Learning symmetries via weight-sharing with doubly stochastic tensors

Putri A. van der Linden [1] Alejandro García Castellanos [1] Sharvaree Vadgama [1] Thijs P. Kuipers [2]
Erik J. Bekkers [1]

Abstract

Traditional group equivariant methods presuppose known groups, an assumption that can be unrealistic for real-world datasets and potentially too restrictive for neural network architectures. Typically, equivariance in neural networks is implemented through group transformations applied to a canonical weight tensor, facilitating weight sharing across a specified group G. In this study, we introduce a method to *learn* such weight-sharing schemes. Our approach involves developing a set of learnable, doubly stochastic matrices that function as soft permutation matrices on canonical weight tensors, accommodating regular group representations as a specific instance. This allows for adaptive kernel transformations that are optimized in conjunction with downstream tasks. Our results demonstrate that when datasets display pronounced symmetries, the learned permutation matrices approximate regular group representations, effectively transforming our weight-sharing networks into standard group convolutional networks.

1. Introduction

Equivariance in deep learning models has proven to be an effective and important inductive bias in various tasks. Constraining the function space that adheres to specific symmetries has been shown to improve generalization and parameter efficiency (Bronstein et al., 2021; Cohen & Welling, 2016; Hoogeboom et al., 2022; Bekkers et al., 2024). The most seminal example of effective equivariance is that of convolutional neural networks (CNNs): which achieve translation equivariance by translating learnable kernels to every position in the input. This design ensures that the weights defining the kernels are shared across all translations, so that if the input is translated, the output features are correspondingly translated; in other words, *equivariance* is achieved through *weight sharing*. In the seminal work by Cohen & Welling (2016), this concept was extended to generalize weight-sharing under any discrete group of symmetries, resulting in the group-equivariant CNN (G-CNN). G-CNNs enable G-equivariance to a broader range of symmetries, such as rotation, reflection, and scale (Cohen & Welling, 2016; Bekkers et al., 2018; Worrall & Welling, 2019; Sosnovik et al., 2021). However, the impact of G-CNNs is closely tied to the presence of specific inductive biases in the data. Yet, for many types of data, including natural images and sequence data, exact symmetries are not present or known, leading to overly constrained models that can suffer in performance (Wang et al., 2022; Romero & Lohit, 2021; Allingham et al., 2024; van der Ouderaa et al., 2023).

In this work, we tackle the challenge of specifying group symmetries upfront by introducing a general weight-sharing scheme. Our method can represent G-CNNs as a special case but is not limited to exact equivariance constraints, offering greater flexibility in handling various symmetries in the data. Inspired by the idea that group equivariance for finite groups can be achieved through weight-sharing patterns on a set of base weights (Ravanbakhsh et al., 2017), we propose learning the symmetries directly from the data on a per-layer basis, requiring no prior knowledge of the possible symmetries.

We leverage the fact that regular group representations act as permutations and that the expectation of random variables defined over this set of permutations is a double stochastic matrix (Birkhoff, 1946). This implies that regular partial group transformation can be approximated by a stack of doubly stochastic matrices which essentially act as (soft) permutation matrices. Consequently, we learn a set of double stochastic matrices through the Sinkhorn operator (Sinkhorn, 1964), resulting in weight-sharing under learnable group transformations. We summarize our contributions as follows:

1. We propose a novel weight-sharing scheme that can adapt to group actions when certain symmetry transformations are present in the data, enhancing model flexibility and

[1]University of Amsterdam, Amsterdam, Netherlands [2]Amsterdam UMC, Amsterdam, Netherlands. Correspondence to: Putri A. van der Linden <p.a.vanderlinden@uva.nl>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at 41st International Conference on Machine Learning, Vienna, Austria. PMLR Vol Number, 2024. Copyright 2024 by the author(s).

performance.

2. We present empirical results on augmented MNIST and CIFAR10, demonstrating the effectiveness of our approach in learning relevant weight-sharing schemes when clear symmetries are present.

3. We provide a thorough analysis of the learned symmetries and their impact on downstream tasks, showcasing the practical benefits of our method.

2. Background

We begin by revisiting group convolutional methods in the context of image processing, followed by their relation to weight-sharing schemes. Some basic familiarity with group theory and representations is assumed and the relevant preliminaries are covered in A.1.

Group convolution Consider feature maps $f : X \to \mathbb{R}^D$ over some domain on which a group action is defined, i.e., over a *G-space*. E.g., for images (signals over $X = \mathbb{R}^2$) we could consider the group $G = (\mathbb{R}^2, +)$ of translations, which acts on X via $gx = x + y$, with $g = (y)$ a translation by $y \in \mathbb{R}^2$. While the group G merely defines how two transformations $g, h \in G$ applied one after the other correspond to a net translation $gh \in G$, a representation ρ concretely describes how data is transformed. E.g., signals $f : X \to \mathbb{R}$ can be transformed via the *left-regular representation* $[\rho(g)f](x) := f(g^{-1}x)$, which in the case of images and the translation group is given by $[\rho(g)f](x) = f(x-y)$.

In general, group convolution is defined as transforming a base kernel under every possible group action, and for every transformation take the inner product with the underlying data via

$$\boxed{\text{G conv inner product form:}} \quad (k \star_G f)(g) = \langle \rho(g)k, f \rangle \,, \quad (1)$$

with $\langle \cdot, \cdot \rangle$ denoting the inner product. For images, which are essential functions over the group $G = (\mathbb{R}^2, +)$, and taking $\langle k, f \rangle := \int_X k(x)f(x)\mathrm{d}x$ the standard inner product, Eq. (1) boils down to the standard *cross-correlation* operator[1]:

$$\boxed{G\text{ conv integral form:}}$$

$$(k \star f)(g) = \int_G k(g^{-1}h)f(h)\mathrm{d}h \qquad (2)$$

$$\boxed{\text{standard } (\mathbb{R}^2, +) \text{ conv:}}$$

$$(k \star f)(x) = \int_{\mathbb{R}^2} k(x' - x)f(x')\mathrm{d}x' \,. \quad (3)$$

Semi-direct product groups When equivariance to larger symmetry groups is desired, e.g. in the case of $G = SE(2)$ roto-translation equivariance for images with domain $X = \mathbb{R}^2$, a *lifting convolution* can be used to generate signals over the group G. In essence it is still of the form of (2), however integration is over X instead of over G:

$$\boxed{G\text{ lifting conv:}}$$

$$(k \star f)(g) = \int_X k(g^{-1}x)f(x)\mathrm{d}x \qquad (4)$$

$$\boxed{SE(2)\text{ lifting conv:}}$$

$$(k \star f)(x, \mathbf{R}) = \int_{\mathbb{R}^2} k(\mathbf{R}^{-1}(x' - x))f(x')\mathrm{d}x' \,, \qquad (5)$$

with $g = (x, \mathbf{R}) \in (\mathbb{R}^2, +) \rtimes SO(2)$. The roto-translation group is an instance of a semi-direct product group (denoted with $\rtimes$) between the translation and rotation group, which has the practical benefit that a stack of rotated kernels can be precomputed (Cohen & Welling, 2016), and the translation part efficiently be taken care of via optimized `Conv2D` operators. Namely via $(k \star f)(x, \mathbf{R}_i) = \texttt{Conv2D}[k_i, f]$, with $k_i := k(\mathbf{R}_i^{-1}x)$. This trick can also be applied for full group convolutions (2).

3. Method

Our goal is to identify the underlying symmetries in datasets where exact symmetries are unknown, ensuring parameter efficiency and avoiding strict group constraints. We revisit regular representations and their essential function in formulating the group convolution equation (1). Transitioning from theoretical constructs to practical implementations, we interpret regular representations as permutations to derive a weight-sharing scheme from a base vector of weights. Our approach, characterized by the use of doubly stochastic matrices, lays the groundwork for developing adaptive weight-sharing layers that intuitively learn dataset symmetries. We cover a positioning of our work within existing literature in Appendix C.

Weight-sharing through learnable representations To achieve weight-sharing over a finite set of symmetries, we define *learnable representations* $\rho : G \to GL(V)$. Specifically, we assign a learnable transformation (permutation matrix) to each element in G. It is important to note that we refer to G and ρ as a "group" and "representation" in a loose sense, as we relax the homomorphism property and do not initially endow G with a group product. Consequently, the collection of linear transformations does not form a group representation a priori. However, our proposed method is capable of modeling this structure in principle.

Regular representations as permutations Consider the case of a continuous group G, e.g. of rotations $SO(2)$, and a real signal $f : G \to \mathbb{R}$ over it. This signal is to be considered an *infinite dimensional vector* with continuous

[1]Due to the equivalence between convolution and correlation via kernel reflection, we henceforth simply refer to operators of the type of (2) as convolution even though technically they are cross-correlations.

"indices" $g \in G$ that index the vector elements $f(g)$. To emphasize the resemblance of the regular representation to permutation matrices we write it as

$$\boxed{\text{Reg. repr. in integral form:}}\ [\rho(g)f](i) = \int_G P_g(i,h)f(h)\mathrm{d}h\,,$$

with $P_g(i,h) = \delta_{g^{-1}i}(h)$ a kernel that for each g maps h to a new "index" $i \in G$, where the Dirac delta essentially codes for the group product as $\delta_{g^{-1}i}(h)$ is non-zero only if $g^{-1}i = h \Leftrightarrow g \cdot h = i$.

The discrete counterpart of such an integral transform is matrix-vector multiplication with a matrix $\rho(g) = \mathbf{P}_g$ with entries $P_{ghi} = 1$ if $gh = i$ and zero otherwise. As a concrete example, consider a signal $f : G \to \mathbb{R}$ over the discrete group $G = C_4$ of cyclic permutations of size 4, i.e., a periodic signal of length four, then we could vectorize it as $\mathbf{v}_g = f(g)$ and the regular representation becomes a simple permutation matrix with

$$\mathbf{P}_0 = \begin{pmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1 \end{pmatrix},\ \mathbf{P}_1 = \begin{pmatrix} 0&0&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0 \end{pmatrix},$$

$$\mathbf{P}_2 = \begin{pmatrix} 0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0 \end{pmatrix},\ \mathbf{P}_3 = \begin{pmatrix} 0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0 \end{pmatrix}.$$

We refer to the collection of permutation matrices $\mathbf{P} \in [0,1]^{|G|\times|G|\times|G|}$ as the **permutation tensor**.

Recall that the regular convolution operator (2) is not only defined for signals over groups G but for G-spaces X, in general, as in Eq. (4). It requires a representation that acts on signals (convolution kernels) over X. However, since the group G acts by automorphisms (bijections) on the space X, we can define the regular representation—as before—using permutation integral with $P_g(x,x') = \delta_{g^{-1}x}(x')$ that effectively sends old "indices" x' to their new location x, or concretely via permutation matrices $\mathbf{P}$ of shape $|G| \times |X| \times |X|$, where X is the domain of the signal that is transformed—which can also be G.

In practice, it is common that we perform a discretization of continuous signals f, e.g., when we work with images, we discretize the continuous signal $f : \mathbb{R}^2 \to \mathbb{R}^C$ to $\mathbf{f} : \mathbb{Z}^2 \to \mathbb{R}^C$. Therefore, the group action over the discretized signal should approximate the group action over the continuous signal. We can see that in some cases, the group representation of the action on the discrete signal can still be implemented using permutation matrices. E.g., 90° rotations (in C_4) applied to images merely permute the pixels. However, for finer discretizations, e.g. using 45° rotations, interpolation can be used as a form of approximate permutations (Lafarge et al., 2021).

Weight sharing In a discrete group setting, we then see that convolution (1) is obtained by multiplications with matrices obtained by stacking permuted base kernel weights $\mathbf{w} \in \mathbb{R}^{|X|}$

$$\boxed{\text{Discrete } G \text{ conv:}}\ \mathbf{f}^{out} = \mathbf{W}\mathbf{f}^{in}\,, \tag{6}$$

$$\text{with} \quad \mathbf{W} = \mathbf{P}\mathbf{w} := \begin{pmatrix} (\mathbf{P}_0\mathbf{w})^T \\ (\mathbf{P}_1\mathbf{w})^T \\ \vdots \end{pmatrix} \in \mathbb{R}^{|G|\times|X|}.$$

E.g., a group convolution over C_4 is implemented with a matrix of the form $\mathbf{W} = \begin{pmatrix} w_0&w_1&w_2&w_3\\w_3&w_0&w_1&w_2\\w_2&w_3&w_0&w_1\\w_1&w_2&w_3&w_4 \end{pmatrix}$. More generally, we can model (partial) equivariance through relaxation of the permutation constraint, by instead employing *double stochasticity*. A more thorough motivation can be found in Appendix B.

Regular representations allow for element-wise activations An important practical element of using regular representations to define group convolutions is that permutations commute with element-wise activations, namely, $[\rho(g)\sigma(f)](i) = \sigma(f)(g^{-1}i) = \sigma(f(g^{-1}i)) = \sigma([\rho(g)f](i))$. In contrast, steerable methods—based on irreducible representations (cf. App. A.4)—require specialized activation functions so as not to break group equivariance. Such activations in practice are not as effective as the classic element-wise activations such as ReLU (Weiler & Cesa, 2019). Hence, when learning weight-sharing schemes—as is our objective—it is preferred to achieve weight-sharing using regular representations without the risk of breaking equivariance by using standard activation functions.

Weight Sharing Convolutional Neural Networks Now everything is in place to define our *Weight Sharing Convolutional Neural Networks* (WSCNNs). We let $\Theta_i^l \in \mathbb{R}^{|X|\times|X|}$ be a collection of learnable parameters that parametrize the *representation stack* of the layer l as $\mathbf{R}^l = (S^K(\Theta_0^l), S^K(\Theta_1^l), ..., S^K(\Theta_N^l))^T \in [0,1]^{|G|\times|X|\times|X|}$. I.e., we parameterize this tensor as a stack of $|G|$ approximate doubly stochastic $|X|$-dimensional matrices, wherein stochasticity is enforced via K applications of the Sinkhorn operator (see Appendix A.3). We also define a set of *learnable base weights* $\boldsymbol{\theta}^l \in \mathbb{R}^{|X|\times C_{out}\times C_{in}}$. The WSCNN layer is then simply given by Eq. (6) with $\mathbf{P}$ and $\mathbf{w}$ respectively replaced by $\mathbf{R}^l$ and $\boldsymbol{\theta}^l$.

We further note that on image data $|X|$ can be large, making the discrete matrix form implementation computationally demanding. Hence, we consider semi-direct product group parametrizations for G, in which we let G be of the form $(\mathbb{R}^n, +) \rtimes H$, with H a learnable (approximate) group. Then, the representation stacks will merely be of shape $|H| \times |X'| \times |X'|$, with $|H| \ll |G|$ the size of the sub-group and $|X'|$ the number of pixels that support the convolution kernel. Hence, in the context of 2D images with convolution kernels of size k, $|X'| = k^2$. A WSCNN layer is then efficiently implemented via a $\texttt{Conv2D}[\mathbf{f}, \mathbf{R}^l\boldsymbol{\theta}^l]$. For group

Table 1. MNIST with different transformations.

Model	Sharing	Test Acc. Rot. MNIST	Test Acc. ScaledMNIST
CNN	Z_2	98.48 ± 0.08	$\mathbf{99.30 \pm 0.01}$
GCNN	$Z_2 \rtimes C_4$	$\mathbf{98.96 \pm 0.14}$	97.50 ± 0.15
WSCNN + norm	learned	97.56 ± 0.07	99.27 ± 0.04
WSCNN + norm + ent	learned	98.04 ± 0.11	99.24 ± 0.01

Table 2. CIFAR10 results.

Model	# Elem.	Test Acc.
CNN 32	-	70.50 ± 0.62
CNN 64	-	76.29 ± 0.57
GCNN	4	76.72 ± 0.26
WSCNN + norm	4	$\mathbf{78.80 \pm 0.46}$
WSCNN + norm + ent	4	76.80 ± 1.40

convolutions (after the lifting layer) the representation stacks will be of shape $|H| \times (|X'| \times |H|) \times (|X'| \times |H|)$.

4. Experiments

We first demonstrate that the proposed weight sharing method can effectively pick up on useful weight sharing patterns when trained on image datasets with different equivariance priors. Training setup, regularizers (norm and ent), model architecture, and model sizes can be found in Appendix D. We then proceed to further analyze the learned weight-sharing structures on a suite of toy datasets.

4.1. Image datasets and equivariance priors

We tested our weight-sharing scheme's ability to recognize data symmetries on augmented datasets, specifically evaluating on rotated and scaled MNIST images, and CIFAR-10 with no added train-time augmentation. MNIST, altered by rotations (full $SO(2)$) and scaling (factors between $[0.3, 1.0]$), represents datasets with known symmetries, whereas CIFAR-10 represents unknown symmetries.

Performance comparisons were made against two models: a C4-group convolutional model, which directly encodes a subgroup of MNIST's symmetries, and a standard CNN with double the channels and no symmetry constraints. The group convolutional model (GCNN), with fixed weight sharing, and the standard CNN, lacking any predefined inductive biases, framed the evaluation spectrum, with our weight-sharing CNN (WSCNN) introducing a learnable, semi-flexible weight-sharing approach.

In cases of misalignment between model and data symmetries, such as the C4-GCNN on scaled MNIST, performance suffered. However, our WSCNN consistently achieved good performance across all datasets, indicating its ability to pick up on relevant weight-sharing patterns without needing fixed group specifications.

Visual inspections and analysis on the rotatedMNIST experiment further confirmed the WSCNN's ability to adapt to data symmetries, with learned kernels effectively rotating in alignment with the data's underlying patterns (see Appendix F.1 for learned transformations applied to kernels), and the representation stack often resembling C_4 permutations, as illustrated in Fig 1).

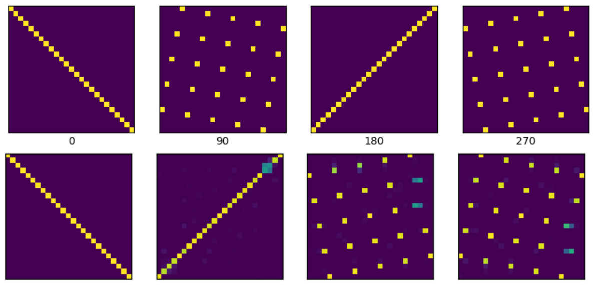


Figure 1. Comparison of C_4 representations and the representation stack learned by the lifting layer on the rotated MNIST dataset.

Table 2 presents the CIFAR-10 results. Notably, when maintaining an equal number of channels, rotational equivariance significantly enhances performance compared to a conventional CNN (referred to as CNN 32) with the same channel capacity. Moreover, our model with a learned weight-sharing scheme surpasses both approaches, even when the CNN's channel dimension is doubled to increase the number of free kernels (referred to as CNN 64). This suggests our model is able to effectively pick up on relevant weight-sharing patterns from training data.

4.2. Learning Partial Equivariances

We tested our model's ability to recognize data symmetries on a series of toy problems featuring noisy G-transformed samples. The data generation details and results are outlined in Appendix E. Additionally, we tested the ability to pick up partial group structures in Appendix F.1. Our experiments used a single-layer setup to learn kernels matching each data sample despite inherent noise, focusing on identifying base kernels and their transformations to accommodate the toy problems' variability.

5. Conclusion

We demonstrated a method for identifying underlying symmetries in data without specific group constraints. Our approach learns symmetries even in partial or approximate forms, reflecting realistic scenarios. By using doubly stochastic matrices and adapting kernel weights for convolutions, we enable flexible learning of the representation stack. For future work, we see an opportunity to enhance our approach by hierarchically weight-sharing across layers and encouraging group equivariance. This involves reusing the learned group structure by identifying the Cayley tensor (akin to (Marchetti et al., 2023)) in each layer and promoting a shared group structure throughout the network.

References

Adams, R. P. and Zemel, R. S. Ranking via sinkhorn propagation, 2011.

Allingham, J. U., Mlodozeniec, B. K., Padhy, S., Antorán, J., Krueger, D., Turner, R. E., Nalisnick, E., and Hernández-Lobato, J. M. A generative model of symmetry transformations, 2024.

Bekkers, E. J., Lafarge, M. W., Veta, M., Eppenhof, K. A. J., Pluim, J. P. W., and Duits, R. Roto-translation covariant convolutional networks for medical image analysis. *CoRR*, abs/1804.03393, 2018. URL http://arxiv.org/abs/1804.03393.

Bekkers, E. J., Vadgama, S., Hesselink, R., der Linden, P. A. V., and Romero, D. W. Fast, expressive $\mathrm{SE}(n)$ equivariant networks through weight-sharing in position-orientation space. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dPHLbUqGbr.

Benton, G. W., Finzi, M., Izmailov, P., and Wilson, A. G. Learning invariances in neural networks. *CoRR*, abs/2010.11882, 2020. URL https://arxiv.org/abs/2010.11882.

Birkhoff, G. Three observations on linear algebra. *Univ. Nac. Tacuman, Rev. Ser. A*, 5:147–151, 1946. URL https://cir.nii.ac.jp/crid/1573387450959988992.

Bronstein, M. M., Bruna, J., Cohen, T., and Velickovic, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021. URL https://arxiv.org/abs/2104.13478.

Cohen, T. S. and Welling, M. Group equivariant convolutional networks. *CoRR*, abs/1602.07576, 2016. URL http://arxiv.org/abs/1602.07576.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d, 2022.

Isbell, J. Infinite Doubly Stochastic Matrices. *Canadian Mathematical Bulletin*, 5(1):1–4, January 1962. ISSN 0008-4395, 1496-4287. doi: 10.4153/CMB-1962-001-4. URL https://www.cambridge.org/core/product/identifier/S0008439500050992/type/journal_article.

Kendall, D. G. On Infinite Doubly-Stochastic Matrices and Birkhoffs Problem 111. *Journal of the London Mathematical Society*, s1-35(1):81–84, 1960. ISSN 1469-7750. doi: 10.1112/jlms/s1-35.1.81. URL https://onlinelibrary.wiley.com/doi/abs/10.1112/jlms/s1-35.1.81. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1112/jlms/s1-35.1.81.

Knigge, D. M., Romero, D. W., and Bekkers, E. J. Exploiting redundancy: Separable group convolutional networks on lie groups. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 11359–11386. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/knigge22a.html.

Lafarge, M. W., Bekkers, E. J., Pluim, J. P., Duits, R., and Veta, M. Roto-translation equivariant convolutional networks: Application to histopathology image analysis. *Medical Image Analysis*, 68:101849, 2021.

Liu, Y. Homomorphism of independent random variable convolution and matrix multiplication, 2023.

Marchetti, G. L., Hillar, C., Kragic, D., and Sanborn, S. Harmonics of learning: Universal fourier features emerge in invariant networks. *arXiv preprint arXiv:2312.08550*, 2023.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Equivariance through parameter-sharing, 2017.

Romero, D. W. and Lohit, S. Learning equivariances and partial equivariances from data. *CoRR*, abs/2110.10211, 2021. URL https://arxiv.org/abs/2110.10211.

Révész, P. A probabilistic solution of problem 111. of G. Birkhoff. *Acta Mathematica Academiae Scientiarum Hungarica*, 13(1):187–198, March 1962. ISSN 1588-2632. doi: 10.1007/BF02033637. URL https://doi.org/10.1007/BF02033637.

Sanborn, S., Shewmake, C., Olshausen, B., and Hillar, C. Bispectral neural networks. *arXiv preprint arXiv:2209.03416*, 2022.

Serre, J.-P. et al. *Linear representations of finite groups*, volume 42. Springer, 1977.

Sinkhorn, R. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Annals of Mathematical Statistics*, 35:876–879, 1964. URL https://api.semanticscholar.org/CorpusID:120846714.

Sosnovik, I., Moskalev, A., and Smeulders, A. W. M. DISCO: accurate discrete scale convolutions. *CoRR*, abs/2106.02733, 2021. URL https://arxiv.org/abs/2106.02733.

Theodosis, E., Helwani, K., and Ba, D. Learning linear groups in neural networks, 2023.

van der Ouderaa, T. F. A., Immer, A., and van der Wilk, M. Learning layer-wise equivariances automatically using gradients, 2023.

Wang, R., Walters, R., and Yu, R. Approximately equivariant networks for imperfectly symmetric dynamics. *CoRR*, abs/2201.11969, 2022. URL https://arxiv.org/abs/2201.11969.

Weiler, M. and Cesa, G. General e (2)-equivariant steerable cnns. *Advances in neural information processing systems*, 32, 2019.

Worrall, D. E. and Welling, M. Deep scale-spaces: Equivariance over scale. *CoRR*, abs/1905.11697, 2019. URL http://arxiv.org/abs/1905.11697.

Yeh, R. A., Hu, Y.-T., Hasegawa-Johnson, M., and Schwing, A. Equivariance discovery by learned parameter-sharing. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I. (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 1527–1545. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/yeh22b.html.

Zhou, A., Knowles, T., and Finn, C. Meta-learning symmetries by reparameterization. *CoRR*, abs/2007.02933, 2020. URL https://arxiv.org/abs/2007.02933.

A. Preliminaries

A.1. Groups

We are interested in (symmetry) groups, which are algebraic constructs that consist of a set G and a group product—which we denote as a juxtaposition—that satisfies certain axioms, such as the existence of an *identity* element $e \in G$ such that for all $g \in G$ we have $eg = ge = g$, *closure* such that for all $g, h \in G$ we have $gh \in G$, the existence of an *inverse* g^{-1} for each g such that $g^{-1}g = e$, and *associativity* such that for all $g, h, i \in G$ we have $(gh)i = g(hi)$.

A.2. Representations

In the context of geometric deep learning (Bronstein et al., 2021), it is most useful to think of groups as transformation groups and the group structure describes how transformations relate to each other. Specifically, *group representations* $\rho : G \to GL(V)$ are concrete operators that transform elements in a vector space V in a way that adheres to the group structure (they are group homomorphisms). That is, to each group element g, we can associate a linear transformation $\rho(g) \in GL(V)$, with $GL(V)$ the set of linear invertible transformations on vector space V.

A.3. Sinkhorn normalization

The Sinkhorn operator (Sinkhorn, 1964; Adams & Zemel, 2011) transforms an arbitrary matrix to a doubly stochastic one through iterative row and column normalization, provided that the number of iterations is large enough. That is, initialize a tensor $\mathbf{X} \in \mathbb{R}^{N \times N}$, then it will converge to a doubly stochastic tensor via the following algorithm:

$$S^0(\mathbf{X}) = \exp(\mathbf{X})\,, \qquad S^l(\mathbf{X}) = T_c(T_r(S^{l-1}(\mathbf{X})))\,, \qquad \mathcal{S}_N \quad \ni \quad \mathbf{S} = \lim_{l \to \infty} S^l(\mathbf{X})\,, \tag{7}$$

with T_c and T_r the normalization operators over the rows and columns, respectively, defined as $T_c = X \oslash \underbrace{\mathbf{1}_N \mathbf{1}_N^T \mathbf{X}}_{\mathrm{sum}_c(\mathbf{X})}$ and $T_r = \mathbf{X} \oslash \underbrace{\mathbf{X}\,\mathbf{1}_N \mathbf{1}_N^T}_{\mathrm{sum}_r(\mathbf{X})}$, where $\oslash$ denotes elementwise division, $\mathrm{sum}_c(\cdot)$, $\mathrm{sum}_r(\cdot)$ perform column-wise and row-wise summation, respectively.

A.4. Irreducible representations

In this section, we closely follow the mathematical preliminaries outlined in (Weiler & Cesa, 2019). For a comprehensive reference on Representation Theory, see (Serre et al., 1977).

Equivalent representations Two representations ρ and ρ' of a group G are considered *equivalent* if there exists a similarity transform such that:

$$\forall g \in G \quad \rho(g) = Q\rho'(g)Q^{-1}$$

where Q represents a change of basis matrix.

Irreps A matrix representation is called *reducible* if it can be decomposed as:

$$\rho(g) = Q^{-1}(\rho_1(g) \oplus \rho_2(g))Q^{-1} = Q \begin{pmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{pmatrix} Q^{-1}$$

where Q is a change of basis matrix. If the sub-representations ρ_1 and ρ_2 cannot be further decomposed, they are termed *irreducible representations* (*irreps*). The set of all irreducible representations of a group G is denoted as $\hat{G}$.

Additionally, any representation $\rho : G \to GL(V)$ of a compact group G can be expressed as:

$$\rho(G) = Q \left[\bigoplus_{j \in \mathcal{I}} \rho_j \right] Q^{-1}$$

where $\mathcal{I}$ is an index set (possibly with repetitions) over $\hat{G}$.

Similarly to what we showed in Section 3, a representation $\rho : G \to \mathbb{R}^{d\times d}$ can be viewed as a collection of d^2 functions over G. The **Peter-Weyl theorem** asserts that the collection of functions formed by the matrix entries of all irreps in $\hat{G}$ spans the space of all square-integrable functions over G. For most groups, these entries form an orthogonal basis, allowing any function $f : G \to \mathbb{R}$ to be written as:

$$f(g) = \sum_{\rho_j \in \hat{G}} \sum_{m,n<d_j} w_{j,m,n} \cdot \sqrt{d_j}[\rho_j(g)]_{mn}$$

where d_j is the dimension of the irrep ρ_j, while m, n index the entries of ρ_j. Note that this expression corresponds to the *inverse Fourier transform* and that the coefficients $w_{j,m,n}$ can be obtained by the *Fourier transform* of f with respect to the basis functions $\{[\rho_j(g)]_{mn}\}_{j\in\mathcal{I}}$.

Connection with regular representations It can be shown that the regular representations can be decomposed using the corresponding irreps as follows:

$$\rho_{\text{reg}}(g) = Q^{-1}\left[\bigoplus_{p_j}\bigoplus^{d_j}\rho_j\right]Q$$

where Q performs the Fourier transform, while Q^{-1} performs the inverse Fourier transform. This implies that when functions $f : G \to \mathbb{R}$ are considered as vectors in $\mathbb{R}^{|G|}$, with a basis where each axis corresponds to a group element, then, as we have seen in Section 3, the group action results in a permutation of these axes. However, applying the Fourier transform changes the basis so that G acts independently on different subsets of the axes, resulting in the action being represented by a block-diagonal matrix, which is the direct sum of irreps.

B. Doubly stochastic matrices as expected regular group representations

Let $\mathcal{S}_\infty$ ($\mathcal{S}_n$) denote respectively the system of infinite ($n \times n$) double stochastic matrices, i.e. matrices $S \equiv \{s_{ij} \in [0,1] : i,j = 1,2,...(,n)\}$ such that $\sum_j s_{ij} = 1$, and $\sum_i s_{ij} = 1$. Let $\mathcal{P}_\infty$ ($\mathcal{P}_n$) denote respectively the system of infinite ($n \times n$) permutation matrices, i.e. matrices $P \equiv \{p_{ij} \in \{0,1\} : i,j = 1,2,...(,n)\}$ such that $\sum_j p_{ij} = 1$, and $\sum_i p_{ij} = 1$. Note that for any permutation tensor $\mathbf{P} \in \mathbb{R}^{|G|\times|X|\times|X|}$, where X is the domain of the signal that is transformed by the group G, then $\mathbf{P}_g \in \mathcal{P}_{|X|}$ for every $g \in G$.

Then, by Birkhoff's Theorem (Birkhoff, 1946), and its extension to infinite dimensional matrices, commonly called Birkhoff's Problem 111 (Isbell, 1962; Révész, 1962; Kendall, 1960), we have that any convex combination of permutation matrices will be equal to a double stochastic matrix, i.e,

$$\sum_{P\in\mathcal{P}_n} \lambda(P)P = S \in \mathcal{S}_n, \text{ with } \sum_{P\in\mathcal{P}_n} \lambda(P) = 1 \quad (\forall n \in \mathbb{N} \cup \{+\infty\})$$

where $\lambda(P)$ gives a probability measure supported on a finite subset of the set of permutation matrices $\mathcal{P}$. Therefore:

Using double stochastic matrices, we can model approximate equivariance as defined in (Romero & Lohit, 2021).

I.e., let S be a random variable over $\{\mathbf{P}_g \in \mathcal{P}_{|X|} \mid g \in G\}$ with a finitely supported probability measure $\mathbb{P}[S = \mathbf{P}_g] = \lambda(\mathbf{P}_g)$ for every $g \in G$, then $\mathbf{S} = \mathbb{E}[S] = \sum_{g\in G} \mathbb{P}[S = \mathbf{P}_g]\mathbf{P}_g$ is a double stochastic matrix. We want to note that $\mathbf{S}$ can be seen as a generalization of the *convolution matrix* presented in (Liu, 2023).

C. Related Works

Partial or relaxed equivariance Methods such as (Romero & Lohit, 2021; Benton et al., 2020; Allingham et al., 2024) learn partial equivariance by learning distributions over transformations, and thereby aim to learn partial or relaxed equivariances from data by sampling some group elements more often than others. (Wang et al., 2022) tries to relax equivariance by introducing learnable equivariance-breaking components. However, these methods require pre-specified sets of symmetry transformations to be known beforehand. In contrast, we aim to pick up the relevant symmetry transformations during training.

Symmetry discovery methods (Sanborn et al., 2022; Marchetti et al., 2023) learn the group structure via (irreducible) group representations. (Sanborn et al., 2022) proposed to learn the Fourier transform of finite compact commutative groups and their corresponding bispectrum by learning to separate orbits on our dataset. This approach can be extended to non-commutative finite groups leveraging advanced unitary representation theory (Marchetti et al., 2023). However, these methods are constrained to finite-dimensional groups and require specific orbit-predicting datasets. In contrast, our approach learns a relaxation of *regular group representations*—as opposed to irreducible representations (see further discussion in Appendix A.4). Moreover, our approach is not merely capable of learning symmetries, it subsequently utilizes them in a regular group-convolution-type architecture.

Weight-sharing methods Previous studies have demonstrated that equivariance to finite groups can be achieved through weight-sharing schemes applied to model parameters. Notably, the works in (Ravanbakhsh et al., 2017), (Zhou et al., 2020), and (Yeh et al., 2022) provide foundational insights into this approach. In (Zhou et al., 2020), weight-sharing patterns are learned by using a matrix that operates on flattened canonical weight tensors, effectively inducing weight sharing. They additionally prove that for finite groups, there are weight-sharing matrices capable of implementing the corresponding group convolution. However, their approach requires learning these patterns through meta-learning and modeling the weight-sharing matrix as an unconstrained tensor. In contrast, our method learns weight sharing directly in conjunction with the downstream task and enforces the matrix to be doubly stochastic, thereby representing soft permutations by design.

(Yeh et al., 2022) presents an approach closely aligned with ours, where a weight-sharing scheme is learned that is characterized by row-stochastic entries. Their method involves both inner- and outer-loop optimization and demonstrates the ability to uncover relevant weight-sharing patterns in straightforward scenarios. However, their approach does not support joint optimization of the canonical weights and weight-sharing pattern, and they acknowledge difficulties in extending their method to higher input dimensionalities. Unlike (Yeh et al., 2022), we enforce both row and column stochasticity. Additionally, we can optimize for the sharing pattern and weight tensors jointly, and successfully apply our approach to more interesting data domains such as image processing.

In (Theodosis et al., 2023), group actions are integrated directly into the learning process of the downstream task. This method involves learning a set of generator matrices that operate via matrix multiplication on flattened input vectors. However, this approach constrains the operators to members of finite cyclic groups, which inherently limits their ability to represent more complex group structures. Furthermore, this restriction precludes the possibility of modeling partial equivariances, reducing the flexibility and applicability of the model to more diverse or complex scenarios.

D. Experiment details

D.1. Regularizers.

We test two regularizers on the representations $\mathbf{R}$, which are similar to those used by (Yeh et al., 2022):

- **Entropy Regularizer**: The primary motivation for using the entropy regularizer is to encourage sparsity in our weight-sharing schemes, which helps the matrices approximate actual permutation matrices rather than simply being doubly stochastic. This approach stems from the intuition for some transformations, the weight-sharing schemes should mimic soft-permutation matrices. The effectiveness of this sparsity depends on the specific group transformations relevant to the task—for example, C4 rotations are typically represented by exact permutation matrices. In contrast, C_N rotations or scale transformations might require interpolation, thus aligning more closely with soft permutation matrices. Our experimental results indicate that the utility of this regularizer varies with the underlying transformations in the data; for instance, it is not beneficial for scale transformations in the MNIST dataset, as anticipated. The entropy regularizer is of the following form:

$$H(\mathbf{R}) = -\sum_{ijk}^{N,D,D} \mathbf{R}_{ijk} \cdot \log(\mathbf{R}_{ijk})$$

- **Normalization Regularizer**: Empirically, we have found the normalization regularizer essential for reducing the number of iterations needed by the Sinkhorn operator to ensure the matrices are row and column-normalized. Without this regularizer, the tensors either fail to achieve double stochasticity or require an excessively high number of Sinkhorn iterations to do so. The normalization regularizer is of the following form:

$$N(\mathbf{R}) = \frac{1}{D}\sum_{i}^{D} \mathrm{sum}_r(\mathbf{R})_i^2 + \mathrm{sum}_c(\mathbf{R})_i^2$$

Where $\mathrm{sum}_r, \mathrm{sum}_c$ are defined as in A.3.

Effect of the entropy regularizer The influence of the entropy regularizer was assessed on the augmented MNIST datasets across 50 training epochs. Results varying the entropy loss weight (λ_{ent}) across different numbers of representation stacks are detailed in Table 3. For the rotated MNIST, where C2 and C4 rotations are expected to match exact permutations, the application of entropy regularization was anticipated to be advantageous. In contrast, for ScaleMNIST, where weight-sharing patterns are not expected to be exact permutation matrices, entropy regularization was found to reduce performance.

Table 3. Effect of different weightings for the entropy regularizing term.

	Rot. MNIST				Scale MNIST
λ_{ent} \ $\lvert G\rvert$	2	3	4	5	4
0	98.076	97.90	97.66	**97.78**	**99.40**
0.001	97.88	97.76	97.61	97.66	99.30
0.01	98.03	97.82	97.74	97.51	99.37
0.1	**98.08**	**97.96**	**97.73**	97.53	99.39

D.2. Architectural details

Model architecture We found it to be beneficial to fix an identity element in the representation stacks, which implies that the first stack of transformed filters corresponds to the raw network filters. We used the following architectures for the MNIST and CIFAR10 datasets:

- **MNIST.** For all MNIST experiments, a simple 5-block CNN was used. Each block uses a kernel size of 5 and is succeeded by instance norm and ReLU activation, respectively. After the final convolution block, any spatial and group dimensions are reduced through a global average pooling operation, and a single linear layer is used as a classification head. We used a group convolution layer and our proposed weight sharing convolution layer as a drop-in replacement for the regular $\mathrm{Conv2d}$ modules in all experiments. For the group convolutional model and our weight sharing model, the default hidden channel dimension in the blocks was set to 32 unless otherwise stated, and 64 in the regular CNN models.
- **CIFAR10.** We used the ResNet architecture as in (Knigge et al., 2022) Appendix B.1, except that we swapped the final global max pooling operator with a global mean pooling. However, in contrast to (Knigge et al., 2022), we use regular discrete kernels instead of continuous kernel parameterizations.

Training details For MNIST, the models used a learning rate of 1e-2 and were trained for 100 epochs. For CIFAR10, following (Knigge et al., 2022), we trained the models for 200 epochs using a learning rate of 1e-4. All the experiments were done on a single GPU with 24GB memory under six hours.

D.3. Network sizes

Table 4. Model sizes for different datasets. For the weight-sharing CNNs we distinguish between the number of free model and kernel parameters and the parameters of the weight sharing scheme with $+$

	MNIST			CIFAR10			
Model	CNN	GCNN	WSCNN	CNN 32	CNN 64	GCNN	WSCNN
Num. Params.	412 K	410 K	410 + 122 K	428 K	1.66 M	1.63 M	1.63 M + 468 K

E. Toy problems

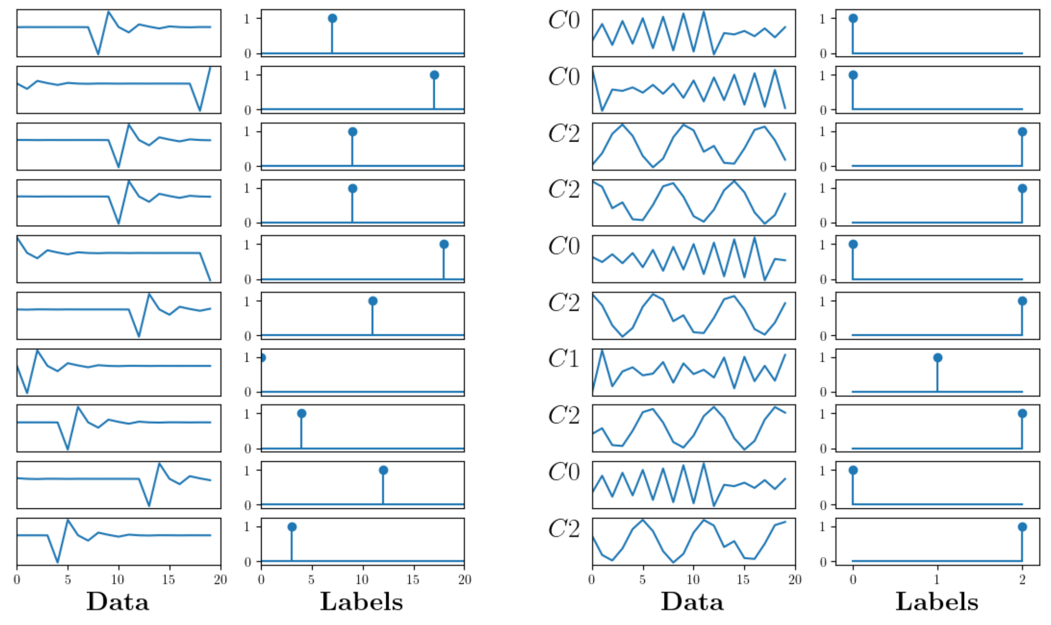
E.1. Data generation processes

For the construction of the toy problems, we look at two types of data-generating processes:

Equivariant data. Assume a canonical vector $\hat{\mathbf{x}}$ and corresponding label $\hat{\mathbf{y}}$. We assume these vectors transform under a known group G, such that the data-generating process is as follows:

$$\begin{aligned}\text{Sample group action:} \quad & g \sim \mu(G), \\ \text{Apply group action to feature with noise:} \quad & \mathbf{x} = \rho^x(g)\hat{\mathbf{x}} + \epsilon \\ \text{Apply group action to label:} \quad & \mathbf{y} = \rho^y(g)\hat{\mathbf{y}}\end{aligned}$$

with ρ^x, ρ^y the representations of G acting on the feature and label space, respectively.



(a) (b)

Figure 2. Samples of the two tasks. *Left*: Equivariant task. The labels transform with the data. *Right*: Invariant task. The labels are fixed.

E.2. Additional results: toy problems

We conducted experiments on various signals subjected to different transformations, including: a 1D signal with cyclic shifts (exemplary samples shown in Fig. 2), a 2D signal with C_8 rotations (illustrated in Fig. 4), and a 3D voxel grid enhanced by 24 cube symmetries. In each scenario, the learned kernel stack accurately matched the data samples, achieving perfect accuracy. Fig. 3 displays the learned representations for localized shifts in the 1D signal, while Fig. 5 presents the learned kernel stack for the 2D signal dataset.

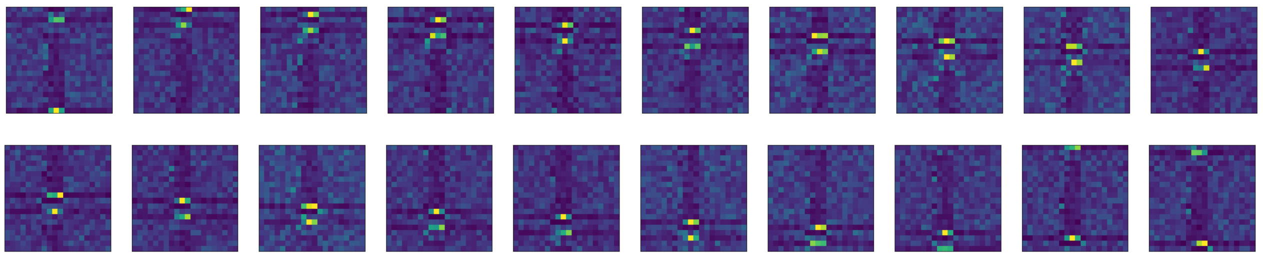


Figure 3. Representations learned for 1D equivariant shift task

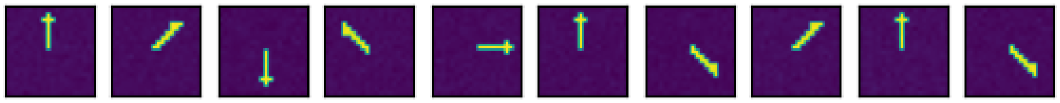


Figure 4. Samples of the 2D-signal dataset.

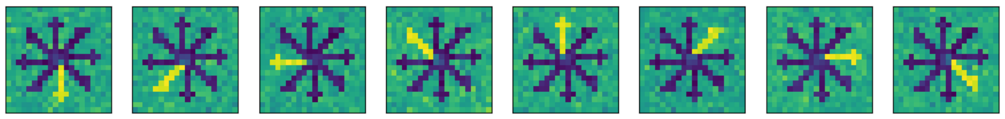


Figure 5. Equivariant task for flattened rotated 2D signals. Learned kernel stack

F. Additional results

F.1. Rotated MNIST: Learned kernels from the lifting layer

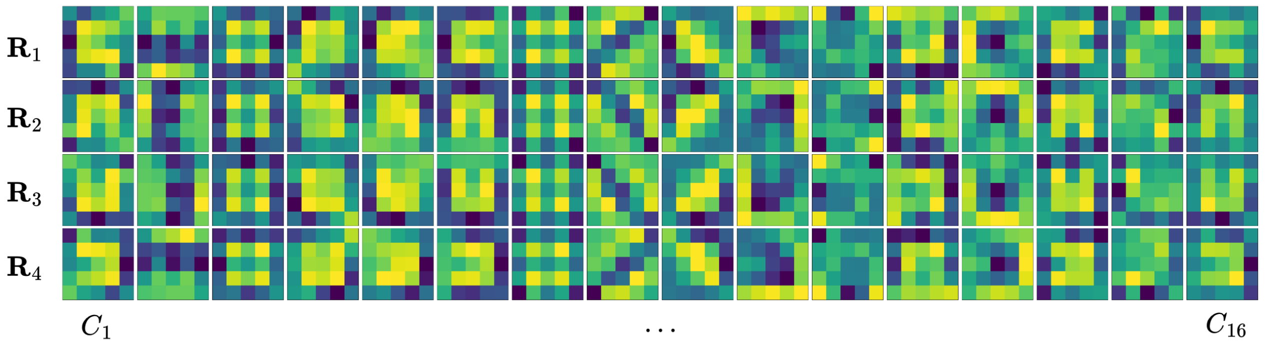


Figure 6. Learned kernels from the lifting layer of WSCNN, applied to rotated MNIST and reshaped to $[\mathrm{No.}, \mathrm{of}, \mathrm{elements}, C_{out}]$. Since $\mathbf{P}_1$ is set as the identity operator, the first column displays the raw kernels. Subsequent columns illustrate the effects of each learned transformation applied to the base tensors.

F.2. Visualization of G-Conv layers

Figure 7 displays the ground truth permutation matrices that implement a shift-twist operator, which is the group transformation that underlies regular group convolution operator for $C4$ rotations. Figure 8 illustrates the corresponding matrices learned for each learnable weight sharing layer on the rotated MNIST dataset. The learned matrices closely show similar patterns as the shift-twist operator, suggesting the model's ability to capture such transformations from training data.

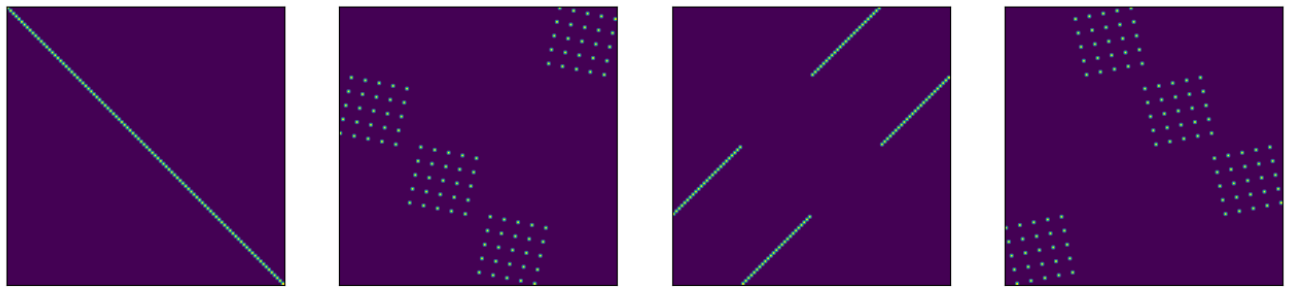


Figure 7. Ground-truth permutation matrices for C_4 rotations for 5×5 spatial kernel, implementing a *shift-twist* operator.

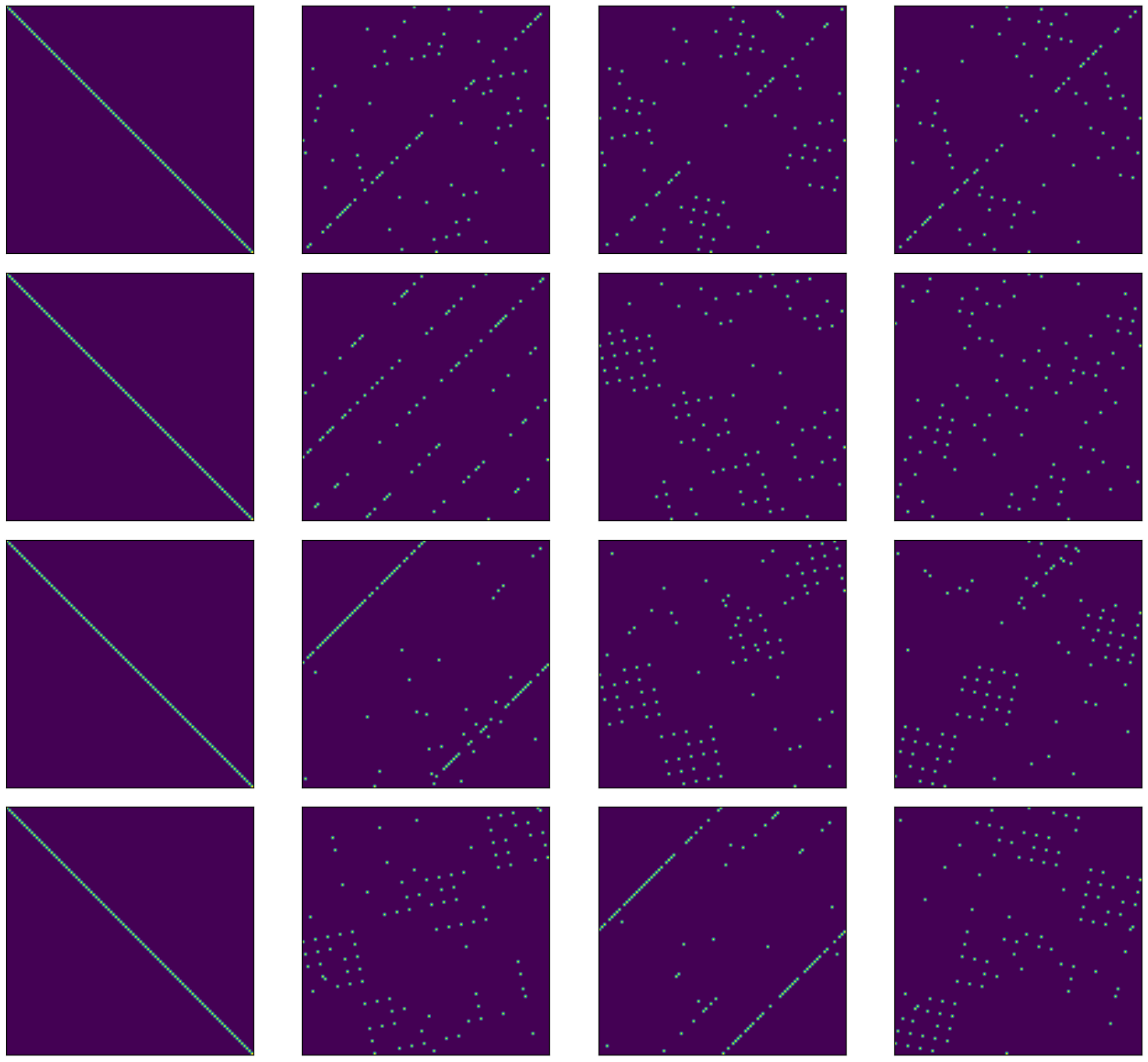


Figure 8. Learned representations for the weight-sharing G-conv layers. Top to bottom: first to last layer learned representation stacks.

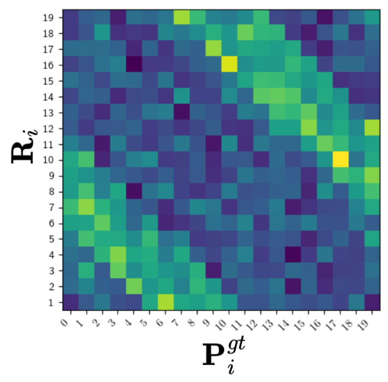
F.3. Learning partial equivariance

To assess whether our method effectively captures the underlying data symmetries, we analyze the learned weight-sharing structure by comparing it to known ground truth patterns. Since our model does not impose associativity or any strict group constraints on the representation stack, it may learn to represent mixtures or interpolations of group elements. Hence, in general, we will not observe a relevant algebraic structure if we produce a Cayley table based on the learned representations as done in (Sanborn et al., 2022; Marchetti et al., 2023).

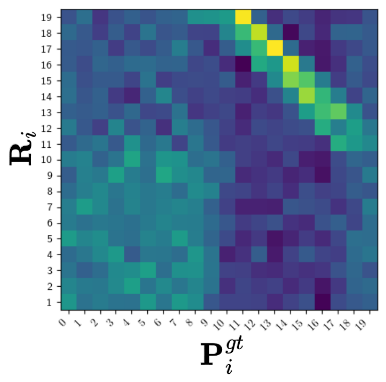
Therefore, we will use an approach that allows us to capture the flexibility of our representations. To this end, we will examine each representation matrix to determine how closely it resembles a *convolution matrix* (Liu, 2023) associated with a random variable defined over some specific group G. We employ the set of group actions from G, represented as approximate permutation tensors $\{\mathbf{P}_k^{gt}\}_{k=1}^{|G|}$, as a reference framework to quantify the fit and alignment of our model's representations with these predefined group actions. As such, for each learned weight sharing tensor $\mathbf{R}_i^l$, we calculate the fit $\hat{\mathbf{P}}_i = \sum_k^{|G|} c_k \mathbf{P}_k^{gt}$ and acquire coefficients $c_k > 0$, such that $\sum_k^{|G|} c_k = 1$ in a constrained linear regression setup by minimizing $||\hat{\mathbf{P}}_i - \mathbf{R}_i^l||_2$.

To verify if our model can detect partial group structures, we assessed it using two datasets: a 1D signal with cyclical shifts, and a $3 \times 3 \times 3$ voxel grid with rotations from $C_4 \times C_4$. The cyclical shifts dataset used the full set of shifts as ground truth.

Figure 9(a, b) displays the coefficient results, where part (a) shows uniform training across group elements and part (b) shows training with the first half of the elements, demonstrating the model's capability to handle partial transformations effectively. Fig. 10 shows the coefficients for the $C_4 \times C_4 \times C_4$ cube symmetries, indicating that our model primarily identifies transformations aligned with the $C_4 \times C_4$ data augmentations.



(a) Shift-dataset with uniformly sampled group elements.



(b) Shift-dataset with partially sampled group elements.

Figure 9. Coefficient responses of learned representations and their base transformations.

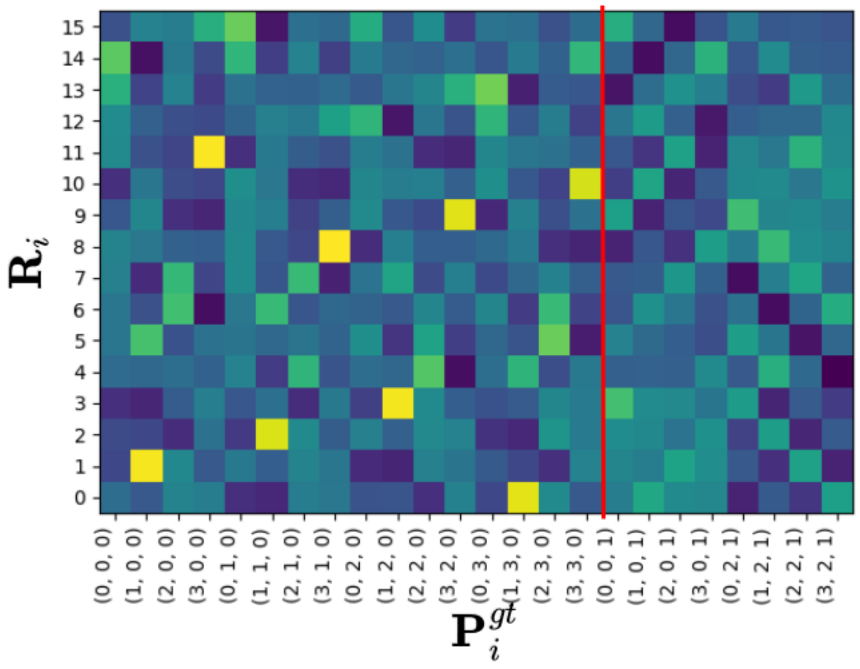


Figure 10. Coefficients for the cube dataset with $C_4 \times C_4$ rotations with $C_4 \times C_4 \times C_4$ base elements. We show the coefficients for the base representations of $C_4 \times C_4 \times C_4$ cube symmetries. The x-axis quantifies the permutation representation for each element consisting of the number of 90-degree flips around each x, y or z axis.