Journal of Machine Learning Research 25 (2024) 1-36 Submitted 8/22; Revised 12/23; Published 1/24

Invariant and Equivariant Reynolds Networks

Akiyoshi Sannai SANNAI.AKIYOSHI.7Z@KYOTO-U.AC.JP
Department of Physics
Kyoto University, RIKEN
Kitashirakawa, Sakyo, Kyoto 606-8502 Japan

Makoto Kawano KAWANO@WEBLAB.T.U-TOKYO.AC.JP
Graduate School of Engineering
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654 Japan

Wataru Kumagai KUMAGAI@WEBLAB.T.U-TOKYO.AC.JP
Graduate School of Engineering
The University of Tokyo, RIKEN
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654 Japan

Editor: Jean-Philippe Vert

Abstract

Various data exhibit symmetry, including permutations in graphs and point clouds. Machine learning methods that utilize this symmetry have achieved considerable success. In this study, we explore learning models for data exhibiting group symmetry. Our focus is on transforming deep neural networks using Reynolds operators, which average over the group to convert a function into an invariant or equivariant form. While learning methods based on Reynolds operators are well-established, they often face computational complexity challenges. To address this, we introduce two new methods that reduce the computational burden associated with the Reynolds operator: (i) Although the Reynolds operator traditionally averages over the entire group, we demonstrate that it can be effectively approximated by averaging over specific subsets of the group, termed the Reynolds design. (ii) We reveal that the pre-model does not require all input variables. Instead, using a select number of partial inputs (Reynolds dimension) is sufficient to achieve a universally applicable model. Employing these methods, which hinge on the Reynolds design and Reynolds dimension concepts, allows us to construct universally applicable models with manageable computational complexity. Our experiments on benchmark data indicate that our approach is more efficient than existing methods.

Keywords: equivariance, graph neural networks, invariant representations, symmetry, Reynolds operator.

1. Introduction

Symmetry is inherent in much of the important data used in machine learning and artificial intelligence. The most important method for handling such data is learning a model that incorporates invariance/equivariance. For example, point clouds and graphs possess symmetries related to symmetric groups, and various researches have studied invariant/equivariant models for them (Murphy et al., 2019; Maron et al., 2019a; Zaheer et al., 2017; Maron et al., 2018, 2020). However, such models lack expressive power or are too computationally expensive. Therefore, in this study, we address the following problem:

Can we construct an invariant/equivariant model with both high expressive power and low computational cost?

Our method improves on the Reynolds operator approach in terms of expressive power (universality) and computational cost. Invariant/equivariant Reynolds operators are defined by

$$\gamma_G(f)(-) = \frac{1}{|G|}\sum_{g\in G} f(g\cdot -),$$
$$\tau_G(f)(-) = \frac{1}{|G|}\sum_{g\in G} g^{-1}\cdot f(g\cdot -)$$

which converts a function f into an invariant/equivariant function, respectively. We call the above f a pre-model, and the model transformed by Reynolds operators a Reynolds model. Yarotsky (2021) adapts this operator to a deep neural network to construct a model of invariant/equivariant functions, which has high expressive power. However, there are two difficulties in implementing this approach.

Difficulty 1. *Computational complexity of Reynolds operators.*

Because the order of the group that describes the symmetry is very large, the computational complexity of the Reynolds operator often becomes significant. In the case of the symmetric group S_n, for example, the computational complexity of the Reynolds operator is $n!$. Therefore, existing methods based on the Reynolds operator struggle to perform calculations on point clouds and graphs with a large n.

In this study, we employ a deep neural net f_θ, parameterized by θ, as a pre-model to derive a Reynolds model $\tau_G(f_\theta)$. Then, we regard θ as a parameter of the Reynolds model $\tau_G(f_\theta)$.

Difficulty 2. *Parameter Redundancy of Reynolds Models.*

The fact that different pre-models can be converted into the same Reynolds model leads to Difficulty 2. For example, for the symmetric group S_n and two pre-models $f_{\theta_1}(x_1, x_2, \dots, x_n) = x_1$ and $f_{\theta_2}(x_1, x_2, \dots, x_n) = x_2$, the Reynolds models of f_{θ_1} and f_{θ_2} are equal since $\gamma_{S_n}(f_{\theta_1}) = \gamma_{S_n}(f_{\theta_2})$. This implies that the two different parameters, θ_1 and θ_2, yield the same Reynolds model, indicating parameter redundancy. In general, there is a significant amount of such redundancy, depending on the order of the groups. This redundancy is a challenge that needs to be addressed, as it can lead to issues such as vanishing differentials and negatively impact the generalization gap inequality.

The primary goal of this study is to resolve these two difficulties. To achieve this, we propose two new concepts: Reynolds design and Reynolds dimension.

Reynolds design. An invariant Reynolds design for a function class $\mathcal{F}$ is a subset H of a group G satisfying

$$\frac{1}{|G|}\sum_{g\in G} f(g\cdot x) = \frac{1}{|H|}\sum_{g\in H} f(g\cdot x)$$

for any function $f \in \mathcal{F}$. An equivariant Reynolds design is similarly defined. Considering the Reynolds design for a given function class, the computational complexity can be greatly reduced. In the case of graphs, for example, we show that the Reynolds design is of order n^2, despite the fact that the order of the group is $n!$ (Theorem 3). With the Reynolds design, we overcome the computational complexity of Reynolds operators in Difficulty 1.

Reynolds dimension. The Reynolds model of a pre-model is often represented by another pre-model with less input dimension. For example, consider the weighted sum of the i-th power of variables $f = a_1 x_1^i + \cdots + a_n x_n^i$, where $\sum a_i = 1/n$, as a pre-model. Then, we have $\gamma_{S_n}(f) = \gamma_{S_n}\left(x_1^i\right)$. Therefore, the only input variable needed to represent $\gamma_{S_n}(f)$ is x_1. The Reynolds dimension d is the minimum input dimension required for functions to represent certain invariant functions through the Reynolds operator:

$$\frac{1}{|G|}\sum_{g\in G} f(g\cdot x) = \frac{1}{|G|}\sum_{g\in G} \tilde{f}_d((g\cdot x)|_d),$$

where $\tilde{f}_d$ is a certain function with input dimension d, and $(g\cdot x)|d$ denotes the vector comprising the first d components of $g \cdot x$. The definition of Reynolds dimension is further elaborated in Section 6. As can be seen in this example, input variables can be reduced in many cases, thereby considerably reducing the parameter redundancy in Difficulty 2.

Our contributions are summarized as follows:

- We introduced a subset of a group, termed the Reynolds design, which significantly reduces the computational complexity of the Reynolds operator.
- We reduced the input space of pre-models while preserving expressive power by considering the Reynolds dimension and the corresponding partial variable inputs.
- We proved the representation theorem of equivariant maps between higher-order tensor spaces, which guarantees the universality of our models.
- Our experiments on benchmark data with models incorporating these two enhancements demonstrated that our method is more efficient than existing approaches.

2. Previous Work

Equivariance and Invariance. Various machine learning tasks aim to approximate a certain target map, such as the labeling function in classification and regression. When symmetries are present in data, the target maps often have invariance or equivariance to the symmetries. In such cases, invariant or equivariant networks are effective and efficient to approximate the target map because model complexity can be significantly reduced compared with neural networks without specific structure for the symmetries. Convolutional neural networks (CNNs) are well-known as a seminal equivariant model for translation symmetry (Krizhevsky et al., 2012). Inspired by the success of CNNs, various equivariant models have been proposed. Besides continuous symmetries such as translation, symmetries of finite groups frequently appear in many machine learning tasks. When sets or point clouds are used as inputs, the target functions are typically invariant to the order of data points. Then, this function has invariance to the permutations group on data points (Qi et al., 2017; Zaheer et al., 2017). In the case where graphs or hyper-graphs are inputs, the symmetry is represented by permutation on a tensor product space. Several researchers generalized convolution to the setting of graphs inspired by CNNs (Bruna et al., 2014; Henaff et al., 2015; Kipf and Welling, 2017; Defferrard et al., 2016). Kondor et al. (2018); Maron et al. (2019a); Chen et al. (2019) have recently investigated graph neural networks. Hartford et al. (2018) consider interaction between sets. Graham et al. (2019) consider relational databases as a generalization of graphs and provides equivariant models to handle relational databases.

An existing work similar to ours is that of Puny et al. (2022). The original definition of the Reynolds operator for functions requires averaging over the entire group G. Both Puny et al. (2022) and our method rewrite the Reynolds operator by the average over G by replacing G with an appropriate subset D. One of the differences between Puny et al. (2022) and our method is that Puny et al. (2022) constructs D depending only on the input space of functions, whereas we construct D depending on the function space of the neural networks. The above construction makes our construction of D being smaller than that of Puny et al. (2022).

Universality. The expressive power of learning models is mathematically validated by universal approximation theorems. Many universal approximation theorems have been proved for different conditions. Invariant models with universal approximation property are provided for point cloud networks and sets network (Qi et al., 2017; Zaheer et al., 2017), networks with matrices and higher-order tensors as inputs (Hartford et al., 2018), graph and hyper-graph networks (Maron et al., 2018), and networks invariant to the actions of finite groups Maron et al. (2019b). The universality of equivariant models for finite groups is proved by Ravanbakhsh et al. (2017). There are also learning models with universality in other settings (Yarotsky, 2021; Keriven and Peyré, 2019; Segol and Lipman, 2019).

3. Reynolds Operators and Reynolds Designs

In this section, we provide the definition of Reynolds operators and introduce the Reynolds design, which contributes to remarkably reducing the computational complexity of Reynolds operators.

Let G be a group. A G-action on the vector space $\mathbb{R}^N$ is the structure-preserving homomorphism $\rho : G \to \mathrm{GL}_N(\mathbb{R})$, where $\mathrm{GL}_N(\mathbb{R})$ denotes the group of $N \times N$ regular matrices. Then, for a vector $x \in \mathbb{R}^N$, we define " $\cdot$ " $: G \times \mathbb{R}^N \to \mathbb{R}^N$ by $g \cdot x = \rho(g)x$. A function $f : \mathbb{R}^N \to \mathbb{R}^M$ is invariant if $f(g \cdot x) = f(x)$ holds for any $g \in G$ and any $x \in \mathbb{R}^N$, and equivariant if $f(g \cdot x) = g \cdot f(x)$ holds for any $g \in G$ and any $x \in \mathbb{R}^N$.

Here, we define the Reynolds operators.

Definition 1 (Reynolds Operator (cf. Mumford et al. (1994), Definition 1.5)) *For a group G, the following are called the equivariant and invariant Reynolds operators, respectively:*

$$\tau_G(f)(x) = \frac{1}{|G|} \sum_{g \in G} g^{-1} \cdot f(g \cdot x), \tag{1}$$

$$\gamma_G(f)(x) = \frac{1}{|G|} \sum_{g \in G} f(g \cdot x). \tag{2}$$

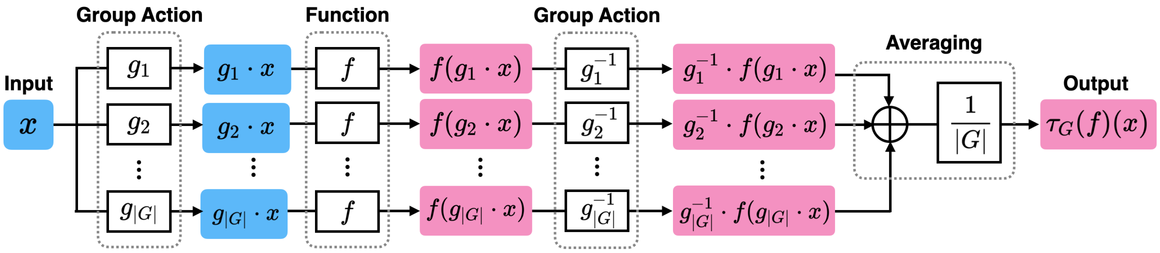


Figure 1: The equivariant Reynolds operator for a function f with a group G.

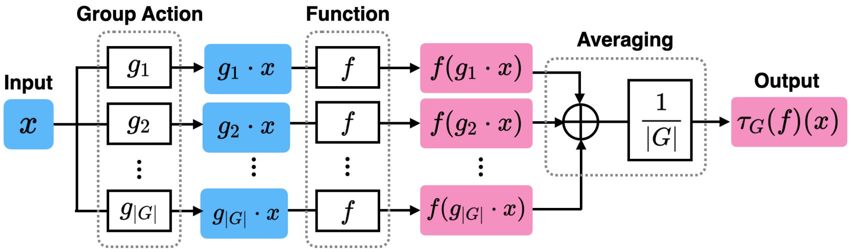


Figure 2: The invariant Reynolds operator for a function f with a group G.

Figures 1 and 2 visually represent the in-out relations of equivariant and invariant Reynolds operators. More generally, for a subset H in G,[*1] we define τ_H and γ_H by replacing G by H in (1) and (2), respectively.

The equivariant Reynolds operator converts an arbitrary map into an equivariant map. In other words, we can use Reynolds operators to construct deep neural nets of equivariant functions. However, the computational complexity of Reynolds operators increases depending on the order of G.

To reduce the computational complexity of Reynolds operators, we introduce the notion of Reynolds design. To the best of our knowledge, this is the first study to propose this notion.

Design theory in mathematics provides a suitable subset that represents some property of the whole. Here, we explain spherical design. Let $\mathcal{P}_t$ be the set of all polynomials of at most degree $t \in \mathbb{N}$ on $\mathbb{R}^d$. Then, there exists a finite subset H of the sphere $\mathbb{S}^{d-1} \subset \mathbb{R}^d$ such that an arbitrary polynomial $p \in \mathcal{P}_t$ satisfies

$$\frac{1}{|\mathbb{S}^{d-1}|} \int_{\mathbb{S}^{d-1}} p(h) dh = \frac{1}{|H|} \sum_{h \in H} p(h).$$

Then, such a subset H is called the t-spherical design.

[*1] The subset H in G may not be a subgroup of G.

Definition 2 (Reynolds Design) *The Reynolds design*[*2] H *of a function* f *is a subset of* G *that satisfies* $\tau_G(f) = \tau_H(f)$, *i.e.,*

$$\frac{1}{|G|}\sum_{g\in G} g^{-1}\cdot f(g\cdot x) = \frac{1}{|H|}\sum_{g\in H} g^{-1}\cdot f(g\cdot x). \tag{3}$$

Furthermore, when (3) holds for all $f \in \mathcal{F}$, *the Reynolds design* H *of a set of functions* $\mathcal{F}$ *is defined.*

We provide two examples of Reynolds' design in the following. First, since an invariant function f satisfies $f = \tau_G(f)$, $H = \{id\}$ is a Reynolds design of invariant functions for any G. Second, since a power of one variable satisfies $\tau_{S_n}\left(x_1^i\right) = \frac{1}{n}(x_1^i + \cdots + x_n^i) = \tau_{C_n}\left(x_1^i\right)$ with the cyclic group C_n of order n, $H = C_n$ is the Reynolds design of a power of one variable for $G = S_n$. The Reynolds design, as demonstrated in these examples, significantly reduces the computational complexity of the Reynolds operator.

4. Representation Theorem for Equivariant Maps

In this section, we provide the representation theorem for equivariant functions. The case of graph representations is an important application of the representation theorem, where a certain Reynolds design H can be of order $O(n^2)$ for graphs with n-nodes. In this case, the computational complexity of the Reynolds operator reduces from $O(n!)$ to $O(n^2)$.

Unless otherwise noted, we suppose that groups are S_n throughout the rest of this paper.

4.1 Hypergraphs and higher order tensors

A hypergraph consists of data (V, X), with V being a set of n nodes and X a tensor of rank m. X is attached to hyperedges, which is the ordered subset of V. The type of hyperedge is indicated by the rank of the tensor X, which is as follows: When $m = 1$, X represents node values, where X_i is the value of the i-th node; When $m = 2$, X represents edge values, where X_{ij} is the value attached to the (i, j) edge; in general, m-th order tensor encodes hyperedge values, where $X_{i_1,\ldots,i_m}$ represents the value of the hyperedge represented by $(i_1, \ldots, i_m)$.

For example, we represent a graph using an adjacency matrix X, where X_{ij} equals one if there is a path from vertex i to vertex j and zero otherwise. We denote the set of m-tensors by $\mathbb{R}^{n^m}$.

We define S_n-action on tensors $\boldsymbol{X} \in \mathbb{R}^{n^m\times a}$ (the last index denoted by a represents feature depth) by $(g\cdot \boldsymbol{X})_{i_1\ldots i_l,\alpha} = \boldsymbol{X}_{g^{-1}(i_1)\ldots g^{-1}(i_l),\alpha}$ for $i_1, \ldots, i_\ell \in [n]$ and $\alpha \in [a]$. Note that this is equivalent to the S_n-action induced by the permutation on the node set V. The orbits of the S_n-action for $n = 3$ and $m \in \{1, 2, 3\}$ are represented as in Figure 3.

4.2 Representation theorem of Equivariant Functions to the Spaces of Matrices.

This subsection presents a special case of the representation theorem presented in the following subsection. To do so, we introduce some notations. When a group G acts on a set X, the stabilizer of an element $x \in X$ is the subset of G defined by $Stab(x) := \{g \in G | g\cdot x = x\}$. In particular, when X is a Euclidean space $\mathbb{R}^N$, $Stab(G)$ is defined by a subset of G that fixes the first coordinate of all vectors in $\mathbb{R}^N$. Similarly, When S is a subset of X, the stabilizer of S is the subset of G

[*2] The notion of Reynolds design is analogous to that of the spherical design, by which the integral over the sphere $\mathbb{S}^{d-1}$ can be reduced to a sum over a small finite subset $H \subset \mathbb{S}^{d-1}$ (Bannai and Bannai, 2009).

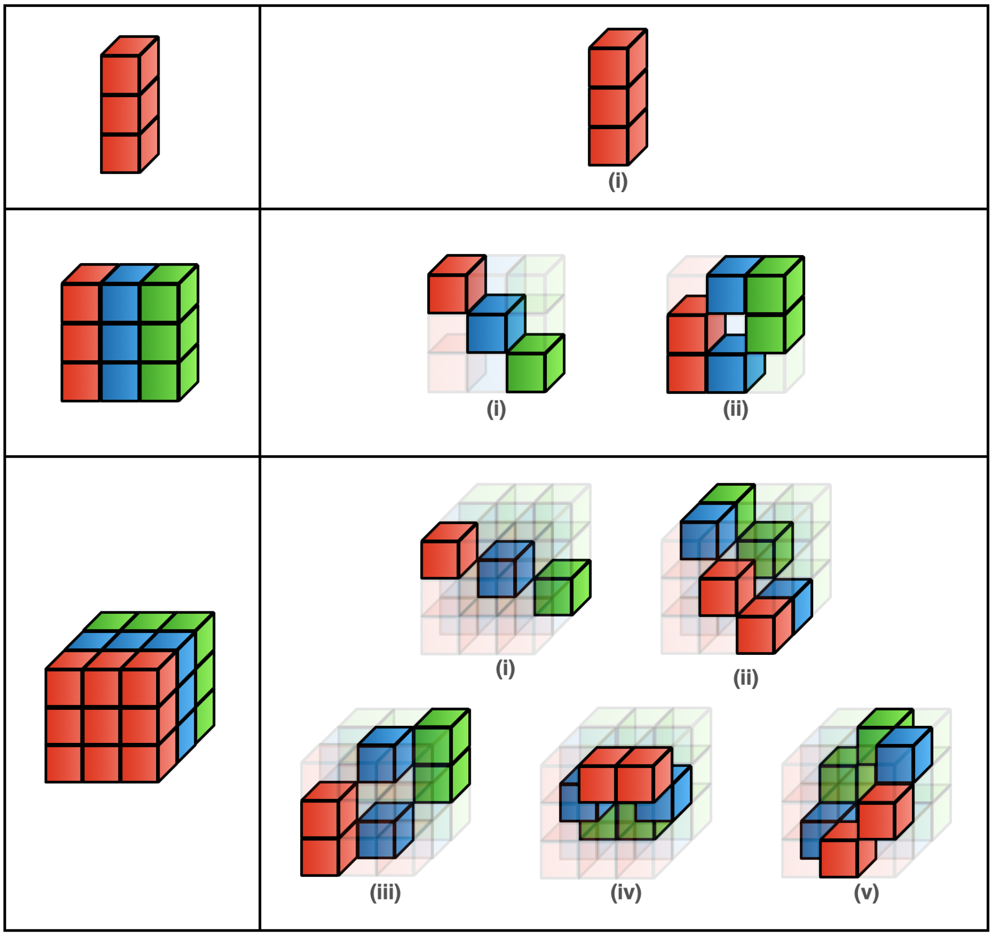


Figure 3: The orbits of group action.

defined by $Stab(S) := \{g \in G | \forall x \in S, g \cdot x = x\}$. In addition, when a group G acts on a set X, G also acts on the space of maps f from X to a set Y as $g \cdot f(x) := f(g^{-1} \cdot x)$. Then, $Stab(f)$ is well-defined (i.e., $Stab(f) := \{g \in G | g \cdot f = f\}$). Next, we define the linear map for basis vectors $\mathbf{e}_{ij} \in \mathbb{R}^{n^2}$ as follows:

$$\hat{\mathbf{e}}_{ij} : \mathbb{R} \ni a \mapsto a\mathbf{e}_{ij} \in \mathbb{R}^{n^2}.$$

Then, the following theorem holds.

Theorem 1 *Let $F : \mathbb{R}^{n^l \times a} \to \mathbb{R}^{n^2}$ be a continuous function, then F is equivariant if and only if there exist two continuous $Stab(\mathbf{e}_{11})$invariant map and $Stab(\mathbf{e}_{12})$invariant map $F_1, F_2 : \mathbb{R}^{n^l \times a} \to \mathbb{R}$ satisfying*

$$F = \tau_{H_1}(\hat{\mathbf{e}}_{11} \circ F_1) + \tau_{H_2}(\hat{\mathbf{e}}_{12} \circ F_2),$$

where H_1, H_2 are Reynolds design of $\hat{\mathbf{e}}_{11} \circ F_1, \hat{\mathbf{e}}_{12} \circ F_2$, respectively. Furthermore, we can show that $|H_1| = n$ and $|H_2| = n(n-1)$.

Theorem 1 has two major implications: a reduction in model complexity and a reduction in computational complexity. To begin, this theorem reduces learning an equivariant function F to learning functions F_1 and F_2. Note that the output spaces of the function significantly decrease from n^2 to 1. Hence, the model complexity is reduced. Second, since H_1 is of order n and H_2 is of order $n(n-1)$, the total computational complexity for the Reynolds operator is n^2. By contrast, the computational complexity of the original Reynolds operator τ_{S_n} is $n!$. Hence, the computational complexity of the Reynolds operator was reduced from $n!$ to n^2, resulting in a significant reduction in the amount of calculation.

4.3 Basis Tableau and Representation Theorem

In subsection 4.2, we proved Theorem 1, in which equivariant functions have computationally efficient representations compared to the original Reynolds operators of S_n in (1) and (2). In this subsection, we delve into the more general combinatorial structure underlying Theorem 1 and present the general version of Theorem 1, employing concepts such as Young diagrams and Young tableaux in combinatorics.

We first define a Young diagram, which is a way to represent the division of a natural number m. Here, division refers expressing a natural number m as a sum of several non-negative integers.

Definition 3 (Young Diagram) *Let $D \in [m]$ be fixed. A vector $\boldsymbol{k} = (k_1, \ldots, k_D)$ of natural numbers $k_1, \ldots, k_D$ is called a Young diagram of depth D if it satisfies $m = k_1 + \ldots + k_D$ and $k_1 \geq \ldots \geq k_D$.*

The partition can be represented by a total of m boxes, consisting of D rows with k_i boxes in the d-th row. Figure 4 provides some examples. Next, we define a basis tableau, which is a Young diagram filled with numbers.

Definition 4 (Basis Tableau) *Let $n \geq m$ and $\boldsymbol{k} = (k_1, \ldots, k_D)$ a Young diagram. A vector $\boldsymbol{T} = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_D)$ of vectors $\boldsymbol{t}_d = (t_{d,1}, \ldots, t_{d,k_d}) \in [n]^{k_d}$ is called basis tableau of depth D if it satisfies the following conditions:*

1. *$t_{d,w} \neq t_{d',w'}$ for $(d, w) \neq (d', w')$.*
2. *$t_{d,1} < t_{d,2} < \ldots < t_{d,k_d}$ for each $d \in [D]$.*
3. *If $k_d = k_{d+1}$, then $t_{d,1} < t_{d+1,1}$ for $d = 1, \ldots, D-1$.*

Figure 4 shows an example of basis tableaux for $m = 3$. We denote the set of basis tableaux for m of depth D by $\mathcal{T}_{m,D}$, and the set $\bigcup_{1 \leq D \leq m} \mathcal{T}_{m,D}$ of basis tableaux with at most depth m by $\mathcal{T}_m$. Basis tableaux determines a set of vectors.

We introduce vectors induced from basis tableaux in the following. To do so, we first introduce some notations. For the standard basis $\mathbf{e}_1, \ldots, \mathbf{e}_n$ of $\mathbb{R}^n$ and $\boldsymbol{u} = (u_1, \ldots, u_m) \in [n]^m$, we set as

$$\mathbf{e}_{\boldsymbol{u}} := \mathbf{e}_{u_1,\ldots,u_m} = \mathbf{e}_{u_1} \otimes \cdots \otimes \mathbf{e}_{u_m} \in \underbrace{\mathbb{R}^n \otimes \cdots \otimes \mathbb{R}^n}_{m} = \mathbb{R}^{n^m}. \tag{4}$$

For any basis tableau $\boldsymbol{T} = [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_D] \in \mathcal{T}_{m,D}$, natural numbers $\boldsymbol{u} = (u_1, \ldots, u_m) \in [n]^m$ are given as $u_\ell := d \in [n]$ *if* $\ell \in \{\boldsymbol{t}_d\}$.[*3] Then, we define the map $\phi : \mathcal{T}_m \to [n]^m$ by $\phi(\boldsymbol{T}) := (u_1, .., u_m) \in [n]^m$.

[*3] For a vector $\boldsymbol{t} = [t_1, \ldots, t_k]$, we set $\{\boldsymbol{t}\} := \{t_1, \ldots, t_k\}$. For example, when $\boldsymbol{t} = [1, 1]$, $\{\boldsymbol{t}\} = \{1\}$.

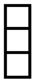
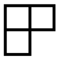
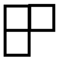
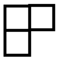
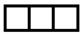
Young Diagram **k**	(1,1,1) 	(2,1) 	(2,1) 	(2,1) 	(3) 
Basis Tableau **T**	1 2 3	1 2 3	1 3 2	2 3 1	1 2 3
Basis Tableau Vector $\mathbf{e_T}$	$\mathbf{e}_{123}$	$\mathbf{e}_{112}$	$\mathbf{e}_{121}$	$\mathbf{e}_{211}$	$\mathbf{e}_{111}$

Figure 4: Young diagrams and basis tableaux for $m = 3$, and the corresponding basis tableau vectors in $\mathbb{R}^{n^3} = \mathbb{R}^n \otimes \mathbb{R}^n \otimes \mathbb{R}^n$.

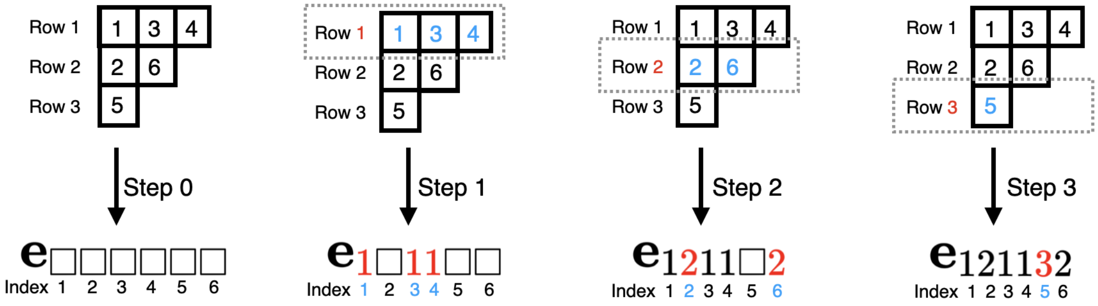


Figure 5: The conversion method from a basis tableau to the corresponding basis tableau vector. When the depth of the basis tableau is D, the number of steps above is D.

Definition 2 *Let $n \geq m$, $D \in [m]$ and $\boldsymbol{T} \in \mathcal{T}_{m,D}$. The basis tableau vector is defined by $\mathbf{e}_{\boldsymbol{T}} := \mathbf{e}_{\phi(\boldsymbol{T})} \in \mathbb{R}^{n^m}$, where the right-hand side is defined by (4).*

Figure 5 shows an example of the process to calculate a basis tableau vector. Moreover, Figure 4 shows examples of basis tableaux vectors when $m = 3$. For $\mathbf{e}_{\boldsymbol{T}} \in \mathbb{R}^{n^m}$, $\mathbf{e}_{\boldsymbol{T}}$ induces the linear map by

$$\hat{\mathbf{e}}_{\boldsymbol{T},b} : \mathbb{R}^b \ni (a_1, \dots, a_b) \mapsto (a_1\mathbf{e}_{\boldsymbol{T}}, \dots, a_b\mathbf{e}_{\boldsymbol{T}}) \in \mathbb{R}^{n^m \times b}. \tag{5}$$

Then, we have the following theorem.

Theorem 3 (Representation Theorem) *Let $n \geq m$ and $G = S_n$. For any continuous map $F : \mathbb{R}^{n^l \times a} \to \mathbb{R}^{n^m \times b}$, F is equivariant if and only if there exist $Stab(\hat{\mathbf{e}}_{\boldsymbol{T},b})$-invariant continuous maps $F_{\boldsymbol{T}} : \mathbb{R}^{n^l \times a} \to \mathbb{R}^b$ indexed by basis tableaux $\boldsymbol{T} \in \mathcal{T}_m$ such that*

$$F = \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \tau_{H_D}(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ F_{\boldsymbol{T}}),$$

where C_{n-i} denote the cyclic group of order $n - i$ on the set $\{i + 1, .., n\}$ and

$$H_D := C_n \circ \cdots \circ C_{n-D+1}$$

$$= \{\sigma_n \cdot \sigma_{n-1} \cdots \sigma_{n-D+1} \mid \sigma_i \in C_i \ (i \in [n-D+1, n])\}.$$

Furthermore, H_D is a Reynolds design of $\hat{\mathrm{e}}_{\boldsymbol{T},b} \circ F_{\boldsymbol{T}}$ for any $\boldsymbol{T} \in \mathcal{T}_{m,D}$.

Theorem 1 corresponds to the case $m = 2$ in Theorem 3. In this case, the Young diagrams are $\boldsymbol{k} = (2)$ and $(1,1)$, whose depth is 1 and 2 respectively. Also, the basis tableaux are uniquely indexed by the above two Young diagrams. As a result, we have two functions, F_1 and F_2, and two bases, e_{11} and e_{12}. The order of H_1 and H_2 follows from the definition directly.

In general, it is known that the generalization bound is improved by $|Stab(G)|^{1/2}$ when using the invariant model compared to when using the fully connected model (Sannai et al., 2021).

5. Proof of Theorem 3

We introduce the following notion to obtain tableau-based representation of elements in $[n]^m$.

Definition 4 (Extended Tableau) *Let $n \geq D$. Let $[n]^D_\#$ be the set of D different natural numbers at most n defined by $[n]^D_\# := \{[j_1, \ldots, j_D] \in [n]^D \mid j_a \neq j_b$ for $a \neq b \in [D]\}$. Then, we call elements in $\tilde{\mathcal{T}}_{m,D} := [n]^D_\# \times \mathcal{T}_{m,D}$ extended tableaux with depth D.*

We denote the set $\bigcup_{1 \leq D \leq m} \tilde{\mathcal{T}}_{m,D}$ of extended tableaux with at most depth m by $\tilde{\mathcal{T}}_m$. The action $g \cdot [j_1, \ldots, j_D] := [g \cdot j_1, \ldots, g \cdot j_D]$ of G on $[n]^D_\#$ is well-defined. Then, we define the action of G on $\tilde{\mathcal{T}}_{m,D}$ by $g \cdot (\boldsymbol{j}, \boldsymbol{T}) := (g \cdot \boldsymbol{j}, \boldsymbol{T})$. In the following, we identify an extended tableau $([1, 2, \ldots, D], \boldsymbol{T})$ with the basis tableau $\boldsymbol{T}$.

Next, we introduce some notations to represent elements in $[n]^m$ by extended tableaux. We first define the partial order $\succ$ of vectors of natural numbers that can have different dimensions. For $\boldsymbol{t} = [t_1, \ldots, t_k] \in \mathbb{N}^k$ and $\boldsymbol{t}' = [t'_1, \ldots, t'_k] \in \mathbb{N}^{k'}$, we denote as $\boldsymbol{t} \succ \boldsymbol{t}'$ if either (i) $k > k'$ or (ii) $k = k'$ and $t_1 > t'_1$. We note that $\boldsymbol{t}_1 \succ \ldots \succ \boldsymbol{t}_D$ holds for a basis tableau $\boldsymbol{T} = [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_D]$ by definition.

Let $\boldsymbol{u} = [u_1, \ldots, u_k] \in [n]^m$. We set $D(\boldsymbol{u}) := |\{\boldsymbol{u}\}|$. We define the multiplicity map $\mathrm{mult}_{\boldsymbol{u}} : \{\boldsymbol{u}\} \to [m]$ by $\mathrm{mult}_{\boldsymbol{u}}(u) := |\{\ell \in [m] \mid u_\ell = u\}|$. For $u \in \{\boldsymbol{u}\}$, we define $\boldsymbol{t}_u := [t_1, \ldots, t_{k_u}]$, where $k_u := \mathrm{mult}_{\boldsymbol{u}}(u)$, $u_{t_1} = \cdots = u_{t_{k_u}} = \boldsymbol{u}$, and $t_1 < \cdots < t_{k_u}$. Here, we note that either $\boldsymbol{t}_u \succ \boldsymbol{t}_{u'}$ or $\boldsymbol{t}_u \prec \boldsymbol{t}_{u'}$ holds for $u \neq u' \in \{\boldsymbol{u}\}$ by definition. Thus, there exist different natural numbers $j_1, \ldots, j_D \subset [n]$ such that $\{j_1, \ldots, j_D\} = \{\boldsymbol{u}\}$ and $\boldsymbol{t}_{j_1} \succ \ldots \succ \boldsymbol{t}_{j_D}$.

Definition 5 (Tableau Representation) *Let $n \geq m$. We define the map $\psi : [n]^m \ni \boldsymbol{u} \mapsto (\boldsymbol{j}, \boldsymbol{T}) \in \tilde{\mathcal{T}}_m$ called tableau representation by*

$$\psi(\boldsymbol{u}) := ([j_1, \ldots, j_D], [\boldsymbol{t}_{j_1}, \ldots, \boldsymbol{t}_{j_D}]) \in \tilde{\mathcal{T}}_{m,D}.$$

We define the map $\phi : \tilde{\mathcal{T}}_m \to [n]^m$ as follows: For any extended tableau $(\boldsymbol{j}, \boldsymbol{T}) \in \tilde{\mathcal{T}}_{m,D}$, natural numbers $\boldsymbol{u} = [u_1, \ldots, u_m] \in [n]^m$ are given as $u_\ell := j_d \in [n]$ *if* $\ell \in \{\boldsymbol{t}_d\}$, and $\phi(\boldsymbol{j}, \boldsymbol{T}) := (u_1, .., u_m) \in [n]^m$.

Lemma 6 *When $n \geq m$, the tableau representation $\psi : [n]^m \to \tilde{\mathcal{T}}_m$ is bijective and $\psi(g \cdot \boldsymbol{u}) = g \cdot \psi(\boldsymbol{u})$ for $g \in H_D$.*

Proof First, from the construction, ψ is injective and $\phi \circ \psi = \mathrm{id}_{[n]^m}$. Since $[n]^m$ and $\mathcal{T}$ are finite sets, if we show that ϕ is injective, then ψ is bijective. For the extended tableau $\boldsymbol{S}, \boldsymbol{T}$, assume that $\phi(\boldsymbol{S}) = \phi(\boldsymbol{T})$. Note that the set of row vectors $\{\boldsymbol{s}_1, ..., \boldsymbol{s}_d\}$ of $\boldsymbol{S}$ is uniquely determined from $\phi(\boldsymbol{S})$. Then, since the extended tableaux satisfy the order $\succ$ in which this goes between the row vectors, the extended tableau $\boldsymbol{S}$ having row vectors $\{\boldsymbol{s}_1, ..., \boldsymbol{s}_d\}$ is unique. Hence, we have $\boldsymbol{S} = \boldsymbol{T}$. ■

Definition 7 (Extended Tableau Vector) *Let $n \geq m$. The extended tableau vector $\mathbf{e}_{\boldsymbol{j},\boldsymbol{T}}$ is defined by $\mathbf{e}_{\boldsymbol{j},\boldsymbol{T}} := \mathbf{e}_{\phi(\boldsymbol{j},\boldsymbol{T})}$.*

For an extended tableau $(\boldsymbol{j}, \boldsymbol{T}) \in \tilde{\mathcal{T}}_m$, the linear map $\hat{\mathbf{e}}_{\boldsymbol{j},\boldsymbol{T},b} : \mathbb{R}^b \to \mathbb{R}^{n^m \times b}$ is defined by replacing $\mathbf{e}_{\boldsymbol{T}}$ by $\mathbf{e}_{\boldsymbol{j},\boldsymbol{T}}$ in (5).

Lemma 8 (Normalization) *Let $n \geq D$. For $\boldsymbol{j} \in [n]^D_{\#}$, there uniquely exists $g \in H_D$ such that $\boldsymbol{j} = g^{-1} \cdot [1, 2, \ldots, D] \in [n]^D_{\#}$. Hence, this correspondence $[n]^D_{\#} \ni \boldsymbol{j} \to g \in H_D$ is bijective.*

Proof There uniquely exists $\sigma_1 \in C_n$ such that $\sigma_1(j_1) = 1$. Inductively, there uniquely exists $\sigma_d \in C_{n-d+1}$ such that $\sigma_d(\sigma_{d-1} \cdot \sigma_{d-2} \cdots \sigma_1(j_d)) = d$ for $d = 2, \ldots, D$. Then, $g := \sigma_D \cdots \sigma_1 \in H_D$ satisfies $g \cdot \boldsymbol{j} = [1, 2, \ldots, D]$ by definition. ■

Lemma 9 *Let T be a basis tableau of depth D. Then, $\mathcal{S}_n = H_D^{-1} \cdot Stab(e_{\boldsymbol{T},b})$ holds, where $e_{\boldsymbol{T},b} := \mathrm{Im}\ \hat{e}_{\boldsymbol{T},b}$.*

Proof The same discussion as in Lemma8 shows that there exists an $h \in H_D$ satisfying $g \cdot \boldsymbol{T} = h^{-1} \cdot \boldsymbol{T}$ for any $g \in \mathcal{S}_n$. This implies that $hg \in Stab(e_{\boldsymbol{T},b})$. Hence, $g = h^{-1} \cdot hg \in H_D^{-1} \cdot Stab(e_{\boldsymbol{T},b})$ holds. ■

From Lemma 8, an extended tableau $(\boldsymbol{j}, \boldsymbol{T}) \in \tilde{\mathcal{T}}_{m,D}$ is uniquely represented by $(\boldsymbol{j}, \boldsymbol{T}) = g^{-1} \cdot ([1, \ldots, D], \boldsymbol{T}) = g^{-1} \cdot \boldsymbol{T}$ as in Figure 6, where $g \in H_D$, and we identified $\boldsymbol{T} \in \mathcal{T}_{m,D}$ with $([1, \ldots, D], \boldsymbol{T}) \in \tilde{\mathcal{T}}_{m,D}$ in the last equation. Thus, we have $\tilde{\mathcal{T}}_{m,D} = \bigcup_{g \in H_D} g^{-1} \mathcal{T}_{m,D}$, and $\tilde{\mathcal{T}}_m = \bigcup_{1 \leq D \leq m} \bigcup_{g \in H_D} g^{-1} \mathcal{T}_{m,D}$. From Lemma 6, for each $\boldsymbol{u} \in [n]^m$, there uniquely exists $g \in H_D$ and $\boldsymbol{T} \in \mathcal{T}_m$ such that $\psi(\boldsymbol{u}) = g^{-1} \cdot \boldsymbol{T}$ (or equivalently $\boldsymbol{u} = \psi^{-1}(g^{-1} \cdot \boldsymbol{T})$). In the following, we omit the bijective ψ for notational simplicity.

In the following, we prove Theorem 3. We can write $F = \sum_{\boldsymbol{u} \in [n]^m} \hat{\mathbf{e}}_{\boldsymbol{u},b} \circ f_{\boldsymbol{u}}$ by maps $f_{\boldsymbol{u}} : \mathbb{R}^{n^l \times a} \to \mathbb{R}^b$. Then, since F is equivariant, we have $\sum_{\boldsymbol{u} \in [n]^m} \hat{\mathbf{e}}_{\boldsymbol{u},b} \circ f_{\boldsymbol{u}}(g \cdot x) = F(g \cdot x) = g \cdot F(x) = \sum_{\boldsymbol{u} \in [n]^m} \hat{\mathbf{e}}_{g \cdot \boldsymbol{u},b} \circ f_{\boldsymbol{u}}(x)$. This implies that

$$\hat{\mathbf{e}}_{\boldsymbol{u},b} \circ f_{\boldsymbol{u}}(g \cdot x) = \hat{\mathbf{e}}_{\boldsymbol{u},b} \circ f_{g^{-1} \cdot \boldsymbol{u}}(x). \tag{6}$$

Then, we obtain the following equations:

$$F(x)$$

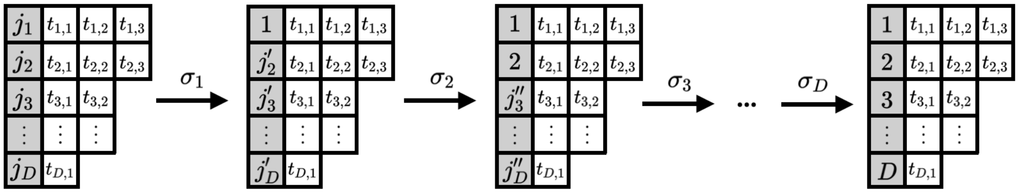


Figure 6: Reduction from an extended tableau to a basis tableau by permutations. An extended tableau $(\boldsymbol{j}, \boldsymbol{T})$ is converted into a basis tableau by multiplying $g = \sigma_D \cdot \sigma_{D-1} \cdots \sigma_1$.

$$
\begin{aligned}
&= \sum_{\boldsymbol{u}\in[n]^m} \hat{\mathbf{e}}_{\boldsymbol{u},b} \circ f_{\boldsymbol{u}}(x) \\
&= \sum_{1\le D\le m} \sum_{g\in H_D} \sum_{\boldsymbol{T}\in\mathcal{T}_{m,D}} \hat{\mathbf{e}}_{g^{-1}\cdot\boldsymbol{T},b} \circ f_{g^{-1}\cdot\boldsymbol{T}}(x) \\
&= \sum_{1\le D\le m} \sum_{\boldsymbol{T}\in\mathcal{T}_{m,D}} \sum_{g\in H_D} \hat{\mathbf{e}}_{g^{-1}\cdot\boldsymbol{T},b} \circ f_{g^{-1}\cdot\boldsymbol{T}}(x) \\
&= \sum_{1\le D\le m} \sum_{\boldsymbol{T}\in\mathcal{T}_{m,D}} \sum_{g\in H_D} \hat{\mathbf{e}}_{g^{-1}\cdot\boldsymbol{T},b} \circ f_{\boldsymbol{T}}(g\cdot x) \\
&= \sum_{1\le D\le m} \sum_{\boldsymbol{T}\in\mathcal{T}_{m,D}} \sum_{g\in H_D} g^{-1}\cdot(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}(g\cdot x)) \\
&= \sum_{1\le D\le m} \sum_{\boldsymbol{T}\in\mathcal{T}_{m,D}} \sum_{g\in H_D} \frac{1}{|H_D|} g^{-1}\cdot(|H_D|\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}(g\cdot x)) \\
&= \sum_{1\le D\le m} \sum_{\boldsymbol{T}\in\mathcal{T}_{m,D}} \tau_{H_D}\left(|H_D|\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}\right)(x) \\
&= \sum_{1\le D\le m} \sum_{\boldsymbol{T}\in\mathcal{T}_{m,D}} \tau_{H_D}\left(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ F_{\boldsymbol{T}}\right)(x),
\end{aligned}
$$

where the fourth equality follows from (6), and the last equality follows by putting $F_{\boldsymbol{T}} := |H_D| f_{\boldsymbol{T}}$. Here, assume that there is a basis tableax T such that F_T is not $Stab(e_{\boldsymbol{T},b})$-invariant. Then we can confirm $s \cdot F \neq F$ for some $s \in Stab(e_{\boldsymbol{T},b})$ by checking T-component of the functions.

Conversely, assume that we have the desired equation and $F_{\boldsymbol{T}}$ is $Stab(e_{\boldsymbol{T},b})$-invariant. To show that F is an equivariant function, it is enough to show that $\sum_{D=1}^{m}\sum_{\boldsymbol{T}\in\mathcal{T}_{m,D}} \tau_G(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ F_{\boldsymbol{T}}) = \sum_{D=1}^{m}\sum_{\boldsymbol{T}\in\mathcal{T}_{m,D}} \tau_{H_D}(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ F_{\boldsymbol{T}})$.

This is equivalent to showing that $\tau_G(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ F_{\boldsymbol{T}}) = \tau_{H_D}(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ F_{\boldsymbol{T}})$ for any T. Using Lemma 14, we can express any $g \in G$ as $h^{-1}s$ for some $h \in H_D$ and $s \in Stab(\hat{\mathbf{e}}_{\boldsymbol{T},b})$. Then we have

$$
\begin{aligned}
\tau_G(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}) &= \sum_{g\in G} g^{-1}\cdot(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}(g\cdot x)) \\
&= \sum_{g\in G} g\cdot(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}(g^{-1}\cdot x)) \\
&= \sum_{h\in H_D, s\in Stab(e_{\boldsymbol{T},b})} h^{-1}s\cdot(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}(s^{-1}h\cdot x))
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{h \in H_D, s \in Stab(e_{\boldsymbol{T},b})} h^{-1} \cdot (\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}(s^{-1}h \cdot x)) \\
&= \sum_{h \in H_D, s \in Stab(e_{\boldsymbol{T},b})} h^{-1} \cdot (\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}(h \cdot x)) \\
&= \sum_{h \in H_D} h^{-1} \cdot (\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}(h \cdot x)) \\
&= \tau_{H_D}(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}),
\end{aligned}
$$

where the fourth equality is derived from the definition of stabilizer subgroups and the fifth equality is derived from the invariance of f_T. ■

6. Invariant/Equivariant ReyNets and Universality

In this section, we provide our equivariant and invariant models and show their universality. First, we introduce multilayer perceptron.

Definition 10 (multilayer perceptron (MLP)) *Let $L \in \mathbb{N}$. A multilayer perceptron $\mathcal{N} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$ with $L + 1$ layers is a composition map of affine maps $(A_1, \ldots, A_L)$ and an activation map ρ represented by*

$$\mathcal{N} := A_L \circ \rho \circ A_{L-1} \circ \cdots \circ \rho \circ A_1,$$

where $A_\ell : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}$ are affine maps and the activation function is applied element-wise.

MLPs are well-known for their universality, provided they have arbitrary depth and width, as demonstrated in Leshno et al. (1993).

In the following, we define equivariant models. Note that not all input variables are required when considering the case of converting with Reynolds operators (see the calculation after Definition 2). Therefore, we introduce an equivariant model with a selection of variables, called d-reduced Equivariant Reynolds Nets.

Definition 11 (d-reduced Equivariant Reynolds Nets for H_D) *We assume that $n \geq m, n^l \geq d$. Let H_D be a subset of G and $\mathcal{P} : \mathbb{R}^{n^l \times a} \to \mathbb{R}^{d \times a}$ be a projection onto d-components. For any basis tableaux $\boldsymbol{T} \in \mathcal{T}_m$, let $\mathcal{N}_{\boldsymbol{T}} : \mathbb{R}^{d \times a} \to \mathbb{R}^b$ be a multilayer perceptron (MLP). The map $\mathcal{E} : \mathbb{R}^{n^l \times a} \to \mathbb{R}^{n^m \times b}$*

$$\mathcal{E} = \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \tau_{H_D}(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ \mathcal{N}_{\boldsymbol{T}} \circ \mathcal{P}),$$

is called a d-reduced equivariant Reynolds network (equivariant ReyNet).

We simply call n^l-reduced equivariant ReyNets (i.e., $\mathcal{P}$ is the identity matrix) equivariant ReyNets. Note that d-reduced ReyNets are determined by the functions $\mathcal{N}_{\boldsymbol{T}}, \boldsymbol{T} \in \mathcal{T}_m$. Theorem 3 guarantees the universal approximation property of this model.

Theorem 12 (Universality) *We assume that $n \geq m$ and $G = S_n$. Let H_D be as in Theorem 3 and $F : \mathbb{R}^{n^l \times a} \to \mathbb{R}^{n^m \times b}$ be a continuous equivariant function. For any compact set $K \subset \mathbb{R}^{n^\ell \times a}$, there exists an equivariant Reynolds network that approximates F to an arbitrary precision on K. Namely, equivariant Reynolds nets are a universal approximator for equivariant functions.*

Proof By Theorem 3, there exist continuous maps $f_{\boldsymbol{T}} : \mathbb{R}^{n^l \times a} \to \mathbb{R}^b$ satisfying

$$F = \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \tau_{H_D}(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}),$$

for standard Young tableaux $\boldsymbol{T} \in \mathcal{T}_{m,D}$. Since K is a compact set and S_n is a finite group, we may assume that K is closed under S_n-action by taking $\bigcup_{g \in S_n} g \cdot K$. Then, for any ε, we have an MLP $\mathcal{N}_{\boldsymbol{T}}$ that approximates $f_{\boldsymbol{T}}$, namely $\|\mathcal{N}_{\boldsymbol{T}} - f_{\boldsymbol{T}}\|_K < \varepsilon$ holds. Hence by the definition of our invariant model, we have

$$\begin{aligned}
&\left\| F - \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \tau_{H_D}\left(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ \mathcal{N}_{\boldsymbol{T}}\right) \right\|_K \\
&= \left\| \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \tau_{H_D}\left(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}\right) - \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \tau_{H_D}\left(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ \mathcal{N}_{\boldsymbol{T}}\right) \right\|_K \\
&\leq \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \left\|\tau_{H_D}\left(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ f_{\boldsymbol{T}}\right) - \tau_{H_D}\left(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ \mathcal{N}_{\boldsymbol{T}}\right)\right\|_K \\
&\leq \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \left\|\tau_{H_D}\left(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ (f_{\boldsymbol{T}} - \mathcal{N}_{\boldsymbol{T}})\right)\right\|_K \\
&\leq \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \left\| \sum_{g \in H_D} \frac{1}{|H_D|} \hat{\mathbf{e}}_{g^{-1} \cdot \boldsymbol{T},b} \circ \left(f_{\boldsymbol{T}} - \mathcal{N}_{\boldsymbol{T}}\right)(g \cdot -) \right\|_K \\
&\leq \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \sum_{g \in H_D} \frac{1}{|H_D|} \left\|\hat{\mathbf{e}}_{g^{-1} \cdot \boldsymbol{T},b} \circ \left(f_{\boldsymbol{T}} - \mathcal{N}_{\boldsymbol{T}}\right)(g \cdot -)\right\|_K \\
&\leq \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \sum_{g \in H_D} \frac{1}{|H_D|} \left\|\hat{\mathbf{e}}_{g^{-1} \cdot \boldsymbol{T},b} \circ \left(f_{\boldsymbol{T}} - \mathcal{N}_{\boldsymbol{T}}\right)(-)\right\|_K \\
&\leq \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \sum_{g \in H_D} \frac{1}{|H_D|} \left\|f_{\boldsymbol{T}} - \mathcal{N}_{\boldsymbol{T}}\right\|_K \\
&\leq \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \sum_{g \in H_D} \frac{1}{|H_D|} \varepsilon \\
&\leq m|\mathcal{T}_{m,D}|\varepsilon.
\end{aligned}$$

By replacing ε, we obtain

$$\left\| F - \sum_{D=1}^{m} \sum_{\boldsymbol{T} \in \mathcal{T}_{m,D}} \tau_{H_D}\left(\hat{\mathbf{e}}_{\boldsymbol{T},b} \circ \mathcal{N}_{\boldsymbol{T}}\right) \right\|_K < \varepsilon.$$

■

Task	Symmetry			
n	3	5	10	20
FNN	1.730e-4	9.180e-4	1.454e-3	3.0583
IEGN (Maron et al., 2018)	6.600e-3	3.786e-3	9.294e-4	4.471e-3
ReyNet (ours)	2.147e-4	3.960e-4	1.408e-3	3.151e-3
4-red ReyNet (ours)	**8.544e-5**	**4.889e-5**	**7.529e-5**	**6.554e-5**
Task	Diagonal			
n	3	5	10	20
FNN	1.295e-4	2.655e-4	1.148e-4	1.081e-1
IEGN (Maron et al., 2018)	2.065e-3	2.266e-3	4.098e-3	4.743e-4
ReyNet (ours)	1.007e-4	2.472e-4	6.635e-4	1.112e-4
4-red ReyNet (ours)	**6.947e-5**	**1.932e-5**	**5.568e-5**	**3.566e-5**

Table 1: Results of comparison to a baseline method

We note that the universality of equivariant ReyNets can also be demonstrated in a similar manner when the MLPs, as defined in Definition 11, are substituted with any models possessing universality for functions defined on compact domains.

Standard invariant models are combined with an MLP after an equivariant model (Zaheer et al., 2017; Maron et al., 2019b). Following this, the invariant model is constructed using the above-defined equivariant model.

Definition 13 (d-reduced Invariant Reynolds Nets) *A d-reduced invariant Reynolds network (d-red ReyNet) is a function $\mathcal{I} : \mathbb{R}^{n^l \times a} \to \mathbb{R}$ defined as*

$$\mathcal{I} = \mathcal{M} \circ \Sigma \circ \mathcal{E},$$

where $\mathcal{E} : \mathbb{R}^{n^l \times a} \to \mathbb{R}^{n^m \times b}$ is a d-reduced equivariant Reynolds network, and Σ is the orbit sum,[*4] *and $\mathcal{M}$ is an MLP.*

Next, we discuss the extent to which reduction is allowed, which is an important discussion because the size of the reduction directly affects the number of parameters. We define this size as Reynolds dimension and explain its relation to the invariant theory. Invariant theory is a field of mathematics that deals with invariant polynomials, and the set of generators of invariant polynomials, which will be introduced next, plays an essential role in invariant theory.

Definition 5 *A finite set of G-invariant polynomials $f_1, .., f_r$ of n variable is called a generator of G-invariant polynomials, if for any G-invariant polynomial f, there exists a polynomial $h(y_1, ..., y_r)$ such that $f(x_1, .., x_n) = h(f_1(x_1, .., x_n), ..., f_r(x_1, .., x_n))$.*

In the case of a general group G, there are cases where there is no generator, but in the case of a finite group, Hilbert (1890) proved that the existence of generators of invariant polynomials. Based on Hilbert's theorem, we define the following Reynolds dimensions.

[*4] The orbit sum $\Sigma : \mathbb{R}^{[n]^m \times b} \to \mathbb{R}^{[n]^m/G \times b}$ is defined by $\Sigma(\boldsymbol{X})_{G\cdot\boldsymbol{u},\beta} := \sum_{g\in G} x_{g\cdot\boldsymbol{u},\beta}$ for $\boldsymbol{X} = [x_{\boldsymbol{u},\beta}] \in \mathbb{R}^{[n]^m \times b}$, $\boldsymbol{u} \in [n]^m$, $\beta \in [b]$ and $G \cdot \boldsymbol{u} \in [n]^m/G$.

Definition 14 (Reynolds dimension) *Let G be a finite group and $r_1, .., r_s$ be generators of invariant polynomials. The smallest natural number d satisfying the following property is the Reynolds dimension of the group G. There exist polynomial $h_1, .., h_s$ of d-variables and an index subset $\{j_1, \ldots, j_d\} \subset [n]$ such that*

$$r_i(x) = \frac{1}{|G|} \sum_{g \in G} h_i((g \cdot x)|_d)$$

holds, where $|d$ denotes the projection onto $\{j_1, \ldots, j_d\}$ components.

For example, when S_n acts by permutation on an n-dimensional space, the Reynolds dimension is 1. However, if the space to be acted on is a space of tensor rank 2 or higher, such as the space of adjacency matrices of a graph, the Reynolds dimension will be 2 or higher even if the group is S_n.

In the following proposition, we construct the Reynolds design corresponding to Reynolds dimensions. For this purpose, let us review some notations. We define $\mathrm{Stab}_G([d])$ to be the set of elements of G for which $x_{j_1}, \ldots, x_{j_d}$ are fixed. In addition, $[G/G']$ a complete system of representatives of G/G' for subgroup $G' \subset G$ is a set of order $|G/G'|$ that satisfies $G = \bigcup_{a \in [G/G']} aG'$. Then we see the following proposition.

Proposition 15 *In the same situation as Definition 14, $[G/\mathrm{Stab}_G([d])]$ is a Reynolds design of h_i.*

Proof Note that h_i is a polynomial of d-variables. Hence, $\mathrm{Stab}_G([d])$ acts trivially on h_i. This implies that $\tau_G(h_i) = \tau_{[G/\mathrm{Stab}_G([d])]}(h_i)$. ■

Proposition 16 *In the same situation as Definition 14 with $G = S_n$, the Reynolds design of h_i is represented as H_d (i.e., $H_d = [G/\mathrm{Stab}_G([d])]$), where H_d was defined in Theorem 3.*

Proof We may assume that $\{j_1, \ldots, j_d\} = [d]$ by replacing the valuables. By the definition of $\mathrm{Stab}_G([d])$, it is enough to show that (1) $|G| = |H_d||\mathrm{Stab}_G([d])|$ and (2) for any $g \in G$, there are $h \in H_d$ and $s \in \mathrm{Stab}_G([d])$ satisfying $g = h \cdot s$. The first condition follows from direct calculation. To see the second condition, we calculate $g^{-1}(i)$. By the construction of H_d (Theorem 3), there is an $h \in H_d$ such that $g^{-1}(i) = h^{-1}(i)$ for any $i > d$. Also, there is an $s \in \mathrm{Stab}_G([d])$ such that $g^{-1}(i) = h^{-1}(i)$ for any $i \leq d$. Note that $\mathrm{Stab}_G([d])$ does not affect $i > d$. Therefore, $g^{-1}(i) = s^{-1} \cdot h^{-1}(i)$ holds for any i. ■

Theorem 17 (Universality) *We assume that $G = S_n$. Let d be the Reynolds dimension of G. Then, the Reynolds invariant nets constructed above for $\mathcal{E} : \mathbb{R}^{n^l \times a} \to \mathbb{R}^{n^d \times b}$ is a universal approximator for invariant functions $f : \mathbb{R}^{n^\ell \times a} \to \mathbb{R}^b$. In other words, the input space of $\mathcal{E}$ can be replaced by a composite $\mathcal{E} \circ \mathcal{Z}$ with the zero padding map $\mathcal{Z} : \mathbb{R}^{d \times a} \to \mathbb{R}^{n^\ell \times a}$.* [*5]

[*5] The zero padding map $\mathcal{Z} : \mathbb{R}^{d \times a} = \mathbb{R}^d \otimes \mathbb{R}^a \to \mathbb{R}^{n^\ell \times a} = \mathbb{R}^{n^\ell} \otimes \mathbb{R}^a$ is the linear map defined by $\mathcal{Z}((x_1, \ldots, x_d) \otimes \mathbf{e}_\alpha) := (x_1, \ldots, x_d, \underbrace{0, \ldots, 0}_{n^\ell - d}) \otimes \mathbf{e}_\alpha$ for $\alpha = 1, 2, \ldots, a$.

Finally, we describe the connection between this Reynolds dimension and universality. This is an analogy to the variable direction of width-bound universality.

Proof Let $f : \mathbb{R}^{n^\ell \times a} \to \mathbb{R}^b$ be a continuous invariant function. By replacing K with $\bigcup_{g \in G} g \cdot K$, we may assume that K is closed under the action of G. Then, by the Stone-Weierstrass theorem, there exists a polynomial $\hat{f} : \mathbb{R}^{n^\ell \times a} \to \mathbb{R}^b$ which approximate f with arbitrary precision on K. Put $\tilde{f} = \gamma_G(\hat{f})$, then

$$\begin{aligned} \left\| f(\boldsymbol{x}) - \gamma_G(\hat{f})(\boldsymbol{x}) \right\|_K &= \frac{1}{|G|} \left\| |G| f(\boldsymbol{x}) - \sum_{g \in G} f(g \cdot \boldsymbol{x}) \right\|_K \\ &\leq \frac{1}{|G|} \sum_{g \in G} \| f(\boldsymbol{x}) - \hat{f}(g \cdot \boldsymbol{x}) \|_K \\ &= \frac{1}{|G|} \sum_{g \in G} \left\| f\left(g \cdot \boldsymbol{x}\right) - \hat{f}(g \cdot \boldsymbol{x}) \right\|_K \\ &\leq \frac{1}{|G|} \sum_{g \in G} \varepsilon = \varepsilon, \end{aligned}$$

where we used the property $f(\boldsymbol{x}) = f(g \cdot \boldsymbol{x})$ in the third equation.

We now use the following theorem by Hilbert.

Theorem 18 (Hilbert finiteness theorem (Hilbert, 1890)) *Let G be a finite group or, more generally, a linearly reductive group. In this case, there is always a generator of G-invariant polynomials.*

By Theorem 18, we have a generator of invariant polynomials $r_1, \dots, r_s$. From the definition of the generator, there exists a polynomial P, and $\tilde{f}$ can be written in the form

$$\tilde{f}\left(x_1, \dots, x_{n^l a}\right) = P(r_1\left(x_1, \dots, x_{n^l a}\right), .., r_s\left(x_1, \dots, x_{n^l a}\right)).$$

By the assumption of Reynolds dimension,

$$r_1\left(x_1, \dots, x_{n^l a}\right), \dots, r_s\left(x_1, \dots, x_{n^l a}\right)$$

are written as

$$\gamma_G\left(h_1\left(x_{j_1}, \dots, x_{j_d}\right)\right), \dots, \gamma_G\left(h_s\left(x_{j_1}, \dots, x_{j_d}\right)\right)$$

for some polynomials

$$h_1\left(x_{j_1}, \dots, x_{j_d}\right), \dots, h_s\left(x_{j_1}, \dots, x_{j_d}\right)$$

of d-variables.

By Proposition 15, H_d is a complete system of representative of $G/\operatorname{Stab}([d])$. We obtain the following decomposition:

$$G = \bigcup_{g \in [G/\operatorname{Stab}_G([d])]} g \cdot \operatorname{Stab}_G([d]) = \bigcup_{g \in H_d} g \cdot \operatorname{Stab}_G([d]) = H_d \cdot \operatorname{Stab}_G([d]).$$

Then, this induces the decomposition of Reynolds operators;

$$\gamma_G = \gamma_{H_d} \circ \gamma_{\operatorname{Stab}([d])},$$

where

$$\begin{aligned}\gamma_{\mathrm{Stab}([d])}:\mathbb{R}\left[x_1,\dots,x_{n^l a}\right]\to\mathbb{R}\left[x_1,\dots,x_{n^l a}\right]^{\mathrm{Stab}([d])},\\ \gamma_{H_d}:\mathbb{R}\left[x_1,\dots,x_{n^l a}\right]^{\mathrm{Stab}([d])}\to\mathbb{R}\left[x_1,\dots,x_{n^l a}\right]^{G},\end{aligned}$$

and $\mathbb{R}\left[x_1,\dots,x_{n^l a}\right],\mathbb{R}\left[x_1,\dots,x_{n^l a}\right]^{\mathrm{Stab}([d])},\mathbb{R}\left[x_1,\dots,x_{n^l a}\right]^{G}$ are the set of polynomials, $\mathrm{Stab}([d])$-invariant polynomials, and invariant polynomials, respectively. This implies

$$\begin{aligned}r_i&=\gamma_G\left(h_i\left(x_{j_1},\dots,x_{j_d}\right)\right)\\&=\gamma_{H_d}\left(\gamma_{\mathrm{Stab}([d])}(h_i\left(x_{j_1},\dots,x_{j_d}\right))\right)\\&=\gamma_{H_d}\left(h_i\left(x_{j_1},\dots,x_{j_d}\right)\right).\end{aligned}$$

Here the last equality follows from the fact that $h_i\left(x_{j_1},\dots,x_{j_d}\right)$ is a $\mathrm{Stab}_G([d])$-invariant polynomial. By contrast, note that the invariant Reynolds operator is equal to the composition of the equivariant Reynolds operator and the orbit sum; $\gamma_G=\Sigma\circ\tau_G$. For the vector valued function $h=(h_1,..,h_s)$, by the universal approximation theorem of fully connected neural nets, we can take a fully connected neural net $\mathcal{Q}$ with which $\|\mathcal{Q}-h\|_K<\varepsilon$ holds. Let $\mathcal{N}$ be a fully connected neural net that approximates P above; $\|\mathcal{N}-P\|_K<\varepsilon$. Then,

$$\begin{aligned}&\mathcal{N}\circ\Sigma\circ\tau_{H_d}\circ Q(x_1,..,x_n)\\&\approx\mathcal{N}\circ\Sigma\circ\tau_{H_d}(h_1(x_1,..,x_n),..,h_s(x_1,..,x_n))\\&\approx\mathcal{N}(\gamma_G(h_1)(x_1,..,x_n),..,\gamma_G(h_s)(x_1,..,x_n))\\&=\mathcal{N}(r_1(x_1,..,x_n),..,r_s(x_1,..,x_n))\\&\approx P(r_1(x_1,..,x_n),..,r_s(x_1,..,x_n))\\&=\tilde{f}(x_1,..,x_n)\\&\approx f(x_1,..,x_n).\end{aligned}$$

Here we denote the approximation by $\approx$. ■

This theorem indicates that in order to have universality, m can be reduced to the minimum number of variables required to represent generators, which implies that the input dimension can be correspondingly reduced. Although the actual number of values depends on the individual case and cannot be written in closed form, we will see later that we experiment with $m=2$ and obtain competitive results.

6.1 A Conjecture on Reynolds Dimension

By Theorem 17, the Reynolds dimension provides sufficient number of input dimensions of the pre-model to guarantee the universality for invariant functions. However, it is hard to determine the Reynolds dimension in general cases. As a special case, when the symmetric group S_n acts on a tensor space $\mathbb{R}^{n^\ell}$ with the tensor rank $\ell=1$, we can verify that the Reynolds dimension is 1 and especially does not depend on n. Based on this fact, we conjecture that the Reynolds dimension is independent of sufficiently large n in the case of higher-order tensor spaces as follows.

Conjecture 6 *Let G be an S_n acting on $\mathbb{R}^{n^\ell}$ as in Section 4. For arbitrary enough large n, the Reynolds dimension $d(n)$ of G is independent of n.*

7. Advantages of Reduced ReyNets

The effect of input dimension reduction. An advantage of reducing the input dimension of pre-models is a tight inequality of the generalization gap. As is shown in Section 10.2 in Anthony and Bartlett (2009), the generalization gap is bounded by the logarithm of the volume of the hyper cube in the input space. Hence, our reduction of the input dimension of pre-models makes the upper bound tight.

Robustness for extrapolation. As a significant advantage of reduced ReyNets, it is worth mentioning that the model gives us an extrapolation method. To understand this, consider the case that we learn graphs of n-nodes with d-reduced ReyNets ($n^2 \geq d$). Here, note that the function $\mathcal{N}_{\boldsymbol{T}}$ of the d-reduced ReyNets does not depend on n by Definition 11. Thus, this function $\mathcal{N}_{\boldsymbol{T}}$ can also be applied to the case of the graphs of n'-nodes for $n' \neq n$ ($(n')^2 \geq d$). Maron et al. (2018) claimed that their models could also extrapolate. Since Maron et al. (2018) extrapolates using parameters, the gap between parameters and functions can have a negative influence. However, since reduced ReyNets transfer the functions directly, reduced ReyNets are robust under the change of n. We confirm the difference between two methods in the experiment (see Section 8.1 and Figure 7).

8. Experiments

We evaluated the performance of ReyNets in equivariant and invariant tasks using multiple data sets. First, we created synthetic data sets for equivariant and invariant tasks and compared ReyNets with fully-connected neural networks (FNNs) and invariant and equivariant Graph Networks (IEGN) (Maron et al., 2018). Then, to verify the performance with real data, we conducted experiments using eight types of graph benchmark data sets. Please refer to the appendix for the details of each experiment. Our code is publicly available at: https://github.com/makora9143/ReyNet.

8.1 Synthetic Datasets

We created four synthetic data sets for comparison. Given the input matrix data $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, each task is defined as:

a) *Symmetry*: projection onto the symmetric matrices $F(\boldsymbol{A}) = \frac{1}{2}(\boldsymbol{A} + \boldsymbol{A}^\top)$, b) *Diagonal*: diagonal extraction $F(\boldsymbol{A}) = diag(\boldsymbol{A})$, c) *Power*: computing each squared element $F(\boldsymbol{A}) = [A_{i,j}^2]$, and d) *Trace*: computing the trace $F(\boldsymbol{A}) = tr(\boldsymbol{A})$, where the task function F is equivariant with *symmetry*, *diagonal*, and *power*, and invariant with *trace*. With reference to Maron et al. (2018), we sampled i.i.d. random matrices $\boldsymbol{A}$ with uniform distribution in $[0, 10]$; then, we transformed $\boldsymbol{A}$ into $F(\boldsymbol{A})$. In our experiments, we provided $n \in \{3, 5, 10, 20\}$, and the size of the training data set and test data set was both 1000. In the experiments, we also compared ReyNet with a variant, 4-red ReyNet.

Objective Function. The squared error function with the ground truth output was used as the objective function for training. By Theorem 3, the squared error between the ReyNet output and all components of the correct matrix is equal to the squared error between the neural network output and only the (1,1) and (1,2) components of the correct matrix.

Since the equivariant task is a regression task, the model attempts to reduce the gap of all the elements between the output matrix and the ground truth matrix; $\ell_{std} \colon \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times n} \to \mathbb{R}$. Meanwhile, by using the result of Theorem 3, ReyNet is not required to calculate the gap of whole

n	MSE	Corner MSE
3	1.438e-4	**1.249e-4**
5	**4.912e-5**	9.375e-5
10	1.157e-4	**6.487e-5**
20	1.608e-4	**8.537e-5**

Table 2: Comparison of objective function

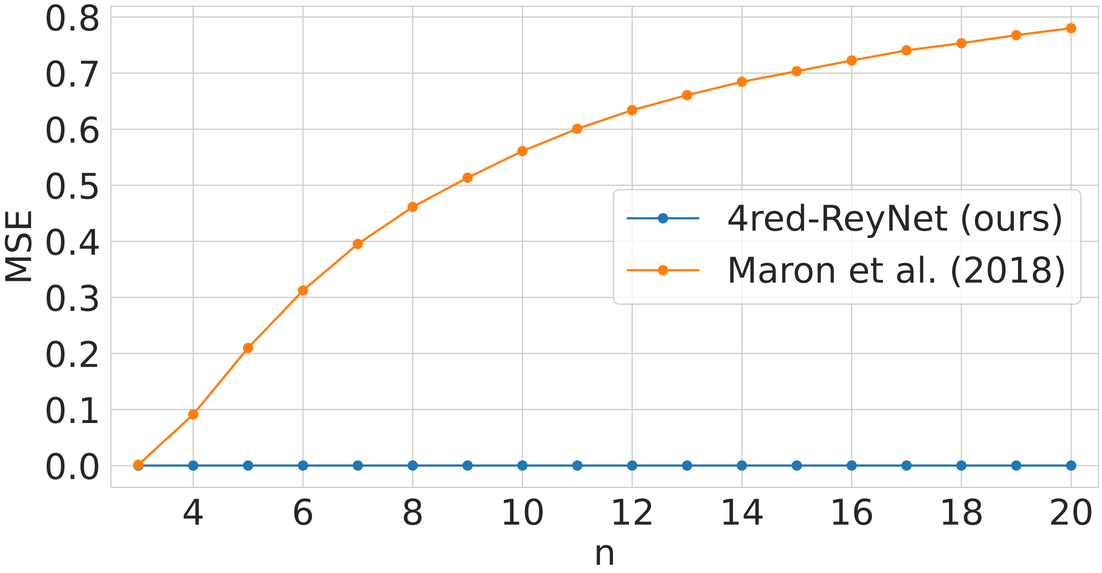


Figure 7: Extrapolation of regression on *symmetry* Synthetic Datasets.

elements but the 1st row of the 1st column element and the 1st row of the 2nd column element: $\ell_{corner}: \mathbb{R}^{1\times 2} \times \mathbb{R}^{1\times 2} \to \mathbb{R}$. In this study, we call the former objective function the standard mean squared error (MSE) loss and the latter the Corner MSE loss. We validated the effect of the Corner MSE loss function prior to conducting synthetic experiments. We trained 4-red ReyNets with standard MSE loss and Corner MSE loss. Table 8.1 shows the result of using each objective function. With the exception of the $n = 5$ case, Corner MSE loss achieved a lower error than standard MSE loss. Therefore, we used Corner MSE loss because it is a good choice for equivariant tasks.

Results. Table 1 shows the result of synthetic data sets, *symmetry* and *diagonal*, which is the average of MSE of five different seeds. The results of the remaining data sets are shown at Appendix. As a result, our 4-red ReyNet outperforms IEGN in Maron et al. (2018).

Extrapolation. Notably, the size of the inputs n has no effect on our 4-red ReyNet. To evaluate the extrapolation performance, we trained 4-red ReyNet with $n_{train} = 3$ data set and then validated the MSE on $n_{test} \in \{3, 4, \dots, 20\}$ data sets. The results are depicted in Figure 7. We can see that our 4-red ReyNet is generalized to the input size. Note that with regard to invariant tasks, we confirmed the model is not generalized to the tasks as reported by Maron et al. (2018).

	MUTAG	PTC	PROTEINS	NCI1	NCI109	COLLAB	IMDB-B	IMDB-M
Graph Model								
PPGN	88.33 ± 7.1	60.59 ± 7.9	73.96 ± 4.6	77.32 ± 2.2	78.98 ± 2.2	75.80 ± 2.0	70.60 ± 4.8	47.40 ± 3.3
GIN	85.83 ± 7.7	56.64 ± 7.0	72.56 ± 5.9	76.84 ± 2.3	73.51 ± 3.0	76.98 ± 2.1	70.60 ± 4.7	44.87 ± 3.9
Invariant Model								
IEGN	78.33 ± 10	55.59 ± 8.6	75.31 ± 5.4	76.06 ± 1.4	73.79 ± 2.9	78.12 ± 2.9	69.40 ± 6.1	47.20 ± 3.3
4-red ReyNet-(i)	89.44 ± 7.1	61.18 ± 5.2	75.41 ± 5.6	77.25 ± 2.0	75.90 ± 2.2	73.62 ± 1.6	70.10 ± 5.1	48.80 ± 2.9
4-red ReyNet-(ii)	88.33 ± 8.9	59.41 ± 7.8	74.60 ± 4.2	77.79 ± 2.1	76.19 ± 2.1	74.31 ± 1.9	70.10 ± 4.6	46.73 ± 3.9

Table 3: Graph classification results (Acc.)

8.2 Graph Classification

As an example of real-world data, we selected eight benchmark data sets from the TU Dortmund data collection (Kersting et al., 2016): five from bioinformatics and three from social networks. Since these data sets are provided for classification tasks, we treated them as invariant tasks. Due to the small size of these data sets, we followed an evaluation protocol that included a 10-fold for the data sets of Yanardag and Vishwanathan (2015). In this experiment, we prepared two variants of 4-red ReyNets with different neural network architectures $\mathcal{N}_{\boldsymbol{T}}$: three hidden units' sizes of (16, 16, 16) for 4-red ReyNet-(i) and (128, 256, 512) for 4-red ReyNet-(ii). We used IEGN (Maron et al., 2018), provably powerful graph networks (PPGN) (Maron et al., 2019a), and graph isomorphism network (GIN) (Xu et al., 2019) as the baseline methods. These are DNN-based methods used for TU Dataset experiments in their studies. Note that PPGN and GIN are practical models that are proposed for graph data, and IEGN and 4-red ReyNet are invariant models that are not limited to graph data. In this study, we re-implemented these methods by referring to available published code. The architecture of our 4-red ReyNets and baselines is adopted from Maron et al. (2018): after respective equivariant layers, the network consists of an invariant max-pooling (i.e. diagonal and off-diagonal) followed by three fully connected layers with hidden units of size (512, 256, #classes). We followed the settings in the published code for the optimizer's learning rate and the scheduler's decay rate in each data set, except when the model training was not convergent, in which case we tuned the learning rate to decrease. For more details, please refer to the appendix.

Results. We report the average accuracy scores and standard deviations of 10-folds in Table 8.1. As a result, our 4-red ReyNet outperformed the existing methods on the five bioinfomatics data

GCN	GIN	ReyNet
0.7363 ± 0.0098	0.7366 ± 0.0268	0.7342 ± 0.0161

Table 4: Results on MOL-HIV (Wu et al., 2018).

sets and achieved slightly worse results on the three social networks data sets. To demonstrate our experimental results, we show the results reported in those benchmark studies with the reproduction results in this study in the appendix.

Effectiveness on Real Data. In order to further validate the effectiveness of ReyNet on real data, we chose and performed Mol-HIV data task from the Open Graph Benchmark Dataset (OGB). The Mol-HIV data task is a classification task to predict the property of the given graph, and is one of the largest in the MoleculeNet data sets. Each graph represents a molecule, where nodes are atoms and edges are chemical bonds. Input node features are of nine dimensions. For more details, please refer to Hu et al. (2020).

Since in OGB, to encode these raw input features, AtomEncoder and BondEncoder modules are provided, we used these modules to encode the input data first. As baseline methods, we chose GCN and GIN (Kipf and Welling, 2017) listed on the leaderboard, and used implementation code available online. Due to computational cost, we set the embedding dimension of AtomEncoder and BondEncoder to 32.[*6] ReyNet was designed in an architecture refer to GIN. The results are shown in Table 4. While ReyNet was comparable to the baseline, it achieved sufficient accuracy, confirming that ReyNet is also effective on real data.

9. Discussion

From the viewpoint of computational complexity, the method proposed in this paper only reduces the computational complexity of the combinatorial explosion level to, say, n^2. Although this reduction is not sufficient from a practical standpoint, it is a solid theoretical step forward. The computational complexity of the proposed method may be improved by sparsification and other methods used in transformer and other applications. From a future perspective, the discovery of symmetry also suggests that, from a deep learning perspective, it is sufficient to discover Reynolds designs instead of discovering the entire group action.

10. Conclusion

We considered invariant/equivariant models over higher order tensor spaces. The method of converting deep neural nets using Reynolds operators had some computational complexity issues that we were able to solve by using our Reynolds designs. Then, we constructed reduced equivariant Reynolds networks (equivariant ReyNets) based on the Reynolds designs and proved their universality.

We also introduced Reynolds dimension in the invariant case. Furthermore, we constructed reduced invariant Reynolds networks (invariant ReyNets) and demonstrated their universality. More-

[*6] While the embedding dimension of GCN and GIN listed on the leaderboard is 300, if the embedding dimension of ReyNet is set to 300 dimensions, the experiment cannot be performed due to memory usage. Therefore we used 32 dimensions.

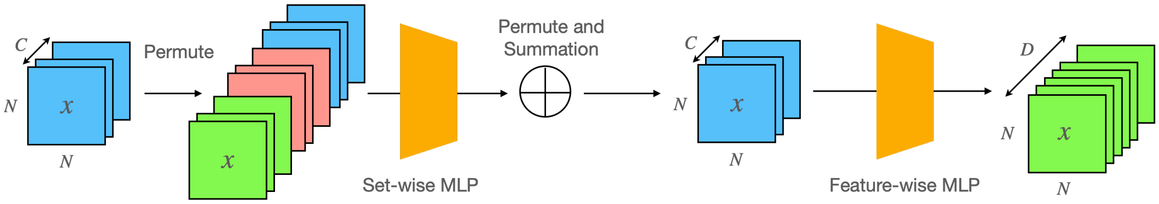


Figure 8: Architecture of invariant ReyNet.

over, we showed that invariant/equivariant ReyNets perform better than or comparable to existing models, including graph-specific models in graph classification tasks, even though ReyNets are applicable to more general tasks. Moreover, we observed that reduced ReyNets with a few input variables can extrapolate well to cases with more input variables.

Acknowledgments

The first author thanks Daigo Matsumaru and D-lab for their advice on improving research activities. The first author was supported in part by JSPS KAKENHI Grant Number JP20K03743, JP23H04484 and JST PRESTO JPMJPR2123. The third author was partially supported by the JSPS KAKENHI Grant Numbers 19H04071 and 23H04974.

Appendix A. ReyNet Implementation

A.1 ReyNet Architecture

Given Definition 11, we can implement the neural network $\mathcal{N}$ such that each layer has $C_{input} \times C_{output} \times d_{input} \times d_{output}$ weight parameters (the bias parameters are omitted for brevity), where C_* represents the number of features (dimension), and d_* represents the number of set size (dimension). For example, if we feed the data $x \in \mathbb{R}^{C\times N\times N}$ to 4-red ReyNet, which contains three fully connected layers, then the weight parameters of the first layer may be $C \times D_1 \times 4 \times d$ and those of second layer is $D_1 \times D_2 \times d \times 2$. However, by implementing these specific FC layers, the number of parameters becomes enormous. To avoid enormous parameters, we refer to the separable convolution layer (Howard et al., 2017), which separates the normal convolution layer into depthwise and pointwise convolution. That is, we separate $\mathcal{N}$ into set-wise MLP and feature-wise MLP as shown in Figure 8. This separation allows us to reduce the number of parameters. Note that although we can adopt skip connections such as Chollet (2017), we cannot observe any experimental advantage. Consequently, we did not employ skip connections in this study.

A.2 Efficient d-reduced ReyNet Implementation

Forward propagation of the d-reduced equivariant ReyNet is implemented based on Definition 11. Given an input x with a set size of N, we need to 1) permute x to $N(N-1)$ patterns using g, 2) feed the permuted input $g \cdot x$ to the neural network $\mathcal{N}_{\boldsymbol{T},b}$, 3) apply g^{-1}, and 4) average all gs. When implementing ReyNet in Python to use the deep learning framework, it is known that the *for* loop is slow due to the nature of Python. Instead of using the *for* loop, the conventional implementation

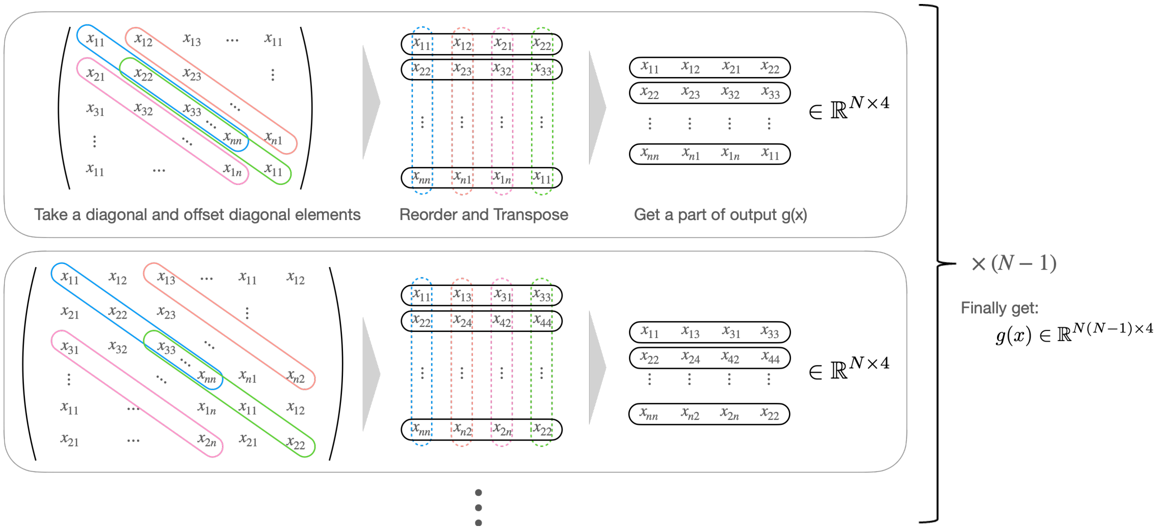

Take a diagonal and offset diagonal elements
Reorder and Transpose
Get a part of output g(x)
$\in \mathbb{R}^{N\times 4}$
$\in \mathbb{R}^{N\times 4}$
$\times (N-1)$
Finally get:
$g(x) \in \mathbb{R}^{N(N-1)\times 4}$

Figure 9: $mapping1(\cdot) = (\mathcal{P} \circ g)(\cdot)$ Implementation Trick.

is calculated in the form of a tensor such as torch.tensor/numpy.ndarray. Thus, Algorithm 1 can be used for a straightforward implementation, which is easy to understand; however, the problem is that it needs space complexity of $\mathcal{O}(N^4)$ and computational complexity $\mathcal{O}(N^2)$ based on the first and second lines, which is not practical.

Meanwhile, there are no learning parameters in the processing from the first line to the fourth line, and the processing depends only on the set size of the input data, and the output of the fourth line can be determined deterministically from the input data. Therefore, we can implement a process from the first line to the fourth line as Algorithm 2. The algorithm is also represented in the Figure 9. Moreover, the processes from the fifth to seventh lines require memory complexity $\mathcal{O}(N^4)$. However, these processes can be assembled into one method $mapping2$ depicted in Figure 10. As a result, the algorithm of our d-reduced equivariant ReyNet becomes Algorithm 3. The space complexity of this algorithm is $\mathcal{O}(N^2)$, and the computational complexity is $\mathcal{O}(N)$.

Algorithm 1 Naïve implementation of the d-reduced equivariant ReyNet (Definition 11)

```
Require: $x \in \mathbb{R}^{C\times N\times N}$, Neural Network $\mathcal{N}\colon \mathbb{R}^d \to \mathbb{R}^2$, Zero-padding $\hat{\mathbf{e}}_b\colon \mathbb{R}^2 \to \mathbb{R}^{N^2}$, Slicing
    $\mathcal{P}\colon \mathbb{R}^{N^2} \to \mathbb{R}^d$
Ensure: $y = \mathcal{E}(x) = \frac{1}{|H|}\sum_{g\in H} g^{-1}\cdot(\hat{\mathbf{e}}_b \circ \mathcal{N} \circ \mathcal{P})(g\cdot x) \in \mathbb{R}^{C\times N\times N}$
x ← insert a dimension                       // $C \times 1 \times N^2$
$y \leftarrow g \cdot x$: permute            // $C \times N(N-1) \times N^2$
$y \leftarrow \mathcal{P}(y)$: slicing       // $C \times N(N-1) \times d$
$y \leftarrow \mathcal{N}(y)$: NN            // $C \times N(N-1) \times 2$
$y \leftarrow \hat{\mathbf{e}}_b(y)$: 0-padding   // $C \times N(N-1) \times N^2$
$y \leftarrow g^{-1} \cdot y$: inv-permute   // $C \times N(N-1) \times N^2$
$y \leftarrow \frac{1}{|H|}\sum_{g\in H} y$: average   // $C \times N^2$
return $y$
```

Algorithm 2 $mapping1(x, N)$

```
Require: $x \in \mathbb{R}^{C\times N\times N}$, Set Size $N$
Ensure: $y \in \mathbb{R}^{C\times N(N-1)\times 2}$
$xx \leftarrow$ copy $x$ twice                 // $C \times 2N \times 2N$
$diagonal \leftarrow diag(x)$                  // $C \times 2N$
$diag1 \leftarrow diagonal[0 : N]$             // $C \times N$
$y = []$
for $i = 0 \ldots N-1$ do
    $diag2 \leftarrow i$ - offset $diag(x)$
    $diag3 \leftarrow -i$ - offset $diag(x)$
    $diag4 \leftarrow diagonal[i : N+i]$
    $z \leftarrow stack([diag1, diag2, diag3, diag4])^\top$
    $y.append(z)$
end for
return $y$
```

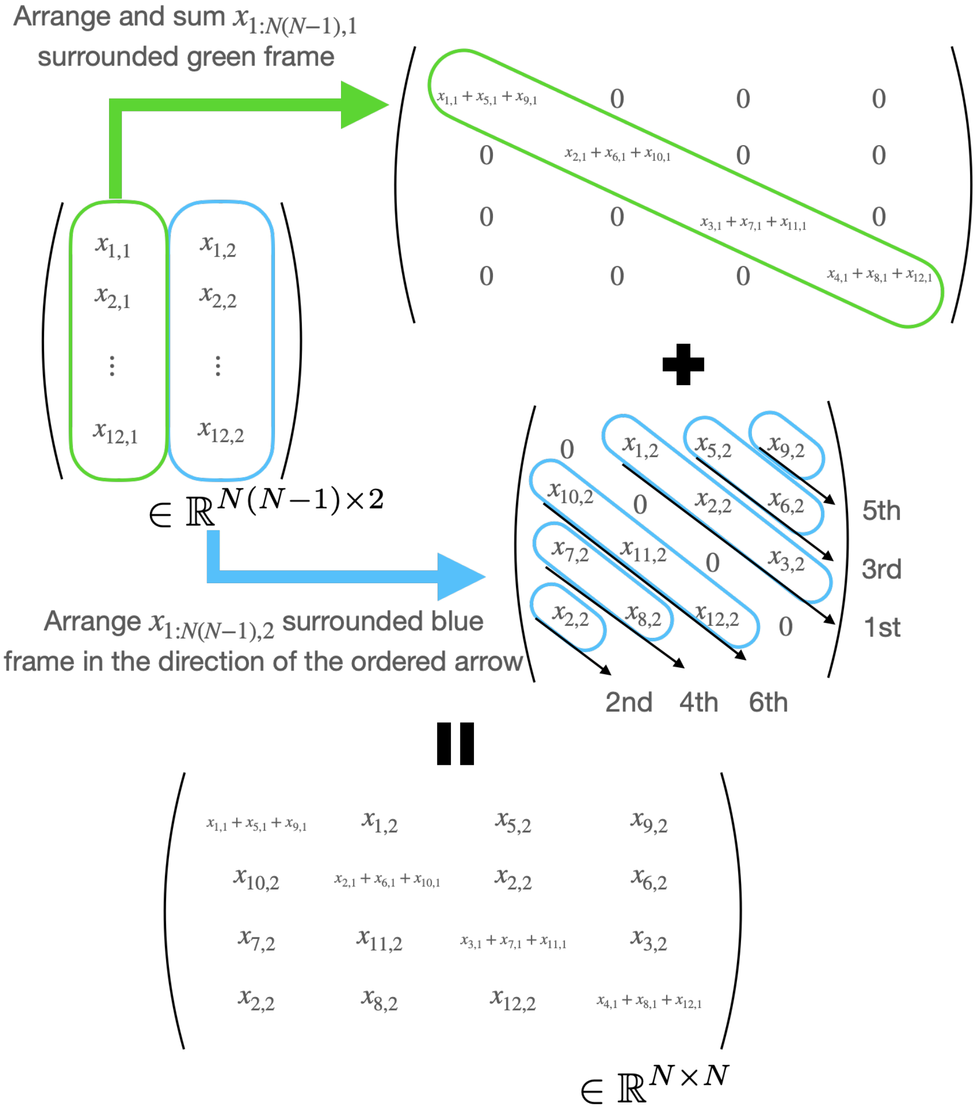

Arrange and sum $x_{1:N(N-1),1}$ surrounded green frame
$\in \mathbb{R}^{N(N-1)\times 2}$
Arrange $x_{1:N(N-1),2}$ surrounded blue frame in the direction of the ordered arrow
5th
3rd
1st
2nd
4th
6th
$\in \mathbb{R}^{N\times N}$

Figure 10: $mapping2(\cdot) = \sum(g^{-1} \circ \hat{e}_b)(\cdot)$ Implementation Trick.

Algorithm 3 Efficient implementation of the d-reduced equivariant ReyNet (Definition 11)

```
Require: $x \in \mathbb{R}^{C\times N\times N}$, Neural Network $f\colon \mathbb{R}^d \to \mathbb{R}^2$
Ensure: $y = \frac{1}{|H|}\sum_{g\in H} g^{-1}\cdot f(g\cdot x) \in \mathbb{R}^{C\times N\times N}$
  $y \leftarrow mapping1(x, N)$          // $C \times N(N-1) \times d$
  $y \leftarrow f(y)$                    // $C \times N(N-1) \times 2$
  $y \leftarrow mapping2(y, N)$          // $C \times N^2$
  return $y$
```

```

def permute(x):
    g_tensor = one_hot(torch.tensor([cyclic_perm_index(swap_positions(list(range
    (N)), 1, i))
                                     for i in range(1, N)]).reshape(-1, N),
                       num_classes=N).unsqueeze(0)  # 1 x n! x n x n
    g_size = g_tensor.size(1)
    # B x C x n x n -> B x C x 1 x n x n
    h = x.unsqueeze(2)
    # 1 x N! x N x N * B x C x 1 x N x N
    h = g_tensor.matmul(h).matmul(g_tensor.transpose(-1, -2))
    return h

def slice(x, dim):
    """
    Args:
        x: Tensor. B x C x N! x N x N
    Return
        Tensor B x C x N! x dim x dim
    """
    return x[..., :dim, :dim]


def inverse_permute():
    """
    Args:

    Return:
        tensor: B x C x N x N
    """
    output = h.new_zeros(B, self.in_features, g_size, N, N)
    output[:, :, :, 0, :2] = h
    output = g_tensor.transpose(-1, -2).matmul(output).matmul(g_tensor)
    output = output.sum(2)
    return


def forward(x):
    coeff = (torch.ones(N, N).fill_diagonal_(0) +
             torch.zeros(N, N).fill_diagonal_(1 / (N - 1))).reshape(1, 1, N, N).
    to(x)

    h = permute(x)
    h = slice(h, dim=2)
    # NN: B x C x n! x 4 -> B x C x n! x 2
    h = NN(h.reshape(B, self.in_features, g_size, -1))
    h = inverse_permute(h)
    return h * coeff.repeat(B, self.in_features, 1, 1)
```

Listing 1: Python Code of ReyNet using PyTorch.

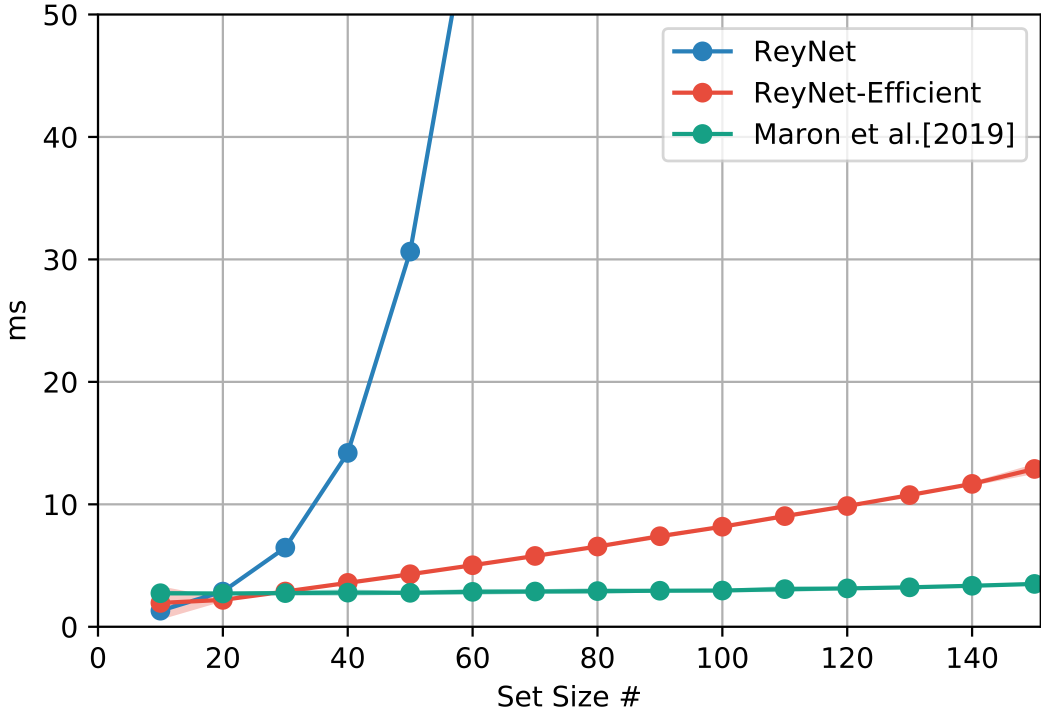


Figure 11: Runtime speed on different number of set sizes

A.3 Runtime Speed

Figures 11 and 12 show the runtime speed of our ReyNet compared with Maron et al. (2018). Figure 11 shows the runtime speed when the size of the set changes without changing the number of input features. Note that the feature size is set to 1. We can confirm that the computational complexity of Reynet is $\mathcal{O}(N^2)$ and that of efficient version is improved to $\mathcal{O}(N)$, while our efficient ReyNet is slightly slower than (Maron et al., 2018). Figure 12 shows the runtime speed when the number of input features is changed and the size of the set is not changed. Note that the input set size is set to 20. In contrast to the runtime speed of the different set sizes, the different number of features does not affect runtime speed.

Appendix B. Synthetic Regression Tasks Details

In the experiments on synthetic data sets, we used two models; Reynolds Networks (ReyNets) and 4-reduced Reynolds Networks (4-red ReyNets), while only 4-reduced Reynolds Networks (4-red ReyNets) for the graph benchmark data set. For synthetic equivariant tasks, we adopted equivariant ReyNets. For synthetic invariant tasks, we implemented invariant ReyNets for which the architecture was adopted from Maron et al. (2018); [*7] an equivariant ReyNet is followed by invariant max pooling,[*8] and fully connected layers. The fully connected layers consist of three layers, and the number of the units are 512, 256, and 1. We adopted the ReLU function as activation. We used

[*7] Please refer to https://github.com/Haggaim/InvariantGraphNetworks/blob/master/models/invariant_basic.py#L14 or our submitted code.

[*8] This operation outputs the max value of diagonal and non-diagonal elements of an input matrix.

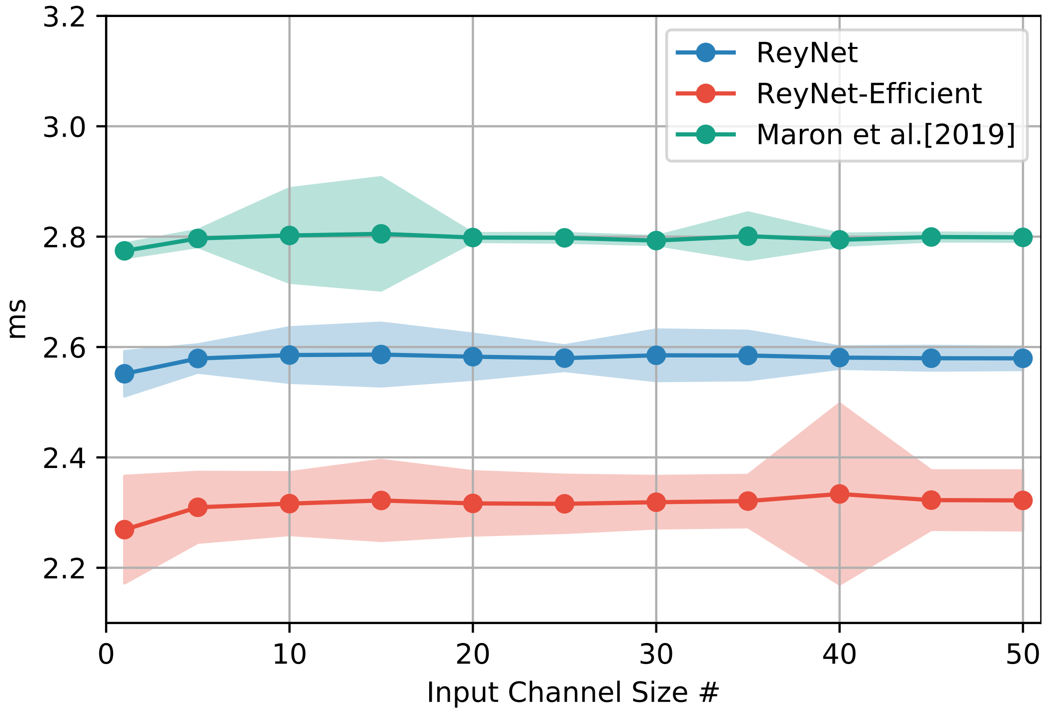


Figure 12: Runtime speed on different number of input features

Adam optimizer and set the learning rate as 1e-3 and weight decay as 1e-5. Batch size was 100. Note that the models of Maron et al. (2018) were reimplemented using PyTorch in reference to the author's implementation in Tensorflow.

B.1 Extrapolation

Figures 13 and 14 show the detailed version of only 4-red ReyNets in Figure 7. In Figure 7 our results seems an almost horizontal straight line, but these figures show that the MSE also slightly increases as n increases.

Appendix C. Graph Classification Tasks Details

C.1 Datasets

Bioinformatics Datasets. MUTAG consists of 188 compounds classified into two types based on their mutagenic effects on bacteria. The nodes representing each compound are labeled with 7 types. PTC is a data set on the carcinogenicity of rats by 344 compounds. Each node is given 19 different labels. PROTEINS is a data set of proteins. The nodes of each graph represent secondary structural elements, and the edges represent amino-acid sequences or neighborhoods in 3D space. Each node is given three types of labels. NCI1 and NCI109 are subsets of the compound data set screened for activity against non-small cell lung cancer cell lines and ovarian cancer cell lines, respectively.

Social Network Datasets. IMDB-BINARY and IMDB-MULTI are movie collaboration data sets. Each graph is an ego-network between performers. The ego network consists of the edges

Task	Symmetry			
n	3	5	10	20
FNN	1.730e-4	9.180e-4	1.454e-3	3.0583
IEGN (Maron et al., 2018)	6.600e-3	3.786e-3	9.294e-4	4.471e-3
ReyNet (ours)	2.147e-4	3.960e-4	1.408e-3	3.151e-3
4-red ReyNet (ours)	**8.544e-5**	**4.889e-5**	**7.529e-5**	**6.554e-5**
Task	Diagonal			
n	3	5	10	20
FNN	1.295e-4	2.655e-4	1.148e-4	1.081e-1
IEGN (Maron et al., 2018)	2.065e-3	2.266e-3	4.098e-3	4.743e-4
ReyNet (ours)	1.007e-4	2.472e-4	6.635e-4	1.112e-4
4-red ReyNet (ours)	**6.947e-5**	**1.932e-5**	**5.568e-5**	**3.566e-5**
Task	Power			
n	3	5	10	20
FNN	2.586	3.091e+1	5.756e+1	6.268e+2
IEGN (Maron et al., 2018)	4.036e-1	3.462e-1	7.062e-1	4.735e-1
ReyNet (ours)	3.798e-1	1.257	3.620	3.065
4-red ReyNet (ours)	**1.204e-1**	**1.330e-1**	**1.217e-1**	**1.165e-1**
Task	Trace			
n	3	5	10	20
FNN	**2.821e-4**	**1.135e-3**	9.529e-3	5.149e-2
IEGN (Maron et al., 2018)	1.241e-3	6.696e-3	3.663e-2	5.527e-2
ReyNet (ours)	1.884e-3	2.949e-3	4.275e-2	5.338e-2
4-red ReyNet (ours)	3.491e-4	1.914e-3	**6.758e-3**	**1.220e-2**

Table 5: Results of comparison to a baseline method

connecting the nodes of the performers who are co-starring and the nodes representing a certain performer. Each graph is annotated with the movie's genre, and the task is to classify this genre. COLLAB is a scientific collaboration data set. Each graph consists of the nodes of the researchers collaborating in three research fields. The task is to classify the graph into this research field.

C.2 Implementation.

The architecture of our ReyNets and baselines was adopted from Maron et al. (2018): after respective equivariant layers, the network consists of an invariant max-pooling (i.e. diagonal and off-diagonal) followed by three fully connected layers with hidden units of size (512, 256, #classes). With regards to the respective equivariant layer, the hidden features(channels) of IEGN and the feature-wise MLP of ReyNet were set to (16, 32, 256), which was determined in Maron et al. (2018). Simultaneously, we prepared two variants for set-wise MLP of ReyNet: (i) (16, 16, 16) and (ii) (128, 256, 512). For GIN and PPGN, we applied their published codes; for GIN, the hidden size

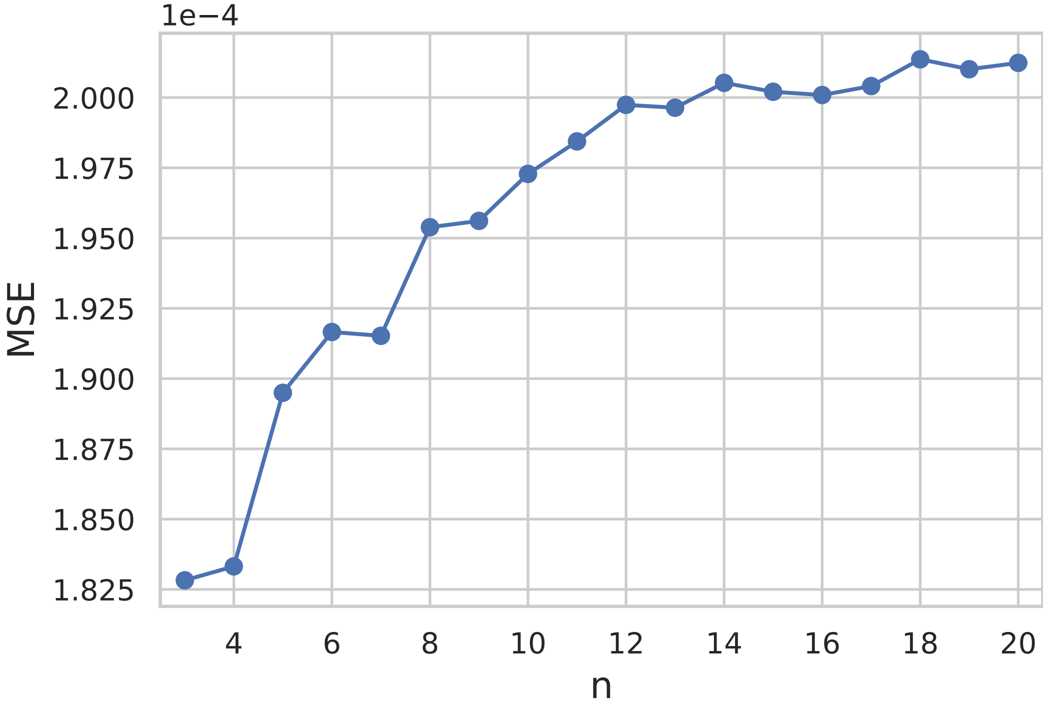


Figure 13: Extrapolation of symmetry regression

IEGN (Maron et al., 2018)	790K
PPGN (Maron et al., 2019a)	1.78M
GIN (Xu et al., 2019)	335K
ReyNet-(i)	666K
ReyNet-(ii)	832K

Table 6: Parameters for MUTAG experiment

was 64, and the number of layers was five; for PPGN, the number of blocks was three, and the size of the hidden units was (256, 256, 256). ReLU was the activation function for all the models. The number of input features is shown in Table C.3. Each model was fed the features and adjacency matrix as the input data, for example, the node of MUTAG has 7 features, and then the models are fed 8 features. Only for GIN, which is assumed to work on a message-passing scheme, we provided synthetic node features; the node degree as one-hot encodings. With the above implementation, the parameter size of each model for MUTAG experiment is represented in Table 6. The batch size was set to 5 according to Maron et al. (2018), except for GIN, which had a batch size of 32. We followed each published code for the optimizer's learning rate and the scheduler's decay rate in each data set, except when the model training was not convergent, in which case we tuned the learning rate to decrease. For ReyNet, we used 1e-4 as learning rate, except for MUTAG, for which the learning rate was 1e-3. We conducted the experiments using an NVIDIA Titan X or an NVIDIA V100.

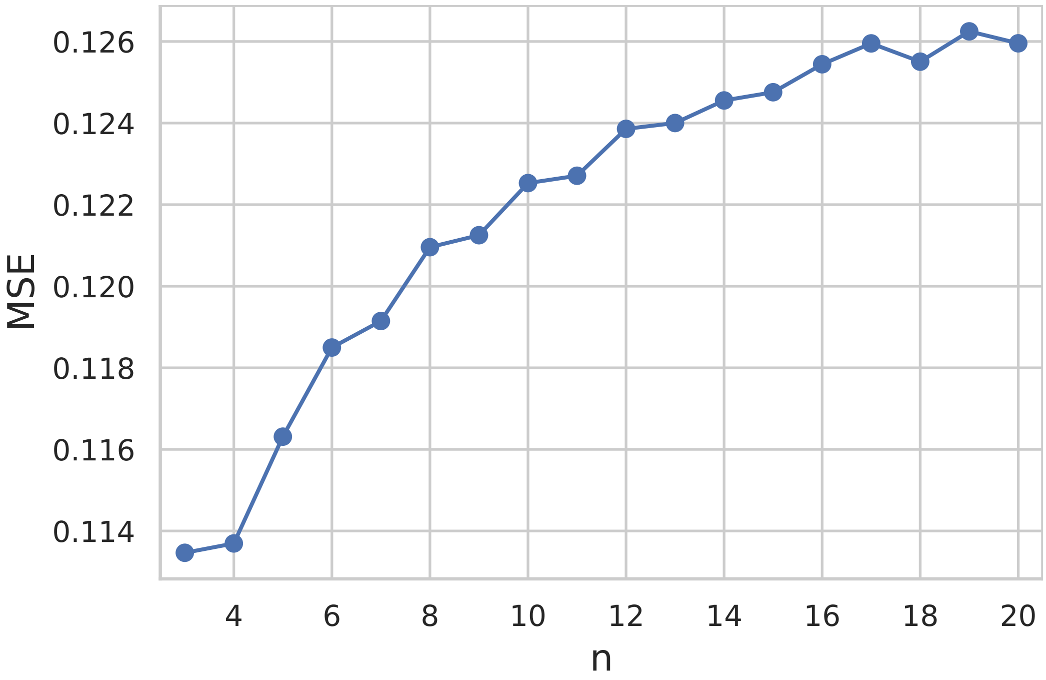


Figure 14: Extrapolation error of power regression

C.3 Results.

Table C.3 displays not just the results of our experiments, but also the results reported in the existing literature for the baseline methods; this includes DGCNN (Zhang et al., 2018), PSCN (Niepert et al., 2016), DGK (Yanardag and Vishwanathan, 2015), GNTK (Du et al., 2019), and GHC (Nguyen and Maehara, 2020).

	MUTAG	PTC	PROTEINS	NCI1
size	188	344	1113	4110
classes	2	2	2	2
avg node #	17.9	25.5	39.1	29.8
features #	7(+1)	22(+1)	3(+1)	37(+1)
Literature				
DGCNN	85.83 ± 1.7	58.59 ± 2.5	75.54 ± 0.9	74.44 ± 0.5
PSCN (k=10)	88.95 ± 4.4	62.29 ± 5.7	75 ± 2.5	76.34 ± 1.7
DGK	87.44 ± 2.7	60.08 ± 2.6	75.68 ± 0.5	80.31 ± 0.5
GHC-Tree	89.28±8.3	52.98 ±1.8	75.23± 1.7	48.8±1.0
GNTK	90.0 ± 8.5	67.9 ± 6.9	75.6 ± 4.2	84.2 ± 1.5
Our implementation				
IEGN	78.33 ± 10	55.59 ± 8.6	75.31 ± 5.4	76.06 ± 1.4
PPGN	88.33 ± 7.1	60.59 ± 7.9	73.96 ± 4.6	77.32 ± 2.2
GIN	85.83 ± 7.7	56.64 ± 7.0	72.56 ± 5.9	76.84 ± 2.3
ReyNet-(i)	89.44 ± 7.1	61.18 ± 5.2	75.41 ± 5.6	77.25 ± 2.0
ReyNet-(ii)	88.33 ± 8.9	59.41 ± 7.8	74.60 ± 4.2	77.79 ± 2.1

	NCI109	COLLAB	IMDB-B	IMDB-M
size	4127	5000	1000	1500
classes	2	3	2	3
avg node #	29.6	74.4	19.7	13
features #	38(+1)	0(+1)	0(+1)	0(+1)
Literature				
DGCNN	NA	73.76 ± 0.5	70.03 ± 0.9	47.83 ± 0.9
PSCN (k=10)	NA	72.6 ± 2.2	71 ± 2.3	45.23 ± 2.8
DGK	80.32 ± 0.3	73.09 ± 0.3	66.96 ± 0.6	44.55 ± 0.5
GHC-Tree	NA	75.23±1.7	72.1±2.6	48.6±4.4
GNTK	NA	83.6 ± 1.0	76.9 ± 3.6	52.8 ± 4.6
Our implementation				
IEGN	73.79 ± 2.9	78.12 ± 2.9	69.40 ± 6.1	47.20 ± 3.3
PPGN	78.98 ± 2.2	75.80 ± 2.0	70.60 ± 4.8	47.40 ± 3.3
GIN	73.51 ± 3.0	76.98 ± 2.1	70.60 ± 4.7	44.87 ± 3.9
ReyNet-(i)	75.90 ± 2.2	73.62 ± 1.6	70.10 ± 5.1	48.80 ± 2.9
ReyNet-(ii)	76.19 ± 2.1	74.31 ± 1.9	70.10 ± 4.6	46.73 ± 3.9

Table 7: Graph benchmark results with the results from literature.

References

Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

Eiichi Bannai and Etsuko Bannai. A survey on spherical designs and algebraic combinatorics on spheres. *European Journal of Combinatorics*, 30(6):1392–1425, 2009.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in Neural Information Processing Systems*, 2019.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, 29:3844–3852, 2016.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in Neural Information Processing Systems*, 32, 2019.

Devon Graham, Junhao Wang, and Siamak Ravanbakhsh. Equivariant entity-relationship networks. *arXiv preprint arXiv:1903.09033*, 2019.

Jason Hartford, Devon Graham, Kevin Leyton-Brown, and Siamak Ravanbakhsh. Deep models of interactions across sets. In *International Conference on Machine Learning*, pages 1909–1918. PMLR, 2018.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

David Hilbert. Über die theorie der algebraischen formen. *Mathematische Annalen*, 1890.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33:22118–22133, 2020.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32:7092–7101, 2019.

Kristian Kersting, Nils M Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels. *URL http://graphkernels. cs. tu-dortmund. de*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, pages 2156–2167, 2019a.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International Conference on Machine Learning*. PMLR, 2019b.

Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On learning sets of symmetric elements. In *International Conference on Machine Learning*. PMLR, 2020.

David Mumford, John Fogarty, and Frances Kirwan. *Geometric invariant theory*, volume 34. Springer Science & Business Media, 1994.

Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *International Conference on Learning Representations*, 2019.

Hoang Nguyen and Takanori Maehara. Graph homomorphism convolution. In *International Conference on Machine Learning*, pages 7306–7316. PMLR, 2020.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pages 2014–2023. PMLR, 2016.

Omri Puny, Matan Atzmon, Edward J Smith, Ishan Misra, Aditya Grover, Heli Ben-Hamu, and Yaron Lipman. Frame averaging for invariant and equivariant network design. In *International Conference on Learning Representations*, 2022.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through parameter-sharing. In *International Conference on Machine Learning*, pages 2892–2901. PMLR, 2017.

Akiyoshi Sannai, Masaaki Imaizumi, and Makoto Kawano. Improved generalization bounds of group invariant/equivariant deep networks via quotient feature spaces. In *Uncertainty in Artificial Intelligence*, pages 771–780. PMLR, 2021.

Nimrod Segol and Yaron Lipman. On universal equivariant set networks. In *International Conference on Learning Representations*, 2019.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.

Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, pages 1–68, 2021.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.