Efficient Equivariant Network

Anonymous Author(s)
Affiliation
Address
email

Abstract

Convolutional neural networks (CNNs) have dominated the field of Computer Vision and achieved great success due to their built-in translation equivariance. Group equivariant CNNs (G-CNNs) that incorporate more equivariance can significantly improve the performance of conventional CNNs. However, G-CNNs are faced with two major challenges: *spatial-agnostic problem* and *expensive computational cost*. In this work, we propose a general framework of equivariant models, which includes G-CNNs and equivariant self-attention layers as special cases. Under this framework, we explicitly decompose the feature aggregation operation into a kernel generator and an encoder, and decouple the spatial and extra geometric dimensions in the computation. Therefore, our filters are essentially dynamic rather than being spatial-agnostic. We further show that our *E*quivariant model is parameter *E*fficient and computation *E*fficient by complexity analysis, and also data *E*fficient by experiments, so we call our model E^4-Net. Extensive experiments verify that our model can significantly improve previous works with smaller model size. Especially, under the setting of training on $1/5$ data of CIFAR10, our model improves G-CNNs by $5\%+$ accuracy, while using only 56% parameters and 68% FLOPs.

1 Introduction

In the past few years, convolutional neural networks (CNNs) have been widely used and achieved superior results on multiple vision tasks, such as image classification [16, 29, 27, 10], semantic segmentation [1], and object detection [22]. A compelling explanation of the good performance of CNNs is that their built-in parameter sharing scheme brings in translation equivariance: shifting an image and then feeding it through a CNN layer is the same as feeding the original image and then shifting the resulted feature maps. In other words, the translation symmetry is preserved by each layer. Motivated by this, Cohen and Welling [5] proposed Group Equivariant CNNs (G-CNNs), showing how convolutional networks can be generalized to exploit larger groups of symmetries. Following G-CNNs, researchers have designed new neural networks that are equivariant to other transformations like rotations [5, 33, 11, 26] and scales [35, 28]. However, G-CNNs still have two main drawbacks: 1) In the implementation, G-CNNs would introduce extra dimensions to encode new transformations, such as rotations and scales, thus have a very high computational cost. 2) As G-CNNs achieve group equivariance by sharing kernels, they lack the ability to adapt kernels to diverse feature patterns with respect to different spatial positions, namely, the spatial-agnostic problem [36, 20, 37, 38].

Some previous works focus on solving these two problems. Cheng et al. [2] proposed to decompose the convolutional filters over joint steerable bases to reduce model size. However, it is essentially G-CNNs which still have the inherent spatial-agnostic problem. To incorporate dynamic filters, one solution is introducing attention mechanism into each convolution layer in G-CNNs without disturbing inherent equivariance [25, 23]. The cost is that they introduce extra parameters and increase the complexity of space and time. Another solution is to replace group convolution layers with stand-

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

alone self-attention layers by designing a specific position embedding to ensure equivariance [24, 12]. However, the self-attention mechanism suffers from quadratic memory and time complexity, because it has to compute the attention score at each pair of inputs.

Actually, Cohen et al. [7] revealed that an equivariant linear layer is essentially a convolution-like operation. Inspired by this, we further discover that a general feature-extraction layer, either linear or non-linear, being equivariant is equivalent to that the feature aggregation mechanism between each pair of inputs only depends on the relative positions of these two inputs. Based on this observation, we propose a generalized framework of equivariant models, which includes G-CNNs and equivariant attention networks as special cases. Under this generalized framework, we design a new equivariant layer to conquer the aforementioned difficulties. Firstly, to avoid quadratic computational complexity, the feature aggregation operator is explicitly decomposed into a kernel generator and an encoder which takes one single feature as the input. Since our kernels are calculated based on input features, they are essentially dynamic rather than being spatial-agnostic. In addition, we decouple the feature aggregation mechanism across spatial and extra geometric dimensions to reduce the inter-channel redundancy in convolution filters [2] and further accelerate computation. Extensive experiments show that our method can process data very efficiently and perform significantly better than previous works using lower computational costs. As our method is parameter *E*fficient, computational *E*fficient, data *E*fficient and *E*quivariant, we name our new layer as E^4-layer.

We summarize our main contributions as follows:

- We propose a generalized framework of equivariant models, which includes G-CNNs and attention-based equivariant models as special cases.
- Under the generalized framework, we explicitly decompose the feature aggregation operator into a kernel generator and an encoder, and further decouple the spatial and extra geometric dimensions to reduce computation.
- Extensive experiments verify that our method is also data efficient and performs competitively with lower computational cost.

2 Related Work

2.1 G-CNNs (Linear)

Conventional CNNs [19] are naturally translation equivariant. In order to exploit more symmetries, Cohen and Welling [5] proposed G-CNNs, succeeding in incorporating rotation and reflection equivariance into conventional CNNs. Some follow-up works focused on exploiting larger rotation groups [11, 31, 26] and deriving more general G-CNNs, namely Steerable CNNs [8, 31]. Beside those methods on 2D images, there are also some works applying the idea of G-CNNs to rotations over the sphere [6, 4] or 3D space [34, 32].

In general, researchers [14, 7] pointed out that an equivariant linear mapping can always be written as a convolution-like integral, i.e., G-CNNs in practice. However, their theory is still limited to linear mappings. In this work, we further extend their theory to a more general case, either linear or non-linear. Besides, G-CNNs are spatial-agnostic and computationally costly due to extra dimensions for encoding more symmetries. Cheng et al. [2] reduced the computation by decoupling different geometric transformations, but the spatial-agnostic problem remained. By contrast, our work addresses both problems simultaneously.

2.2 Equivariant Attention Networks (Non-linear)

Different from conventional CNNs, dynamic filters adjust or predict filter values based on input features [36, 20, 37, 38]. Particularly, attention mechanism [30] can be viewed as a special kind of dynamic filters that has been widely used. Romero et al. [25, 23] directly incorporated attention into G-CNNs and obtained non-linear equivariant models. However, compared with G-CNNs, these methods introduce extra parameters and operations, resulting in an even heavier computational burden. Also, some works [24, 12] proposed group equivariant stand-alone self-attention [21]. Fuchs et al. [9] incorporated transformers into 3D equivariant networks and proposed SE(3)-Transformers. However, since their filters are essentially calculated based on a pair of inputs, the computational complexity is

quadratic. By contrast, our dynamic filters only depend on one single input feature, which reduces the computation significantly.

3 A Unified Framework of Group Equivariant Models

In this section, we first briefly review two representative group equivariant models: the linear model G-CNNs [5], and the non-linear model equivariant self-attention [24, 12]. Then, we propose a general framework of equivariant models based on the inner relationship among these specific models.

3.1 Equivariance

Equivariance indicates that the outputs of a mapping transform in a predictable way with the transformation of the inputs. Formally, a group equivariant map Ψ satisfies that

$$\forall u \in \mathcal{G}, \quad \Psi\left[\mathcal{T}_u[\mathbf{f}]\right] = \mathcal{T}'_u[\Psi[\mathbf{f}]], \tag{1}$$

where $\mathcal{G}$ is a transformation group, $\mathbf{f}$ is an input feature map, and $\mathcal{T}_u$ and $\mathcal{T}'_u$ are group actions, indicating how the transformation u acts on the input and output features, respectively. Besides, since we hope that two transformations $u, v \in \mathcal{G}$ acting on the feature maps successively is equivalent to the composition of transformations $uv \in \mathcal{G}$ acting on the feature maps directly, we require that $\mathcal{T}_u\mathcal{T}_v = \mathcal{T}_{uv}$, where uv is the group product of u and v. The same is the case with $\mathcal{T}'_u$.

Now we examine the specific form of the transformation group $\mathcal{G}$. In this work, we focus on the analysis of 2D images defined on $\mathbb{R}^2$. Consequently, we are most interested in the groups of the form $\mathcal{G} = \mathbb{R}^2 \rtimes \mathcal{A}$, resulting from the semi-product ($\rtimes$) between the translation group $\mathbb{R}^2$ and a group $\mathcal{A}$ acts on $\mathbb{R}^2$, e.g., rotations, scalings and mirrorings. This family of groups is referred to as affine groups and their group product rule is:

$$uv = (x_u, a_u)(x_v, a_v) = (x_u + a_u x_v, a_u a_v), \tag{2}$$

where $u = (x_u, a_u)$ and $v = (x_v, a_v)$, in which $x_u, x_v \in \mathbb{R}^2$ and $a_u, a_v \in \mathcal{A}$. For ease of implementation, following [5], we take $\mathcal{A}$ as the cyclic group $\mathcal{C}4$ or the dihedral group $\mathcal{D}4$, then $\mathcal{G}$ becomes $p4$ or $p4m$. As for the group action, we employ the most common regular group action in this work, i.e.,

$$\mathcal{T}_u[\mathbf{f}](v) = \mathbf{f}(u^{-1}v). \tag{3}$$

Here, we only care about the group action over the feature maps defined on $\mathcal{G}$, because we always use a lifting operation to lift the input images defined on $\mathbb{R}^2$ to the feature maps on $\mathcal{G}$, where the equivariance can be preserved properly, as will be shown in Section 3.2.

3.2 G-CNNs

Let $\mathbf{f}^{(l)} : \mathcal{X} \to \mathbb{R}^{C_l}$ and $W : \mathcal{G} \to \mathbb{R}^{C_{l+1} \times C_l}$ be the input feature and the convolutional filter in the l-th layer, respectively, where C_l denotes the channel number of the l-th layer. $\mathcal{X}$ is taken as $\mathbb{R}^2$ for the first layer, and taken as $\mathcal{G}$ for the following layers. Then for any $g \in \mathcal{G}$, the group convolution [14, 7] of $\mathbf{f}^{(l)}$ and W on $\mathcal{G}$ at g is given by

$$\mathbf{f}^{(l+1)}(g) = \Psi[\mathbf{f}^{(l)}](g) = \int_{\mathcal{X}} W(g^{-1}\widetilde{g})\mathbf{f}^{(l)}(\widetilde{g})d\mu(\widetilde{g}), \tag{4}$$

where $\mu(\cdot)$ is the Haar measure. When $\mathcal{X}$ is discrete, Eqn. (4) can be rewritten as

$$\mathbf{f}^{(l+1)}(g) = \sum_{\widetilde{g} \in \mathcal{X}} W(g^{-1}\widetilde{g})\mathbf{f}^{(l)}(\widetilde{g}). \tag{5}$$

G-CNNs essentially generalize the translation equivariance of conventional convolution to a more general group $\mathcal{G}$.

In fact, the first layer maps the 2D images to a function defined on $\mathcal{G}$, while the following layers map one feature map on $\mathcal{G}$ to another. As a result, the computational complexity of the first layer and the following layers are of the order $O(k^2|\mathcal{A}|)$ and $O(k^2|\mathcal{A}|^2)$, respectively, where k is the kernel size in the spatial space. As a result, G-CNNs have a much larger computational cost when $\mathcal{A}$ is large, especially for the intermediate layers. In this work, we employ the first layer of G-CNNs as a lifting operation, and focus on reducing the computation of the latter layers.

3.3 Equivariant Attention Networks

Group Equivariant Self-Attention (G-SA) [24, 12] is a representative method of equivariant attention networks, whose form can be simplified as follows:

$$\mathbf{f}^{(l+1)}(g) = \sum_{\widetilde{g}\in\mathcal{G}} \mathrm{Softmax}_{\widetilde{g}}[h_Q^T(\mathbf{f}^{(l)}(g))(h_K(\mathbf{f}^{(l)}(\widetilde{g})) + P_{g^{-1}\widetilde{g}})]h_V(\mathbf{f}^{(l)}(\widetilde{g})), \tag{6}$$

where $h_V : \mathbb{R}^{C_l} \to \mathbb{R}^{C_{l+1}}$, and $h_Q, h_K : \mathbb{R}^{C_l} \to \mathbb{R}^d$ are the embedding functions of values, querys and keys, respectively, which are neural networks in the most general case. d is the dimension of the low dimensional embeddings, and $P_{g^{-1}\widetilde{g}} \in \mathbb{R}^d$ encodes the relative positions of the query $\mathbf{f}^l(g)$ and the key $\mathbf{f}^{(l)}(\widetilde{g})$.

3.4 Generalized Equivariant Framework

As more and more group equivariant structures emerge, researchers start to deduce the most general equivariant structures. To this end, Cohen et al. [7] and Kondor et al. [14] proposed a general theory of linear group equivariant structures, which indicates that G-CNNs are the most general equivariant linear layers. Besides, a lot of non-linear equivariant structures appear recently, such as equivariant self-attention layers [24, 12]. This motivates us to investigate a more general framework that can include both linear and non-linear cases.

In all, with only slight modification, most of layers in a neural network can be viewed as a kind of feature aggregation as follows:

$$\mathbf{f}^{(l+1)}(g) = \sum_{\widetilde{g}\in\mathcal{G}} H_{g,\widetilde{g}}(\mathbf{f}^{(l)}(g), \mathbf{f}^{(l)}(\widetilde{g})), \tag{7}$$

where the feature aggregation operator $H_{g,\widetilde{g}}(\cdot,\cdot) : \mathbb{R}^{C_l} \times \mathbb{R}^{C_l} \to \mathbb{R}^{C_{l+1}}$ is a mapping indexed by two locations g and $\widetilde{g}$, which describes how to aggregate the input features $\mathbf{f}(g)$ and $\mathbf{f}(\widetilde{g})$. In general, the above layer is not equivariant. However, we can find a general constraint for $H_{g,\widetilde{g}}(\cdot,\cdot)$ to make this layer equivariant over $\mathcal{G}$.

Theorem 1 *The layer formulated as Eqn.(7) is group equivariant if and only if there is a mapping $\widetilde{H}_{\hat{g}} : \mathbb{R}^{C_l} \times \mathbb{R}^{C_l} \to \mathbb{R}^{C_{l+1}}$ which is indexed by a single group element $\hat{g}$, and $\forall g, \widetilde{g}$,*

$$H_{g,\widetilde{g}}(\cdot,\cdot) = \widetilde{H}_{g^{-1}\widetilde{g}}(\cdot,\cdot). \tag{8}$$

Proof *Firstly, $\forall u, g$ and $\widetilde{g} \in \mathcal{G}$,*

$$\mathcal{T}_u\mathbf{f}^{(l+1)}(g) = \mathbf{f}^{(l+1)}(u^{-1}g) = \sum_{\widetilde{g}\in\mathcal{G}} H_{u^{-1}g,\widetilde{g}}(\mathbf{f}^{(l)}(u^{-1}g), \mathbf{f}^{(l)}(\widetilde{g})).$$

On the other hand,

$$\begin{aligned}
&\sum_{\widetilde{g}\in\mathcal{G}} H_{g,\widetilde{g}}(\mathcal{T}_u\mathbf{f}^{(l)}(g), \mathcal{T}_u\mathbf{f}^{(l)}(\widetilde{g})) \\
=&\sum_{\widetilde{g}\in\mathcal{G}} H_{g,\widetilde{g}}(\mathbf{f}^{(l)}(u^{-1}g), \mathbf{f}^{(l)}(u^{-1}\widetilde{g})) \\
=&\sum_{\widetilde{g}\in\mathcal{G}} H_{g,u\widetilde{g}}(\mathbf{f}^{(l)}(u^{-1}g), \mathbf{f}^{(l)}(\widetilde{g})).
\end{aligned}$$

As a result, we have that

$$\mathcal{T}_u\mathbf{f}^{(l+1)}(g) = \sum_{\widetilde{g}\in\mathcal{G}} H_{g,\widetilde{g}}(\mathcal{T}_u\mathbf{f}^{(l)}(g), \mathcal{T}_u\mathbf{f}^{(l)}(\widetilde{g})) \iff H_{u^{-1}g,\widetilde{g}}(\cdot,\cdot) = H_{g,u\widetilde{g}}(\cdot,\cdot).$$

Let $g \to ug$, we can find that $H_{g,\widetilde{g}}(\cdot,\cdot) = H_{ug,u\widetilde{g}}$, then, let u be g^{-1}, and denote $H_{e,g^{-1}\widetilde{g}}(\cdot,\cdot)$ as $\widetilde{H}_{g^{-1}\widetilde{g}}(\cdot,\cdot)$, we can find that $H_{g,\widetilde{g}}(\cdot,\cdot) = \widetilde{H}_{g^{-1}\widetilde{g}}(\cdot,\cdot)$.

■

From the theorem, we can get a group equivariant layer:

$$\mathbf{f}^{(l+1)}(g) = \sum_{\widetilde{g} \in \mathcal{G}} \widetilde{H}_{g^{-1}\widetilde{g}}(\mathbf{f}^{(l)}(g), \mathbf{f}^{(l)}(\widetilde{g})), \tag{9}$$

which is also the only equivariant form of Eqn. (7). Actually, the above theorem also reveals the essence of equivariance in previous works, i.e., if the relative positions of (g_1,$\widetilde{g_1}$) and (g_2,$\widetilde{g_2}$) are the same, *i.e.,* ${g_1}^{-1}\widetilde{g_1} = g_2^{-1}\widetilde{g_2} = \hat{g}$, the feature pairs located at the two tuples should be processed equally. In other words, we should employ the same function $\widetilde{H}_{\hat{g}}$ to act on these two input feature pairs.

From this perspective, we can readily see that both the kernel sharing used in G-CNNs, Eqn. (4), and the relative position encoding adopted in the G-SA, Eqn. (6), utilizes the above rule. According to Theorem 1, designing a group equivariant layer becomes much more easily and flexibly than ever, as we only need to design a new function $\widetilde{H}_{\hat{g}}$. In addition, the new formulation provides a more general perspective on the group equivariant layer, i.e., sharing the parameters of function $\widetilde{H}_{\hat{g}}$, which generalizes the kernel sharing schemes in G-CNNs. Based on the above understanding, we can see that if we replace the feature vector in the right hand side of Eqn. (9) with the local patch at group element g and $\widetilde{g}$, respectively, it is still equivariant, *i.e.,*

$$\mathbf{f}^{(l+1)}(g) = \sum_{\widetilde{g} \in \mathcal{G}} \widetilde{H}_{g^{-1}\widetilde{g}}(\mathcal{F}_{\mathcal{N}(g)}, \mathcal{F}_{\mathcal{N}(\widetilde{g})}) \tag{10}$$

where the $\mathcal{F}_{\mathcal{N}(g)}$ and $\mathcal{F}_{\mathcal{N}(\widetilde{g})}$ denote the local patches of g and $\widetilde{g}$, in which $\mathcal{N}(g)$ represent g's neighborhood $\{gg' | g' \in \mathcal{N}(e)\}$ and $\mathcal{N}(e)$ is the neighborhood of the identity element $e \in \mathcal{G}$. We acquire the local patches by concatenating features in the neighborhoods of g and $\widetilde{g}$ in a predefined order on $\mathcal{N}(e)$ respectively. We denote the concatenation operator as $\bigcup$, and will discuss the above in detail in Section 4.1, which shows that concatenating features can not only make our framework more flexible, but also help to reduce the computational burden of our newly proposed equivariant layer.

4 Efficient Equivariant Layer

A straight-forward and easy case of Eqn. (10) is to adopt $\widetilde{H}_{\hat{g}}$, $\forall \hat{g} \in \mathcal{G}$, as a multi-layer perceptron (MLP), where the subscript $\hat{g}$ is used to identify different MLPs. However, in Eqn. (10), we have to compute a mapping from two high dimensional vectors to another high dimensional one for each input pair of g and $\widetilde{g}$, which is very expensive. A similar issue exists in computing the attention score in self-attention. To deal with this problem, we decompose $\widetilde{H}$ into the following form to reduce the computation, *i.e.,*

$$\forall \hat{g} \in \mathcal{G}, \quad \widetilde{H}_{\hat{g}}(x, y) = K_{\hat{g}}(x) \odot V(y) \tag{11}$$

where $\odot$ means element-wise product, and $K_{\hat{g}} : \mathbb{R}^{C_l|\mathcal{N}(e)|} \rightarrow \mathbb{R}^{C_{l+1}}$ is a kernel generator and $V : \mathbb{R}^{C_l|\mathcal{N}(e)|} \rightarrow \mathbb{R}^{C_{l+1}}$ is an encoder. We use $|\cdot|$ to denote the numbers of elements in a set. Hence, we can compute $K_{\hat{g}}(x)$ and $V(y)$ separately. In addition, to further save computation, we split the kernel into several slices along the channels, such that $K_{\hat{g}}$ is shared across these slices, i.e., $\forall\, 1 \leq i, j \leq C_{l+1}$, $K^i_{\hat{g}} = K^j_{\hat{g}}$ if $i \equiv j \pmod{s}$, where s is the number of slices, and i and j are channel indexes. The $K_{\hat{g}}$ is essentially a dynamic filters which is adaptive to features around g, avoiding the spatial-agnostic problem in G-CNNs. Unlike conventional dynamic filters, which are matrices, the output of $K_{\hat{g}}$ is a vector, which can be viewed as a depth wise kernel [3]. This can decouple channel dimension with spatial dimension during feature aggregation to reduce the computational cost. Position information is implicitly encoded in the organized output form of our kernel generator, rather than using explicit positional embedding in the group self-attention layer [12, 24].

In practice, we can view the whole kernel family $\{K_{\hat{g}}\}_{\hat{g} \in \mathcal{G}}$ as the output of a single mapping, i.e., $\widetilde{K}$: $\mathbb{R}^{C_l} \rightarrow \mathbb{R}^{|\mathcal{G}|C_{l+1}}$. Then, we resize the output of $\widetilde{K}$ to be a $|\mathcal{G}| \times C_{l+1}$ matrix, with different rows represent different $K_{\hat{g}}$. Namely, if we adopt $\widetilde{K}$ as an MLP, the computations and parameters used for hidden layer are shared across $K_{\hat{g}}$ for different $\hat{g}$, which is another merit of the Eqn. (11). However, there is still a large search space for $\widetilde{H}_{\hat{g}}$, as Eqn (11) is only a special structue of $\widetilde{H}_{\hat{g}}$, we leave a more complete study of $\widetilde{H}_{\hat{g}}$ in the future work.

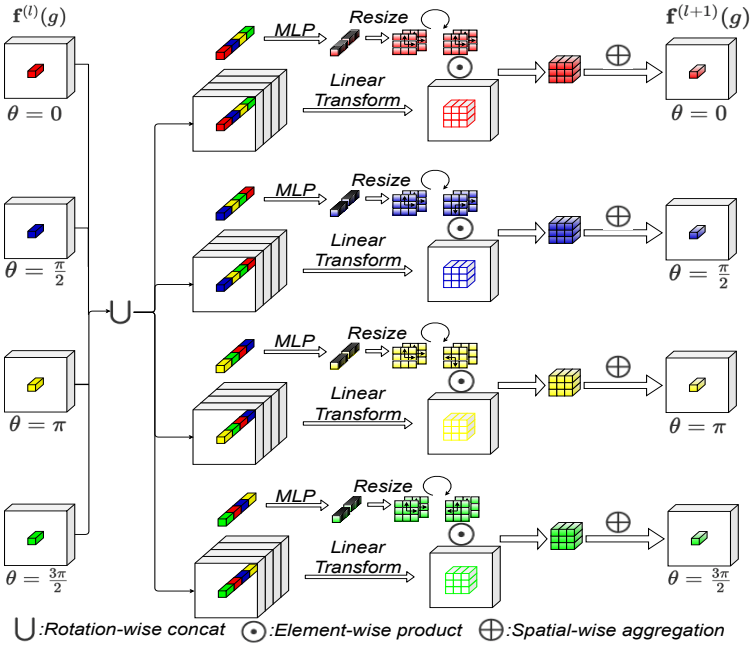

Figure 1: An example of our E^4-layer on $p4$ group. We firstly concatenate features along rotation dimension in a predefined order on the $\mathcal{C}_4$ and then pass them through the MLP and the linear layer to generate kernel and encode features, respectively. After this, the element-wise product is carried out to compute $\widetilde{H}_{g^{-1}\widetilde{g}}$, and finally spatial-wise aggregation is performed to acquire the output. Note that when computing output features at different rotation dimensions, the generated kernel should be rotated by a specific degree to keep the correct relative position.

4.1 Implementation on Affine Group

In this section, we design a very efficient equivariant layer based on Eqn. (11) for affine group $\mathbb{R}^2 \rtimes \mathcal{A}$. The computation of the operator is:

$$\mathbf{f}^{(l+1)}(g) = \sum_{\widetilde{g} \in \mathcal{N}_1(g)} K_{g^{-1}\widetilde{g}} \left(\bigcup_{g' \in \mathcal{N}_2(g)} \mathbf{f}^{(l)}(g') \right) \odot V \left(\bigcup_{\widetilde{g}' \in \mathcal{N}_2(\widetilde{g})} \mathbf{f}^{(l)}(\widetilde{g}') \right). \tag{12}$$

Following the standard practice in computer vision, aggregation is done only on the local neighborhood of g, $\mathcal{N}_1(g)$. To save computation, we choose $\mathcal{N}_1(g)$ to be only spatial-wise neighborhood, *i.e.,* $\mathcal{N}_1(g) = \{g(v, e_{\mathcal{A}}) \mid v \in \Omega\}$, where $\Omega \in \mathbb{R}^2$ and $e_{\mathcal{A}}$ is the identity element of group $\mathcal{A}$. However, aggregating information along spatial neighborhood only discards the information interaction along $\mathcal{A}$, which could lead to a drop in performance [17]. We alleviate the issue by concatenating the feature map along $\mathcal{A}$, *i.e.*, we choose $\mathcal{N}_2(g)$ to be $\{g(\mathbf{0}, a) | a \in \mathcal{A}\}$. The order of concatenation is predefined on $\mathcal{A}$. As will be shown in the later experiments, this concatenation does not introduce much computation but can significantly improve performance. Compared to group convolution, such a design enables us to decouple the feature aggregation across the spatial dimension and the $\mathcal{A}$ dimension to further reduce computational costs.

In practice, we adopt the $\widetilde{K}$ as a two layer MLP: $\widetilde{K}(x) = W_2 \mathrm{Relu}(W_1 x)$, where $W_1 \in \mathbb{R}^{C_l/r \times C_l|\mathcal{A}|}, W_2 \in \mathbb{R}^{|\Omega| s \times C_l/r}$, and r is the reduction ratio which saves both parameters and computation. For 2D images, Ω is usually adopted as a $k \times k$ square mesh grids and $|\Omega| = k^2$, where k is the kernel size. For better illustration, we visualize a concrete layer of Eqn. (12) by choosing $\mathcal{G}$ as $p4$ in Figure 1.

4.2 Computational Complexity Analysis

In practice, the feature map is defined on discrete mesh grids. We use h and w to denote the height and the width of mesh grids. As the numbers of the input and output channels are usually the same, we assume $C_l = C_{l+1} = c$.

Parameter Analysis The number of learnable parameters of E^4-layer (12) is $c^2|\mathcal{A}|(1 + 1/r) + csk^2/r$. As $s \ll c$, parameter counts are dominated by the first term when k is not too large, and increasing kernel size will not significantly increase parameter counts, which is shown in later experiments. The parameters count of group convolution layer is $c^2k^2|\mathcal{A}|$. Notice that $(1+1/r) \ll k^2$ and $s/r \ll c|\mathcal{A}|$, parameters count of our E^4-layer is significantly less than that of group convolution layer.

Time Complexity Analysis The FLOPs of E^4-layer and group convolution layer are $(1 + 1/r)c^2|\mathcal{A}|^2hw + (1 + s/r)k^2c|\mathcal{A}|hw$ and $k^2c^2|\mathcal{A}|^2hw$, respectively. Similarly, as $(1 + 1/r) \ll k^2$ and $(1 + s/r) \ll c|\mathcal{A}|$, the FLOPs of E^4-layer is significantly lower than that of group convolutional layer.

It can be observed that both the parameter count and FLOPs of our E^4-layer are composed of two terms, one depending on k^2 and the other not relying on k, which is a result of disentangling across spatial dimension with both channels and $\mathcal{A}$ during feature aggregation.

5 Experiments

In this section, we conduct extensive experiments to study and demonstrate the performance of our model. The experimental results show that our model has a greater capacity than the group-convolution-based one in terms of parameter efficiency, computational efficiency, data efficiency and accuracy. On the MNIST-rot dataset, we detailedly study the effect of hyperparameters on the number of parameters, computation FLOPs and performance of our model. All the experiments are done on the GeForce RTX 3090 GPU.

5.1 Rotated MNIST

The MNIST-rot dataset [18] is the most widely used benchmark to test the equivariant models. It contains 62k 28×28 randomly rotated gray-scale handwritten digits. Images in the dataset are split into 10k for training, 2k for validation and 50k for testing. Random rotation of digits and only 20 percent of training data of the standard MNIST dataset increases the difficulty of classification.

Table 1: Test error on rot-MNIST(with standard deviation under 5 random seed variations)

Model	Test error (%)	Params	FLOPs
$p4$_SA [24]	2.54±0.052	44.67K	400M
$p4$_CNN [5]	1.79±0.043	77.54K	46.2M
α_p4_CNN[23]	1.69±0.021	73.13K	27.0M
E^4-Net (Ours)	**1.29±0.023**	**18.8K**	**17M**
E^4-Net(Large)(Ours)	**1.17±0.019**	**41.1K**	**36.9M**

For a fair comparison, we keep both training settings and architectures of our model as close as possible to previous works [5, 24]. In addition, we adopt the $p4$ group to construct all our models in this section. In our first experiment, we adopt our E^4-Net given in the appendix to make a comparison to previous works. This is a very lightweight model which contains only 18.8K learnable parameters. It is composed of one group convolutional layer which lifts the image to the $p4$ group, six E^4-layers and one fully connected layer. Two 2×2 max-pooling layers are inserted after the first and the third E^4-layer to downsample feature maps. The last E^4-layer is followed by a global max group pooling layer [5], which takes the maximum response over the entire group, to ensure the predictions invariant to rotations.

Our model is trained using the Adam optimizer [13] for 200 epochs with a batch size of 128. The learning rate is initialized as 0.02 and is reduced by 10 at the 60th, 120th and 160th epochs. The weight decay is set as 0.0001 and no data augmentation is used during training. The results are listed in Table 1. Our models significantly outperform G-CNNs [5] using only about 25% parameters and 40% FLOPs. For G-SA [24], which is a group equivariant stand-alone self-attention model, even performs inferiorly to G-CNNs with much more computational cost. The α-p4-CNN model [23] further

introduces the attention mechanism to group convolution along both spatial and channel dimensions to enhance the expressiveness of G-CNNs, while our E^4-Net still significantly outperforms it with less computation costs. We also experiment with a larger model to further demonstrate the capacity of our model, which is listed in the last line of Table 1.

Ablation Study of Concatenation: In the E^4-layer (12), we introduce the concatenation operation to enable the disentanglement across the rotation and the spatial information interaction. To study the importance of concatenation, we carry out experiments on the case that neither $K_{\hat{g}}$ nor V in Eqn. (12) use concatenation, i.e., $\mathcal{N}_2(g) = g$, $N_3(\widetilde{g}) = \widetilde{g}$. As shown in the first line of Table 2, this leads to a significant drop in performance. This is because if aggregation in Eqn.(12) is done merely in the spatial neighborhoods without concatenation, there is no information interaction along the rotation dimensions. We also experiment the cases using concatenation only in $K_{\hat{g}}$ or V, and the performance of both is better than the case without concatenation but is still inferior to the case with concatenation in both $K_{\hat{g}}$ and V. This further illustrates the importance of concatenation along $\mathcal{A}$.

Table 2: The effect of concatenation

Concate	Test error (%)	Params	FLOPs
None	4.10±0.085	9.9K	8.9M
only K	1.96±0.045	14.4K	13M
only V	1.52±0.036	14.4K	13M
K&V	**1.29±0.023**	18.8K	17M

Hyperparameters Analysis: We investigate the effect of various hyperparameters used in the E^4-layer. The reduction ratio r and the slice number s in the $K_{\hat{g}}$ and kernel size k control the computations and parameters of the layer. Based on the baseline model, we vary the three hyperparameters respectively. As shown in the Table 3, improvement is observed when decreasing the reduction ratio and increasing the slice number, with the cost of computational burden increasing. Especially, the improvement of $s = 2$ over $s = 4$ and $r = 1$ over $r = 2$ is marginal, which is attributes to redundancy in the kernel [2]. In conclusion, appropriately increasing the reduction ratio r and decreasing the slice number s can help to reduce computational costs while preserving performance. Keeping other hyperparameters fixed, we study the effect of kernel size on our model. In Table 3, the performance peaks when kernel size equals 7. In general, a larger kernel size leads to improved performance due to a larger receptive field. In addition, as explained in Section 4.2, increasing kernel size does not dramatically increase parameters and FLOPs as standard convolution.

Table 3: Hyperparameters Analysis

Hyperparam	Test error (%)	Params	FLOPs
s=1	1.45±0.022	16.3K	14.9M
s=2	1.29±0.023	18.8K	17M
s=4	1.24±0.026	23.9K	21.2M
r=1	1.29±0.023	18.8K	17M
r=2	1.33±0.026	13.0K	12M
r=4	1.37±0.025	10.1K	9.5M
k=3	1.46±0.031	15.6K	14.3M
k=5	1.29±0.023	18.8K	17M
k=7	1.27±0.021	23.8K	21.1M

5.2 Natural Image Classification

In this section, we evaluate the performance of our model on the two common natural image datasets, CIFAR10 and CIFAR100 [15]. The CIFAR-10 and the CIFAR100 datasets consist of 32×32 images belonging to 10 and 100 classes, respectively. Both of the datasets contain 50k training data and 10k testing data. Before training, images are normalized according to the channel means and standard deviations.

In this experiment, we adopt ResNet-18 [10] as the baseline model(short as R18), which is composed of an initial convolution layer, followed by 4 stage Res-Blocks and one final classification layer. Following the standard practice in [5], we replace all the conventional layers with $p4$ ($p4m$) convolutions in R18 and increase the width of each layer by $\sqrt{4}$ ($\sqrt{8}$) to keep the learnable parameters approximately the same. We denote the resulting models as $p4$-R18 ($p4m$-R18). We replace the second group convolution layer in each Res-Block of $p4$-R18 ($p4m$-R18) with our E^4-layer, resulting in the $p4$-E^4R18 ($p4m$-E^4R18). For a fair comparison, all the above models are trained under the same training settings. We use the stochastic gradient descent with an initial learning rate of 0.1, a Nesterov momentum of 0.9 and a weight decay of 0.0005. The learning rate is reduced by 5 at 60th, 120th, and 160th epochs. Models are trained for 200 epochs using 128 batch size. *No data augmentation* is used during training.

Table 4: Test error on CIFAR10 and CIFAR100. (with standard deviation under 5 random seed variations)

Model	CIFAR10 (%)	CIFAR100 (%)	Params	FLOPs
R18	9.7±0.43	34±0.76	11M	0.56G
$p4$-R18	7.53±0.21	27.96±0.56	11M	2.99G
$p4$-E^4R18(Ours)	**6.42±0.14**	**26.59±0.36**	**5.8M**	**1.85G**
$p4m$-R18	5.83±0.17	24.95±0.42	10.8M	5.63G
$p4m$-E^4R18(Ours)	**4.96±0.16**	**22.18±0.46**	**6.0M**	**3.87G**

The classification accuracy, parameters count and FLOPs of all models on CIFAR10 and CIFAR100 are reported in Table 4. We can see that models incorporating more symmetry achieve better performance, *i.e.,* R18 $\leq$ $p4$-R18 $\leq$ $p4m$-R18. Our $p4$ and $p4m$ models significantly outperform their counterparts on both CIFAR10 and CIFAR100. Furthermore, our model decreases the parameter count and FLOPs by 45% and 32%, respectively. Notice that the model size reduction is purely caused by the introduction of our E^4-layers, as topological connections and width of each layer of E^4 model and its counterparts are the same.

Data Efficiency: To further study the performance of our model, we train all the models listed in Table 4 on CIFAR10 with different sizes of training data. To be specific, we consider 5 settings, where 1k, 2k, 3k, 4k and 5k training data of each class are randomly sampled from the CIFAR10 training set. Testing is still performed on the original test set of CIFAR10. Other training settings are identical to the above. We visualize the results in Figure 2. Numerical results are listed in the appendix.

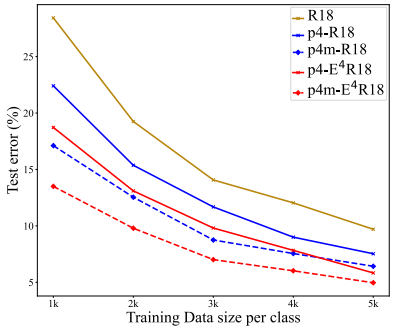

Figure 2: Trend of test error(%) on various training data sizes.

It is observed that the performance gap between $p4$, $p4m$ and $\mathbb{R}^2$ models tend to increase as we reduce the training data. This is mainly because that the prior that the label is invariant to rotations is more important when training data are fewer. The trend is also observed in the gap between our models and their counterparts. For instance, the gap between $p4$-E^4-R18 and $p4m$-R18 is 0.87% when training data of each class is 5k, while it is enlarged to 5.22% when training data of each class is reduced to $1k$. Especially, we observe the line of $p4$-E^4R18 intersects with the one of $p4m$-R18, which further indicates that our model is much more data efficient than G-CNNs. As indicated above, symmetry prior is more important when training data are fewer, and the data efficiency of our model implies that $p4$-E^4R18 and $p4m$-E^4R18 can better exploit the symmetry of data.

6 Conclusions

In this work, we propose a general framework of group equivariant models which delivers a unified understanding on the previous group equivariant models. Based on the new understanding, we propose a novel efficient and powerful group equivariant layer which can serve as a drop-in replacement for convolutional layers. Extensive experiments demonstrate the E^4-layer is more powerful, parameter efficient and computational efficient than group convolution layers and their variants. Through a side by side comparison with G-CNNs, we demonstrate our E^4-layer can significantly improve data efficiency of equivariant models, which show great potential for reducing the cost of collecting data, especially in the areas where acquiring data is very expensive, such as medical diagnosis. In future research, we expect to extend our work to 3D space groups to relieve the huge computational burden of previous works.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *TPAMI*, 40(4):834–848, 2017.

[2] Xiuyuan Cheng, Qiang Qiu, Robert Calderbank, and Guillermo Sapiro. RotDCF: Decomposition of convolutional filters for rotation-equivariant deep networks. In *ICLR*, 2018.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[4] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *ICML*, pages 1321–1330, 2019.

[5] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *ICML*, pages 2990–2999, 2016.

[6] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *ICLR*, 2018.

[7] Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. In *NeurIPS*, 2019.

[8] Taco S Cohen and Max Welling. Steerable CNNs. In *ICLR*, 2017.

[9] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D roto-translation equivariant attention networks. *NeurIPS*, 33, 2020.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[11] Emiel Hoogeboom, Jorn WT Peters, Taco S Cohen, and Max Welling. HexaConv. In *ICLR*, 2018.

[12] Michael Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. LieTransformer: Equivariant self-attention for lie groups. *arXiv preprint arXiv:2012.10885*, 2020.

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

[14] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *ICML*, pages 2747–2755, 2018.

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25:1097–1105, 2012.

[17] Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. In *CVPR*, pages 289–297, 2016.

[18] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pages 473–480, 2007.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. Weightnet: Revisiting the design space of weight networks. In *ECCV*, 2020.

[21] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019.

[22] Shaoqing Ren, Kaiming He, Ross B Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[23] David Romero, Erik Bekkers, Jakub Tomczak, and Mark Hoogendoorn. Attentive group equivariant convolutional networks. In *ICML*, pages 8188–8199. PMLR, 2020.

[24] David W Romero and Jean-Baptiste Cordonnier. Group equivariant stand-alone self-attention for vision. In *ICLR*, 2021.

[25] David W Romero and Mark Hoogendoorn. Co-attentive equivariant neural networks: Focusing equivariance on transformations co-occurring in data. In *ICLR*, 2020.

[26] Zhengyang Shen, Lingshen He, Zhouchen Lin, and Jinwen Ma. PDO-eConvs: Partial differential operator based equivariant convolutions. In *ICML*, 2020.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[28] Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. In *ICLR*, 2019.

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[31] Maurice Weiler and Gabriele Cesa. General E(2)-equivariant steerable CNNs. In *NeurIPS*, pages 14334–14345, 2019.

[32] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. In *NeurIPS*, pages 10381–10392, 2018.

[33] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant CNNs. In *CVPR*, pages 849–858, 2018.

[34] Daniel Worrall and Gabriel Brostow. CubeNet: Equivariance to 3D rotation and translation. In *ECCV*, pages 567–584, 2018.

[35] Daniel Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. *NeurIPS*, 32:7366–7378, 2019.

[36] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *NeurIPS*, 2019.

[37] Yikang Zhang, Jian Zhang, Qiang Wang, and Zhao Zhong. Dynet: Dynamic convolution for accelerating convolutional neural networks. *arXiv preprint arXiv:2004.10694*, 2020.

[38] Jingkai Zhou, Varun Jampani, Zhixiong Pi, Qiong Liu, and Ming-Hsuan Yang. Decoupled dynamic filter networks. In *CVPR*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See sec.4
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The datasets, experimental settings and instructions are provided, but the codes are proprietary.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see section 5
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] see section 5
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see section 5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]