Neural Symmetry Detection for Learning Neural Network Constraints

author names withheld

Under Review for the Workshop on High-dimensional Learning Dynamics, 2024

Abstract

Neural symmetry detection can be defined as the deep learning-aided task of recovering both the nature of the transformation that relates points in a data set and the distribution with respect to the magnitude of the transformation. Applications range from automatic data augmentation to model selection. In this work, we investigate how the matrix exponential can be leveraged to recover the correct symmetry transformation, encoded as a generator of a Lie group for various transformations, both affine and non-affine. In order to make the calculation of the matrix exponential tractable, this operation is performed in a low-dimensional latent space. Additionally, a loss term is introduced to enforce matching the generator in latent space to the one in pixel-space.

1. Introduction

In the depths of the transcendental deduction, Kant argues that at the basis of all understanding of sensory input lies the ability to apply *rules*, rules which act according to the primal possibility to define direction and magnitude through the fundamental properties of time and space [10]. Combined in various ways, these produce the elementary building blocks for our understanding, the categories, which are not unlike elementary logical operators. In machine learning, a similar philosophy persists. From early vision scientists [9], to computational neurobiologists [23], and currently geometric deep learning [2], the importance of designing and exploiting data-informed structural priors has been identified with the presence of symmetries in certain tasks and raw data.

Modern approaches to symmetry detection focus on learning the most likely symmetry group that relates points in a data set. In previous work [6, 27], the matrix exponential required to quantify the continuous symmetry transformation has been approximated in various ways, which sacrifices accuracy for tractability. We propose instead to evaluate the matrix exponential exactly while avoiding the computationally expensive operation in pixel space by going to a low-dimensional latent space, using a deep autoencoder-like architecture. Additionally, having more control over the behavior of latent vectors and features in general by enforcing continuous transformations such as rotations (i.e., example of compact groups) between them has shown to be advantageous for explainability of deep models [16]. Using neural networks for the purposes of finding underlying invariances is just one approach to what could be called *neural symmetry detection*. This has the added benefit of learning suitable representations and has a possible extension to defining the connectivity matrix in the context of structural prior learning. For more related work, we refer the reader to Appendix A.

2. Theory: Symmetry Generators and Structural Priors

We focus on one-parameter groups for two reasons: Ease of implementation and the fact that one such inductive bias is incredibly powerful already. One need not look further than CNNs to conclude that identifying translation as a symmetry of a dataset immediately leads to equivariant models that are superbly successful in practice. Multiple transformations also require additional considerations that relate to the algebra itself, such as closure under commutators [24], an extension we leave for future work.

2.1. Connectivity Matrices and Equivariance

Learning connectivity matrices for deep equivariant models, whether the symmetry group is given or not, is not new [7, 13, 32, 37]. It is worth noting that generators can be related to the connectivity matrix, explicitly so for translations, where the shift matrix determines a power series that tiles the weight matrix accordingly. Formally, for the one-pixel shift matrix $S = e^{\partial_x}$, we can write:

$$f_\theta^L = \sum_{i \in \mathbb{Z}} \theta_i^L S^i. \tag{1}$$

The above equation defines the weight matrix of one such convolutional layer L, with updatable weights θ_i^L. A collection of multiple power series applied in succession and interlaced with non-linear activation functions *is* the neural network. Schematically, we have

$$f_\theta(\,\cdot\,) = \bigcirc_{L=1}^{\Lambda} \sigma_L \left(f_\theta^L(\,\cdot\,) + \boldsymbol{b}_L \right), \tag{2}$$

with a total number of layers Λ, biases $\boldsymbol{b}_L$, and activation functions σ. Note that this does not work for densely-connected layers, as all the elements in the basis need to be related by matrix powers.

In previous work, a main issue was overcoming the computational complexity associated with high pixel count, especially in learning the exponent of a matrix exponential. A second potential issue is introducing strong spatial correlations as a constraint *a priori*, defeating the purpose of learning transformations from scratch. If it is already known that the data is spatially structured, one could introduce continuous coordinates [20] or use spatial convolutions [13] immediately, without conveniently ignoring the possibility of spatially unstructured data. This issue is partially alleviated with the matrix exponential method, as learning a generator that corresponds to the zeros matrix leads to the identity matrix, a trivial operation.

3. Method: Matrix Exponentials and Latent Flow

Our dataset consists of MNIST and CIFAR-10 data, paired with an augmented version of the original image. The original labels are not used in the current setting. The goal of this work is two-fold: (i) extract the type of transformation G that was applied, and (ii) estimate the distribution of *parameters* t of the seen transformations in the data, which are the magnitudes of the transformation (e.g., rotation angle, scaling factor, etc.).

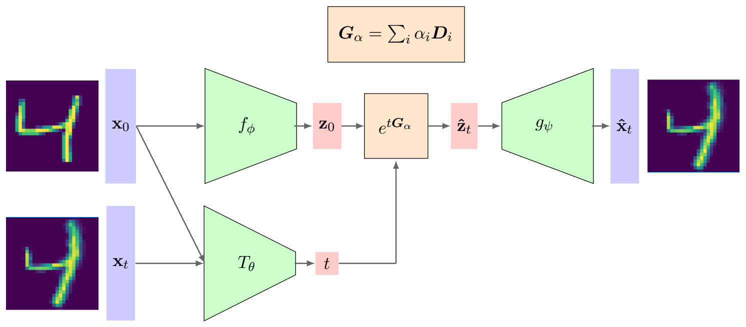

Figure 1: Latent model architecture. The image pair consists of an MNIST digit, its transformation under a SCT, and the expected output.

3.1. Parametrizing the Generator

We can parametrize the generator with any given basis for the functional form of its components, to allow for modelling a broad range of symmetry transformations [8]. In other words, regression is performed on the coefficients of a basis, which can be chosen freely. We wish to have the ability to detect the “typical" symmetries considered in the symmetry detection literature, such as rotation, scaling, and translation, which will be referred to as the *canonical symmetries*, a subset of the affine transformations. Therefore, we pick a *quadratic basis*, which includes affine transformations, such that the functions Γ_i (from equation 8, see Appendix B for mathematical details) have the following form:

$$\begin{aligned}\Gamma_x &= \alpha_c^{(x)} + \alpha_x^{(x)}\, x + \alpha_y^{(x)}\, y + \alpha_{xx}^{(x)}\, x^2 + \alpha_{xy}^{(x)}\, xy + \alpha_{yy}^{(x)}\, y^2,\\ \Gamma_y &= \alpha_c^{(y)} + \alpha_x^{(y)}\, x + \alpha_y^{(y)}\, y + \alpha_{xx}^{(y)}\, x^2 + \alpha_{xy}^{(y)}\, xy + \alpha_{yy}^{(y)}\, y^2,\end{aligned} \tag{3}$$

with learnable coefficients α_i. The above quadratic basis can capture the canonical symmetries but others, such as shears, compositions, or special conformal transformations (which are manifestly non-affine) as well. Note that one can pick an arbitrarily complicated basis for the expressions given above. This is the major appeal of this approach, and we hope to explore the expressibility of different bases in future work.

3.2. Latent Model

In our model, the autoencoder design allows for the model to keep relevant information in the latent space and transform this according to a transformation, a result of the exponential map, it shares with all other input pairs (Figure 1). The model takes an image as input and reconstructs the transformed image as output. In order to get an estimate for the parameter (e.g. rotation angle) a separate network is trained together with the autoencoder. The parameter estimate is then passed to the matrix exponential function that is then used to matrix multiply the latent patch(es). Finally, the latent patch(es) are decoded and a reconstruction loss is enforced on the output-transformed image pair.

The loss function consists of a part that ensures both the n-dimensional pixel-space and d-dimensional latent-space vectorized data pairs transform to each other. For images $\boldsymbol{x}_0, \boldsymbol{x}_t \in \mathbb{R}^{n^2}$ and

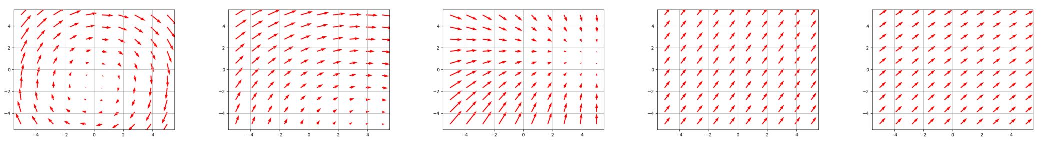

Figure 2: Latent flow learned for various transformations on MNIST. From left to right: small rotation angle, large rotation angle, scaling, translation, uniform and bimodal.

$\boldsymbol{G}_\alpha \in \mathbb{R}^{d^2 \times d^2}$, one can write these as:

$$\mathcal{L}_T^X(\boldsymbol{x}_0, \boldsymbol{x}_t) = \|g_\psi \circ e^{T_\theta(\boldsymbol{x}_0, \boldsymbol{x}_t)\boldsymbol{G}_\alpha} \circ f_\phi(\boldsymbol{x}_0) - \boldsymbol{x}_t\|^2, \tag{4}$$

$$\mathcal{L}_T^Z(\boldsymbol{x}_0, \boldsymbol{x}_t) = \|e^{T_\theta(\boldsymbol{x}_0, \boldsymbol{x}_t)\boldsymbol{G}_\alpha} \circ f_\phi(\boldsymbol{x}_0) - f_\phi(\boldsymbol{x}_t)\|^2. \tag{5}$$

The model allows for various numbers of latent patches (cf. channels) to be transformed in parallel in the latent space.

$\boldsymbol{\alpha}$-Matching Next to the above, we include a loss term that enforces the generator in the latent space to be close to the one in pixel space is introduced. Since calculating the matrix exponential in pixel space does not scale well w.r.t. data size, a different approach needs to be used (See Appendix D). We also include the standard reconstruction loss term for each of the inputs in the data pair individually, $\mathcal{L}_R(\boldsymbol{x}) = \|g_\psi \circ f_\phi(\boldsymbol{x}) - \boldsymbol{x}\|^2$, to train the autoencoder. This loss term simply encourages the encoder to learn to reconstruct individual image inputs well, as usually done with autoencoders. Thus, this term ignores the exponential on the latent space. The total loss, therefore, is:

$$\mathcal{L}(\boldsymbol{x}_0, \boldsymbol{x}_t) = \mathcal{L}_T^X(\boldsymbol{x}_0, \boldsymbol{x}_t) + \lambda_Z \mathcal{L}_T^Z(\boldsymbol{x}_0, \boldsymbol{x}_t) + \lambda_R[\mathcal{L}_R(\boldsymbol{x}_0) + \mathcal{L}_R(\boldsymbol{x}_t)] + \lambda_\alpha \mathcal{L}_\alpha + \lambda_L||\boldsymbol{\alpha}||_1. \tag{6}$$

4. Results

A transformation and parameter distribution are chosen and fed to the data generator. This produces data pairs $\{\boldsymbol{x}_0, \boldsymbol{x}_t\}$ in which the first image is the original and the second is the transformed image. The transformation is applied using the `affine` function from the `torchvision` library [17] and our own implementation of more complicated non-affine transformations. The magnitude of the transformation is the parameter sampled from the chosen distribution. This procedure allows for a lot of flexibility in testing a neural symmetry detector, as the distribution can be arbitrary and, in theory, so can the transformations. In these experiments, we will focus on detecting combinations of various affine transformations and a non-affine transformation, the SCT. Additional implementation details of the MLPs can be found in the appendix D.

4.1. Generator Prediction and Latent Flow

Investigating the quality of the learned generator depends on what final coefficients the model converged to. We visualize the latent flow defined by the generator using the natural connection to differential equations (Equation 9). This describes the flow in latent space, i.e., it shows how the latent 5×5 image is transformed in order to obtain the transformed image in pixel space (Figure 2). Note that generator relates to the direction of the flow, the magnitude and orientation (i.e., forwards or backwards) is determined by the value and sign of the parameter t, respectively.

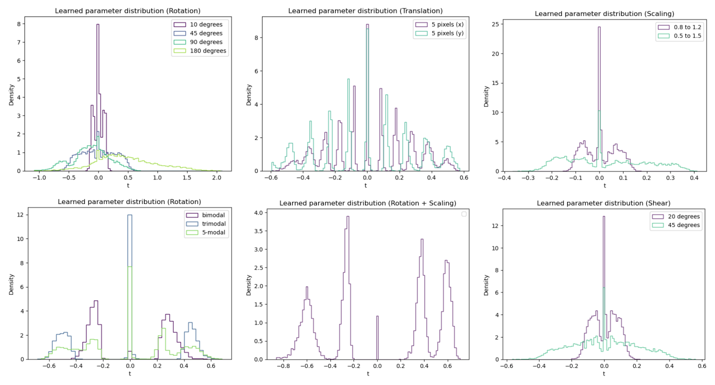

Figure 3: Learned parameter distribution for various transformations and uniform sampling (top row). Parameters for more complicated distributions, compositions, and non-canonical transformations (bottom row). All of these results were applied to augmented MNIST.

4.2. Parameter Distribution

The parameters predicted by the t-network augmented MNIST are plotted on a histogram during training (Figures 3). Beyond multimodal distributions, which the model can learn well, it seems like the distributions capture aliasing artifacts in the small angle and translation setting. Note the broadening of the range (in radians) of the learned distribution when sampling angles in the rotation setting, which breaks down for larger values. Peaks are also visible at the origin, even when no pair relates to such an identity transformation.

4.3. Discussion

In most cases, in particular augmented MNIST, it seems like the final generator has some correct values. This is probably due to the α-matching term, which pushes the values of the coefficient towards the values expected from the first order Taylor expansion in pixel space. It is expected that this term helps the coefficients reach a basin where the correct pixel-space transformations are reachable, although placing too much weight on this term might be counter-productive for large values of the parameters. Additionally, it is worth noting that the parameter distributions mostly match the correct ones. Finally, we note that the quality of the reconstructions are not always as crisp (attaining about 0.12 MSE for CIFAR in particular) (Figure 6). This is probably due to the MLP not being suitable for autoencoding image data. More results are shown in Appendix E.

5. Conclusion

Symmetry detection tasks rely on identifying transformations of data points that keep some task-related quality, such as classification label, identical. In this work, we proposed a latent variable framework for learning one-parameter subgroups of Lie group symmetries from observations. Our method uses a neural network to predict the one-parameter of every transformation that has been applied to datapoints, and the coefficients of a linear combination of a pre-specified basis of (affine and non-affine) generators. We show that our method can learn the correct generators for a variety of transformations as well as characterize the distribution of the parameter that has been used for transforming the dataset.

References

[1] Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew G Wilson. Learning invariances in neural networks from training data. *Advances in neural information processing systems*, 33: 17605–17616, 2020.

[2] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021. URL https://arxiv.org/abs/2104.13478.

[3] Taco Cohen and Max Welling. Learning the irreducible representations of commutative lie groups. In *International Conference on Machine Learning*, pages 1755–1763. PMLR, 2014.

[4] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. In *International conference on Machine learning*, pages 1321–1330. PMLR, 2019.

[5] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

[6] Nima Dehmamy, Robin Walters, Yanchen Liu, Dashun Wang, and Rose Yu. Automatic symmetry discovery with lie algebra convolutional network. *Advances in Neural Information Processing Systems*, 34:2503–2515, 2021.

[7] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups, 2021.

[8] Alex Gabel, Rick Quax, and Efstratios Gavves. Data-driven lie point symmetry detection for continuous dynamical systems. *Machine Learning: Science and Technology*, 5(1):015037, mar 2024. doi: 10.1088/2632-2153/ad2629. URL https://dx.doi.org/10.1088/2632-2153/ad2629.

[9] J.J. Gibson. *The Senses Considered as Perceptual Systems*. Bloomsbury Academic, 1966. ISBN 9780313239618. URL https://books.google.nl/books?id=J9ROAAAAMAAJ.

[10] Immanuel Kant. *Critique of Pure Reason*. The Cambridge Edition of the Works of Immanuel Kant. Cambridge University Press, New York, NY, 1998. Translated by Paul Guyer and Allen W. Wood.

[11] Hamza Keurti, Hsiao-Ru Pan, Michel Besserve, Benjamin F. Grewe, and Bernhard Schölkopf. Homomorphism autoencoder – learning group structured representations from observed transitions, 2023.

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

[13] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups, 2018.

[14] Sven Krippendorf and Marc Syvaeri. Detecting symmetries with neural networks, 2020. URL https://arxiv.org/abs/2003.13679.

[15] Ziming Liu and Max Tegmark. Machine-learning hidden symmetries. *CoRR*, abs/2109.09721, 2021. URL https://arxiv.org/abs/2109.09721.

[16] Sindy Löwe, Phillip Lippe, Francesco Locatello, and Max Welling. Rotating features for object discovery. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 59606–59635. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/bb36593e5e438aac5dd07907e757e087-Paper-Conference.pdf.

[17] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589336. doi: 10.1145/1873951.1874254. URL https://doi.org/10.1145/1873951.1874254.

[18] Ning Miao, Tom Rainforth, Emile Mathieu, Yann Dubois, Yee Whye Teh, Adam Foster, and Hyunjik Kim. Learning instance-specific augmentations by capturing local invariances, 2023.

[19] Xu Miao and Rajesh PN Rao. Learning the lie groups of visual invariance. *Neural computation*, 19(10):2665–2693, 2007.

[20] Artem Moskalev, Anna Sepliarskaia, Ivan Sosnovik, and Arnold Smeulders. Liegg: Studying learned lie group generators. In *Advances in Neural Information Processing Systems*, 2022.

[21] Francesco Oliveri. Lie symmetries of differential equations: Classical results and recent contributions. *Symmetry*, 2, 06 2010. doi: 10.3390/sym2020658.

[22] P.J. Olver. *Applications of Lie Groups to Differential Equations*. Graduate Texts in Mathematics. Springer New York, 1993. ISBN 9780387950006. URL https://books.google.nl/books?id=sI2bAxgLMXYC.

[23] Rajesh Rao and Daniel Ruderman. Learning lie groups for invariant visual perception. *Advances in neural information processing systems*, 11, 1998.

[24] Alexander Roman, Roy T. Forestano, Konstantin T. Matchev, Katia Matcheva, and Eyup B. Unlu. Oracle-preserving latent flows, 2023.

[25] Sophia Sanborn, Christian Shewmake, Bruno Olshausen, and Christopher Hillar. Bispectral neural networks. *arXiv preprint arXiv:2209.03416*, 2022.

[26] Utkarsh Singhal, Carlos Esteves, Ameesh Makadia, and Stella X. Yu. Learning to transform for generalizable instance-wise invariance, 2024.

[27] Jascha Sohl-Dickstein, Ching Ming Wang, and Bruno A Olshausen. An unsupervised algorithm for learning lie group transformations. *arXiv preprint arXiv:1001.1027*, 2010.

[28] Ivan Sosnovik, Artem Moskalev, and Arnold Smeulders. Disco: accurate discrete scale convolutions, 2021.

[29] Riccardo Valperga, Kevin Webster, Dmitry Turaev, Victoria Klein, and Jeroen Lamb. Learning reversible symplectic dynamics. In *Learning for Dynamics and Control Conference*, pages 906–916. PMLR, 2022.

[30] Tycho F. A. van der Ouderaa and Mark van der Wilk. Learning invariant weights in neural networks, 2022.

[31] Tycho F. A. van der Ouderaa, David W. Romero, and Mark van der Wilk. Relaxing equivariance constraints with non-stationary continuous filters, 2022.

[32] Tycho FA van der Ouderaa, Alexander Immer, and Mark van der Wilk. Learning layer-wise equivariances automatically using gradients. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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

[34] Daniel Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 32, 2019.

[35] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[36] Yan Zhang, David W. Zhang, Simon Lacoste-Julien, Gertjan J. Burghouts, and Cees G. M. Snoek. Multiset-equivariant set prediction with approximate implicit differentiation, 2022.

[37] Allan Zhou, Tom Knowles, and Chelsea Finn. Meta-learning symmetries by reparameterization. *arXiv preprint arXiv:2007.02933*, 2020.

[38] Xinqi Zhu, Chang Xu, and Dacheng Tao. Commutative lie group vae for disentanglement learning, 2021.

Appendix A. Related Work

In this section, we present related work, including a succinct list of our contributions.

Equivariance A lot of work has been dedicated to designing neural networks that are equivariant with respect to a given transformation [7, 13]. Transformations of interest beyond translation are scaling [28, 34], rotation on spheres [5], local gauge transformations [4] and the Euclidean group [33], as well as discrete transformations like permutations of sets [35, 36] and time-reversal [29]. Research in these areas shows improved performance on tasks that are symmetric w.r.t. the transformation under consideration, but nonetheless requires knowledge about the symmetries a priori.

Symmetry Detection Early work on detecting symmetries from observations was performed by [23] and [19], who use methods to learn transformations for small parameter values. [27] propose a smoothing operation of the transformation space to overcome the issue of a highly non-convex reconstruction objective that includes an exponential map. These methods are close to ours in that we also make use of the exponential map to obtain group elements from their Lie algebra, although their work being focused on video patches and using EM-algorithms to find the parameters and the generator. [3] focus on disentangling and learning the distributions of multiple compact “toroidal" one-parameter groups in the data.

Neural Symmetry Detection Techniques from Lie theory and generators have been used in conjunction with deep learning methods in order to identify symmetries of a task, although usually only for small angles or in a supervised setting [6]. These are also of interest to physicists, as this can simplify the process of identifying conservation laws or picking the right theoretical model for a given problem [14, 15]. Probabilistic approaches are also of interest, especially in relation to our work, in which learning the distribution over the parameters as well as the symmetry is performed. This can be for inference [30, 31] or automatic data augmentation [1, 18, 26]. Another method is found in [25], where a group invariant function known as the bispectrum is used to learn group-equivariant and group-invariant maps from data. [1] consider a task similar to ours, attempting to learn groups with respect-to-which the data is invariant, however, the objective places constraints directly on the network parameters as well as the distribution of transformation parameters with which the data is augmented.

Latent Transformations Learning transformations of a one-parameter subgroup in latent space (whether that subgroup is identical to the one in pixel space or not) has been visited by [11, 24, 38]. Nevertheless, these works either presuppose local structure in the data by using CNNs instead of fully-connected networks like we do, or they focus on disentangling interpretable features instead of directly learning generators that can be used as an inductive bias for a new model.

In contrast to the above, we propose a model that is able to:

- perform symmetry detection in pixel-space, without assuming strong spacial inductive biases,
- efficiently parametrize the latent space generator for a wide range of parameter values and different symmetry transformations,
- learn both the generator and the parameter distributions simultaneously in an unsupervised manner.

To the best of our knowledge, previous works have only been able to estimate symmetries from images in a supervised way [6], on small image patches [27] or focusing on downstream validation without directly evaluating the symmetries [37], specifically for automatic data augmentation [1, 18, 26] unfortunately rendering them unsuitable for baseline comparisons (Table 1).

	MNIST		**CIFAR-10**		**Param.**	**Transferable**
	Affine	Non-aff.	Affine	Non-aff.	**Distributions**	**Generators**
Dehmamy *et al.* [6]	X	X	X	X	X	✓
Augerino [1]	✓	X	✓	X	✓(uniform)	X
Singhal *et al.* [26]	✓	X	✓	X	✓(2,3-modal)	X
Ours	✓	✓	✓	✓	✓(n-modal)	✓

Table 1: Comparison of model properties across different works.

Appendix B. Lie Theory

The transformations that describe the symmetries are assumed to form *Lie groups*. This means they are sufficiently smooth (k-times differentiable, where k is usually chosen to be infinity), closed under composition, associative, have a neutral element, and have smooth inverse. These transformations can be defined by the way in which they *act* on objects, namely $\mathbf{H} : X \times \mathbb{R} \to X$, with object $\boldsymbol{x} \in X \subset \mathbb{R}^n$. This also introduces a parameter, $t \in \mathbb{R}$, which is related to the magnitude of the transformation, forming what is usually called a *one-parameter group*. For rotations, this parameter will correspond to the angle, for translations, it will be the distance, etc. Because of continuity in the parameter, we can perform a Taylor expansion of the transformation $\mathbf{H}$ for small values of t:

$$\mathbf{H}(\mathbf{x}, t) \approx \mathbf{x} + t\boldsymbol{\Gamma}(\boldsymbol{x}), \quad \boldsymbol{\Gamma}(\boldsymbol{x}) := \left.\frac{\partial \mathbf{H}(\mathbf{x}, t)}{\partial t}\right|_{t=0}. \tag{7}$$

We apply the First Fundamental Theorem of Lie [21, 22] in order to make the following claim: $\boldsymbol{\Gamma}(\boldsymbol{x})$ defines the transformation and is related to what is known as the *generator* of the transformation. Intuitively, this correspondence between action and generator is due to the constraints imposed on the transformation function being a Lie group. The generator can thus be written as a differential operator as follows:

$$G = \sum_{i=1}^{n} \Gamma_i(\boldsymbol{x}) \frac{\partial}{\partial x_i} \tag{8}$$

That is, if one solves the differential equations $\frac{\partial \mathbf{H}(\mathbf{x},t)}{\partial t}|_{t=0}$ that characterize the generator, the original transformation function is obtained. More specifically, the solution is the family of functions topologically connected to the identity transformation through continuity in t. The generator is an element of the *Lie algebra* of the transformation group and is related to the original transformation by what is called the *exponential map*. This nomenclature emphasizes the connection between the differentiation performed in Equation 7 and exponentiation, easily seen when solving the characteristic equation [22], i.e.,

$$\frac{d\mathbf{H}(\boldsymbol{x}, t)}{dt} = G\boldsymbol{H}(\boldsymbol{x}, t), \quad \boldsymbol{H}(\boldsymbol{x}, 0) = \boldsymbol{x}, \tag{9}$$

as the solution is $\boldsymbol{H}(\boldsymbol{x}, t) = e^{tG}\boldsymbol{x}$ with $e^{tG} := \sum_{k=0}^{\infty} \frac{1}{k!} t^k G^k$, where the integer power of G is defined by applying it iteratively. The inverse procedure, which extracts the generator from the action as shown in Equation 7, is also referred to as the *logarithmic map*.

Appendix C. Interpolation Scheme

To apply the operators to a grid, one must write the partial derivatives as matrices. We use the *Shannon-Whittaker interpolation*, as is done in [23] and [6]. This automatically assumes the function to be interpolated is periodic, although other interpolation schemes could have been chosen. We note that this scheme introduces a substantial amount of "leakage", an effect known to cause some aliasing for transformations of low-resolution images, and forms one of the notable limitations of the current model. One could investigate other choices, such as bicubic interpolation, although deriving the differential operator for this scheme requires some additional analysis and is left for future work. Let I be some real-valued signal. For a discrete set of n points on the real line and $I(i+n) = I(i)$ for all samples i from 1 to n, the Shannon-Whittaker interpolation reconstructs the signal for all $r \in \mathbb{R}$ as

$$\begin{aligned} I(r) &= \sum_{i=0}^{n-1} I(i) Q(r-i), \\ Q(r) &= \frac{1}{n}\left[1 + 2\sum_{p=1}^{n/2-1} \cos\left(\frac{2\pi p r}{n}\right)\right]. \end{aligned} \tag{10}$$

To obtain numerical expressions (matrices) for ∂_x, Q can be differentiated with respect to its input. This then describes continuous changes in the one dimensional spatial coordinate at all n points, i.e., $[\boldsymbol{D}_{\mathbb{R}}]_{ab} = \partial_a Q(a-b)$. The above can be extended to two dimensions by performing the Kronecker product of the result obtained for one dimension with the identity matrix, $\boldsymbol{D}_x = \boldsymbol{D}_{\mathbb{R}} \otimes \mathbb{I}$ and $\boldsymbol{D}_y = \mathbb{I} \otimes \boldsymbol{D}_{\mathbb{R}}$, mirroring the flattening operation applied to the input images. The parametrized generator for the 2D affine case, for example, looks like:

$$\boldsymbol{G}_\alpha = \sum_{i=1}^{6} \alpha_i \boldsymbol{D}_i, \tag{11}$$

where the $\boldsymbol{D}_i \in \mathbb{R}^{n^2 \times n^2}$ are the matrices that represent the operators $\partial_x, x\partial_x, y\partial_x, \partial_y, x\partial_y$, and $y\partial_y$, respectively. This can easily be extended to arbitrarily dimensional data by adding more factors to the above matrices, as was done above for the quadratic basis. One can see that performing this operation in pixel space scales poorly with signal length (or image width) n.

Appendix D. Experimental Details

For augmented MNIST, the encoder and decoders are fully-connected MLPs with 512, 256, 128, 64 neurons in each hidden layer with ReLU activation functions. The output layer of the encoder and therefore the input layer of the decoder has size 25. The Adam optimizer [12] was used with a learning rate of 0.001, batch size of 512 and a StepLR scheduler with stepsize 50 and $\gamma = 0.1$. All the results are shown for 1 channel in latent space.

For the CIFAR task, LeakyReLU (negative slope 0.2), wider and deeper MLPs, and various channels in latent space were all checked (1,4,16, and 64). 16 seemed to work best for MSE loss on the reconstructions, but there was no dramatic increase in predictive performance regarding the generators. LayerNorm and batch sizes of 512 and 1024 were used.

Regarding the loss function, the α-*matching* term uses the learned α_i from the latent space (Equation 3) and places them in a generator for the pixel-space, using a Taylor expansion, to compare its effect on the vectorized input, formally

$$\mathcal{L}_\alpha = \left\| \left[\mathbb{I} + t\boldsymbol{G}'_\alpha + \frac{1}{2}t^2(\boldsymbol{G}'_\alpha)^2 + \dots \right] \boldsymbol{x}_0 - \boldsymbol{x}_t \right\|^2, \tag{12}$$

where the prime denotes the basis $\boldsymbol{D}'_i$ evaluated in pixel-space, i.e. $\boldsymbol{G}'_\alpha, \boldsymbol{D}'_i \in \mathbb{R}^{n^2 \times n^2}$. A sparsity loss (LASSO) can also be added in order to enforce the correct behavior in the symbolic regression portion of the symmetry detection.

In order to avoid the ambiguity in the exponent of the matrix exponential (namely, $t\boldsymbol{G} = st \times \boldsymbol{G}/s, \forall s \in \mathbb{R}_0$), the generator is normalized by enforcing the coefficient vector to have unit norm during training. I.e., $||\boldsymbol{\alpha}||^2 = 1$, where $\boldsymbol{\alpha}$ is the vector made up of the coefficients α_i.

Appendix E. Additional Results

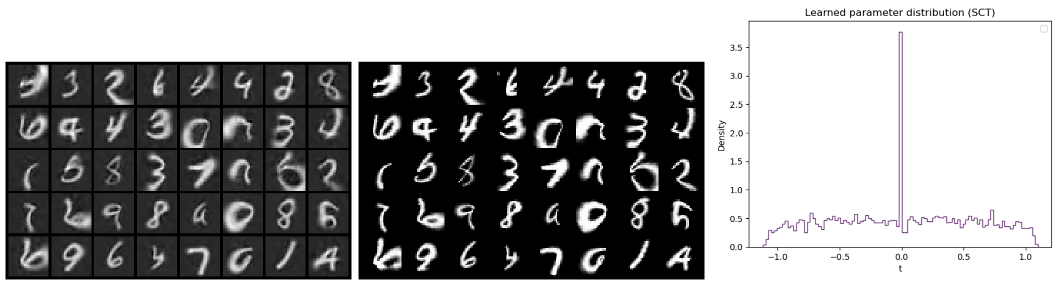

Figure 4: Additional results for SCT applied to MNIST (uniform sampling $b_x \in [-0.8, 0.8]$).

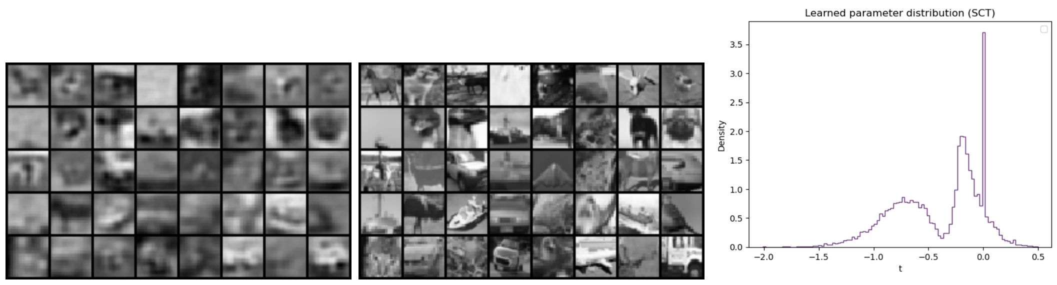

Figure 5: Additional results for scaling applied to CIFAR-10 (sampling $s \in 1.2, 1.8$).

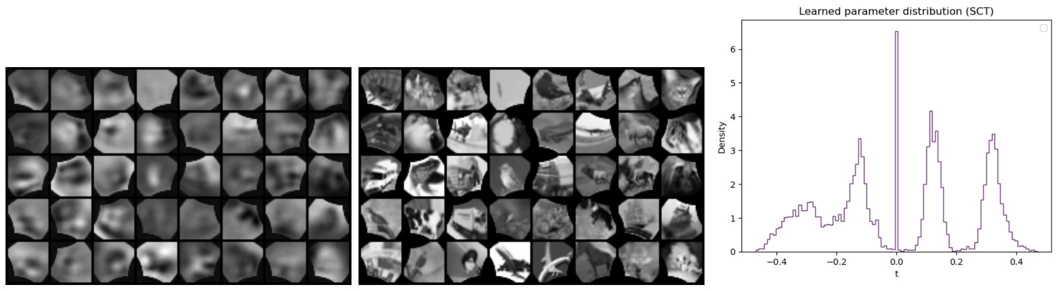

Figure 6: Output and learned histogram for 4-modal SCT transforms on CIFAR-10.

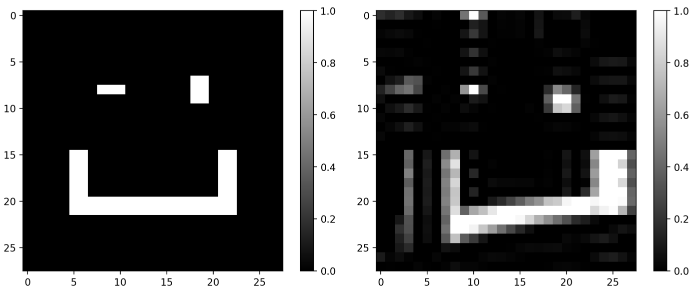

Figure 7: Application of a SCT on a smiley face.