INTRINSIC COMPUTATIONAL COMPLEXITY OF EQUIVARIANT NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Equivariant neural networks have shown significant advantages in learning on data with intrinsic symmetries represented by groups. A major concern is on the high computational costs in the cases of large-scale groups, especially in the inference stage. This paper studies the required computational complexity of equivariant neural networks in inference for achieving a desired expressivity. We theoretically compare three classes of ReLU networks: (1) two-layer group-averaging networks (TGNs); (2) two-layer layer-wise equivariant networks (TENs); and (3) two-layer networks without any equivariant constraints (TNs), with a new notion *intrinsic computational complexity* for better characterizing computational costs. We prove that (1) TGNs/TENs have equal and full expressivities to represent any invariant function that can be learned by a TN, where the TGNs and TENs have equal intrinsic computational complexities; (2) a TGN/TEN requires at most double the intrinsic computational complexity of a TN; and (3) a TEN can achieve the inference speed coincident with its intrinsic computational complexity, while TGNs are strictly slower, which justifies the computational advantages of layer-wise equivariant architectures over group averaging. Our theory rules out the existence of equivariant networks with group-scale-independent computational costs, summarized in a new no-free-lunch theorem: when more equivariance is desired, more computation is required.

1 INTRODUCTION

Equivariant neural networks are designed to process data with intrinsic symmetries (Cohen & Welling, 2016) and have shown significant advantages in 3D point cloud (Li et al., 2018; Chen et al., 2021), chemistry (Faber et al., 2016; Eismann et al., 2021), astronomy (Ntampaka et al., 2016; Ravanbakhsh et al., 2016), and drug discovery (Hoogeboom et al., 2022). The groups in typical practical scenarios may be of prohibitively large scales (Romero & Cordonnier, 2020). Take the example in 2D point cloud processing, the group can have a scale up to 1000 (Hutchinson et al., 2021). However, experiments show that existing algorithms usually have considerable dependence on the group scale. This casts a shadow over the computational efficiency, and further development and application of equivariant neural networks. Unfortunately, few efforts are seen in the literature on the theoretical understanding of equivariant neural networks' computational complexity.

This paper studies the required computational complexity of equivariant neural networks in inference for achieving a desired expressivity. To our best knowledge, this is the first work on this topic.

To characterize the required computational complexity, we define a new *intrinsic computational complexity* as the computational complexity of a neural network that merges the inner product operations between the same-direction channel weight vectors and the input vector. In major cases of two-layer group-averaging networks defined below, a significant number of channels have same-direction weight vectors. These channels yield a considerable volume of repeating computation; see Example 4.5. Intrinsic computational complexity thus has remarkable advantages in characterizing the required computational complexity in practice.

We then theoretically compare the intrinsic computational complexity of three canonical classes of ReLU equivariant networks: (1) two-layer group-averaging networks (TGNs); (2) two-layer layer-wise equivariant networks (TENs); and (3) two-layer networks without any equivariant constraints

(TNs). To ensure a fair comparison, we fix the expressivity invariant in all cases, which is characterized as that they have the same invariant output function in the inference stage. Then, under the same output function, we compare their intrinsic computational complexities.

We first prove that any invariant output function of a TN can be represented by a TGN/TEN, where the TGN and the TEN are of equal intrinsic computational complexities. This indicates that both TENs and TGNs have *equal* and *full* expressivities to represent all invariant functions that can be learned by a two-layer neural network. The proof consists of three stages based on constructions. We first apply group averaging to an arbitrary TN to obtain a TGN. Since any invariant function is *invariant* after performing group averaging, the TN and the TGN share the same invariant output function. Then, we construct a TEN of the same intrinsic computational complexity with the TGN, which represent the same invariant output function as the TGN. Eventually, we prove that all TENs and TGNs of the same output function always have equal intrinsic computational complexities.

We then prove that the intrinsic computational complexity of a TGN/TEN would not be larger than double the complexity of the corresponding TN. The proof depends on a new notion *active hyperplane* defined as the boundary of one or more linear pieces of the piecewise linear output function in the ReLU network, where the piecewise linearity is guaranteed by the ReLU activations. When the TN and the TGN/TEN have the same invariant output function, we then show that the intrinsic computational complexity of the TN is larger than the cardinality of active hyperplanes, and is then larger than double the intrinsic computational complexity of the TGN/TEN. Additionally, we construct an exemplary network that meets the "double intrinsic computational complexity" upper bound exactly, which verifies the tightness of our bound.

Moreover, we theoretically compare the intrinsic computational complexities of TENs/TGNs with their empirical computational costs in practice. We prove that the intrinsic computational complexity of TENs rigorously coincides with their empirical computational cost under scenarios with common quotient representations. In contrast, the empirical computational complexity of TGNs is often strictly larger than their intrinsic computational complexity. These are guaranteed by the existence of phenomena that TGNs may fail to merge the inner product computations involving the channel weight vectors of the same direction, while TENs are sufficiently efficient in merging the inner products. This shows a remarkable gain in computing efficiency of adopting layer-wise equivariance over directly employing group averaging.

Our theory rules out the possibility of designing equivariant neural networks of group-scale-independent computational complexity. We show that the intrinsic computational complexity of a TEN relies on the group scale. Further, to represent an arbitrary invariant function, a TN requires at least half the intrinsic computational complexity of the corresponding TEN, which is dependent on the group scale. We summarize this as an equivariance no-free-lunch theorem: when more equivariant constraints are desired, more computation is required.

2 Related Work

Designing equivariant neural networks. Remarkable advances have been seen in designing equivariant network architectures for the intrinsic symmetries in data. Typical approaches for obtaining an equivariant neural network can be categorized into three steams: designing equivariant architectures, introducing equivariant constraints into the optimization, and applying group-averaging to backbone neural networks.

(1) Equivariant architectures. A typical designing paradigm is designing equivariant neural architectures based on classic neural networks. For example, Cohen & Welling (2016) design group convolutional neural networks in the group spaces as the equivariant variants of the convolutional neural networks (Shin et al., 2016), which is then extended to the homogeneous spaces by Cohen et al. (2019). Similar approaches are also seen in designing equivariant graph neural networks (Klicpera et al., 2020), equivariant transformers (Hutchinson et al., 2021; He et al., 2021), equivariant diffusion (Hoogeboom et al., 2022), equivariant point network (Chen et al., 2021), equivariant harmonic networks (Worrall et al., 2017), etc.

(2) Equivariant-constrained optimization. In this stream, equivariant constraints on the the parameters of the networks are introduced in the optimization to ensure the equivariance of the learned model, usually called steerable neural networks Cohen & Welling (2017). For examples,

methods surge to compute the solution space satisfying the equivariance constraints for different groups and network architectures, including the symmetry group S_N in steerable graph networks (Maron et al., 2018) and the rotation and Euclidean groups $\{C_N, E_2\}$ in steerable convolutional neural networks (Weiler et al., 2018; Weiler & Cesa, 2019). A major obstacle is the high computational costs to compute the equivariant basis. Finzi et al. (2021) address this issues with an algorithm with polynomial computational complexity on the sum of the number of discrete generators and the dimension of the group, which divides the problem into some independent subproblems and adopts Krylov method to compute nullspaces.

(3) Group-averaging on neural networks. Group-averaging apply to any backbone neural network for obtaining the corresponding equivariant neural network (Schulz-Mirbach, 1994). It may generally project an arbitrary hypothesis space of neural networks to the corresponding set of equivariant networks. Huang et al. (2022) adopt group averaging over the subgroup in each layer to obtain a equivariant graph network. Puny et al. (2022) propose a variant approach, frame averaging, that calculates the average over the subgroup $\mathcal{F}(x)$ with respect to the input x. When the frame function $\mathcal{F}$ is equivariant, the average over the subgroup $\mathcal{F}(x)$ is also ensured to be equivariant.

Theoretical analyses and benefits. Theoretical studies have shown that equivariant neural networks have significant advantages from the aspects of generalizability, convergence, and approximation. Sannai et al. (2021) prove improved generalization error bounds of equivariant models . Lawrence et al. (2022) prove that linear group convolutional neural networks trained by gradient descent for binary classification converge to solutions with low-rank Fourier matrix coefficients based on the results via implicit bias. Zaheer et al. (2017); Maron et al. (2019); Yarotsky (2021) prove the universal approximation ability of equivariant models by various approximation techniques, while Ravanbakhsh (2020) utilizes group averaging to obtain an equivariant approximator to prove the universal approximation ability. In addition, Elesedy & Zaidi (2021) prove that invariant/equivariant models have a smaller expected loss when the target function is invariant/equivariant.

3 PRELIMINARIES

This section introduces necessary preliminaries in group representation theory (Serre, 1977).

Define $[k] = \{1, 2, \ldots, k\}$. For a linear space V, define $\mathrm{GL}(V)$ as the group of all invertible linear isomorphisms on V. The notations G and H refer to a group and a subgroup, respectively. Define $[G : H] = |G|/|H|$, where G is a group, H is a subgroup, and $|\cdot|$ is the (sub-)group scale. The inner product between any $x, y \in \mathbb{R}^n$ is defined as $\langle x, y\rangle = \sum_{i=1}^n x_i y_i$. Moreover, we define the norm of a vector $x \in \mathbb{R}^n$ as $\|x\| = \sqrt{\langle x, x\rangle}$ for any $n \in \mathbb{N}$.

A group representation ψ on the linear space $\mathbb{R}^m$ is defined as a homomorphism from the group G to the isomorphism group $\mathrm{GL}(\mathbb{R}^m)$. For any $g \in G$, ψ_g is an invertible matrix; and for any $g, h \in G$, we have $\psi_{gh} = \psi_g \psi_h$. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is defined as G-equivariant under group representations $\psi : G \to \mathrm{GL}(\mathbb{R}^m)$ and $\rho : G \to \mathrm{GL}(\mathbb{R}^n)$, if $\psi_g \circ f = f \circ \rho_g$ for all $g \in G$. For the simplicity, for the cases $\psi = 1$ and $f \circ \rho_g = f$ for all $g \in G$, we call it invariant under the group representation ρ. Additionally, we define the transformed set $\rho_g X$ as $\{\rho_g x : x \in X\}$ for any set $X \in \mathbb{R}^n$.

A permutation representation ψ is defined as a group representation where ψ_g is a permutation matrix for all $g \in G$. If every nonzero entry in ψ_g is not constrained to be 1 only, ψ_g is called a generalized permutation matrix, and correspondingly, ψ is called a generalized permutation representation. Moreover, generalized permutation representations always commute with ReLU nonlinearity.

Quotient representation is defined as a special permutation representation. Given a subgroup H of group G, the quotient representation $\mathrm{Ind}_H^G 1$ permutates $[G : H]$ channels indexed by the cosets Hg. In particular, it acts on axes e_{Hg} as defined by $\mathrm{Ind}_H^G 1(g') \circ e_{Hg} = e_{Hgg'}$.

Every permutation matrix in $\mathbb{R}^{m\times m}$ functions as permutating the coordinates of the vector. Then, any permutation representation $P : G \to \mathbb{R}^{m\times m}$ can be extended to $P : G \to S_m$, where each P_g is a permutation on the set $[m]$. Moreover, P divides $[m]$ into some orbits, where P is transitive in each orbit. We denote the orbit partition of $[m]$ as $[m] = \bigcup_{i=1}^{\ell} C_i$. We also define $\mathrm{Stab}(k) = \{g \in G : P_g(k) = k\}$ as the stabilizer subgroup with respect to $k \in [m]$.

4 MAIN RESULTS

This section presents our main results. We first define the intrinsic computational complexity and the three canonical families of neural networks in our comparisons, TENs, TGNs, and TNs. Then, we prove that TENs and TGNs can represent all possible invariant output functions of TNs, where the TENs and the TGNs have equal intrinsic computational complexities. We further study the required intrinsic computational complexity of TENs and TGNs. Furthermore, we compare the intrinsic computational complexity of TENs and TGNs with their empirical computational costs. Lastly, we prove an equivariance no-free-lunch theorem, which rules out the existence of group-scale-independent equivariant networks.

4.1 THEORETICAL PROBLEM SETTINGS AND MEASUREMENTS

In this paper, we compare the required computational complexity for achieving a desired expressivity of the following three canonical families of equivariant neural networks.

Two-layer neural networks without any equivariant constraints (TNs). Two-layer networks are feedforward neural networks of a hidden layer with weight matrix $W^{(1)}$ and an output layer with weight matrix $W^{(2)}$. The output function is $F(x) = W^{(2)}\sigma(W^{(1)}x)$, where $x \in \mathbb{R}^n$ is the input and $\sigma(\cdot)$ is the nonlinear activation. In this paper, we consider the settings where all activations are ReLU functions. For simplicity, we denote $W^{(2)}$ and $W^{(1)}$ by $(\beta_1, \beta_2, \dots, \beta_m)$ and $(\alpha_1, \alpha_2, \dots, \alpha_m)^T$, respectively, where $\beta_i \in \mathbb{R}^d$ and $\alpha_i \in \mathbb{R}^n$ for all $i \in [m]$. In addition, we assume without any loss of generality that $\beta_i \neq 0$ and define its unit vector $\sigma_i = \beta_i / \|\beta_i\|$. Consequently, the output function can be formulated as follows,

$$F(x) = W^{(2)}\sigma(W^{(1)}x) = \sum_{i=1}^{m} \beta_i \sigma\Big(\langle \alpha_i, x \rangle\Big) = \sum_{i=1}^{m} \sigma_i \sigma\Big(\langle \|\beta_i\|\alpha_i, x \rangle\Big) = \sum_{i=1}^{m} \sigma_i \sigma\Big(\langle \tilde{\alpha}_i, x \rangle\Big).$$

The empirical computational complexity is defined as the number of channels m.

Two-layer layer-wise equivariant networks (TENs). Given two group representations $\psi : G \to \mathrm{GL}(\mathbb{R}^n)$ and $\rho : G \to \mathrm{GL}(\mathbb{R}^m)$, TENs require for any $g \in G$ that

$$W^{(2)} \circ \psi_g = W^{(2)},\ \psi_g \circ \sigma = \sigma \circ \psi_g,\ \text{and } \psi_g \circ W^{(1)} = W^{(1)} \circ \rho_g.$$

Then, the output function is invariant under the group representation ρ, *i.e.*, for all input $x \in \mathbb{R}^n$ and group element $g \in G$,

$$W^{(2)}\sigma(W^{(1)}\rho_g x) = W^{(2)}\sigma(\psi_g W^{(1)}x) = W^{(2)}\psi_g\sigma(W^{(1)}x) = W^{(2)}\sigma(W^{(1)}x).$$

Moreover, we define a group representation ψ as admitted, if it commutes with ReLU.

Two-layer group-averaging networks (TGNs). Group averaging acting on a function f is defined as $[\mathcal{Q}f] = \frac{1}{|G|}\sum_{g\in G} \psi_g^{-1} \circ f \circ \rho_g$, where ρ and ψ are two given group representations. Particularly, for the cases where $\psi = 1$, group averaging becomes $[\mathcal{Q}f](x) = \frac{1}{|G|}\sum_{g\in G} f(\rho_g x)$, which calculates the average over the outputs of all transformed inputs. Then, given a two-layer network with output function $F(x)$, the group averaging transfers the output function of TGNs into $\tilde{F}(x) = [\mathcal{Q}F](x) = \frac{1}{|G|}\sum_{g\in G} F(\rho_g x)$. Moreover, group averaging is a projection to the equivariant function space, *i.e.*, $\psi_g \circ [\mathcal{Q}f] = [\mathcal{Q}f] \circ \rho_g$ for all $g \in G$ and $\mathcal{Q}$ fixes all equivariant functions. Thus, the output function of TGNs are invariant under given group representation ρ.

Intrinsic computational complexity. Equivariant neural networks ordinarily have channels whose weight vectors are of the same directions, which is particularly normal in TGNs. This phenomenon causes a considerable amount of repeated computations, which can be merged in practice. This consequently contributes a large discrepancy between the theoretical computational complexity in terms of conventional measures and the computing costs in practice.

To address this issue, we define an *intrinsic computational complexity* which theoretically measures the computing costs after merging the computations in channels with the same-direction weight vectors. In this way, the discrepancy aforementioned is rectified.

Consider an output function $\sum_{i=1}^{m} \beta_i \sigma(\langle \alpha_i, x \rangle)$ of a two-layer network, where $\beta_i \neq 0 \in \mathbb{R}^d$ and $\alpha_i \neq 0 \in \mathbb{R}^n$. If there are two weight vectors α_i and α_j that are of the same direction, *i.e.*, $\alpha = \alpha_i/|\alpha_i| = \alpha_j/|\alpha_j|$, we have that

$$\beta_i \sigma(\langle \alpha_i, x \rangle) + \beta_j \sigma(\langle \alpha_j, x \rangle) = (\beta_i \|\alpha_i\| + \beta_j \|\alpha_j\|) \sigma(\langle \alpha, x \rangle),$$

which indicates that we can construct an "equivalent" channel to replace the original two channels with the same output, which merges the computation of the two channels. This merge is repeated until no channel pairs in the networks have weight vectors of the same direction. After the merges, the output function becomes $\sum_{i=1}^{\ell} \sigma_i(\langle \gamma_i, x \rangle)$, where $\sigma_i \in \mathbb{R}^d$ with $\|\sigma_i\| = 1$, $\gamma_i \neq 0 \in \mathbb{R}^n$, and $\gamma_i/\|\gamma_i\| \neq \gamma_j/\|\gamma_j\|$ for all $i \neq j$. In this way, we may significantly reduce the computational complexity, while this "merged" computational complexity is then defined as *intrinsic computational complexity* as follows.

Definition 4.1 (Intrinsic computational complexity)**.** For a two-layer network of output function $F(x) = \sum_{i=1}^{m} \beta_i \sigma(\langle \alpha_i, x \rangle)$ with $\beta_i \neq 0 \in \mathbb{R}^d$ and $\alpha_i \neq 0 \in \mathbb{R}^n$, we define its intrinsic computational complexity $\mathcal{C}$ as the scale of the direction set $\{\alpha_i/\|\alpha_i\| : i \in [m]\}$.

4.2 "Universal" Expressivity and Equal Intrinsic Computation Complexities of TENs and TGNs

In this subsection, we construct a TGN and a TEN to represent any invariant output function of a TN, where the TGN and the TEN have equal intrinsic computational complexities.

We first prove that TENs and TGNs have equal and "universal" expressivity, shown in the following theorem.

Theorem 4.2. *Any invariant output function of a TN can be represented by a TGN or a TEN, where the TGN and the TEN have equal intrinsic computational complexities.*

The detailed constructions are presented in Section 5.1.

Then, we prove that all TENs and TGNs of the same invariant output function have equal intrinsic computational complexities. It further justifies the essential position of intrinsic computational complexity in characterizing the required computational cost for equivariant networks to achieve a desired expressivity. Suppose the invariant output function of a TGN is as follows, $F(x) = \frac{1}{|G|} \sum_{i=1}^{m} \sigma_i \sum_{g \in G} \sigma(\langle \rho_g^T \alpha_i, x \rangle)$. Similarly from the definition of intrinsic computational complexity, the TGN's intrinsic computational complexity is exactly $\mathcal{C} = |\{\rho_g^T \alpha_i / \|\rho_g^T \alpha_i\| : i \in [m], g \in G\}|$.

We then have the following theorem to characterize the output functions of TENs.

Theorem 4.3. *A TEN's output function is as follows,*

$$F(x) = \sum_{i=1}^{\ell} \sigma_i \sum_{k \in C_i} \sigma\Big(\Big\langle \sum_{g \in G} \rho_g^T \tilde{\alpha}_{P_g(k)}, x \Big\rangle\Big) = \sum_{i=1}^{\ell} \sigma_i \sum_{g \in G} \sigma\Big(\Big\langle \rho_g^T \tilde{\gamma}_i, x \Big\rangle\Big),$$

where ℓ is the number of the orbits $\{C_i : i \in [\ell]\}$ induced by P defined in Lemma 5.2, $\sigma_i \in \mathbb{R}^d$ with $\|\sigma_i\| = 1$, $\tilde{\alpha}_k = \frac{|\beta_k|}{|G|} * \alpha_k$, and $\tilde{\gamma}_i = \frac{|Stab(k_i)|}{|G|} \sum_{g \in G} \rho_g^T \tilde{\alpha}_{P_g(k_i)}$ for some fixed k_i in each orbit C_i.

The proof will be sketched in Section 5.2 and full details will be given in Appendix A.5.

Theorem 4.3 implies that TEN's output function has the same form of the TGN, where each channel weight vector is $\sum_{g \in G} \rho_g^T \tilde{\alpha}_{P_g(k)}$. Additionally, we have that

$$\Big\{ \sum_{g \in G} \rho_g^T \tilde{\alpha}_{P_g(k)} / |\sum_{g \in G} \rho_g^T \tilde{\alpha}_{P_g(k)}| : k \in C_i \Big\} = \{\rho_g^T \tilde{\gamma}_i / |\rho_g^T \tilde{\gamma}_i| : g \in G\}.$$

Thus, the intrinsic computational complexity of the TEN is $\mathcal{C} = |\{\rho_g^T \tilde{\gamma}_i / \|\rho_g^T \tilde{\gamma}_i\| : i \in [\ell], g \in G\}|$, exactly equal with the TGNs'. Eventually, we have shown that TENs and TGNs have equal intrinsic computational complexity when they represent the same output function.

4.3 UPPER BOUNDS OF INTRINSIC COMPUTATIONAL COMPLEXITY OF TENS AND TGNS

In this subsection, we prove that the required intrinsic computational complexity of TENs and TGNs would not be larger than double the computational complexity of the corresponding TN, as shown in the following theorem.

Theorem 4.4. *Suppose a TN is of an invariant output function whose intrinsic computational complexity is m. Then, its output function is as follows, $F(x) = \sum_{i=1}^{\ell} \sigma_i \sum_{g\in G} \sigma\langle \rho_g^T \gamma_i, x\rangle$, where $\sigma_i \in \mathbb{R}^d$ with $\|\sigma_i\| = 1$, $\gamma_i \neq 0 \in \mathbb{R}^n$, and $|\{\rho_g^T\gamma_i/\|\rho_g^T\gamma_i\| : g \in G, i \in [\ell]\}| \leq 2m$.*

Thus, given any TN, we may apply group averaging to it to obtain a TGN with the same output function $\sum_{i=1}^{\ell} \sigma_i(\langle \gamma_i, x\rangle)$. The intrinsic computational complexity of this TGN is $|\{\rho_g^T\gamma_i/\|\rho_g^T\gamma_i\| : g \in G, i \in [\ell]\}| \leq 2m$. As shown in the previous subsection, all TENs and TGNs have the same intrinsic computational complexities if they represent the same output function. We then have that the required intrinsic computational complexity of TENs/TGNs is no larger than double the computational complexity of the corresponding TN.

We also construct the following example to show that the "double intrinsic computational complexity" upper bound is tight.

Example 4.5. *Denote $G = \langle g, h | g^2 = h^2 = e, gh = hg\rangle$. Suppose a group representation ρ is as follows*

$$\rho_g = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \text{ and } \rho_h = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}.$$

We define $F(x, y) = \sigma(x) + \sigma(-y) + \sigma(-x + y)$, which is the output function of a TN of intrinsic computational complexity 3. For a TEN/TGN of the output function F, we can prove that its intrinsic computational complexity is 6 and the TGN's empirical computational complexity is at least 8. We leave the proof in Appendix A.7.

4.4 WHICH OF TENS AND TGNS CAN REALIZE THEIR INTRINSIC COMPUTATIONAL COMPLEXITY?

In this subsection, we compare the intrinsic computational complexity and the empirical computational complexity of TENs and TGNs. Since intrinsic computational complexity is not larger than empirical computational complexity, a question then rises: when do they equal?

Given any output function $F(x) = \sum_{i=1}^{\ell} \sigma_i \sum_{g\in G} \sigma(\langle \rho_g^T \alpha_i, x\rangle)$, we have proven previously that all TENs and TGNs have equal intrinsic computational complexities if they represent this output function. By the constructed TEN of Theorem 4.2 in Section 5.1, we have the following corollary.

Corollary 4.6. *For any invariant output function of a TN, there must exist a TEN of the same output function whose intrinsic computational complexity and empirical computational complexity are equal.*

For TGNs, in contrast, the empirical computational complexity can be strictly larger than its intrinsic computational complexity. Take Example 4.5, the empirical computational complexity of the TGN is at least 8, while its intrinsic computational complexity is only 6.

For theoretical analyses, when the output function is $F(x) = \sum_{i=1}^{\ell} \sigma_i \sum_{g\in G} \sigma(\langle \rho_g^T \alpha_i, x\rangle)$, we find that the TGN must compute $\langle \rho_g^T \alpha_i, x\rangle$ for all $g \in G$, while TEN can compute the channels of same-direction weight vectors only once. As a result, the TEN decreases the empirical computational complexity $\sum_{i=1}^{\ell} |G|$ to $\sum_{i=1}^{\ell} |M_i|$. This finding shows a significant gain in computing efficiency of adopting layer-wise equivariance over directly employing group averaging.

4.5 EQUIVARIANCE NO-FREE-LUNCH THEOREM

This subsection rules out the fairy tale of designing equivariant neural networks with group-scale-independent computational complexity, beyond the restrictions of nonlinearity ReLU. This is suggested in the following equivariance no-free-lunch theorem.

Theorem 4.7 (Equivariance no-free-lunch theorem)**.** *The required computational complexity of any ReLU network to achieve a G-equivariant function has a positive correlation with the scale of the*

group G which is also the number of equivariant constraints. Thus, more equivariant constraints require more computation.

This theorem is proved based on a new notion, *active hyperplane*, as the boundary of one or more linear parts of the piecewise linear output function. Active hyperplane help discover the connection between computational complexity and group scale.

Intuitively, the computational complexity of a network has a positive correlation with the number of active hyperplanes, as also indirectly shown in Raghu et al. (2017). Moreover, this "active" property maintains during group action: for any active hyperplane M and group G, all the transformed hyperplanes $\rho_G M = \{\rho_g M : g \in G\}$ are still active. Thus, the number of active hyperplanes has a positive correlation with the scale of group G. Further, any $g \in G$ serves an equivariant constraint. Therefore, when more equivariance constraints are required, more active hyperplanes exist, and more computation is required.

5 Proof Sketch

In this section, we give the proof sketches for Theorems 4.2, 4.3, and 4.4.

5.1 Proof Sketch of Theorems 4.2

This theorem can be proved via two stages of constructions as shown below.

Stage 1. Construct the TGN of the same output function with any TN.

Suppose $F(x) = \sum_{i=1}^{m} \sigma_i \sigma(\langle \alpha_i, x \rangle)$ is an invariant output function of a TN. Applying group averaging to this TN, we obtain a TGN. This TGN inherits the invariant output function $[\mathcal{Q}F] = F$ of the TN, because invariant functions are *invariant* after performed group averaging. Then the output function can be formulated as below,

$$F(x) = [\mathcal{Q}F](x) = \frac{1}{|G|} \sum_{g \in G} \sum_{i=1}^{m} \sigma_i \sigma\Big(\Big\langle \rho_g^T \alpha_i, x \Big\rangle\Big) = \frac{1}{|G|} \sum_{i=1}^{m} \sigma_i \sum_{g \in G} \sigma\Big(\Big\langle \rho_g^T \alpha_i, x \Big\rangle\Big),$$

where $\sigma_i \in \mathbb{R}^d$ with $\|\sigma\| = 1$, $\alpha_i \neq 0$, and $\alpha_i \in \mathbb{R}^n$. Thus, the intrinsic computational complexity equals the scale of the direction set $\{\rho_g^T \alpha_i / \|\rho_g^T \alpha_i\| : g \in G, i \in [m]\}$.

We denote $M_i = \{\rho_g^T \alpha_i / \|\rho_g^T \alpha_i\| : g \in G\}$. Then the intrinsic computational complexity becomes $|\bigcup_{i=1}^{m} M_i|$. The following lemma helps further characterize the intrinsic computational complexity.

Lemma 5.1. *For all $i \neq j \in [m]$, we have $M_i = M_j$ or $M_i \cap M_j = \emptyset$.*

From calculating definition of intrinsic computational complexity, all same-direction channel weight vectors have been merged. We thus may assume for any $i \neq j$ that $M_i \cap M_j$. Therefore, the intrinsic computational complexity of the TGN satisfies $\mathcal{C} = \sum_{i=1}^{m} |M_i|$. This result indicates we can efficiently compute the intrinsic computational complexity without comparing different M_is.

Stage 2. Construct the TEN of the same output function with the TGN.

To this end, we should determine the weight matrices $W^{(1)}$ and $W^{(2)}$ to construct a network of the same output function, and the admitted group representation ψ in the hidden layer to ensure that it is a TEN such that $W^{(2)} \circ \psi_g = W^{(2)}$ and $\psi_g \circ W^{(1)} = W^{(1)} \circ \rho_g$ for all $g \in G$. We define that

$$W^{(1)} = \Big[\rho_{H_1 g_1}^T \alpha_1, \ldots, \rho_{H_1 g_{|M_1|}}^T \alpha_1, \ldots, \rho_{H_m g_1}^T \alpha_m, \ldots, \rho_{H_m g_{|M_m|}} \alpha_m\Big]^T,$$

and

$$W^{(2)} = \Big[\frac{\sigma_1}{|M_1|}, \ldots, \frac{\sigma_1}{|M_1|}, \ldots, \frac{\sigma_m}{|M_m|}, \ldots, \frac{\sigma_m}{|M_m|}\Big],$$

where $\{\rho_{H_1 g_j}^T \alpha_i : j \in [|M_i|]\}$ is the set of all distinct weight vectors $\widetilde{M_i} = \{\rho_g^T \alpha_i : g \in G\}$, and $H_i = \{g \in G : \rho_g^T \alpha_i = \alpha_i\}$ is the stabilizer subgroup with respected to vector α_i. Then, we

construct a two-layer neural network of equal empirical computational complexity and intrinsic computational complexity $\mathcal{C} = \sum_{i=1}^{m} |M_i|$, with the following output function,

$$W^{(2)}\sigma(W^{(1)}x) = \sum_{i=1}^{m} \frac{\sigma_i}{|M_i|} \sum_{j\in[|M_i|]} \sigma\Big(\Big\langle \rho_{H_i g_j}^T \alpha_i, x \Big\rangle\Big) = \sum_{i=1}^{m} \frac{\sigma_i}{|G|} \sum_{g\in G} \sigma\Big(\Big\langle \rho_g^T \alpha_i, x \Big\rangle\Big) = F(x),$$

which proves that the constructed two-layer network has the same output function with the TN.

Then, we define a group representation ψ as the direct sum of some quotient representations as below,

$$\psi = \bigoplus_{i=1}^{m} \mathrm{Ind}_{H_i}^{G} 1 : G \to \mathbb{R}^{\sum_{i=1}^{m} |M_i|}, \text{ where } \psi_g \rho_{H_i g_j}^T \alpha_i = \rho_{H_i(g_j g)}^T \alpha_i$$

which characterizes how the channels is permutated. We further have for all $g \in G$ that $W^{(2)} \circ \psi_g = W^{(2)}$ and $\psi_g \circ W^{(1)} = W^{(1)} \circ \rho_g$, which verify that the constructed two-layer network is a TEN. A detailed verification can be found in Appendix A.2.

5.2 Proof sketch of Theorem 4.3

We start by proving the following lemma that characterizes all admitted group representations that commute with ReLU nonlinearities.

Lemma 5.2. *A group representation ψ is admitted if and only if $\psi_g = diag(\lambda_g^1, ..., \lambda_g^m) P_g$ is a generalized permutation matrix, where P_g is a permutation matrix and λ_g^k is positive for all $k \in [m]$ and $g \in G$. Moreover, the P is also a permutation representation.*

In addition, the invariance relationship $W^{(2)} \circ \psi_g = W^{(2)}$ implies that $\beta_i(\psi_g)_{ij} = \beta_j$ for all $(\psi_g)_{ij} \neq 0$ and $g \in G$. To ensure a solution $W^{(2)}$ with no zero-weight vector β_i, the constraint $\beta_i(\psi_g)_{ij} = \beta_j$ requires all nonzero $(\psi_g)_{ij}$ to be equal for all $g \in G$. We summarize this constraint in the following lemma.

Lemma 5.3. *There exists a $W^{(2)}$ with no zero-weight vector β_i such that $W^{(2)} \circ \psi_g = W^{(2)}$ for all $g \in G$, if and only if the admitted group representation ψ satisfies all the following conditions:*

(1) For any $g \in G$, ψ_g is a generalized permutation matrix with nonnegative entries.

(2) The value of every nonzero entry only depends on its position in the matrix and is independent of the group element g. Thus, we can use the notation ψ_{ij} to denote the nonzero entry in the i-th row and j-th column of ψ_g if there exists g such that $(\psi_g)_{ij} \neq 0$.

From the definition of the orbit C_r induced by the permutation representation P in Lemma 5.2, we may derive that there exists $g \in G$ such that $(\psi_g)_{ij} \neq 0$, if and only if i and j are in the same orbit C_r. Therefore, we use the notation ψ_{ij} when i and j are in the same orbit for the briefy.

Now we are ready to prove Theorem 4.3 by proving following two equations,

$$F(x) = \sum_{i=1}^{\ell} \sigma_i \sum_{k\in C_i} \sigma(\langle \rho_g^T \tilde{\alpha}_{P_g(k)}, x\rangle) \text{ and } \sum_{i=1}^{\ell} \sigma_i \sum_{k\in C_i} \sigma(\langle \rho_g^T \tilde{\alpha}_{P_g(k)}, x\rangle) = \sum_{i=1}^{\ell} \sigma_i \sum_{g\in G} (\langle \rho_g^T \tilde{\gamma}_i, x\rangle).$$

Step 1. Since $W^{(2)}\psi_g = W^{(2)}$, we have that $\beta_i \psi_{ij} = \beta_j$ for all i and j in the same orbit. It implies that $\|\beta_i\|\psi_{ij} = \|\beta_j\|$ and $\beta_i/\|\beta_i\| = \beta_j/\|\beta_j\|$. Since all weight vectors β_j have the same direction $\beta_j/\|\beta_j\|$ for all $j \in C_i$, we may define the same direction as $\sigma_i \in \mathbb{R}^k$ without any loss of generality.

Then, we formulate $W^{(1)}$ as $\mathcal{Q}W^{(1)} = \frac{1}{|G|}\sum_{g\in G} \psi_g^{-1} W^{(1)} \rho_g$, since the equivariant $W^{(1)}$ is maintained in group averaging. Thus, the output function becomes as follows,

$$F(x) = \frac{1}{|G|} \sum_{k=1}^{m} \beta_k \sigma\Big(\Big\langle \sum_{g\in G} \frac{\rho_g^T \alpha_{P_g(k)}}{\psi_{P_g(k),k}}, x \Big\rangle\Big) = \sum_{i=1}^{\ell} \sigma_i \sum_{k\in C_i} \sigma\Big(\Big\langle \sum_{g\in G} \frac{\rho_g^T \|\beta_{P_g(k)}\| \alpha_{P_g(k)}}{|G|}, x \Big\rangle\Big).$$

Denote $\tilde{\alpha}_k$ as $\frac{|\beta_k|}{|G|} * \alpha_k$. Then we have completed the proof for the first equation.

Step 2. Denote γ_i as $\sum_{g\in G}\rho_g^T\tilde{\alpha}_{P_g(k_i)}$ for some fixed k_i in every orbit C_i. Since P is transitive in any orbit, then for any $k\in C_i$, there exists group element $h\in G$ such that $P_{h^{-1}}(k_i)=k$. Hence, we have $\rho_h^T\gamma_i=\sum_{g\in G}\rho_g^T\alpha_{P_g[P_{h^{-1}}(k_i)]}=\sum_{g\in G}\rho_g^T\alpha_{P_g(k)}$. This implies that

$$\left\{\rho_g^T\gamma_i : g\in G\right\}=\left\{\sum_{g\in G}\rho_g^T\tilde{\alpha}_{P_g(k)} : k\in C_i\right\} \text{ for all } i\in[\ell].$$

The number of group elements g that satisfies $P_g^{-1}(k_i)=k$ is exactly equal to $|G|/|\text{Stab}(k_i)|$, which is a constant independent of the choice of k. Then, we have that

$$\sum_{k\in C_i}\sigma\left(\left\langle\sum_{g\in G}\rho_g^T\tilde{\alpha}_{P_g(k)},x\right\rangle\right)=\frac{|\text{Stab}(k_i)|}{|G|}\sum_{g\in G}\sigma\left(\left\langle\rho_g^T\gamma_i,x\right\rangle\right)=\sum_{g\in G}\sigma\left(\left\langle\rho_g^T\frac{|\text{Stab}(k_i)|\gamma_i}{|G|},x\right\rangle\right).$$

Denote $\tilde{\gamma}_i$ as $\frac{|\text{Stab}(k_i)|}{|G|}*\gamma_i$. Then, the proof for the second equation is completed.

5.3 Proof sketch of Theorem 4.4

We first define two new notions, *feature gap* and *active hyperplane*.

We divide the whole input space $\mathbb{R}^n$ into some cells such that the output function is linear in each cell. The output function is $F(x)=\langle W,x\rangle$ in each cell, where W is defined as the *feature* of the cell. Then, we can define the feature function G as $G(x)=W$ to output the feature of the cell. To characterize the boundary of the cells, we define *feature gap* as follows.

Definition 5.4 (Feature gap)**.** Given a hyperplane M of dimension $n-1$ and one of its normal vectors $x\in\mathbb{R}^n$, the feature gap $\Delta_G(M,x)$ is defined as $\mathbb{E}_{y\in M}\Big[\lim_{z\to 0}G(y+z*x)-G(y-z*x)\Big]$, where $z\in\mathbb{R}$, and the expectation is taken over any continuous probability on the hyperplane M.

The feature gap characterizes how the feature changes when the point crosses through the hyperplane M. It characterizes the active hyperplanes as a whole. Moreover, the feature gap $\Delta_G(M,x)$ is nonzero, if and only if the hyperplane M is a boundary of some cells. Therefore, we define a hyperplane M as active if it is of nonzero feature gap $\Delta_G(M,x)$.

Then, we may present the formal definition of active hyperplane is as follows.

Definition 5.5 (Active hyperplane)**.** A hyperplane is defined as active if it is of nonzero feature gap.

From the definition, a TN of intrinsic computational complexity m has at most m active hyperplanes, because the TN only has m channels, and each channel causes only one active hyperplane (some active hyperplanes may be the same). In contrast, a TGN of intrinsic computational complexity $\mathcal{C}=\ell$ has at least $\ell/2$ active hyperplanes. If two channels introduce the same active hyperplane, they have anti-parallel vectors. Then, at most two channels introduce the same active hyperplane, and thus, a TGN of intrinsic computational complexity ℓ has at least $\ell/2$ active hyperplanes.

When a TN and a TGN represent equal output functions, they have equal numbers of active hyperplanes. Thus, the intrinsic computational complexity of the TN is larger than the number of active hyperplanes, while the number of active hyperplanes is larger than half the TGN's intrinsic computational complexity. It implies that the TGN's intrinsic computational complexity is at most double that of the TN. The proof is completed.

6 Conclusion

This paper studies the required computational complexities in inference of equivariant networks to achieve a desired expressivity. We compare three classes of ReLU networks: two-layer layer-wise equivariant networks (TENs), two-layer group-averaging networks (TGNs), and two-layer networks without any equivariant restrictions (TNs), based on a new notion intrinsic computational complexity. We prove that any invariant output function of TNs can be represented by a TEN or a TGN. The TEN and the TGN aforementioned have equal intrinsic computational complexities which are at most double intrinsic computational complexity of the TN. Then, we prove that TENs can achieve the inference speeds coincident with their intrinsic computational complexities, while TGNs have strictly larger computing costs in practice. We also prove an equivariance no-free-lunch theorem: when more equivariance is desired, more computation is required.

REFERENCES

Haiwei Chen, Shichen Liu, Weikai Chen, Hao Li, and Randall Hill. Equivariant point network for 3d point cloud analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*. PMLR, 2016.

Taco S. Cohen and Max Welling. Steerable CNNs. In *International Conference on Learning Representations (ICLR)*, 2017.

Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. *Neural Information Processing Systems (NeurIPS)*, 2019.

Stephan Eismann, Raphael JL Townshend, Nathaniel Thomas, Milind Jagota, Bowen Jing, and Ron O Dror. Hierarchical, rotation-equivariant neural networks to select structural models of protein complexes. *Proteins: Structure, Function, and Bioinformatics*, 2021.

Bryn Elesedy and Sheheryar Zaidi. Provably strict generalisation benefit for equivariant models. *International Conference on Machine Learning (ICML)*, 2021.

Felix A Faber, Alexander Lindmaa, O Anatole Von Lilienfeld, and Rickard Armiento. Machine learning energies of 2 million elpasolite (a b c 2 d 6) crystals. *Physical review letters*, 2016.

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International Conference on Machine Learning (ICML)*. PMLR, 2021.

Lingshen He, Yiming Dong, Yisen Wang, Dacheng Tao, and Zhouchen Lin. Gauge equivariant transformer. *Neural Information Processing Systems (NeurIPS)*, 2021.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning (ICML)*. PMLR, 2022.

Zhongyu Huang, Yingheng Wang, Chaozhuo Li, and Huiguang He. Going deeper into permutation-sensitive graph neural networks. *International Conference on Machine Learning (ICML)*, 2022.

Michael J Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. Lietransformer: Equivariant self-attention for lie groups. In *International Conference on Machine Learning (ICML)*. PMLR, 2021.

Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *International Conference on Learning Representations (ICLR)*, 2020.

Hannah Lawrence, Kristian Georgiev, Andrew Dienes, and Bobak Kiani. Implicit bias of linear equivariant networks. *International Conference on Machine Learning (ICML)*, 2022.

Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

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

Michelle Ntampaka, Hy Trac, Dougal J Sutherland, Sebastian Fromenteau, Barnabás Póczos, and Jeff Schneider. Dynamical mass measurements of contaminated galaxy clusters using machine learning. *The Astrophysical Journal*, 2016.

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

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *International Conference on Machine Learning (ICML)*. PMLR, 2017.

Siamak Ravanbakhsh. Universal equivariant multilayer perceptrons. In *International Conference on Machine Learning (ICML)*. PMLR, 2020.

Siamak Ravanbakhsh, Junier Oliva, Sebastian Fromenteau, Layne Price, Shirley Ho, Jeff Schneider, and Barnabás Póczos. Estimating cosmological parameters from the dark matter distribution. In *International Conference on Machine Learning (ICML)*. PMLR, 2016.

David W Romero and Jean-Baptiste Cordonnier. Group equivariant stand-alone self-attention for vision. *arXiv preprint arXiv:2010.00977*, 2020.

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 (UAI)*. PMLR, 2021.

Hanns Schulz-Mirbach. Constructing invariant features by averaging techniques. In *International Conference on Pattern Recognition (ICPR)*. IEEE, 1994.

Jean-Pierre Serre. *Linear representations of finite groups*. Springer, 1977.

Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 2016.

Maurice Weiler and Gabriele Cesa. General e (2)-equivariant steerable cnns. *Neural Information Processing Systems (NeurIPS)*, 2019.

Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

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

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

A APPENDIX

This appendix collects all detailed proofs.

A.1 PROOF OF LEMMA 5.1

Denote $M = \{\rho_g^T\alpha/\|\rho_g^T\alpha\| : g \in G\}$ and $N = \{\rho_g^T\beta/\|\rho_g^T\beta\| : g \in G\}$, where α and β are two nonzero vectors in $\mathbb{R}^n$. Then we prove that $M \cap N = \emptyset$ or $M = N$.

If $M \cap N \neq \emptyset$, there exist group elements $g_0, h_0 \in G$ such that $\rho_{g_0}^T\alpha/\|\rho_{g_0}^T\alpha\| = \rho_{h_0}^T\beta/\|\rho_{h_0}^T\beta\|$. Then, for any element $\rho_g^T\alpha/\|\rho_g^T\alpha\| \in M$, there exists $\rho_{h_0g_0^{-1}g}^T\beta/\|\rho_{h_0g_0^{-1}g}^T\beta\| \in N$ such that

$$\rho_{h_0g_0^{-1}g}^T\beta = \rho_g^T\rho_{g_0^{-1}}^T\rho_{h_0}^T\beta = \frac{\|\rho_{h_0}^T\beta\|}{\|\rho_{g_0}^T\alpha\|}\cdot\rho_g^T\alpha \text{ and } \frac{\rho_{h_0g_0^{-1}g}^T\beta}{\|\rho_{h_0g_0^{-1}g}^T\beta\|} = \frac{\rho_g^T\alpha}{\|\rho_g^T\alpha\|}.$$

It implies that $M \subset N$. Inversely, we can prove that $N \subset M$ in the same way. Hence, combining $M \subset N$ and $N \subset M$, we have $M = N$ and thus complete this proof.

Moreover, when denoting $\widetilde{M} = \{\rho_g^T\alpha : g \in G\}\}$, we can also prove that $|\widetilde{M}| = |M|$. If there exist $g, h \in G$ such that $\rho_g^T\alpha/\|\rho_g^T\alpha\| = \rho_h^T\alpha/\|\rho_h^T\alpha\|$, we have $\rho_g^T\alpha = c\rho_h^T\alpha$ with some positive number c since they have the same direction. Then, we have $\rho_{gh^{-1}}^T\alpha = c\alpha$ and

$$(\rho_{gh^{-1}}^T)^{|G|}\alpha = (\rho_{gh^{-1}}^T)^{|G|-1}c\alpha = (\rho_{gh^{-1}}^T)^{|G|-2}c^2\alpha = \cdots = c^{|G|}\alpha.$$

Meanwhile, since $(gh^{-1})^{|G|} = e \in G$, we have $(\rho_{gh^{-1}}^T)^{|G|}\alpha = \rho_e^T\alpha = \alpha$. It implies that $c^{|G|} = 1$ and thus $c = 1$. Finally, we have that $\rho_g\alpha = \rho_h\alpha \Leftrightarrow \frac{\rho_g\alpha}{\|\rho_g\alpha\|} = \frac{\rho_h\alpha}{\|\rho_g\alpha\|}$, which indicates that $|\widetilde{M}| = |M|$. The proof is completed.

A.2 PROOF OF THEOREM 4.2

We do not repeat the proof shown in Section 5.1 and only prove that the constructed two-layer network is a TEN, *i.e.*, we prove the equivariant constraints $\psi_g \circ W(1) = W^{(1)} \circ \rho_g$ and $W^{(2)} \circ \psi_g = W^{(2)}$. Given the group representation ψ as

$$\psi = \bigoplus_{i=1}^{m} \text{Ind}_{H_i}^G 1 : G \to \mathbb{R}^{\sum_{i=1}^m |M_i|}, \text{ where } \psi_g\rho_{H_ig_j}^T\alpha_i = \rho_{H_i(g_jg)}^T\alpha_i,$$

we have that

$$\psi_g \circ W^{(1)} = \psi_g \circ \begin{bmatrix} \alpha_1^T\rho_{H_1g_1} \\ \alpha_1^T\rho_{H_1g_2} \\ \vdots \\ \alpha_1^T\rho_{H_1g_{|M_1|}} \\ \vdots \\ \alpha_m^T\rho_{H_mg_1} \\ \alpha_m^T\rho_{H_mg_2} \\ \vdots \\ \alpha_m^T\rho_{H_mg_{|M_m|}} \end{bmatrix} = \begin{bmatrix} \alpha_1^T\rho_{H_1g_1g} \\ \alpha_1^T\rho_{H_1g_2g} \\ \vdots \\ \alpha_1^T\rho_{H_1g_{|M_1|}g} \\ \vdots \\ \alpha_m^T\rho_{H_mg_1g} \\ \alpha_m^T\rho_{H_mg_2g} \\ \vdots \\ \alpha_m^T\rho_{H_mg_{|M_m|}g} \end{bmatrix} = \begin{bmatrix} \alpha_1^T\rho_{H_1g_1} \\ \alpha_1^T\rho_{H_1g_2} \\ \vdots \\ \alpha_1^T\rho_{H_1g_{|M_1|}} \\ \vdots \\ \alpha_m^T\rho_{H_mg_1} \\ \alpha_m^T\rho_{H_mg_2} \\ \vdots \\ \alpha_m^T\rho_{H_mg_{|M_m|}} \end{bmatrix} \circ \rho_g = W^{(1)} \circ \rho_g,$$

and

$$\begin{aligned} W^{(2)} \circ \psi_g &= \left[(\tfrac{\sigma_1}{|M_1|} \cdots \tfrac{\sigma_1}{|M_1|})\text{Ind}_{H_1}^G 1 \ldots (\tfrac{\sigma_m}{|M_m|} \cdots \tfrac{\sigma_m}{|M_m|})\text{Ind}_{H_m}^G 1\right] \\ &= \left[\tfrac{\sigma_1}{|M_1|} \cdots \tfrac{\sigma_1}{|M_1|} \cdots \tfrac{\sigma_m}{|M_m|} \cdots \tfrac{\sigma_m}{|M_m|}\right] = W^{(2)}. \end{aligned}$$

The proof is completed.

A.3 PROOF OF LEMMA 5.2

For simplicity, we denote $e_i \in \mathbb{R}^n$ as the vector $(0, \ldots, 1, \ldots, 0)^T$, where the i-th entry is 1 and other entries are all 0. Since $\psi_g \circ \sigma = \sigma \circ \psi_g$, we have $\psi\sigma(x) = \sigma(\psi x)$ for all $x \in \mathbb{R}^n$.

If there are two nonzero entries $\psi_{ij} < \psi_{ik}$ in the same row of ψ_g, we can set $x = e_j - e_k$ and then we have $(\psi_g\sigma(x))_i = (\psi_g e_j)_i = \psi_{ij}$ while $(\sigma(\psi_g x))_i = \sigma(\psi_{ij} - \psi_{ik}) = 0$. That implies $\psi_{ij} = 0$, which contradicts. Thus, each row has at most one nonzero entry. Hence, there are at most n nonzero entries of ψ_g.

Since ψ_g is invertible, each row and each column of ψ_g have at least one nonzero entry. It implies that there are at least n nonzero entries of ψ_g. Thus, there are exactly n nonzero entries of ψ_g and ψ_g is a generalized permutation matrix $\text{diag}(\lambda_g^1, \ldots, \lambda_g^m)P_g$.

Moreover, when $\psi_{ij} \neq 0$, we can set $x = e_j$. Then $\sigma(\psi x) = \psi\sigma(x)$ implies that $\sigma(\psi_{ij}) = \psi_{ij}$. From the definition of ReLU, we have that $\psi_{ij} > 0$. The proof is completed.

A.4 PROOF OF LEMMA 5.3

The equivariant constraint $W^{(2)} \circ \psi_g = W^{(2)}$ implies that $\beta_i(\psi_g)_{ij} = \beta_j$ for all $(\psi_g)_{ij} \neq 0$. For any i and j in the same orbit, there exists group element $g \in G$ such that $(\psi_g)_{ij} \neq 0$. Thus, if there is some β_i equal to 0, then $\beta_j = 0$ for all js in the same orbit with i. Therefore, we can remove all there channels and get a new equivariant network of the same output function and a smaller computational complexity. In this sense, we can assume without loss of generality that $\beta_i \neq 0$ for all $i \in [m]$.

Besides, the constraint $\beta_i(\psi_g)_{ij} = \beta_j$ implies that $\beta_i/\|\beta_i\| = \beta_j/\|\beta_j\|$ and $(\psi_g)_{ij} = \frac{\|\beta_j\|}{\|\beta_i\|}$ for all i, j in the same orbit and $(\psi_g)_{ij} \neq 0$. Hence, since the value of the nonzero entry $(\psi_g)_{ij}$ is independent of the group element g, we can denote it by ψ_{ij} for simplicity.

The proof is completed.

A.5 PROOF OF THEOREM 4.3

Since we can extend the group representation P to act on the coordinate set $[m]$, the nonzero entry of ψ_g can be characterized as $\psi_{i,P_g^{-1}(i)}$ for all $i \in [m]$. Then the output function of TENs becomes

$$
\begin{aligned}
&F(x) = W_2\sigma(W_1 x) = W_2\sigma(\mathcal{Q}W_1 x) \\
=&\frac{1}{|G|}\left[\beta_1 \ldots \beta_m\right]\sigma\Bigg(\sum_{g\in G}\psi_g^{-1}\begin{bmatrix}\alpha_1^T\\ \vdots \\ \alpha_m^T\end{bmatrix}\rho_g x\Bigg) \\
=&\frac{1}{|G|}\left[\beta_1 \ldots \beta_m\right]\sigma\Bigg(\sum_{g\in G}\begin{bmatrix}\frac{\alpha_{P_g(1)}^T}{\psi_{P_g(1),1}}\\ \vdots \\ \frac{\alpha_{P_g(m)}^T}{\psi_{P_g(m),m}}\end{bmatrix}\rho_g x\Bigg) \\
=&\frac{1}{|G|}\sum_{k=1}^m \beta_k\sigma\sum_{g\in G}\Big\langle\frac{\alpha_{P_g(k)}}{\psi_{P_g(k),k}}, \rho_g x\Big\rangle \\
=&\frac{1}{|G|}\sum_{k=1}^m \frac{\beta_k}{\|\beta_k\|}\sigma\sum_{g\in G}\Big\langle\frac{\|\beta_k\|\alpha_{P_g(k)}}{\psi_{P_g(k),k}}, \rho_g x\Big\rangle \\
=&\frac{1}{|G|}\sum_{k=1}^m \frac{\beta_k}{\|\beta_k\|}\sigma\sum_{g\in G}\Big\langle\|\beta_{P_g(k)}\|\alpha_{P_g(k)}, \rho_g x\Big\rangle.
\end{aligned}
$$

Moreover, since β_k has the same direction $\frac{\beta_k}{\|\beta_k\|}$ for all ks in the same orbit, we have

$$\begin{aligned}
&\frac{1}{|G|}\sum_{k=1}^{m}\frac{\beta_k}{\|\beta_k\|}\sigma\sum_{g\in G}\Big\langle\|\beta_{P_g(k)}\|\alpha_{P_g(k)},\rho_g x\Big\rangle\\
=&\frac{1}{|G|}\sum_{i=1}^{\ell}\sigma_i\sum_{k\in C_i}\sigma\sum_{g\in G}\Big\langle\|\beta_{P_g(k)}\|\alpha_{P_g(k)},\rho_g x\Big\rangle\\
=&\sum_{i=1}^{\ell}\sigma_i\sum_{k\in C_i}\sigma\Big\langle\sum_{g\in G}\rho_g^T\frac{\|\beta_{P_g(k)}\|\alpha_{P_g(k)}}{|G|},x\Big\rangle\\
=&\sum_{i=1}^{\ell}\sigma_i\sum_{k\in C_i}\sigma\Big\langle\sum_{g\in G}\rho_g^T\tilde{\alpha}_{P_g(k)},x\Big\rangle,
\end{aligned}$$

where $\tilde{\alpha}_k=\frac{\|\beta_k\|}{|G|}*\alpha_k$ for all $k\in[m]$.

Moreover, we fix a k_i in each orbit C_i, and denote the weight vector $\sum_{g\in G}\rho_g^T\tilde{\alpha}_{P_g(k_i)}$ by γ_i. Then we have

$$\rho_h^T\gamma_i=\rho_h^T\sum_{g\in G}\rho_g^T\tilde{\alpha}_{P_g(k_i)}=\sum_{g\in G}\rho_{gh}^T\tilde{\alpha}_{P_{gh}P_h^{-1}(k_i)}=\sum_{g\in G}\rho_g^T\tilde{\alpha}_{P_g[P_h^{-1}(k_i)]}.$$

Denote $\text{Stab}(k_i)=\{g\in G:P_g(k_i)=k_i\}$ as the stabilizer subgroup with respect to k_i. Then the set $H_i(k)=\{g\in G:P_g^{-1}(k_i)=k\}$ is of scale $|G|/|\text{Stab}(k_i)|$ for all $k\in C_i$. Hence, we have

$$\sum_{g\in G}\sigma\langle\rho_g^T\gamma_i,x\rangle=\sum_{k\in C_i}\sum_{g\in H_i(k)}\sigma\langle\rho_g^T\gamma_i,x\rangle=\sum_{k\in C_i}\frac{|G|}{|\text{Stab}(k_i)|}\sigma\Big\langle\sum_{g\in G}\rho_g^T\tilde{\alpha}_{P_g(k)},x\Big\rangle.$$

Finally, we have

$$\sum_{i=1}^{\ell}\sigma_i\sum_{k\in C_i}\sigma\Big\langle\sum_{g\in G}\rho_g^T\tilde{\alpha}_{P_g(k)},x\Big\rangle=\sum_{i=1}^{\ell}\sigma_i\sum_{g\in G}\sigma\Big\langle\rho_g^T\tilde{\gamma}_i,x\Big\rangle,$$

where $\tilde{\gamma}_i$ is $\frac{|\text{Stab}(k_i)|}{|G|}*\gamma_i$ for all $i\in[\ell]$. This proof is completed.

A.6 Proof of Theorem 4.4

We first verify that the feature gap $\Delta_G(\cdot,\cdot)$ is well-defined for two-layer networks. Besides, it is worth noting that we only define the feature $G(x)$ for the inner point x of each (open) cell. The feature $G(x)$ for x in the boundary can be any arbitrary value, which does not change our results.

Lemma A.1. *For the function $F(x)=\beta\sigma\langle\alpha,x\rangle$ with $\beta\in\mathbb{R}^d$ and $\alpha\in\mathbb{R}^n$, we have*

$$\Delta_G(M,x)=\begin{cases}\beta\alpha^T & \text{if } \alpha \text{ is a normal vector of } M \text{ and } x/\|x\|=\alpha/\|\alpha\|,\\ -\beta\alpha^T & \text{if } \alpha \text{ is a normal vector of } M \text{ and } x/\|x\|=-\alpha/\|\alpha\|,\\ 0 & \text{otherwise.}\end{cases}$$

Proof. When α is a normal vector of M, then we have

$$\Delta_G(M,\alpha)=\mathbb{E}_{y\in M}[\lim_{z\to 0}G(y+z*\alpha)-G(y-z*\alpha)]=\mathbb{E}_{y\in M}[\beta\alpha^T-0]=\beta\alpha^T,$$

and

$$\Delta_G(M,-\alpha)=\mathbb{E}_{y\in M}[\lim_{z\to 0}G(y-z*\alpha)-G(y+z*\alpha)]=\mathbb{E}_{y\in M}[0-\beta\alpha^T]=-\beta\alpha^T.$$

When α is a normal vector of $M_0\neq M$, the intersection of two hyperplanes $M\cap M_0$ is of dimension $(n-2)$. Since M is of dimension $n-1$, we have $\mathbb{E}_{y\in M}[\cdot]=\mathbb{E}_{y\in M\setminus(M\cap M_0)}[\cdot]$. When

$y \in M \backslash (M \cap M_0)$, we have $\lim_{z\to 0} G(y+z*x)-G(y-z*x) = 0$ since $G(y+z*x) = G(y-z*x)$ for sufficiently small $z \in \mathbb{R}$. Then we have

$$\begin{aligned}\Delta_G(M, x) &= \mathbb{E}_{y\in M}\big[\lim_{z\to 0} G(y + z * x) - G(y - z * x)\big] \\ &= \mathbb{E}_{y\in M\backslash(M\cap M_0)}\big[\lim_{z\to 0} G(y + z * x) - G(y - z * x)\big] = 0.\end{aligned}$$

The proof is completed. □

Lemma A.2. *The feature function $G(F)$ and the feature gap $\Delta_{G(F)}$ are additive, i.e.,*

$$G\Big(\sum_{i=1}^m F_i\Big) = \sum_{i=1}^m G(F_i) \ \textit{and} \ \Delta_{G(\sum_{i=1}^m F_i)} = \sum_{i=1}^m \Delta_{G(F_i)}.$$

Proof. From the definition of the feature function G, we have

$$\sum_{i=1}^m F_i(x) = \Big\langle \Big[G\Big(\sum_{i=1}^m F_i\Big)\Big](x), x\Big\rangle \text{ and } \sum_{i=1}^m F_i(x) = \sum_{i=1}^m \big\langle [G(F_i)](x), x\big\rangle = \Big\langle \sum_{i=1}^m [G(F_i)](x), x\Big\rangle.$$

Thus, we have that $G\Big(\sum_{i=1}^m F_i\Big) = \sum_{i=1}^m G(F_i)$.

Hence, for the feature gap Δ_G, we have

$$\begin{aligned}\Delta_{G(\sum_{i=1}^m F_i)}(M, x) &= \mathbb{E}_{y\in M}\Big[\lim_{z\to 0}\Big[G\Big(\sum_{i=1}^m F_i\Big)\Big](y + z * x) - \Big[G\Big(\sum_{i=1}^m F_i\Big)\Big](y - z * x)\Big] \\ &= \sum_{i=1}^m \mathbb{E}_{y\in M}\Big[\lim_{z\to 0}[G(F_i)](y + z * x) - [G(F_i)](y - z * x)\Big] = \sum_{i=1}^m \Delta_{G(F_i)}(M, x).\end{aligned}$$

The proof is completed. □

Moreover, for the function $F(x) = \sum_{i=1}^m \beta_i \sigma(\langle \alpha_i, x\rangle)$, a hyperplane M is of nonzero feature gap $\Delta_G(M, x)$ if and only one of the weight vectors α_i is its normal vector.

For the output function $F_1(x) = \sum_{i=1}^m \beta_i \sigma(\langle \alpha_i, x\rangle)$ of a TN of intrinsic computational complexity $\mathcal{C}_1 = m$, it has at most $\mathcal{C}_1$ active hyperplanes.

Besides, we consider the function $F_2(x) = \sum_{i=1}^\ell \beta_i \sum_{g\in G} \sigma(\langle \rho_g^T \alpha_i, x\rangle)$ of a TGN of intrinsic computational complexity $\mathcal{C}_2$. From the definition of intrinsic computational complexity, we can assume no two weight vectors $\rho_g^T \alpha_i$ and $\rho_h^T \alpha_j$ are of the same direction for all distinct $i \neq j$. Since the normal vectors of a hyperplane M have at most two different directions, the function F_2 with $\mathcal{C}_2$ weight vectors of different directions has at least $\mathcal{C}_2/2$ active hyperplanes.

We denote the number of active hyperplanes by C when $F_1 = F_2$. Then, combining the above two results, we have

$$\mathcal{C}_1 \geq C \geq \mathcal{C}_2/2.$$

It implies that the intrinsic computational complexity of the TGN is no larger than double that of the TN. The proof is completed.

A.7 Proof for Example 4.5

We prove that the intrinsic computational complexity of the corresponding TEN/TGN is $\mathcal{C} = 6$ and the empirical computational complexity of the TGN is at least 8.

Given the function $F(x, y) = \sigma(x) + \sigma(-y) + \sigma(-x + y)$, we have

$$[\mathcal{Q}F](x, y) = (\sigma(x) + \sigma(-x) + \sigma(y) + \sigma(-y) + \sigma(-x + y) + \sigma(x - y))/2.$$

The constructed TGN has an intrinsic computational complexity $\mathcal{C} = 6$. Thus all TENs/TGNs of the same output function have equal intrinsic computational complexities $\mathcal{C} = 6$. Besides, since the empirical computation complexity is a multiple of the group's scale 4, thus it is no smaller than 8.