Learning Partial Equivariances from Data

Anonymous Author(s)
Affiliation
Address
email

Abstract

Group Convolutional Neural Networks (G-CNNs) constrain learned features to respect the symmetries in the selected group, and lead to better generalization when these symmetries appear in the data. However, if this is not the case equivariance leads to overly constrained models and worse performance. Frequently, transformations occurring in data can be better represented by a subset of a group than by the group as a whole, e.g., rotations in $[-90°, 90°]$. In such cases, a model that respects equivariance *partially* is better suited to represent the data. In addition, relevant transformations may differ for low and high-level features. For instance, full rotation equivariance is useful to describe edge orientations in a face, but partial rotation equivariance is better suited to describe face poses relative to the camera. In other words, the optimal level of equivariance may differ per layer. In this work, we introduce *Partial G-CNNs*: G-CNNs able to learn layer-wise levels of partial and full equivariance to discrete, continuous groups and combinations thereof as part of training. Partial G-CNNs retain full equivariance when beneficial, e.g., for rotated MNIST, but adjust it whenever it becomes harmful, e.g., for classification of 6 / 9 digits or natural images. We empirically show that partial G-CNNs pair G-CNNs when full equivariance is advantageous, and outperform them otherwise.

1 Introduction

The translation equivariance of Convolutional Neural Networks (CNNs) [28] has proven an important inductive bias for good generalization on vision tasks. This is achieved by restricting learned features to respect the translation symmetry encountered in visual data, such that if an input is translated, its features are also translated, but not modified. Group equivariant CNNs (G-CNNs) [6] extend equivariance to other symmetry groups. Analogously, they restrict the learned features to respect the symmetries in the group considered such that if an input is transformed by an element in the group, e.g., a rotation, its features are also transformed, e.g., rotated, but not modified.

Nevertheless, the group to which G-CNNs are equivariant must be fixed prior to training, and imposing equivariance to symmetries not present in the data leads to overly constrained models and worse performance [5]. The latter comes from a difference in the data distribution, and the family of distributions the model can describe. Consequently, the group must be selected carefully, and it should correspond to the transformations that appear naturally in the data.

Frequently, transformations appearing in data can be better represented by a subset of a group than by the group as a whole, e.g., rotations in $[-90°, 90°]$. For instance, natural images much more likely show an elephant standing straight or slightly rotated than an elephant upside-down. In some cases, group transformations even change the desired response, e.g., in the classification of the digits 6 and 9. In both examples, the data distribution is better represented by a model that respects rotation equivariance *partially*. That is, a model equivariant to some, but not all rotations.

Moreover, the optimal level of equivariance may change per layer. This results from changes in the likelihood of some transformations for low and high-level features. For instance, the orientations of

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

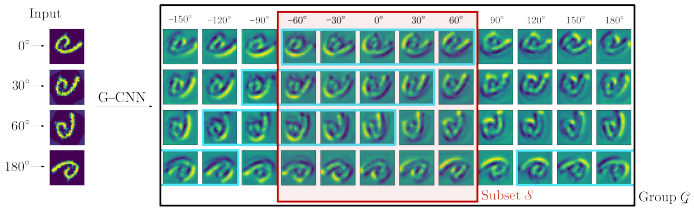

Figure 1: Partial group convolution. In a group convolution, the domain of the output is the group $\mathcal{G}$. Consequently, all output components are part of the output for any group transformation of the input. In a partial group convolution, however, the domain of the output is a learned subset $\mathcal{S}$ and all values outside of $\mathcal{S}$ are discarded. As a result, parts of the output in a partial group convolution can change for different group transformations of the input. In this figure, the output components within $\mathcal{S}$ for a $0°$ rotation (outlined in blue) gradually leave $\mathcal{S}$ for stronger transformations of the input. For strong transformations –here a $180°$ rotation–, the output components within $\mathcal{S}$ are completely different. This difference allows partial group convolutions to distinguish among input transformations. By controlling the size of $\mathcal{S}$, the level of equivariance of the operation can be adjusted.

edges in an human face are properly described with full rotation equivariance, whereas the pose of human faces relative to the camera are better represented by rotations in a subset of the circle.

The previous observations indicate that constructing a model with different levels of equivariance at each layer may be advantageous. Weiler and Cesa [42] empirically observed that manually tuning the level of equivariance at different layers leads to accuracy improvements for non-fully equivariant tasks. Nevertheless, manually tuning layer-wise levels of equivariance is not straightforward and requires iterations over several possible combinations of equivariance levels. Consequently, it is desirable to construct a model able to learn optimal levels of equivariance directly from data.

In this work, we introduce *Partial Group equivariant Convolutional Neural Networks*: a family of equivariant models able to *learn* layer-wise levels of equivariance directly from data. This is achieved by learning a probability distribution over group elements at each group convolutional layer in the network. Instead of sampling group elements uniformly from the group during the group convolution as is done in G-CNNs, Partial G-CNNs sample group elements from the learned distribution. This allows Partial G-CNNs to adjust their level of equivariance at every layer during training.

We evaluate Partial G-CNNs on illustrative toy tasks and on challenging benchmark datasets. We show that whenever full equivariance is beneficial, e.g., for rotated MNIST, Partial G-CNNs learn to remain fully equivariant. However, if harmful, Partial G-CNNs learn to adjust equivariance to a subset of the group to improve accuracy, e.g., for classification of 6 / 9 digits and natural images. Our method improves upon conventional group equivariant networks when equivariance reductions are advantageous, and matches their performance whenever their design is optimal.

In summary, our **contributions** are:

- We present a novel architectural design for equivariant networks, with which group convolutional networks are able to learn layer-wise partial or full equivariances from data.
- We show empirically that Partial G-CNNs improve upon conventional G-CNNs for tasks for which full equivariance is harmful. Nevertheless, if full equivariance is beneficial, Partial G-CNNs learn to be fully equivariant and match the performance of G-CNNs.

2 Background

Group equivariance. Group equivariance is the property of a map to respect the transformations in a group. We say that a map is equivariant to a group if whenever the input is transformed by elements of the group, the output of the map is equally transformed but not modified. Formally, for a group $\mathcal{G}$ with elements $g \in \mathcal{G}$ acting on a set $\mathcal{X}$, and a map $\phi : \mathcal{X} \to \mathcal{X}$, we say that ϕ is equivariant to $\mathcal{G}$ if:

$$\phi(g \cdot x) = g \cdot \phi(x), \quad \forall x \in \mathcal{X}, \ \forall g \in \mathcal{G}. \tag{1}$$

For example, the convolution of a signal $f : \mathbb{R} \to \mathbb{R}$ and a kernel $\psi : \mathbb{R} \to \mathbb{R}$ is *translation equivariant* because $\mathcal{L}_t(\psi * f) = \psi * \mathcal{L}_t f$, where $\mathcal{L}_t$ translates the function by t: $\mathcal{L}_t f(x) = f(x - t)$. That is, if the input is translated, its numerical descriptors are also translated but not modified.

The group convolution. In order to construct neural networks equivariant to a given group $\mathcal{G}$, it is crucial to define an operation that respects the symmetries in the group. The *group convolution* is such

a mapping. It generalizes the convolution for equivariance to general symmetry groups. Formally, for any $u \in \mathcal{G}$, the group convolution of a signal $f : \mathcal{G} \to \mathbb{R}$ and a kernel $\psi : \mathcal{G} \to \mathbb{R}$ is given by:

$$h(u) = (\psi * f)(u) = \int_{\mathcal{G}} \psi(v^{-1}u) f(v) \, \mathrm{d}\mu_{\mathcal{G}}(v), \tag{2}$$

where $\mu_{\mathcal{G}}(\cdot)$ is the (invariant) Haar measure of the group. The group convolution is $\mathcal{G}$-equivariant. That is, it holds that, for all $u, v, w \in \mathcal{G}$:

$$(\psi * \mathcal{L}_w f)(u) = \mathcal{L}_w(\psi * f)(u), \text{ with } \mathcal{L}_w f(u) = f(w^{-1}u).$$

The lifting convolution. Regularly, the input of a neural network is not readily defined on the group of interest $\mathcal{G}$, but on a sub-domain $\mathcal{X}$, i.e., $f : \mathcal{X} \to \mathbb{R}$. For instance, medical images are still functions of $\mathbb{R}^2$ although equivariance to 2D-translations and planar rotations is often desirable. In this case $\mathcal{X}$=$\mathbb{R}^2$ and the group of interest is $\mathcal{G}$=$\mathrm{SE}(2)$. Consequently, we must first *lift* the input signal from its domain $\mathcal{X}$ to the group $\mathcal{G}$ in order to use group convolutions. This is achieved with a *lifting convolution*. Formally, for $u \in \mathcal{G}$, $v \in \mathcal{X}$, the lifting group convolution is defined as:

$$(\psi *_{\text{lift}} f)(u) = \int_{\mathcal{X}} \psi(v^{-1}u) f(v) \, \mathrm{d}\mu_{\mathcal{G}}(v). \tag{3}$$

s **Practical implementation of the group convolution.** The group convolution requires integration over a continuous domain. Hence, it cannot be computed in finite time and must be approximated. Two main strategies exist to approximate group convolutions with regular group representations: (*i*) group discretization [6], and (*ii*) Monte Carlo approximations [15].

Group discretization approximates the group convolution using a fixed discretization of the group. However, this approximation is *only* equivariant to transformations in the discretization and not to the continuous group. A better solution is to use a Monte Carlo approximation of the group convolution by sampling transformations uniformly from the entire group during each forward pass:

$$(\psi \hat{*} f)(u_i) = \textstyle\sum_j \psi(v_j^{-1}u_i) f(v_j) \mu_{\mathcal{G}}(v_j), \tag{4}$$

where $\{u_i\}$, $\{v_j\}$ are the discretizations of the output and input domain of the operation, respectively.

Importantly, Monte Carlo approximations require the convolutional kernel ψ be defined on the *continuous group*. As the domain cannot be enumerated, ψ cannot be parameterized by independent weights. Instead, Finzi et al. [15] parameterize convolutional kernels with small neural networks, i.e., $\psi(x)$=$\texttt{MLP}(x)$. This parameterization maps all elements $v_j^{-1}u_i$ to a well-defined kernel value.

3 Partial Group Equivariant Networks

3.1 (Approximate) partial group equivariance

Before defining the partial group convolution, we first formalize what we mean by partial group equivariance. In contrast to group equivariance, we say that a map ϕ is partially equivariant to $\mathcal{G}$, if it is equivariant to transformations in a subset of the group $\mathcal{S} \subset \mathcal{G}$, but not necessarily to all transformations in the group $\mathcal{G}$. That is, if:

$$\phi(g \cdot x) = g \cdot \phi(x) \quad \forall x \in \mathcal{X}, \forall g \in \mathcal{S}. \tag{5}$$

Different from equivariance to a *subgroup* of $\mathcal{G}$: a subset of the group which also fulfill the group axioms, we do not restrict the subset $\mathcal{S}$ to be itself a group.

As explained in detail in Sec. 3.3, partial equivariance holds, in general, only approximately, and it is exact only if $\mathcal{S}$ is a subgroup of $\mathcal{G}$. This results from the set $\mathcal{S}$ not being necessarily closed under group actions. Hence, partial equivariance is a relaxation of group equivariance similar to *soft invariance* [40]: the property of a map to be approximately invariant. We opt for the word *partial* in the equivariance setting to emphasize that (approximate) partial group equivariance arises by restricting the domain of the signals in a group convolution to a subset, i.e., a part, of the group.

3.2 The partial group convolution

Let $\mathcal{S}^{(1)}, \mathcal{S}^{(2)}$ be subsets of a group $\mathcal{G}$ and $\mathrm{p}(u)$ be a probability distribution on the group, which is non-zero only on $\mathcal{S}^{(2)}$. We define the partial group convolution from a function $f : \mathcal{S}^{(1)} \to \mathbb{R}$ to a function $h : \mathcal{S}^{(2)} \to \mathbb{R}$ as:

$$h(u) = (\psi * f)(u) = \int_{\mathcal{S}^{(1)}} \mathrm{p}(u) \psi(v^{-1}u) f(v) \, \mathrm{d}\mu_{\mathcal{G}}(v); \; u \in \mathcal{S}^{(2)}, v \in \mathcal{S}^{(1)}. \tag{6}$$

In contrast to group convolutions whose input and output functions are defined on the entire group, i.e., $f, h : \mathcal{G} \to \mathbb{R}$, the domain of the input and output functions of the partial group convolution can

be defined on *subsets of the group*. By learning these subsets during training, the model can learn to be (*a*) fully equivariant $\big(\mathcal{S}^{(1)}{=}\mathcal{G}, \mathcal{S}^{(2)}{=}\mathcal{G}\big)$; (*b*) partially equivariant $\big(\mathcal{S}^{(1)}{\neq}\mathcal{G}, \mathcal{S}^{(2)}{\neq}\mathcal{G}\big)$, or (*c*) forget some equivariances $\big($with $\mathcal{S}^{(2)}$ a subgroup of $\mathcal{G}\big)$.

3.3 From group convolutions to partial group convolutions

In this section, we show how group convolutions can be generalized to describe partial equivariances. Vital to our analysis is the equivariance proof of the group convolution [8]. In addition, we must distinguish between the domains of the input and output signals of the group convolution, i.e., the domains of f and h in Eq. 2. This distinction is important because they may be different for partial group convolutions. From here on, we refer to these as ***input domain*** and ***output domain***.

Proposition 3.1. *Let $\mathcal{L}_w f(u){=}f(w^{-1}u)$. The group convolution is $\mathcal{G}$-equivariant in the sense that:*

$$(\psi * \mathcal{L}_w f)(u) = \mathcal{L}_w(\psi * f)(u), \textit{ for all } u, v, w \in \mathcal{G}. \tag{7}$$

Proof. [8]

$$\begin{aligned}(\psi * \mathcal{L}_w f)(u) &= \int_{\mathcal{G}} \psi(v^{-1}u) f(w^{-1}v)\, \mathrm{d}\mu_{\mathcal{G}}(v) = \int_{\mathcal{G}} \psi(\bar{v}^{-1}w^{-1}u) f(\bar{v})\, \mathrm{d}\mu_{\mathcal{G}}(\bar{v}) \\ &= (\psi * f)(w^{-1}u) = \mathcal{L}_w(\psi * f)(u).\end{aligned}$$

In the first line, the change of variables $\bar{v}{:=}w^{-1}v$ is used. This is possible because the group convolution is a map from the group to itself, and thus if $w, v \in \mathcal{G}$, so does $w^{-1}v$. Since the Haar measure is an invariant measure on the group, we have that $\mu_{\mathcal{G}}(v){=}\mu_{\mathcal{G}}(\bar{v})$, for all $v, \bar{v} \in \mathcal{G}$. □

Going from the group $\mathcal{G}$ to a subset $\mathcal{S}$. Crucial to the proof of Proposition 3.1 is the fact the group convolution is an operation from input signals to output signals on the group. As a result, $w^{-1}u$ is a member of the output domain for any $w \in \mathcal{G}$ applied to the input domain. Consequently, a group transformation applied to the input can be reflected by an equivalent transformation on the output.

Now, consider the case in which the output domain is not the group $\mathcal{G}$, but instead an arbitrary subset $\mathcal{S} \subset \mathcal{G}$, e.g., rotations in $[-\frac{\pi}{2}, \frac{\pi}{2}]$. Following the proof of Proposition 3.1 with $u \in \mathcal{S}$, and $v \in \mathcal{G}$, we observe that the operation is equivariant to transformations $w \in \mathcal{G}$ as long as $w^{-1}u$ is a member of $\mathcal{S}$. However, if $w^{-1}u$ does not belong to the output domain $\mathcal{S}$, the output of the operation cannot reflect an equivalent transformation to that of the input, and thus equivariance is not guaranteed. By tuning the size of $\mathcal{S}$, partial group convolutions are able to adjust their level of equivariance (Fig. 1).

Note that equivariance is *only* obtained if Eq. 7 holds for *all* elements in the output domain. That is, if $w^{-1}u$ is a member of $\mathcal{S}$, for all elements $u \in \mathcal{S}$. This is not the case in general as the output domain $\mathcal{S}$ is not necessarily closed under group transformations. Nevertheless, we can precisely quantify how much the output response will change for any input transformation given an output domain $\mathcal{S}$. Intuitively, this difference is given by the difference in the parts of the output feature representation that go in and out of $\mathcal{S}$ by the action of input group transformations. The stronger the transformation and the smaller the size of $\mathcal{S}$, the larger the equivariance difference in the output is (Fig. 1). The formal treatment and derivation are provided in the supplement.

Going from a subset $\mathcal{S}^{(1)}$ to another subset $\mathcal{S}^{(2)}$. Now, consider the case in which the domain of the input and the output are both subsets of the group, i.e., $v \in \mathcal{S}^{(1)}$ and $u \in \mathcal{S}^{(2)}$. Analogous to the previous case, equivariance to input transformations $w \in \mathcal{G}$ holds at positions $u \in \mathcal{S}^{(2)}$ for which $w^{-1}u$ are also members of $\mathcal{S}^{(2)}$. In addition now, the input domain is not restricted to be closed anymore. Consequently, the input function can change for different group transformations.

To see this, consider a partial group convolution defined from an input subset $\mathcal{S}^{(1)}$ to the group $\mathcal{G}$, i.e., $\mathcal{S}^{(2)}{=}\mathcal{G}$ (Fig. 2). Even though the output domain is the group itself, differences in the output feature map can be observed. This results from differences in the input feature map f for group transformations of the input.

Similar to the previous case, we can precisely quantify how much the output response changes for an arbitrary subset $\mathcal{S}^{(1)}$ in the input domain. Intuitively, the difference is given by the change in the parts of the input feature representation that go in and out of $\mathcal{S}^{(2)}$ by the action of the input group transformation. The stronger the transformation and the smaller the size of $\mathcal{S}^{(1)}$, the larger the difference in the output is (Fig. 2). The formal treatment and derivation of this quantity are provided in the supplement.

Special case: lifting convolutions. To conclude, we note that the lifting convolution (Eq. 3) can be seen as an special case of a partial group convolution with $\mathcal{S}^{(1)}{=}\mathcal{X}$ ad $\mathcal{S}^{(2)}{=}\mathcal{G}$. Note however, that the lifting convolution is fully group equivariant. This is because the domain of the input signal $\mathcal{X}$

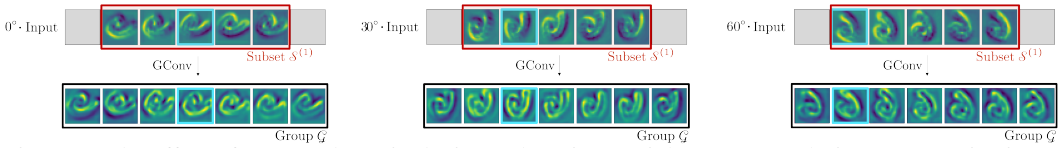

Figure 2: The effect of group subsets in the input domain. Partial group convolutions can receive input functions whose domain is not the group $\mathcal{G}$ but a subset $\mathcal{S}^{(1)}$, e.g., in a mid layer of a Partial G-CNN. Consequently, even if the output domain is the group, i.e., $\mathcal{S}^{(2)}$=$\mathcal{G}$, a partial group convolution does not produce exactly equivariant outputs by design. The difference from exact equivariance in the output response is proportional to the strength of the input transformation and the size of $\mathcal{S}^{(1)}$.

remains equal for all group transformations of the input. Consequently, no input component leaves the subset for any group transformation of the input.

3.4 Learning group subsets via probability distributions on group elements

So far, we have discussed the properties of the partial group convolution in terms of group subsets without specifying how these subsets can be learned. In this section, we describe how to do so by learning a certain probability distribution on the group. We provide specific examples for discrete groups, continuous groups, and combinations thereof.

Vital to our approach is the Monte Carlo approximation to the group convolution presented in Sec. 2:

$$(\psi \hat{*} f)(u_i) = \sum_j \psi(v_j^{-1} u_i) f(v_j) \bar{\mu}_{\mathcal{G}}(v_j).$$

As shown in the supplement, this approximation is equivariant to $\mathcal{G}$ in expectation if the elements in the input and output domain are uniformly sampled from the Haar measure, i.e., $u_i, v_j \sim \mu_{\mathcal{G}}(\cdot)$.[1]

Approach. Our main observation is that we can prioritize sampling specific group elements during the group convolution by learning a probability distribution $\mathrm{p}(u)$ over the elements of the group. When group convolutions draw elements uniformly from the entire group, each group element is drawn with equal probability and thus, the resulting approximation is fully equivariant in expectation. However, we can also define a different probability distribution that draws some samples with larger probability. For instance, we can sample from a predefined region, e.g., rotations in $[-\frac{\pi}{2}, \frac{\pi}{2}]$, by defining a probability distribution $\mathrm{p}(u)$ which is uniform in this range, but zero otherwise. The same idea can be used to forget an equivariance by letting this distribution collapse to a single point, e.g., the identity, along the corresponding group dimension.

In other words, learning a probability distribution $\mathrm{p}(u)$ on the group which is non-zero *only* in a subset of the group can be used to effectively learn this subset. Specifically, we define a probability distribution $\mathrm{p}(u)$ on the output domain of the group convolution in order to learn a subset of the group $\mathcal{S}^{(2)}$ upon which partial equivariance is defined. Note that we only need a distribution on the output domain at each layer. This is because neural networks apply layers sequentially, and thus the subset of the input domain is learned by a distribution on the output domain of the previous layer.

Distributions for one-dimensional continuous groups. We take inspiration from Augerino [2], an use the reparameterization trick [25] to parameterize continuous distributions. In particular, we use the reparameterization trick on the Lie algebra of the group [14] to define a distribution which is uniform over a connected set of group elements $[u^{-1}, \ldots, e, \ldots, u]$, but zero otherwise. To this end, we define a distribution $\mathfrak{u} \cdot [-1, 1]$ with learnable $\mathfrak{u}$ on the Lie algebra $\mathfrak{g}$, and map it to the group via the pushforward of the exponential map $\exp : \mathfrak{g} \to \mathcal{G}$. This give us a distribution which is uniform over a connected set of elements $[u^{-1}, \ldots, e, \ldots, u]$, but zero otherwise.[2]

For instance, we can learn a distribution on the rotation group $\mathrm{SO}(2)$, which is uniform between $[-\theta, \theta]$ and zero otherwise by defining a probability distribution $\theta \cdot [-1, 1]$ with learnable θ on the Lie algebra, and mapping it to the group. If we parameterize group elements as scalars $g \in [-\pi, \pi)$, the exponential map is the identity, and thus $\mathrm{p}(g)$=$\mathcal{U}(\theta \cdot [-1, 1))$. If we sample group elements from this distribution during the calculation of the group convolution, the output domain will only contain elements in $[-\theta, \theta)$ and the output feature map will be partially equivariant.

[1] Finzi et al. [15] show a similar result where u_i and v_j are the same points and thus $v_j \sim \mu_{\mathcal{G}}(\cdot)$ suffices.

[2] Note that an exp-pushforwarded local uniform distribution is locally equivalent to the Haar measure, and thus we can still use the Haar measurement for integration on group subsets.

Distributions for one-dimensional discrete groups. We can define a probability distribution on a discrete group as the probability of sampling from all possible element combinations. For instance, for the mirroring group $\{1, -1\}$, this distribution assigns a probability to each of the combinations $\{0, 0\}$, $\{0, 1\}$, $\{1, 0\}$, $\{1, 1\}$ indicating whether the corresponding element is sampled (1) or not (0). For a group with elements $\{e, g_1, \ldots, g_n\}$, however, this means sampling from 2^{n+1} elements, which is computationally expensive and potentially unstable to train. Instead, we define element-wise Bernoulli distributions over the elements $\{g_1, \ldots, g_n\}$, and learn the probabilities $\{p_i\}$ of sampling the elements $\{g_i\}$. The probability distribution on the group can then be formulated as the joint probability of the element-wise Bernoulli distributions $\mathrm{p}(e, g_1, \ldots, g_n) = \prod_{i=1}^{n} \mathrm{p}(g_i)$.

To learn the element-wise distributions, we use the Gumbel-Softmax trick [24, 31], and use the Straight-Through Gumbel-Softmax to back-propagate through sampling. If all the probabilities equal 1, i.e., $\{p_i{=}1\}$, the group convolution will be fully equivariant. Whenever probabilities start declining, group equivariance becomes partial. In the limit, if all probabilities become zero, i.e., $\{p_i{=}0\}$, then only the identity is sampled and this equivariance is effectively forgotten.

Probability distributions for multi-dimensional groups. There exist several multi-dimensional groups with important applications. Examples are the orthogonal group $\mathrm{O}(2)$ parameterized by rotations and mirroring, or the dilation-rotation group parameterized by scaling and rotations.

For multi-dimensional groups, we construct a probability distribution over the entire group as a combination of *independent probability distributions along each of the group axes*. For a group $\mathcal{G}$ with elements g decomposable along n dimensions $g{=}(d_1, ..., d_n)$, we decompose the probability distribution as: $\mathrm{p}(g){=}\prod_{i=1}^{n} \mathrm{p}(d_i)$, where the probability $\mathrm{p}(d_i)$ is defined given the type of space, i.e., continuous or discrete. For instance, for the orthogonal group $\mathrm{O}(2)$ defined by rotations r and mirroring m, i.e., $g = (r, m)$, $r \in \mathrm{SO}(2), m \in \{\pm 1\}$, we define the probability distribution on the group as $\mathrm{p}(g){=}\mathrm{p}(r) \cdot \mathrm{p}(m)$, where $\mathrm{p}(r)$ is a continuous distribution, and $\mathrm{p}(m)$ is a discrete one.

3.5 Partial Group Equivariant Networks

To conclude this section, we illustrate the general structure of Partial G-CNNs. We build upon the work of Finzi et al. [15] and extend their continuous G-CNNs to discrete groups. This is achieved by combining kernel parameterizations on Lie groups, and actions of discrete groups on the group representation of the kernels. In addition, we replace the isotropic lifting layer of Finzi et al. [15] by a layer with lifting convolutions (Eq. 3). Inspired by Romero et al. [35], we parameterize convolutional kernels as implicit neural representations with SIRENs [37]. The proposed kernel parameterization shows better expressivity and convergence speed, and outperforms ReLU, LeakyReLU and Swish `MLP` parameterizations used so far for continuous G-CNNs, e.g., Schütt et al. [36], Finzi et al. [15]. Our network structure is shown in Fig. 3, and varies only in the number of blocks and channels.

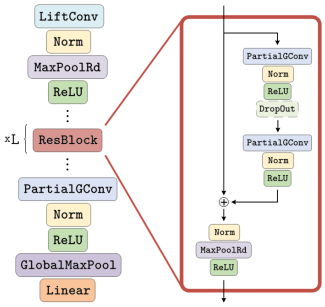

Figure 3: Partial G-CNN architecture.

4 Related work

Group equivariant neural networks. The seminal work of Cohen and Welling [6] has inspired several methods equivariant to many different groups. Existing convolutional methods show equivariance to planar rotations [10, 46, 44], spherical rotations [43, 7, 11–13], scaling [45, 38, 34], and more general symmetry groups [32, 3, 17]. Group equivariant self-attention has also been proposed [18, 33, 22]. Common to these approaches is that they are fully equivariant and the group must be fixed prior to training. In contrast, Partial G-NNs learn their level of equivariance from data and can represent full, partial and no equivariances.

Invariance learning. Learning the right amount of global invariance from data has been explored by learning a probability distribution over continuous test-time augmentations [2], or by using the marginal likelihood of a Gaussian process [40, 39]. Contrary to these approaches, Partial G-CNNs aim to learn the right level of equivariance at every layer and do not require additional loss terms. Partial G-CNNs relate to Augerino [2] in the form that probability distributions are defined on continuous groups. However, Partial G-CNNs are intended to learn partial layer-wise equivariances and are able to learn probability distributions on discrete groups. There also exist learnable data augmentation strategies [30, 19, 29, 4] that can find transformations of the input that optimize the

Table 1: Results on MNIST6-180 and MNIST6-M.

Base Group	Dataset	G-CNN	Partial G-CNN
$\mathrm{SE}(2)$	MNIST6-180	50.0	**100.0**
Mirroring	MNIST6-M	50.0	**100.0**
$\mathrm{E}(2)$	MNIST6-180	50.0	**100.0**
	MNIST6-M	50.0	**100.0**

Table 2: Augerino vs. Partial G-CNNs. Terms in parentheses show accuracy of Partial G-CNNs.

Base Group	No. Elems	Classification Accuracy (%)		
		RotMNIST	CIFAR10	CIFAR100
$\mathrm{SE}(2)$	8	99.17 (**99.23**)	82.38 (**88.59**)	52.98 (**57.26**)
	16	**99.25** (99.18)	82.53 (**88.59**)	51.47 (**57.31**)
$\mathrm{E}(2)$	8	**99.12** (97.78)	84.29 (**89.00**)	52.59 (**55.22**)
	16	**99.18** (98.35)	83.54 (**90.12**)	54.76 (**61.46**)

task loss. We can also view Augerino as learning a smart data augmentation which we compare with. In contrast, our work tries to find optimal partial equivariances at each layer.

Equivariance learning. Learning equivariant mappings from data has been explored by meta-learning of weight-tying matrices encoding symmetry equivariances [47, 1] and by learning the Lie algebra generators of the group jointly with the parameters of the network [9]. These approaches utilize the same learned symmetries across layers. MSR [47] is only applicable to (small) discrete groups, and requires long training times. L-Conv [9] is only applicable to continuous groups and is not fully compatible with current deep learning components, e.g., pooling, normalization. Unlike these approaches, Partial G-CNNs can learn levels of equivariance at every layer, are fully compatible with current deep learning components, and are applicable for discrete groups, continuous groups and combinations thereof. We note, however, that Zhou et al. [47], Dehmamy et al. [9] learn the structure of the group from scratch. Contrarily, Partial G-CNNs start from a (very) large group and allows layers in the network to constrain their equivariance levels to better fit the data.

Concurrently, Finzi et al. [16] incorporate soft equivariance constraints by combining outputs of equivariant and non-equivariant layers running in parallel. Differently, Partial G-CNNs aim to learn from data the optimal amount of partial equivariance directly on the group manifold.

5 Experiments

Experimental details. We parameterize all our convolutional kernels as 3-layer SIRENs [37] with 32 hidden units. All our networks are constructed with two residual blocks of 32 channels each, batch normalization [23] and follow the structure shown in Fig. 3. Our networks are intentionally selected to be simple to better assess the effect of partial equivariance. We avoid learning probability distributions on the translation part of the considered groups, and assume all spatial positions to be sampled. This allows us to use fast `PyTorch` convolution primitives in our implementation.

Toy tasks: MNIST6-180 and MNIST6-M. First, we validate whether Partial G-CNNs can learn partial equivariances. To this end, we construct two illustrative datasets: *MNIST6-180*, and *MNIST6-M*. *MNIST6-180* is constructed by extracting the digits of the class 6 from the MNIST dataset [28], and rotating them on the circle. The goal is to predict whether the number is a six, i.e., a rotation in $[-90°, 90°]$ was applied, or a nine otherwise. Similarly, we construct *MNIST6-M* by mirroring digits over the y axis. The the goal is to predict whether a digit was mirrored or not.

As shown in Tab. 1, G-CNNs are unable to solve these tasks as discrimination among group transformations is required. Specifically, $\mathrm{SE}(2)$-CNNs are unable to solve MNIST6-180, and Mirror-CNNs are unable to solve MNIST6-M. Furthermore, $\mathrm{E}(2)$-CNNs cannot solve any of the two tasks, because $\mathrm{E}(2)$-CNNs incorporate equivariance to both rotations and mirroring. Partial G-CNNs, on the other hand, easily solve both tasks with corresponding base groups. This indicates that Partial G-CNNs learn to adjust the levels of equivariance of the networks in order to solve the tasks.

We verify the learned levels of equivariance for a Partial $\mathrm{SE}(2)$-CNN on MNIST6-180. To this end, we plot the probability of assigning the label 6 to test samples of MNIST6-180 rotated on the whole circle. Fig 4 shows that the network learns to predict "6" for rotated samples in $[-90°, 90°]$, and "9" otherwise. Note that Partial G-CNNs learn the expected levels of partial equivariance without adding regularization loss terms to encourage them as in Benton et al. [2].

Benchmark image datasets. Next, we validate Partial G-CNNs on classification datasets: RotMNIST [27], CIFAR-10 and CIFAR-100 [26]. Results on PatchCam [41] can be found in the supplement.

We construct Partial G-CNNs with base groups $\mathrm{SE}(2)$ and $\mathrm{E}(2)$, and varying number of elements used in the Monte Carlo approximation of the group convolution and compare them to equivalent G-CNNs and ResNets (equivalent to $\mathrm{T}(2)$-CNNs). Our results (Tab. 3) show that Partial G-CNNs are

Table 3: Accuracy on vision benchmark datasets.

Base Group	No. Elems	Partial Equiv.	Classification Accuracy (%)		
			RotMNIST	CIFAR10	CIFAR100
T(2)	1	-	97.23	83.11	47.99
SE(2)	4	✗	99.10	83.73	52.35
		✓	**99.13**	**86.15**	**53.91**
	8	✗	99.17	86.08	55.55
		✓	**99.23**	**88.59**	**57.26**
	16	✗	**99.24**	86.59	51.55
		✓	99.18	**89.11**	**57.31**
E(2)	8	✗	**98.14**	85.55	54.29
		✓	97.78	**89.00**	**55.22**
	16	✗	98.35	88.95	57.78
		✓	**98.58**	**90.12**	**61.46**

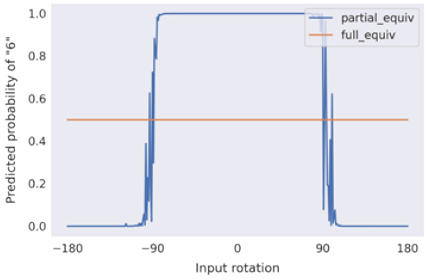

Figure 4: Learned equivariances for a "6" on MNIST6-180. Partial G-CNNs become equivariant to rotations on the semi-circle. Regular G-CNNs cannot solve the task as they cannot set group transformations apart.

competitive with G-CNNs where full-equivariance is needed (rotated MNIST and PatchCamelyon). However, for tasks in which the data does not naturally exhibit full rotation equivariance (CIFAR-10 and CIFAR-100), Partial G-CNNs consistently outperform fully equivariant G-CNNs.

The need for learning partial layer-wise equivariances. Next, we evaluate (*i*) the effect of learning partial equivariances instead of soft invariances, and (*ii*) the effect of learning layer-wise levels of equivariances instead of a single level of partial equivariance for the entire network.

For the former, we compare Partial G-CNNs to equivalent ResNets with Augerino [2] (Tab. 2). We extend our strategy to learn distributions on discrete groups to the Augerino framework to allow it to handle groups with discrete components, e.g., $\mathrm{E}(2)$. For the latter, we construct regular G-CNNs and replace either the final group convolutional layer by a $\mathrm{T}(2)$ convolutional layer, or the global max pooling layer at the end by a learnable MLP (Tab. 4). If determining the level of equivariance at the end of the network is sufficient, these models should perform comparably to Partial G-CNNs.

Tab. 2 shows that Augerino is competitive to Partial G-CNNs on rotated MNIST, but falls behind by a large margin on CIFAR-10 and CIFAR-100. This result can be explained by how these datasets are constructed. RotMNIST is constructed by rotating MNIST digits globally, thus it is not surprising that a model able to encode global invariances can match Partial G-CNNs. The invariance and equivariance relationships in natural images, however, can be local as well. Consequently, tackling different levels of equivariance at each layer using Partial G-CNNs leads to benefits over using single level of global invariance for the entire network.

In Tab. 4, we observe that Partial G-CNNs perform consistently better than the alternatives. This indicates that modifying the level of equivariance at the end of the model is not sufficient and that learning layer-wise partial equivariances is beneficial. Moreover, Partial G-CNNs can become fully-, partial-, and non-equivariant during training, whereas the alternative models are either unable to become fully equivariant ($\mathrm{T}(2)$ models) or very unlikely to do so in practice (MLP models).

SIRENs as group convolution kernels. Next, we validate SIRENs as parameterization for group convolutional kernels. Tab. **??** shows that $\mathrm{SE}(2)$-CNNs with SIREN kernels outperform $\mathrm{SE}(2)$-CNNs with ReLU, $\mathrm{LeakyReLU}$ and Swish kernels by a large margin on all datasets considered.

6 Discussion

Memory consumption in partial equivariant settings. G-CNNs fix the number of samples used to approximate the group convolution prior to training. In Partial G-CNNs we fix a maximum number of samples and adjust the number used at every layer based on the group subset learned. Consequently, a partial group convolution with a learned distribution $\mathrm{p}(u)$=$\mathcal{U}(\frac{\pi}{2}[-1, 1])$ uses half of the elements used in a corresponding group convolution. This reduction in memory and execution time leads to improvements in training and inference time for Partial G-CNNs on partial equivariant settings. We observe reductions up to 2.5× in execution time and memory usage on CIFAR-10 and CIFAR-100.

Better kernels with implicit neural representations. Inspired by Romero et al. [35], we replace $\mathrm{LeakyReLU}$, ReLU and Swish kernels used so far for continuous group convolution kernels with a SIREN [37]. Our results show that SIRENs are better at modelling group convolutional kernels.

Table 4: Accuracy of G-CNNs with a MLP instead of global pooling, G-CNNs with a final $\mathrm{T}(2)$-conv. layer, and our proposed Partial G-CNNs.

Base Group	No. Elems	Net Type	Classification Accuracy (%)		
			RotMNIST	CIFAR10	CIFAR100
SE(2)	16	T(2)	99.04	82.76	52.51
		MLP	99.00	86.25	56.29
		Partial	**99.18**	**87.45**	**57.31**
E(2)	16	T(2)	97.98	86.68	57.61
		MLP	**99.02**	87.43	58.87
		Partial	98.58	**90.12**	**61.46**

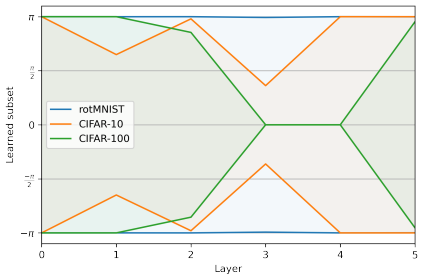

Figure 5: Example group subsets learned by Partial G-CNNs.

Sampling per batch element. In our experiments, we sample once from the learned distribution $\mathrm{p}(u)$ at every layer, and use this sample for all elements in the batch. A better estimation of $\mathrm{p}(u)$ can be obtained by drawing a sample for every batch element. Though this method may lead to faster convergence and better estimations of the learned distributions, it comes at a prohibitive memory cost. Consequently, our results are restricted to a single sample per layer.

Going from a small group subset to a larger one, e.g., the group $\mathcal{G}$. What does it mean and why is it advantageous? In Sec. 3.3 we described that a partial group convolution can go from a group subset $\mathcal{S}^{(1)}$ to a larger group subset $\mathcal{S}^{(2)}$, e.g., the whole group $\mathcal{G}$. Nevertheless, once partial equivariance is chosen, full-equivariance cannot reappear even for $\mathcal{S}^{(2)}$=$\mathcal{G}$. Interestingly, we observe that Partial G-CNNs often learn to disrupt equivariance halfway in the network, and return to the whole group afterwards (Fig. 5). Doing so actually brings advantages as described below.

Full equivariance restricts group convolutions to apply the same mapping on the entire group. As a result, once the input is transformed, the output remains equal up to the same group transformations. In partial equivariance settings, Partial G-CNNs output different feature representations for different input transformations. Consequently, Partial G-CNNs can use the group dimension to encode different mappings. Specifically, some kernel values can be used for some rotations, and different ones can be used for others. This means that when Partial G-CNNs go back to a larger group subset from a smaller one, they are able to use the group axis to encode additional transformation-dependent features. This results in increased expressivity of the partial equivariant model.

In the supplement, we also test the effect of enforcing that the group subsets cannot become larger as the depth increases. The corresponding results show that enforcing such monotonicity actually produces slightly worse results compared to the unconstrained case.

7 Limitations and future work

Partial equivariances for other group representations. The theory of learnable partial equivariances proposed here is only applicable to architectures using regular group representations, e.g., Cohen and Welling [6], Romero and Cordonnier [33]. Nevertheless, other type of representations exist which provide exact equivariance to continuous groups: irreducible representations [46, 44, 42]. Extending partial equivariances to irreducible representations is a valuable extension of our work. However, this is a non-trivial endeavor which we must leave for future work.

Unstable training on discrete groups. Although we can model partial equivariance on discrete groups with our proposed discrete probability distribution parameterization, we observed that these distributions were unstable to train. Finding a good way of learning discrete distributions is an active field of research [20, 21], and advances in this field could be used to define more stable training of partial equivariances on discrete groups.

Future work. Our method relies on a good parameterization of the probability distribution defined over group elements. Finding better representations and learning methods for these distributions, specially for distributions on discrete variables, is an important direction for future research. Other interesting avenues include the extension of partial equivariances to irreducible group representations, and the definition of input-dependent partial equivariances.

References

[1] F. Alet, D. Doblar, A. Zhou, J. Tenenbaum, K. Kawaguchi, and C. Finn. Noether networks: meta-learning useful conserved quantities. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] G. Benton, M. Finzi, P. Izmailov, and A. G. Wilson. Learning invariances in neural networks from training data. *Advances in Neural Information Processing Systems*, 33:17605–17616, 2020.

[3] A. Bogatskiy, B. Anderson, J. Offermann, M. Roussi, D. Miller, and R. Kondor. Lorentz group equivariant neural network for particle physics. In *International Conference on Machine Learning*, pages 992–1002. PMLR, 2020.

[4] E. Chatzipantazis, S. Pertigkiozoglou, E. Dobriban, and K. Daniilidis. Learning augmentation distributions using transformed risk minimization. *arXiv preprint arXiv:2111.08190*, 2021.

[5] S. Chen, E. Dobriban, and J. H. Lee. A group-theoretic framework for data augmentation. *Journal of Machine Learning Research*, 21(245):1–71, 2020.

[6] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[7] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical cnns. In *International Conference on Learning Representations*, 2018.

[8] T. S. Cohen, M. Geiger, and M. Weiler. A general theory of equivariant cnns on homogeneous spaces. *Advances in neural information processing systems*, 32, 2019.

[9] N. Dehmamy, R. Walters, Y. Liu, D. Wang, and R. Yu. Automatic symmetry discovery with lie algebra convolutional network. *Advances in Neural Information Processing Systems*, 34, 2021.

[10] S. Dieleman, J. De Fauw, and K. Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *International conference on machine learning*, pages 1889–1898. PMLR, 2016.

[11] C. Esteves, A. Sud, Z. Luo, K. Daniilidis, and A. Makadia. Cross-domain 3d equivariant image embeddings. In *International Conference on Machine Learning*, pages 1812–1822. PMLR, 2019.

[12] C. Esteves, Y. Xu, C. Allen-Blanchette, and K. Daniilidis. Equivariant multi-view networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1568–1577, 2019.

[13] C. Esteves, A. Makadia, and K. Daniilidis. Spin-weighted spherical cnns. *Advances in Neural Information Processing Systems*, 33:8614–8625, 2020.

[14] L. Falorsi, P. de Haan, T. R. Davidson, and P. Forré. Reparameterizing distributions on lie groups. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3244–3253. PMLR, 2019.

[15] M. Finzi, S. Stanton, P. Izmailov, and A. G. Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176. PMLR, 2020.

[16] M. Finzi, G. Benton, and A. G. Wilson. Residual pathway priors for soft equivariance constraints. *Advances in Neural Information Processing Systems*, 34, 2021.

[17] M. Finzi, M. Welling, and A. G. Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International Conference on Machine Learning*, pages 3318–3328. PMLR, 2021.

[18] F. Fuchs, D. Worrall, V. Fischer, and M. Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

[19] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama. Faster autoaugment: Learning augmentation strategies using backpropagation. In *European Conference on Computer Vision*, pages 1–16. Springer, 2020.

[20] E. Hoogeboom, J. Peters, R. Van Den Berg, and M. Welling. Integer discrete flows and lossless compression. *Advances in Neural Information Processing Systems*, 32, 2019.

[21] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling. Argmax flows and multinomial diffusion: Towards non-autoregressive language models. *arXiv preprint arXiv:2102.05379*, 2021.

[22] M. J. Hutchinson, C. Le Lan, S. Zaidi, E. Dupont, Y. W. Teh, and H. Kim. Lietransformer: equivariant self-attention for lie groups. In *International Conference on Machine Learning*, pages 4533–4543. PMLR, 2021.

[23] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[24] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[25] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[26] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[27] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480, 2007.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Y. Li, G. Hu, Y. Wang, T. M. Hospedales, N. M. Robertson, and Y. Yang. DADA: differentiable automatic data augmentation. 2020.

[30] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim. Fast autoaugment. *Advances in Neural Information Processing Systems*, 32, 2019.

[31] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[32] D. Romero, E. Bekkers, J. Tomczak, and M. Hoogendoorn. Attentive group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 8188–8199. PMLR, 2020.

[33] D. W. Romero and J.-B. Cordonnier. Group equivariant stand-alone self-attention for vision. In *International Conference on Learning Representations*, 2020.

[34] D. W. Romero, E. J. Bekkers, J. M. Tomczak, and M. Hoogendoorn. Wavelet networks: Scale equivariant learning from raw waveforms. *arXiv preprint arXiv:2006.05259*, 2020.

[35] D. W. Romero, A. Kuzina, E. J. Bekkers, J. M. Tomczak, and M. Hoogendoorn. Ckconv: Continuous kernel convolution for sequential data. 2020.

[36] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *arXiv preprint arXiv:1706.08566*, 2017.

[37] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[38] I. Sosnovik, M. Szmaja, and A. Smeulders. Scale-equivariant steerable networks. 2020.

[39] T. F. van der Ouderaa and M. van der Wilk. Learning invariant weights in neural networks. *Workshop in Uncertainty & Robustness in Deep Learning, ICML*, 2021.

[40] M. van der Wilk, M. Bauer, S. John, and J. Hensman. Learning invariances using the marginal likelihood. *Advances in Neural Information Processing Systems*, 31, 2018.

[41] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pages 210–218. Springer, 2018.

[42] M. Weiler and G. Cesa. General e (2)-equivariant steerable cnns. *Advances in Neural Information Processing Systems*, 32, 2019.

[43] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, 2018.

[44] M. Weiler, F. A. Hamprecht, and M. Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018.

[45] D. Worrall and M. Welling. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 32, 2019.

[46] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

[47] A. Zhou, T. Knowles, and C. Finn. Meta-learning symmetries by reparameterization. In *International Conference on Learning Representations*, 2020.

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]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (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]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (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]
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? [Yes]
   (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? [N/A]

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]