoGCN outperforms other GCN models for all settings. This

Table 2: Summary of the test losses (mean squared error $\pm$ the standard error of the mean) of the differential operator dataset: $0 \rightarrow 1$, $0 \rightarrow 2$, $1 \rightarrow 0$, and $1 \rightarrow 2$ correspond to the scalar field to the gradient field, the scalar field to the Hessian field, the gradient field to the Laplacian field, the gradient field to the Hessian field, respectively. Here, if "$\boldsymbol{x}$" is "Yes", it means $\boldsymbol{x}$ is also in the input feature. We show only the best setting for each method. For a full table, see Appendix E.

Method	**# hops**	$\boldsymbol{x}$	**Loss of $0 \rightarrow 1$** $\times 10^{-5}$	**Loss of $0 \rightarrow 2$** $\times 10^{-6}$	**Loss of $1 \rightarrow 0$** $\times 10^{-6}$	**Loss of $1 \rightarrow 2$** $\times 10^{-6}$
GIN	5	Yes	147.07 ± 0.51	47.35 ± 0.35	404.92 ± 1.74	46.18 ± 0.39
GCNII	5	Yes	151.13 ± 0.53	31.87 ± 0.22	280.61 ± 1.30	39.38 ± 0.34
SGCN	5	Yes	151.16 ± 0.53	55.08 ± 0.42	127.21 ± 0.63	56.97 ± 0.44
GCN	5	Yes	151.14 ± 0.53	48.50 ± 0.35	542.30 ± 2.14	25.37 ± 0.28
Cluster-GCN	5	Yes	146.91 ± 0.51	26.60 ± 0.19	185.21 ± 0.99	18.18 ± 0.20
IsoGCN (Ours)	2	No	$\mathbf{2.67} \pm 0.02$	$\mathbf{6.37} \pm 0.07$	$\mathbf{7.18} \pm 0.06$	$\mathbf{1.44} \pm 0.02$
	5	No	14.19 ± 0.10	21.72 ± 0.25	34.09 ± 0.19	8.32 ± 0.09

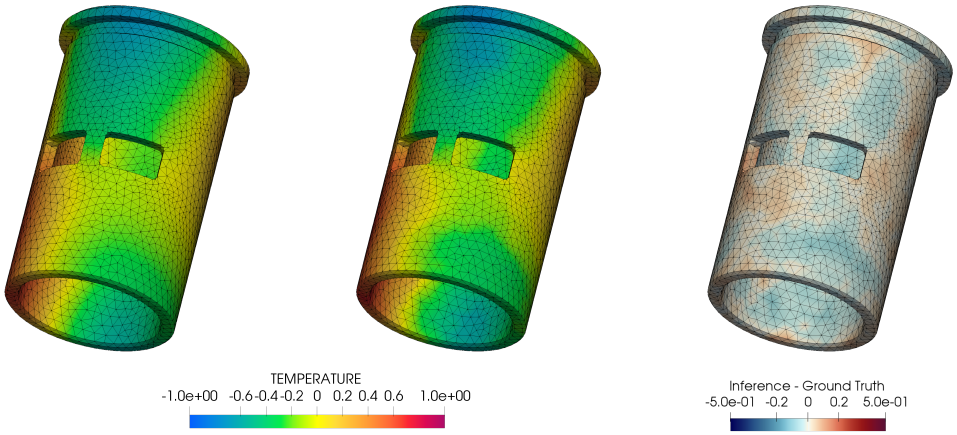

Figure 2: Qualitative comparison for randomly sampled test dataset points between (left) the ground truth computed through using an FEA, (center) an IsoGCN with $m = 2$ hops, and (right) the error of the inference. (right) error vector between the ground truth and the prediction. The error is exaggerated by the factor of 2 for clear visualization.

is because it has the information on the relative position of the adjacency vertices, and thus understands the direction of the gradient, whereas the others cannot distinguish where the adjacencies are, making it nearly impossible to predict the gradient directions. Adding the vertex positions to the input feature to other GCN models showed performance improvement, however as the vertex position was not a translation invariant feature, it could degrade the predictive performance of models. Thus, we did not input $\boldsymbol{x}$ as a vertex feature to IsoGCN model to keep isometric transformation invariant and equivariant nature of them. As mentioned in Section 3.3, $\tilde{\mathbf{D}}$ corresponds to the gradient operator, which is now confirmed in practice.

4.2 Anisotropic nonlinear heat equation dataset

To apply the proposed model to a real problem, we adopted the anisotropic nonlinear heat equation. We considered the task of predicting the time evolution of the temperature field based on the initial temperature field, material property, and mesh geometry information as inputs. We selected 82 CAD shapes by randomly considering the first 200 shapes of the ABC dataset (Koch et al., 2019), generate first-order tetrahedral meshes using a mesher program Gmsh (Geuzaine & Remacle, 2009), randomly set the initial temperature and anisotropic thermal conductivity, and finally conducted a

Table 3: Summary of the test losses (mean squared error $\pm$ the standard error of the mean) of the anisotropic nonlinear heat dataset. Here, if "$\boldsymbol{x}$" is "Yes", it means $\boldsymbol{x}$ is also in the input feature. We show only the best setting for each method except for the IsoGCN models. For the full table, see Appendix E.

Method	**# hops**	$\boldsymbol{x}$	**Loss** $\times 10^{-3}$
GIN	2	No	16.921 ± 0.040
GCN	2	No	10.427 ± 0.028
GCNII	5	No	8.377 ± 0.024
Gluster-GCN	2	No	7.266 ± 0.021
SGCN	5	No	6.426 ± 0.018
IsoGCN (Ours)	2	No	4.674 ± 0.014
	5	No	$\mathbf{2.470} \pm 0.008$

Table 4: Comparison of computation time between an IsoGCN and a FEA using FrontISTR with different numbers of timesteps. To generate the test data, we sampled CAD data from the test dataset and then generated the mesh such that the graph gets large with keeping the element volume sizes almost the same. The initial temperature field and the material properties are set randomly using the same methodology as the dataset sample generation. The compuration was run on Intel Xeon E5-2695 v2@2.40GHz using one core. We excluded file I/O time from the measured time to enable a fair comparison.

		$\lvert\mathcal{V}\rvert = 21,289$		$\lvert\mathcal{V}\rvert = 155,019$	
Method	**# time step**	**Loss** ($\times 10^{-4}$)	**Time [s]**	**Loss** ($\times 10^{-4}$)	**Time [s]**
FrontISTR	1	10.9	16.7	6.1	181.7
	2	0.8	30.5	0.4	288.0
IsoGCN	N/A	8.1	**5.5**	4.9	**63.0**

finite element analysis (FEA) using the FEA program FrontISTR[2] (Morita et al., 2016; Ihara et al., 2017).

For this task, we set $w_{ij} = V_j^{\text{effective}}/V_i^{\text{effective}}$, where $V_i^{\text{effective}}$ denotes the effective volume of the ith vertex (equation 45.) Similar to the differential operator dataset, we tested the number of hops $m = 2, 5$. However, because we put two IsoAM operations in one propagation block and the model has two propagation blocks, the number of hops visible from the model is 8 ($m = 2$) or 20 ($m = 5$). For more details about the used dataset and the model, see Appendix F.

Figure 2 and Table 3 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 other GCN variants. Moreover, owing to the isometric transformation invariant nature of IsoGCN, it also achieved a high prediction performance for the meshes that had a significantly larger graph than those considered in the training dataset. IsoGCN can scales up to 1M vertices, which is a practical scale and considerably greater than that reported in Sanchez-Gonzalez et al. (2020). Therefore, we conclude that the IsoGCN models could be trained on a relatively smaller meshes to save the training time and then used to apply an inference of larger meshes without observing significant performance deterioration. Because NT & Maehara (2019) has outlined that GNNs can be considered to be low-pass filters, it was deemed reasonable that the IsoGCN model can learn the heat equation appropriately. However, because the problem is anisotropic, the IsoGCN model can learn the direction for applying low-pass filters more strongly.

Table 4 reports the computation time of inference using the IsoGCN model with $m = 2$ as the number of hops and FEA using FrontISTR 5.0.0. For the FEA computation, we varied the number of steps to compute 1s time evolution, thus resulting in the different computation time and error compared to an FEA with time steps of 100, which were considered as the ground truth. It can be seen that IsoGCN is 3- to 5- times faster than the FEA with the same level of accuracy. This is due to the simplicity coming from a GCN-like operation and high accuracy coming from its isometric transformation invariant feature.

5 Conclusion

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

[2] `https://github.com/FrontISTR/FrontISTR`. We applied a private update to FrontISTR to deal with anisotropic heat problem, which will be also made available online.

REFERENCES

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, Mateusz 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.

Kai-Hung Chang and Chin-Yi Cheng. Learning to simulate and design for structural engineering. *arXiv preprint arXiv:2003.09103*, 2020.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. *arXiv preprint arXiv:2007.02133*, 2020.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 257–266, 2019.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *7nternational 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.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 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.

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.

Naoki Morita, Kazuo Yonekura, Ichiro Yasuzumi, Mitsuyoshi Tsunori, Gaku Hashimoto, and Hiroshi Okuda. Development of 3×3 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'Alché-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.

Blair Swartz and Burton Wendroff. Generalized finite-difference schemes. *Mathematics of Computation*, 23(105):37–49, 1969.

Tasuku Tamai and Seiichi Koshizuka. Least squares moving particle semi-implicit method. *Computational Particle Mechanics*, 1(3):277–305, 2014.

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.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

A NOTATION

$\mathcal{G}$	A graph
$\mathcal{V}$	A vertex set
$\lvert\mathcal{V}\rvert$	The number of vertices
$\mathcal{E}$	An edge set
$\mathbb{Z}^+$	The positive integers
d	The dimension of the Euclidean space
$\boldsymbol{x}_i$	The position of the ith vertex
x_{ik}	Element k of $\boldsymbol{x}_i$
$\mathbf{G} \in \mathbb{R}^{\lvert\mathcal{V}\rvert\times\lvert\mathcal{V}\rvert\times d}$	The isometric adjacency matrix (IsoAM)(equation 1)
$\mathbf{G}_{ij:} \in \mathbb{R}^d$	1-D slice of $\mathbf{G}$ (equation 1)
$\mathbf{G}_{ijk} \in \mathbb{R}$	Element (i, j, k) of $\mathbf{G}$
${}^p\mathbf{H} \in \mathbb{R}^{n\times f\times d^p}$	A collection of rank-p tensors ($n, f \in \mathbb{Z}^+$)
${}^p\mathbf{H} * {}^0\mathbf{H}'$	Convolution of collections of rank-p tensors ${}^p\mathbf{H}$ and rank-0 tensors ${}^0\mathbf{H}'$ (equation 3, equation 8)
${}^p\mathbf{H} \odot {}^p\mathbf{H}'$	Contraction of collections of rank-p tensors ${}^p\mathbf{H}$ and ${}^p\mathbf{H}'$ (equation 4, equation 9)
${}^p\mathbf{H} \otimes {}^q\mathbf{H}'$	Tensor product of collections of rank-p tensors ${}^p\mathbf{H}$ and rank-q tensors ${}^q\mathbf{H}'$ (equation 6)
${}^p\mathbf{H}^{(l)}$	The input feature of the lth layer with rank-p tensors
$\sigma^{(l)}$	The activation function of the lth layer
$\boldsymbol{W}^{(l)}$	The trainable parameter matrix of the lth layer
$\boldsymbol{A}$	An adjacency matrix
δ_{ij}	The Kronecker delta
$V_i^{\mathrm{effective}}$	The effective volume of the ith vertex (equation 45)
V_i^{mean}	The mean volume of the ith vertex (equation 46)
$\tilde{\mathbf{D}}$	An IsoAM (equation 12)

B PROOFS OF PROPOSITIONS

In this section, we present the proofs of the propositions described in Section 3. Let $\mathbb{R}^3 \ni \boldsymbol{g}(\boldsymbol{x}_l, \boldsymbol{x}_k) = (\boldsymbol{x}_k - \boldsymbol{x}_l)$. Note that $\mathbf{G}$ is expressed using $\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as $\mathbf{G}_{ij:} = \sum_{k,l\in\mathcal{V},k\neq l} \boldsymbol{T}_{ijkl}\boldsymbol{g}(\boldsymbol{x}_l, \boldsymbol{x}_k)$.

B.1 PROOF OF PROPOSITION 3.1

Proof. First, we demonstrate the invariance with respect to the translation with $\forall \boldsymbol{t} \in \mathbb{R}^d$. $\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is transformed invariantly as follows under translation:

$$\begin{aligned}\boldsymbol{g}(\boldsymbol{x}_i + \boldsymbol{t}, \boldsymbol{x}_j + \boldsymbol{t}) &= [\boldsymbol{x}_j + \boldsymbol{t} - (\boldsymbol{x}_i + \boldsymbol{t})] \\ &= (\boldsymbol{x}_j - \boldsymbol{x}_i) \\ &= \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{x}_j). \end{aligned} \tag{15}$$

By definition, $\boldsymbol{T}_{ijkl}$ is also translation invariant. Thus,

$$\begin{aligned}\sum_{k,l\in\mathcal{V},k\neq l} \boldsymbol{T}_{ijkl}\boldsymbol{g}(\boldsymbol{x}_l+\boldsymbol{t},\boldsymbol{x}_k+\boldsymbol{t}) &= \sum_{k,l\in\mathcal{V},k\neq l} \boldsymbol{T}_{ijkl}\boldsymbol{g}(\boldsymbol{x}_l,\boldsymbol{x}_k)\\ &= \mathbf{G}_{ij:}. \end{aligned} \tag{16}$$

We then show an equivariance with respect to the orthogonal transformation with $\forall \boldsymbol{U} \in \mathrm{O}(d)$. $\boldsymbol{g}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ is transformed as follows under an orthogonal transformation:

$$\begin{aligned}\boldsymbol{g}(\boldsymbol{U}\boldsymbol{x}_i,\boldsymbol{U}\boldsymbol{x}_j) &= \boldsymbol{U}\boldsymbol{x}_j-\boldsymbol{U}\boldsymbol{x}_i\\ &= \boldsymbol{U}(\boldsymbol{x}_j-\boldsymbol{x}_i)\\ &= \boldsymbol{U}\boldsymbol{g}(\boldsymbol{x}_i,\boldsymbol{x}_j). \end{aligned} \tag{17}$$

By definition, $\boldsymbol{T}_{ijkl}$ is transformed to $\boldsymbol{U}\boldsymbol{T}_{ijkl}\boldsymbol{U}^{-1}$ under orthogonal transformation. Thus,

$$\begin{aligned}\sum_{k,l\in\mathcal{V},k\neq l} \boldsymbol{U}\boldsymbol{T}_{ijkl}\boldsymbol{U}^{-1}\boldsymbol{g}(\boldsymbol{U}\boldsymbol{x}_l,\boldsymbol{U}\boldsymbol{x}_k) &= \sum_{k,l\in\mathcal{V},k\neq l} \boldsymbol{U}\boldsymbol{T}_{ijkl}\boldsymbol{U}^{-1}\boldsymbol{U}\boldsymbol{g}(\boldsymbol{x}_l,\boldsymbol{x}_k)\\ &= \boldsymbol{U}\mathbf{G}_{ij:}. \end{aligned} \tag{18}$$

Therefore, $\mathbf{G}$ is both translation invariant and orthogonal transformation equivariant. □

B.2 Proof of permutation equivariance of the IsoAM

Proposition B.1. *Let ${}^0\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times f}$ ($f \in \mathbb{Z}^+$) denote a collection of rank-0 tensors. The convolution of the IsoAM and that collection $(\mathbf{G} * {}^0\mathbf{H})_{ilk} = G_{ijk}{}^0H_{jl}$ is permutation equivariant, i.e., for the permutation matrix $\boldsymbol{P}_\pi$ corresponding to any permutation π and $\forall k \in \{1,\dots d\}$,*

$$\pi : (\mathbf{G} * {}^0\mathbf{H})_{::k} \mapsto \boldsymbol{P}_\pi \mathbf{G}_{::k} * {}^0\mathbf{H}. \tag{19}$$

Proof. Let $\boldsymbol{P}_\pi$ be the permutation matrix corresponding to π, and define ${}^0\mathbf{H} = \boldsymbol{H}$. The convolution of vertex features with IsoAM $\mathbf{G} * \boldsymbol{H}$ is transformed as follows under the permutation π: $\forall k \in \{1,\dots d\}$,

$$\begin{aligned}(\mathbf{G} * {}^0\mathbf{H})_{::k} &\mapsto (\pi(\mathbf{G}) * \pi({}^0\mathbf{H}))_{::k}\\ &= \boldsymbol{P}_\pi \mathbf{G}_{::k}\boldsymbol{P}_\pi^T\boldsymbol{P}_\pi\boldsymbol{H}\\ &= \boldsymbol{P}_\pi \mathbf{G}_{::k}\boldsymbol{H}. \end{aligned} \tag{20}$$

It should be noted that $\mathbf{G}_{::k}$ can be regarded as a matrix. Therefore, $\mathbf{G}*\boldsymbol{H}$ is permutation equivariant. □

B.3 Proof of Proposition 3.2

Proof. Here, $\mathbf{G} \odot \mathbf{G}$ is translation invariant because $\mathbf{G}$ is translation invariant. We prove rotation invariance under an orthogonal transformation $\forall \boldsymbol{U} \in \mathrm{O}(n)$. In addition, $\mathbf{G}\odot\mathbf{G}$ is transformed under the $\boldsymbol{U}$ as follows:

$$\begin{aligned}\sum_{j,k} G_{ijk}G_{jlk} &\mapsto \sum_{k,l,m,n} \boldsymbol{U}_{km}G_{ijm}\boldsymbol{U}_{kn}G_{jln}\\ &= \sum_{k,l,m,n} \boldsymbol{U}_{km}\boldsymbol{U}_{kn}G_{ijm}G_{jln}\\ &= \sum_{k,l,m,n} \boldsymbol{U}^T_{mk}\boldsymbol{U}_{kn}G_{ijm}G_{jln}\\ &= \sum_{l,m,n} \delta_{mn}G_{ijm}G_{jln} && (\because \text{ property of the orthogonal matrix})\\ &= \sum_{l} G_{ijm}G_{jlm}\\ &= \sum_{k} G_{ijk}G_{jlk}. && (\because \text{ Change the dummy index } m \to k) \end{aligned} \tag{21}$$

Therefore, $\mathbf{G} \odot \mathbf{G}$ is isometric transformation invariant. □

B.4 PROOF OF PROPOSITION 3.3

Proof. $\mathbf{G} \otimes \mathbf{G}$ is transformed under $\forall \boldsymbol{U} \in \mathrm{O}(n)$ as follows:

$$\begin{aligned}\sum_j \mathsf{G}_{ijk}\mathsf{G}_{jlm} &\mapsto \sum_{n,o} \boldsymbol{U}_{kn}\mathsf{G}_{ijn}\boldsymbol{U}_{mo}\mathsf{G}_{jlo} \\ &= \sum_{n,o} \boldsymbol{U}_{kn}\mathsf{G}_{ijn}\mathsf{G}_{jlo}\boldsymbol{U}^T_{om}. \end{aligned} \tag{22}$$

By regarding $\mathsf{G}_{ijn}\mathsf{G}_{jlo}$ as one matrix $\boldsymbol{H}_{no}$, it follows the coordinate transformation of rank-2 tensor $\boldsymbol{U}\boldsymbol{H}\boldsymbol{U}^T$. □

C PHYSICAL INTUITION OF $\tilde{\mathbf{D}}$

In this section, we discuss the connection between the concrete example of IsoAM $\tilde{\mathbf{D}}$ and the differential operators such as the gradient, the divergence, the Laplacian, the Jacobian, and the Hessian operators.

Let $\phi_i \in \mathbb{R}$ denote a rank-0 tensor (scalar) ath the ith vertex. Let us assume a partial derivative model of a rank-0 tensor ϕ at the ith vertex regarding the kth axis $(\partial\phi/\partial x_k)_i \in \mathbb{R}$ $(k \in \{1, \ldots, d\})$, that is based on the gradient model in the least squares moving particle semi-implicit method (Tamai & Koshizuka, 2014).

$$\left(\frac{\partial \phi}{\partial x_k}\right)_i := \boldsymbol{M}_i^{-1} \sum_j \frac{\phi_j - \phi_i}{\|x_{jk} - \boldsymbol{x}_{ik}\|} \frac{\boldsymbol{x}_j - \boldsymbol{x}_i}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|} w_{ij} A_{ij}(m) \tag{23}$$

$$= \sum_j \mathsf{D}_{ijk}(\phi_j - \phi_i), \tag{24}$$

$$\boldsymbol{M}_i = \sum_l \frac{\boldsymbol{x}_l - \boldsymbol{x}_i}{\|\boldsymbol{x}_l - \boldsymbol{x}_i\|} \otimes \frac{\boldsymbol{x}_l - \boldsymbol{x}_i}{\|\boldsymbol{x}_l - \boldsymbol{x}_i\|} w_{il} A_{il}(m). \tag{25}$$

Although one could define w_{ij} as a function of the distance $\|\boldsymbol{x}_j - \boldsymbol{x}_i\|$, we kept w_{ij} constant with respect to the distance required to maintain the simplicity of the model with fewer hyperparameters.

C.1 GRADIENT

$\tilde{\mathbf{D}}$ can be viewed as a Laplacian matrix based on $\mathbf{D}$; however, $\tilde{\mathbf{D}} * {}^0\mathbf{H}$ can be interpreted as the gradient within the Euclidean space. Let $\nabla\, {}^0\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times f \times d}$ be an approximation of the gradient of ${}^0\mathbf{H}$. Using equation 24, the gradient model can be expressed as follows:

$$\left(\nabla\, {}^0\mathbf{H}\right)_{igk} = \frac{\partial\, {}^0\mathsf{H}_{ig}}{\partial x_k} \tag{26}$$

$$= \mathsf{D}_{ijk}({}^0\mathsf{H}_{jg} - {}^0\mathsf{H}_{ig}). \tag{27}$$

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

$$
\begin{aligned}
\left(\tilde{D} * {}^0\mathbf{H}\right)_{igk} &= \sum_j \tilde{D}_{ijk}\, {}^0H_{jg} && (28)\\
&= \sum_j (D_{ijk} - \delta_{ij} \sum_l D_{ilk})\, {}^0H_{jg} \\
&= \sum_j D_{ijk}\, {}^0H_{jg} - \sum_{j,l} \delta_{ij} D_{ilk}\, {}^0H_{jg} \\
&= \sum_j D_{ijk}\, {}^0H_{jg} - \sum_l D_{ilk}\, {}^0H_{ig} \\
&= \sum_j D_{ijk}\, {}^0H_{jg} - \sum_j D_{ijk}\, {}^0H_{ig} \qquad (\because \text{Change the dummy index } l \to j) \\
&= \sum_j D_{ijk}({}^0H_{jg} - {}^0H_{ig}) \\
&= \left(\nabla\, {}^0\mathbf{H}\right)_{igk}. && (29)
\end{aligned}
$$

Therefore, $\tilde{\mathbf{D}}*$ can be interpreted as the gradient operator within a Euclidean space.

C.2 DIVERGENCE

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

$$
\begin{aligned}
\left(\nabla \cdot {}^1\mathbf{H}\right)_{ig} &= \left(\sum_k \frac{\partial\, {}^1\mathbf{H}}{\partial x_k}\right)_{ig} && (30)\\
&= \sum_{j,k} D_{ijk}({}^1H_{jgk} - {}^1H_{igk}). && (31)
\end{aligned}
$$

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

$$
\begin{aligned}
(\tilde{\mathbf{D}} \odot {}^1\mathbf{H})_{ig} &= \sum_{j,k} \tilde{\mathbf{D}}_{ijk}\, {}^1H_{igk} \\
&= \sum_{j,k} \left(\mathbf{D}_{ijk} - \delta_{ij} \sum_l \mathbf{D}\right)\, {}^1H_{igk} \\
&= \sum_{j,k} \mathbf{D}_{ijk}\, {}^1\mathbf{H}_{jgk} - \sum_{l,k} \mathbf{D}_{ilk}\, {}^0\mathbf{H} igk \\
&= \sum_{j,k} D_{ijk}({}^1H_{jgk} - {}^1H_{igk}) \qquad (\because \text{Change the dummy index } l \to j) \\
&= (\nabla \cdot {}^1\mathbf{H})_{ig}. && (32)
\end{aligned}
$$

C.3 LAPLACIAN OPERATOR

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

Using equation 24, the Laplacian model $\nabla^2\ {}^0\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times f}$ can be expressed as follows:

$$
\begin{aligned}
\left(\nabla^2\ {}^0\mathbf{H}\right)_{ig} &:= \sum_k \left[\frac{\partial}{\partial x_k}\left(\frac{\partial \mathbf{H}}{\partial x_k}\right)_i\right]_{ig} \\
&= \sum_{j,k} D_{ijk}\left[\left(\frac{\partial \mathbf{H}}{\partial x_k}\right)_{jg} - \left(\frac{\partial \mathbf{H}}{\partial x_k}\right)_{ig}\right] \\
&= \sum_{j,k} D_{ijk}\left[\sum_l D_{jlk}({}^0H_{lg} - {}^0H_{jg}) - \sum_l D_{ilk}({}^0H_{lg} - {}^0H_{ig})\right] \\
&= \sum_{j,k,l} D_{ijk}(D_{jlk} - D_{ilk})({}^0H_{lg} - {}^0H_{jg}). \qquad (33)
\end{aligned}
$$

Then, $(\tilde{\mathbf{D}} \odot \tilde{\mathbf{D}})^0\mathbf{H}$ is

$$
\begin{aligned}
\left((\tilde{\mathbf{D}} \odot \tilde{\mathbf{D}})^0\mathbf{H}\right)_{ig} &= \sum_{j,k,l} \tilde{D}_{ijk}\tilde{D}_{jlk}{}^0H_{lg} \\
&= \sum_{j,k,l}\left(D_{ijk} - \delta_{ij}\sum_m D_{imk}\right)\left(D_{jlk} - \delta_{jl}\sum_n D_{jnk}\right){}^0H_{lg} \\
&= \sum_{j,k,l} D_{ijk}D_{jlk}{}^0H_{lg} - \sum_{j,k,n} D_{ijk}D_{jnk}{}^0H_{jg} \\
&\quad - \sum_{k,l,m} D_{imk}D_{ilk}{}^0H_{lg} + \sum_{k,m,n} D_{imk}D_{ink}{}^0H_{ig} \\
&= \sum_{j,k,l} D_{ijk}D_{jlk}{}^0H_{lg} - \sum_{j,k,n} D_{ijk}D_{jnk}{}^0H_{jg} \\
&\quad - \sum_{k,l,j} D_{ijk}D_{ilk}{}^0H_{lg} + \sum_{k,j,n} D_{ijk}D_{ink}{}^0H_{ig} \\
&\qquad (\because \text{Change the dummy index } m \to j \text{ for the third and fourth terms}) \\
&= \sum_{j,k,l} D_{ijk}(D_{jlk} - D_{ilk})({}^0H_{lg} - {}^0H_{jg}) \\
&\qquad (\because \text{Change the dummy index } n \to l \text{ for the second and fourth terms}) \\
&= \left(\nabla^2\ {}^0\mathbf{H}\right)_{ig}. \qquad (34)
\end{aligned}
$$

C.4 Jacobian and Hessian operators

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

$$
\begin{aligned}
\left(\boldsymbol{J}[{}^1\mathbf{H}]\right)_{igkl} &= \left(\frac{\partial\ {}^1\mathbf{H}}{\partial x_l}\right)_{igk} \qquad (35) \\
&= \sum_j D_{ijl}({}^1H_{jgk} - {}^1H_{igk}) \qquad (36) \\
&= (\tilde{D} \otimes {}^1\mathbf{H})_{iglk}. \qquad (37)
\end{aligned}
$$

For the Hessian model, $\mathrm{Hess}[{}^0\mathbf{H}] \in \mathbb{R}^{|\mathcal{V}| \times f \times d \times d}$,

$$\left(\mathrm{Hess}[{}^0\mathbf{H}]\right)_{igkl} = \left(\frac{\partial}{\partial x_k}\frac{\partial}{\partial x_l}\ {}^0\mathbf{H}\right)_{ig} \tag{38}$$

$$= \sum_{j,m} \mathsf{D}_{ijk}[\mathsf{D}_{jml}({}^0\mathsf{H}_{mg} - {}^0\mathsf{H}_{lg}) - \mathsf{D}_{iml}({}^0\mathsf{H}_{mg} - {}^0\mathsf{H}_{ig})] \tag{39}$$

$$= \left[(\tilde{\mathbf{D}} \otimes \tilde{\mathbf{D}}) *\ {}^0\mathbf{H}\right]_{igkl}. \tag{40}$$

D IsoGCN modeling details

To achieve isometric transformation invariance and equivariance, there are several rules to follow. Thus, here we describe where to focus when constructing an IsoGCN model. In this section, a rank-p tensor and σ denotes a tensor whose rank is $p \geq 1$ and nonlinear activation function, respectively. $\boldsymbol{W}$ and $\boldsymbol{b}$ are a trainable weight matrix and a trainable bias.

D.1 Activation and bias

As nonlinear activation function is not isometric transformation equivariant, one cannot apply nonlinear activation to rank-p tensors, while one can apply any activation to rank-0 tensors. In addition, adding bias is also not isometric transformation equivariant, one cannot add bias when performing an affin transformation to rank-p tensors. Again, one can add bias to rank-0 tensors.

Thus, for instance, if one perform conversion from rank-0 tensors ${}^0\mathbf{H}$ to rank-1 tensors using IsoAM $\mathbf{G}$, $\mathbf{G} * \sigma({}^0\mathbf{H}\boldsymbol{W} + \boldsymbol{b})$ and $(\mathbf{G} * \sigma({}^0\mathbf{H}))\boldsymbol{W}$ are isometric equivariant functions, however $(\mathbf{G} * {}^0\mathbf{H})\boldsymbol{W} + \boldsymbol{b}$ and $\sigma\left((\mathbf{G} * \sigma({}^0\mathbf{H}))\boldsymbol{W}\right)$ is not due to the bias and the nonlinear activation, respectively. Likewise, regarding a conversion from rank-1 tensors ${}^1\mathbf{H}$ to rank-0 tensors, $\sigma\left((\mathbf{G} \odot {}^1\mathbf{H})\boldsymbol{W} + \boldsymbol{b}\right)$ and $\sigma\left(\mathbf{G} \odot ({}^1\mathbf{H}\boldsymbol{W})\right)$ are isometric transformation invariant functions, however $\mathbf{G} \odot ({}^1\mathbf{H}\boldsymbol{W} + \boldsymbol{b})$ and $(\mathbf{G} \odot \sigma({}^1\mathbf{H}))\boldsymbol{W} + \boldsymbol{b}$ are not.

To convert rank-p tensors to rank-q tensors ($q \geq 1$), one can apply neither bias nor nonlinear activation. To add nonlinearity to such a conversion, we can multiply the converted rank-0 tensors $\sigma((\bigotimes^p \mathbf{G} \odot {}^p\mathbf{H})\boldsymbol{W} + \boldsymbol{b})$ to the input tensors ${}^p\mathbf{H}$ or the output tensors ${}^q\mathbf{H}$.

D.2 Preprocessing of input feature

Similar to the discussion regarding the biases, we have to take care of the preprocessing of rank-p tensors to keep isometric transformation invariance because adding a constant array and component-wise scaling could distort the tensors, resulting in broken isometric transformation equivariance.

For instance, ${}^p\mathbf{H}/\mathrm{Std}_{\mathrm{all}}\left[{}^p\mathbf{H}\right]$ is a valid transformation to keep isometric transformation equivariance, assuming $\mathrm{Std}_{\mathrm{all}}\left[{}^p\mathbf{H}\right] \in \mathbb{R}$ is a standard deviation of all components of ${}^p\mathbf{H}$. However, conversion such as ${}^p\mathbf{H}/\mathrm{Std}_{\mathrm{component}}\left[{}^p\mathbf{H}\right]$ and ${}^p\mathbf{H} - \mathrm{Mean}\left[{}^p\mathbf{H}\right]$ are not isometric transformation equivariant, assuming $\mathrm{Std}_{\mathrm{component}}\left[{}^p\mathbf{H}\right] \in \mathbb{R}^{d^p}$ is a component-wise standard deviation.

D.3 Scaling

Because the concrete instance of IsoAM $\tilde{\mathbf{D}}$ corresponds to the differential operator, the scale of the output after operations regarding $\tilde{D}$ can be considerably huge. Thus, we rescale $\tilde{\mathbf{D}}$ using the scaling factor $\left[\mathrm{Mean}_{\mathrm{sample,i}}(\tilde{\mathsf{D}}_{ii1}^2 + \tilde{\mathsf{D}}_{ii2}^2 + \tilde{\mathsf{D}}_{ii3}^2)\right]^{1/2}$, where $\mathrm{Mean}_{\mathrm{sample,i}}$ denotes mean over samples and vertices.

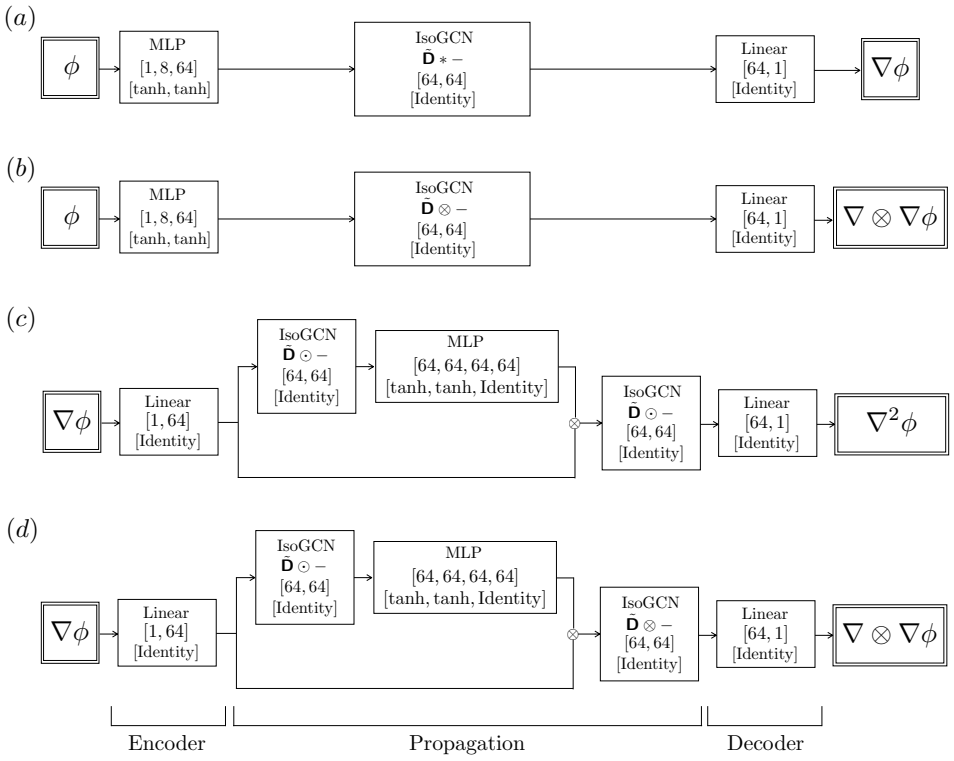

Figure 3: The IsoGCN model used for (a) the scalar field to the gradient field, (b) the scalar field to the Hessian field, (c) the gradient field to the Laplacian field, (d) the gradient field to the Hessian field of the gradient operator dataset. The numbers in each box denote the number of units. Below the unit numbers, the activation function used for each layer is also shown. $\otimes$ denotes multiplication in the feature direction.

E EXPERIMENTS DETAILS: DIFFERENTIAL OPERATOR DATASET

E.1 MODEL ARCHITECTURES

Figure 3 represents the IsoGCN model used for the differential operator dataset. We used the $\tanh$ activation function as a nonlinear activation function 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). For GCN and its variants, we simply replaced the IsoGCN layers to the corresponding ones. We stacked $m(= 2, 5)$ layers for GCN, GIN, GCNII, Cluster-GCN. We used m hop adjacency matrix for SGCN.

E.2 RESULT DETAILS

Table 5 represents the detailed comparison of training results. The proposed model achieved the best performance compared with the considered baseline models. Therefore, it can be found out the proposed model has a strong expressive power to express differential regarding space.

Table 5: Summary of the test losses (mean squared error $\pm$ the standard error of the mean) of the differential operator dataset. Here, if "" is "Yes", it means $\boldsymbol{x}$ is also in the input feature. $0 \to 1$, $0 \to 2$, $1 \to 0$, and $1 \to 2$ correspond to the scalar field to the gradient field, the scalar field to the Hessian field, the gradient field to the Laplacian field, the gradient field to the Hessian field, respectively.

Method	**# hops**	$\boldsymbol{x}$	**Loss of $0 \to 1$** $\times 10^{-5}$	**Loss of $0 \to 2$** $\times 10^{-6}$	**Loss of $1 \to 0$** $\times 10^{-6}$	**Loss of $1 \to 2$** $\times 10^{-6}$
GIN	2	No	151.19 $\pm$ 0.53	49.10 $\pm$ 0.36	542.52 $\pm$ 2.14	59.65 $\pm$ 0.46
	2	Yes	147.10 $\pm$ 0.51	47.56 $\pm$ 0.35	463.79 $\pm$ 2.08	50.73 $\pm$ 0.40
	5	No	151.18 $\pm$ 0.53	48.99 $\pm$ 0.36	542.54 $\pm$ 2.14	59.64 $\pm$ 0.46
	5	Yes	147.07 $\pm$ 0.51	47.35 $\pm$ 0.35	404.92 $\pm$ 1.74	46.18 $\pm$ 0.39
GCNII	2	No	151.18 $\pm$ 0.53	43.08 $\pm$ 0.31	542.74 $\pm$ 2.14	59.65 $\pm$ 0.46
	2	Yes	151.14 $\pm$ 0.53	40.72 $\pm$ 0.29	194.65 $\pm$ 1.00	45.43 $\pm$ 0.36
	5	No	151.11 $\pm$ 0.53	32.85 $\pm$ 0.23	542.65 $\pm$ 2.14	59.66 $\pm$ 0.46
	5	Yes	151.13 $\pm$ 0.53	31.87 $\pm$ 0.22	280.61 $\pm$ 1.30	39.38 $\pm$ 0.34
SGCN	2	No	151.17 $\pm$ 0.53	50.26 $\pm$ 0.38	542.90 $\pm$ 2.14	59.65 $\pm$ 0.46
	2	Yes	151.12 $\pm$ 0.53	49.96 $\pm$ 0.37	353.29 $\pm$ 1.49	59.61 $\pm$ 0.46
	5	No	151.12 $\pm$ 0.53	55.02 $\pm$ 0.42	542.73 $\pm$ 2.14	59.64 $\pm$ 0.46
	5	Yes	151.16 $\pm$ 0.53	55.08 $\pm$ 0.42	127.21 $\pm$ 0.63	56.97 $\pm$ 0.44
GCN	2	No	151.23 $\pm$ 0.53	49.59 $\pm$ 0.37	542.54 $\pm$ 2.14	59.64 $\pm$ 0.46
	2	Yes	151.14 $\pm$ 0.53	47.91 $\pm$ 0.35	542.68 $\pm$ 2.14	59.60 $\pm$ 0.46
	5	No	151.18 $\pm$ 0.53	50.58 $\pm$ 0.38	542.53 $\pm$ 2.14	59.64 $\pm$ 0.46
	5	Yes	151.14 $\pm$ 0.53	48.50 $\pm$ 0.35	542.30 $\pm$ 2.14	25.37 $\pm$ 0.28
Cluster-GCN	2	No	151.19 $\pm$ 0.53	33.39 $\pm$ 0.24	542.54 $\pm$ 2.14	59.66 $\pm$ 0.46
	2	Yes	147.23 $\pm$ 0.51	32.29 $\pm$ 0.24	167.73 $\pm$ 0.83	17.72 $\pm$ 0.17
	5	No	151.15 $\pm$ 0.53	28.79 $\pm$ 0.21	542.51 $\pm$ 2.14	59.66 $\pm$ 0.46
	5	Yes	146.91 $\pm$ 0.51	26.60 $\pm$ 0.19	185.21 $\pm$ 0.99	18.18 $\pm$ 0.20
IsoGCN (Ours)	2	No	**2.67** $\pm$ 0.02	**6.37** $\pm$ 0.07	**7.18** $\pm$ 0.06	**1.44** $\pm$ 0.02
	5	No	14.19 $\pm$ 0.10	21.72 $\pm$ 0.25	34.09 $\pm$ 0.19	8.32 $\pm$ 0.09

F EXPERIMENTS DETAILS: ANISOTROPIC NONLINEAR HEAT EQUATION DATASET

F.1 DATASET

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

$$\Omega \subset \mathbb{R}^3, \tag{41}$$

$$\frac{\partial T(\boldsymbol{x}, t)}{\partial t} = \nabla \cdot \boldsymbol{C}(T(\boldsymbol{x}, t)) \nabla T(\boldsymbol{x}, t), \text{in } \Omega, \tag{42}$$

$$T(\boldsymbol{x}, t = 0) = T_{0.0}(\boldsymbol{x}), \text{in } \Omega, \tag{43}$$

$$\nabla T(\boldsymbol{x}, t)|_{\boldsymbol{x} = \boldsymbol{x}_b} \cdot \boldsymbol{n}(\boldsymbol{x}_b) = 0, \text{on } \partial\Omega, \tag{44}$$

where T, $T_{0.0}$, $\boldsymbol{C} \in \mathbb{R}^{d \times d}$, and $\boldsymbol{n}(\boldsymbol{x}_b)$ are the temperature field, initial temperature field, an anisotropic diffusion tensor, and the normal vector at $\boldsymbol{x}_b \in \partial\Omega$, respectively. Here, $\boldsymbol{C}$ depends on temperature thus the equation is nonlinear. We randomly generate $\boldsymbol{C}(T = -1)$ such that it is positive semidefinite symmetric tensor whose eigenvalues varies from 0.0 to 0.02. Then we defined linear temperature dependency whose slope is $-\boldsymbol{C}(T = -1)/4$. The function of the anisotropic diffusion tensor is uniform for each one sample. The task is defined to predict temperature field at $t = 0.2, 0.4, 0.6, 1.0$ ($T_{0.2}, T_{0.4}, T_{0.6}, T_{0.8}, T_{1.0}$) from the given initial temperature field, material property, and mesh geometry. However, the performance is evaluated only with $T_{1.0}$ to focus on predictive performance. We put other output fearues just to stabilize the trainings. Accordingly, the diffusion number of this problem is $\boldsymbol{C}\Delta t/(\Delta x)^2 \simeq 10.0^4$ with assuming $\Delta x \simeq 10.0^{-3}$.

Figure 4 represents the process of generating the dataset. We generated up to 9 FEA results for each CAD shape. To avoid data leakage in terms of the CAD shapes, we first split them into training, validation, and test datasets, and then applied the following process.

Using one CAD shape, we generated up to three meshes using clscale (a 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 into a cube with an edge length equal to 1.

Using one mesh, we generated three initial conditions rondomly using a Fourier series of the 2nd to 10th orders. We then applied an FEA for each initial condition and material property determinded rondomly as described above. We applied an implicit method to solve time evolutions and a direct method to solve the linear equations. The time step of the FEA was set equal to 0.01.

During this process, some of the meshes or FEA results might not be available owing to an excessive computation time or non-convergence. Therefore, the size of the dataset was not exactly equal to the number multiplied by 9. Finally, we obtained 439 FEA results for the training dataset, 143 FEA results for the validation dataset, and 140 FEA results for the test dataset.

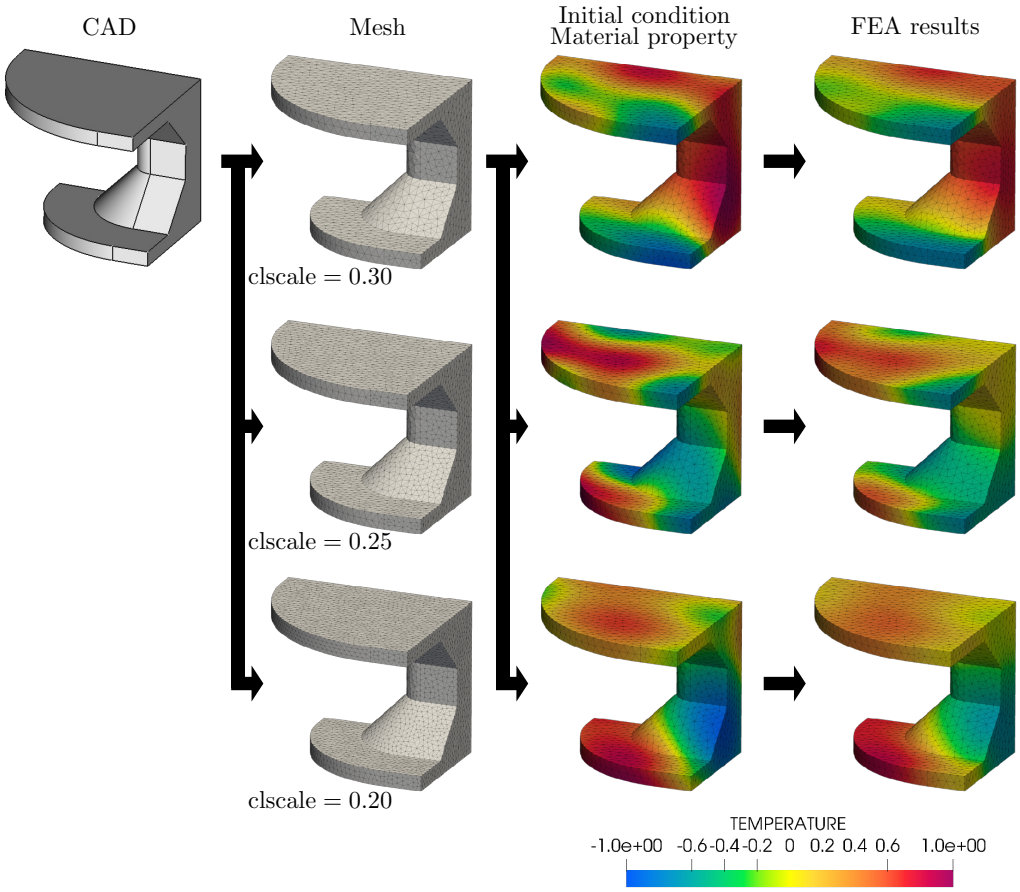

Figure 4: The process of generating the dataset. A smaller clscale parameter generates smaller meshes.

F.2 Input features

To express the geometry information, we extracted the effective volume of the ith vertex $V_i^{\text{effective}}$ and the mean volume of the ith vertex V_i^{mean}, which are defined as follows:

$$V_i^{\text{effective}} = \sum_{e \in \mathcal{N}_i^e} \frac{1}{4} V_e, \tag{45}$$

$$V_i^{\text{mean}} = \frac{\sum_{e \in \mathcal{N}_i^e} V_e}{|\mathcal{N}_i^e|}, \tag{46}$$

where $\mathcal{N}_i^e$ is the set of elements including the ith vertex.

For baseline models, we tested several combinations of input vertex features $T_{0.0}$, $\boldsymbol{C}$, $V^{\text{effective}}$, V^{mean}, and $\boldsymbol{x}$ (Table 6). For the IsoGCN model, inputs were $T_{0.0}$, $\boldsymbol{C}$, $V^{\text{effective}}$, and V^{mean}.

F.3 MODEL ARCHITECTURES

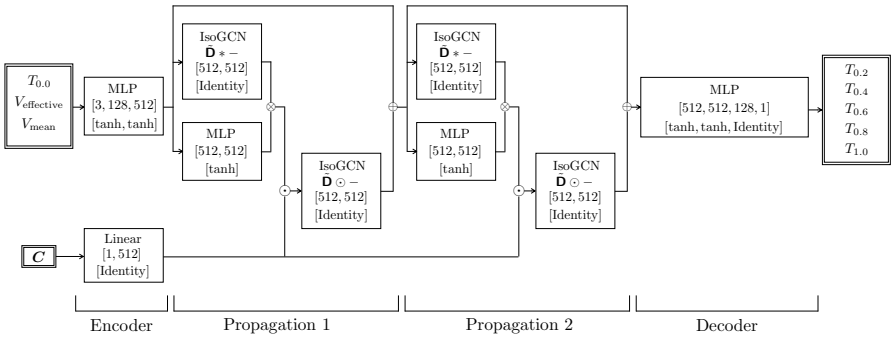

Figure 5: The IsoGCN model used for the anisotropic nonlinear heat equation dataset. The numbers in each box denote the number of units. Below the unit numbers, the activation function used for each layer is also shown. $\otimes$, $\odot$, and $\oplus$ denote multiplication in the feature direction, contraction, and addition in the feature direction, respectively.

Figure 5 represents the IsoGCN model used for the anisotropic nonlinear heat equation dataset. We used the $\tanh$ activation function as a nonlinear activation function 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). Although the model looks complicated, one propagation block corresponds to the first-order Taylor expansion $T(t+\Delta t) \simeq \nabla \boldsymbol{C} \odot \nabla T(t) + T(t)$ because the propagation block is expressed as $\tilde{\mathbf{D}} \odot \boldsymbol{C} \odot \mathrm{MLP}(T)\tilde{\mathbf{D}} * T + T$, where T denotes the rank-0 tensor input to the propagation block. By stacking this propagation block p times, one can approximate the pth order Taylor expansion of the anisotropic nonlinear heat equation.

For GCN and its variants, we simply replaced the IsoGCN layers to the corresponding ones. We stacked $m(=2,5)$ layers for GCN, GIN, GCNII, Cluster-GCN. We used m hop adjacency matrix for SGCN.

F.4 RESULT DETAILS

Table 6 shows a detailed comparison of the training results. Including $\boldsymbol{x}$ in the input features of the baseline models did not improve the performance. In addition, if $\boldsymbol{x}$ is included in the input features, it might lead to a loss of the generalization capacity for larger shapes compared to the training dataset, as it extrapolates. The proposed model achieved the best performance compared with the considered baseline models considered. Therefore, we concluded that the essential features regarding the mesh shapes are included in $\tilde{\mathbf{D}}$.

Table 6: Summary of the test losses (mean squared error $\pm$ the standard error of the mean) of the anisotropic nonlinear heat dataset. Here, if "$\boldsymbol{x}$" is "Yes", it means $\boldsymbol{x}$ is also in the input feature.

Method	**# hops**	$\boldsymbol{x}$	**Loss** $\times 10^{-3}$
GIN	2	No	16.921 ± 0.040
	2	Yes	18.483 ± 0.025
	5	No	22.961 ± 0.056
	5	Yes	17.637 ± 0.046
GCN	2	No	10.427 ± 0.028
	2	Yes	11.610 ± 0.032
	5	No	12.139 ± 0.031
	5	Yes	11.404 ± 0.032
GCNII	2	No	9.595 ± 0.026
	2	Yes	9.789 ± 0.028
	5	No	8.377 ± 0.024
	5	Yes	9.172 ± 0.028
Cluster-GCN	2	No	7.266 ± 0.021
	2	Yes	8.532 ± 0.023
	5	No	8.680 ± 0.024
	5	Yes	10.712 ± 0.030
SGCN	2	No	7.317 ± 0.021
	2	Yes	9.083 ± 0.026
	5	No	6.426 ± 0.018
	5	Yes	6.519 ± 0.020
IsoGCN (Ours)	2	No	4.674 ± 0.014
	5	No	$\mathbf{2.470} \pm 0.008$