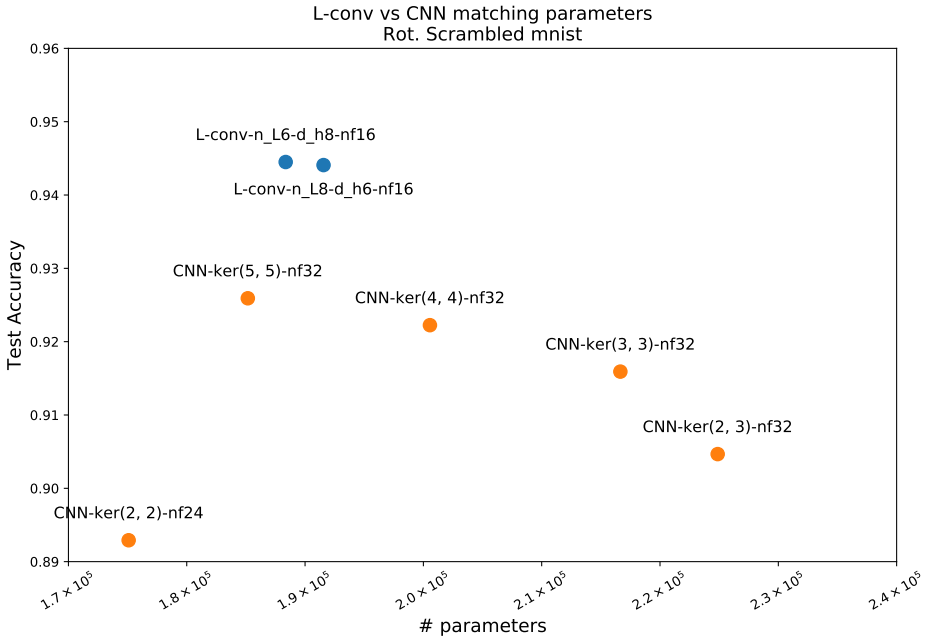

Figure 4: Matching number of parameters in CNN and L-conv, we observe that L-conv still performs better on Rotated and Scrambled MNIST.

A L-CONV EXPERIMENTS

Matching number of parameters in CNN To verify that the difference in the number of parameters between CNN and L-conv was not responsible for the improved performance, we ran experiment where we allowed the kernel-size of L-conv and CNN to differ and tried to match the number of parameters between the two. Fig. 4 shows that on rotated and scrambled MNIST L-conv still performs better than CNN even after the latter has been allowed to have the same or more number of parameters than L-conv.

Hardware and Implementation We implemented L-conv in Keras and Tensorflow 2.2 and ran our tests on a system with a 6 core Intel Core i7 CPU, 32GB RAM, and NVIDIA Quadro P6000 (24GB RAM) GPU. The L-conv layer did not require significantly more resources than CNN and ran only slightly slower.

In Figure 6 we compare the performance of a single layer of L-conv on a classification task on scrambled rotated MNIST, where pixels have been permuted randomly and images have been rotated between -90 to $+90$ degrees. The models consisted of a final classification layer preceded by either one L-conv (blue), or one CNN (orange), or multiple fully-connected (FC, green) layers with similar number of neurons as the L-conv, but without weight sharing. We see that most L-conv configurations had the highest performance without a too many trainable parameters. Note that, parameters in FC layers are much higher than comparable L-conv, but yield worse results. The dots are labeled to show the configurations, with $L[32]h[6](k[6])$ meaning $k = 6$ as number of L_i, 32 output filters, and $h = 6$ hidden dimensions for low-rank encoding of L_i. The y-axis shows the test accuracy and the x-axis the number of trainable parameters. The grey lines show the performance of L-conv with fixed random L_i, but trainable shared wights, showing that indeed the learned L_i improve the performance quite significantly.

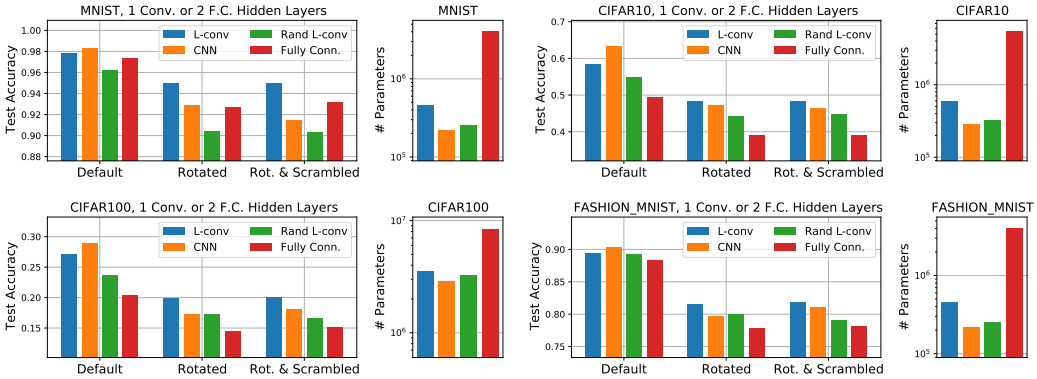

Figure 5: Test results on four datasets with three variant: "Default" (unmodified dataset), "Rotated" and "Rotated and scrambled". On the Default dataset, CNN performs best, but L-conv is always the second best. For Rotated and Rot. & Scrambled, in all cases L-conv performed best. In MNIST, FC and CNN layers come close, but using 5x more parameters.

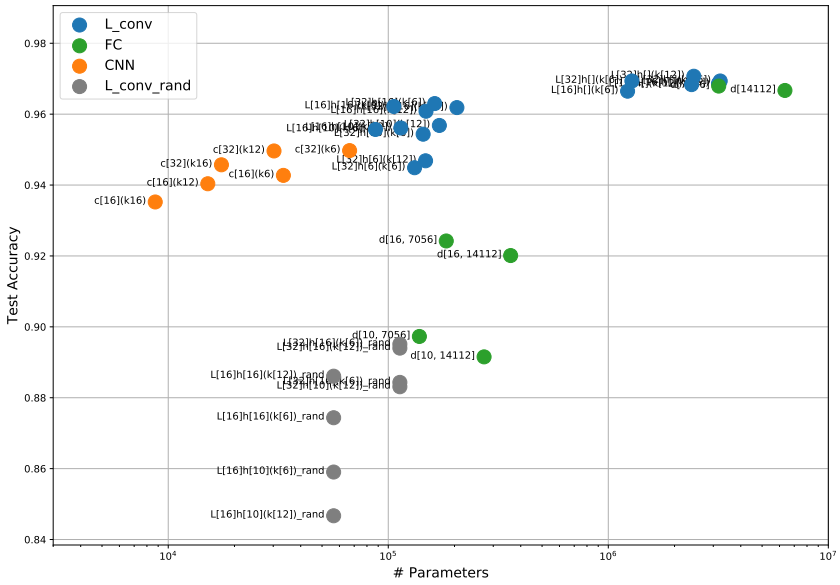

Figure 6: Training low-rank L-conv layer during training.

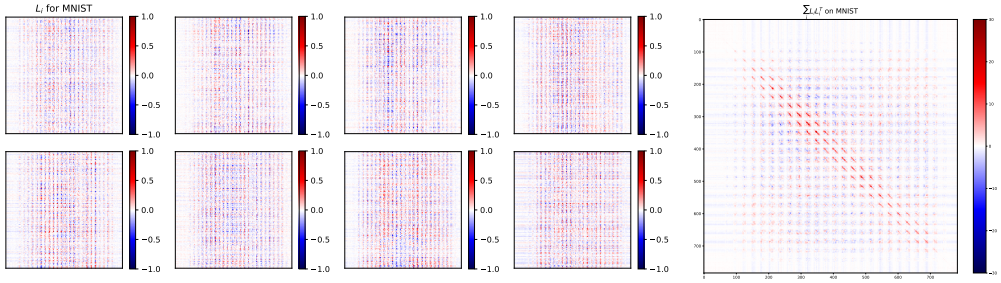

Figure 7: Visualization of the L_i found on MNIST and theit covariance $LL^T = \sum_i L_i L_i^T$.

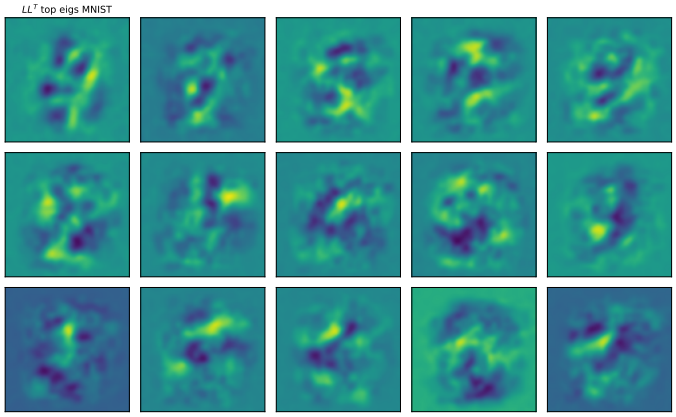

Figure 8: Visualization of the top eigenvectors of $\sum_i L_i L_i^T$. They show some resemblance to the eigenvectors of the covariance matrix $H = XX^T$.

B Structure of Learned L_i

C Pooling

Pooling operations, such maxpooling, generally yield a significant performance boost in CNN. Cohen & Welling (2016a) showed a relation between pooling and coset of subgroups. They also state that the strides in CNN pooling are like subsampling the group to a subgroup $H \subset G$, resulting in outputs which are equivariant only under H and not the full G.

Lastly, unlike an actual Lie algebra basis, we did not include regularizers enforcing orthogonality among L_i.

C.1 Max and Average Pooling

In a 1D CNN, a maxpooling $MP(\cdot)$ with size k and stride s acts on the CNN output $f(x)^a_\mu$ as

$$\begin{aligned} MP(f(x))^a_\mu &= \max\left\{ f(x)^a_{\nu-i} \middle| i \in \{1, \cdots k\}, \nu = s\mu \right\} \\ &= \max_{i \in \{1,\cdots k\}} \left\{ \sigma\left(\sum_{j,c} W_j^{ac} x^c_{s\mu - i - j} + b^a \right) \right\} \end{aligned} \tag{16}$$

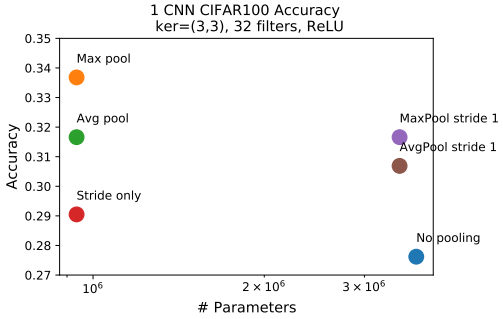

Figure 9: Effect of different pooling schemes. Max pooling yields the best performance, followed by average pooling, which is still significantly higher than no pooling. To check that the increase in accuracy is not simply due to dimensional reduction, we also tested simple strides of (2,2), the same as the strides in pooling, and saw only minor improvement, but the stride combined with pooling has the largest effect.

Using equation 10 and equation 11 we have

$$\begin{aligned} MP(f(x))^a_\mu &= \max_{i\in\{1,\cdots k\}} \left\{ \sigma\left(\sum_{j,c,\nu} W_j^{ac} {L^{(j)}}^{\nu}_{s\mu} x^c_{\nu-i} + b^a \right) \right\} \\ &= \max_{i\in\{1,\cdots k\}} \left\{ \sigma\left(\sum_{j,c,\nu,\eta} W_j^{ac} {L^{(j)}}^{\nu}_{s\mu} {L^{(i)}}^{\eta}_{\nu} x^c_\eta + b^a \right) \right\} \end{aligned} \tag{17}$$

Now, we note that most commonly used activation functions, such as ReLU, sigmoid, tanh, etc are all monotonically increasing. This mean that the max operation and the activation function commute, meaning for a set of inputs $S = \{x_1, \cdots x_n\}$

$$\sigma\left(\max\left[S\right]\right) = \max[\sigma(S)] \tag{18}$$

Using equation 18 in equation 17 we get

$$MP(f(x))^a_\mu = \sigma\left(\max_{i\in\{1,\cdots k\}} \left\{ \sum_{j,c,\nu,\eta} W_j^{ac} {L^{(j)}}^{\nu}_{s\mu} {L^{(i)}}^{\eta}_{\nu} x^c_\eta \right\} + b^a \right). \tag{19}$$

Adding and subtracting $W_j^{ac} {L^{(i)}}^{\nu}_{s\mu} {L^{(j)}}^{\eta}_{\nu}$, we notice that

$$\sum_\nu W_j^{ac} {L^{(j)}}^{\nu}_{s\mu} {L^{(i)}}^{\eta}_{\nu} = \left(W_j^{ac} \left[L^{(j)}, L^{(i)} \right] + \frac{1}{2} W_j^{ac} \left\{ L^{(j)}, L^{(i)} \right\} \right)^{\eta}_{s\mu} \tag{20}$$

where $\{A, B\} = AB + BA$ is the anti-commutator.

D Linear Regression

Consider a linear regression problem on inputs $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ and labels $\boldsymbol{Y} \in \mathbb{R}^{c\times n}$ like above. We are looking for the linear function $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{b}$. For brevity, we will absorb $\boldsymbol{b}$ into $\boldsymbol{A}$ as its last row and append a row of 0 to the the end of $\boldsymbol{X}$. Assuming this problem is equivariant under a group G, and using the same group element u in two representations $u \in T_d(G)$ and $u_c \in T_c(G)$ acting on $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively, equation 1 becomes

$$u_c \boldsymbol{Y} = u_c \boldsymbol{A}\boldsymbol{X} = \boldsymbol{A} u \boldsymbol{X} \qquad \Rightarrow \boldsymbol{A} = u_c \boldsymbol{A} u^{-1} \tag{21}$$

Assuming that the number of samples is much greater than features, $n \gg d$, and unbiased data, $\boldsymbol{X}^T\boldsymbol{X}$ will be full rank $d \times d$ and that its inverse exists. The solution to the linear regression $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ is given by $\boldsymbol{A} = \boldsymbol{Y}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1}$. Defining the covariance matrix $\boldsymbol{H}$, the condition equation 21 becomes

$$\boldsymbol{H} \equiv \frac{1}{n}\boldsymbol{X}\boldsymbol{X}^T, \qquad \boldsymbol{A} = \frac{1}{n} u_c \boldsymbol{Y}\boldsymbol{X}^T u^T (u \boldsymbol{H} u^T)^{-1} \tag{22}$$

Instead of the most general solutions to equation 21, let us consider a subset of them which are naturally separated by equation 22 into a supervised and unsupervised part as

$$u\boldsymbol{H}u^T = \boldsymbol{H} \qquad\qquad u_c\boldsymbol{Y}\boldsymbol{X}^T u^T = \boldsymbol{Y}\boldsymbol{X}^T. \tag{23}$$

The first condition states that u preserves the covariance $\boldsymbol{H}$ as a bilinear form, while the second condition relates u_c and u through overlap of input and labels $\boldsymbol{Y}\boldsymbol{X}^T$.

D.1 Deformed Orthogonal Group as Symmetries of Linear Regression

Since $\boldsymbol{H}$ is positive definite, the u satisfying $u\boldsymbol{H}u^T = \boldsymbol{H}$ define an orthogonal group, which we denote by $O_H(d)$. The familiar orthogonal group $O(d)$, where $uu^T = I$ corresponds to $\boldsymbol{H} = I$. Similar to $O(d)$, we have

$$\det(u\boldsymbol{H}u^T) = \det(\boldsymbol{H}) \quad \Rightarrow \quad \det(u)^2 = 1 \tag{24}$$

$O_H(d)$ is simply a reparametrization of $O(d)$ and their group manifolds are isomorphic, having two identical component like $\pm SO(d)$, the special orthogonal group. The only difference is that the manifold $O_H(d)$ is deformed. To see this, define $u' = \boldsymbol{H}^{-1/2}u\boldsymbol{H}^{1/2}$. From equation 23, we have $u'u'^T = I$, meaning $u' \in O(d)$. We will first focus on the subgroup $SO_H(d)$ where $\det(u) = 1$.

D.2 Lie Algebra of Linear Regression Symmetries

From equation 23, we have $u'u'^T = I$, meaning $u' \in O(d)$. We will first focus on the subgroup $SO_H(d)$ where $\det(u) = 1$. Note that since $\boldsymbol{H}$ is positive-definite, $SO_H(d)$ is a compact group, isomorphic to $SO(d)$. Using generators of $SO_H(d)$, infinitesimal elements near the identity can be written as $u \approx I + \epsilon \cdot L$, which yields $u\boldsymbol{H}u^T \approx \boldsymbol{H} + \epsilon \cdot \left(L\boldsymbol{H} + \boldsymbol{H}L^T\right) = \boldsymbol{H}$, and $u\overline{\boldsymbol{X}}_a = \overline{\boldsymbol{X}}_a$ leads to $\epsilon \cdot L\overline{\boldsymbol{X}}_a = 0$. As a result, the conditions in equation 23 require that the generators satisfy

$$L_i\boldsymbol{H} = -\boldsymbol{H}L_i^T, \qquad\qquad L_i\overline{\boldsymbol{X}}_a = 0, \qquad\qquad \mathrm{Tr}\,[L_i] = 0 \tag{25}$$

where the $\mathrm{Tr}\,[L_i] = 0$ condition follows from $\det(u) = 1$, which can be seen by diagonalizing u. The L_i form the Lie algebra of $SO_H(d)$, denoted as $so_H(d)$, which is isomorphic to $so(d)$. The generators of $so(d)$ are $d(d-1)/2$ skew-symmetric matrices. Similar to the isomorphism $SO_H(d) \cong SO(d)$, the generators can be mapped by realizing $L'_i = \boldsymbol{H}^{-1/2}L_i\boldsymbol{H}^{1/2}$ satisfies $L'_i = -{L'_i}^T$ and

$$[L'_i, L'_i] = {f_{ij}}^k L'_k = \boldsymbol{H}^{-1/2}[L_i, L_j]\boldsymbol{H}^{1/2} \tag{26}$$

showing that the structure constants ${f_{ij}}^k$ of $so(d)$ also being identical to $so_H(d)$. We can therefore choose the canonical basis used for $SO(d)$ to build generators for $SO_H(d)$. Thus we have

$$L_i = \boldsymbol{H}^{1/2}L'_i\boldsymbol{H}^{-1/2} \qquad\qquad L'_i \in so(d) \tag{27}$$

In addition to this, only the subset of generators of $SO_H(d)$ which preserve the average input $\overline{\boldsymbol{X}}_a$ for each class a are acceptable as symmetries. Below, we show how these symmetries can be found by optimization.

D.3 Restricted Search Space and Classification

The second condition in equation 23 relating u_c and u is more involved. In principle, for any choice of representation of group action u on labels $\boldsymbol{Y}$, there will a subset of $SO_H(d)$ which satisfies the second condition in equation 23. To study a concrete case and constrain the search space, we will derive the conditions for the case where the $u_c\boldsymbol{Y} = \boldsymbol{Y}$, meaning the group action on labels is trivial and keeps the labels invariant. Another question of interest is whether symmetries found in the linear learning problem can be employed in nonlinear settings. For instance, would incorporating linear symmetries into a nonlinear neural network architecture help in a classification problem? Hence, instead of a general linear regression problem, we consider a case where we use label-invariant symmetries of linear regression on a dataset with discrete categorical labels.

D.4 EQUIVARIANCE AND INVARIANCE WHEN LABELS ARE CATEGORICAL

We will first show that if G is continuous and connected, then in problems with categorical labels such as classification, the only valid equivariances either keep the labels invariant, or only include a discrete subgroup of G.

Lemma 1. *The only representation of a connected Lie group G on $\mathbb{Z}_n$ are constant.*

Proof: A representation is a smooth homomorphism, meaning that it is continuous and infinitely differentiable. Since G is a connected Lie and hence topological group, it has a connected continuous manifold. A function $T : G \to \mathbb{Z}_n$ is continuous functions if its pre-image in G is open. This implies that if $T(u) = z$, there exist an open ball $u \in B_u$ on which T is constant ($T(u') = z, \forall u' \in B_u$. Consider two $u, v \in G$ for which $T(u) \neq T(v)$, belonging to open balls $u \in B_u$ and $v \in B_v$ over which T is constant. Since $T(u) \neq T(v)$ we must have $B_u \cap B_v = \emptyset$. Therefore, any element in the boundary of B_u cannot be in any other B_v, meaning the domain of T is not all elements in G and hence it cannot be a smooth homomorphism. Thus, the only smooth $T : G \to \mathbb{Z}_n$ are constant functions. □

Corollary 2. *In supervised learning with continuous inputs $\boldsymbol{x} \in \mathbb{R}^d$ and categorical outputs $y \in \mathbb{Z}_2^m$, if the dataset is equivariant under a connected Lie group G the only possible equivariance is label invariance, or equivariance under discrete subgroups of G.*

Proof: From Lemma 1, if G is continuous and connected the only representations $T : G \to \mathbb{Z}_2$ must be constant, meaning the labels are kept invariant. For discrete subgroups of G this restriction does not exist. □

D.5 LABEL-INVARIANT SYMMETRIES IN LINEAR REGRESSION

When $n \gg c$ and the dataset is fairly unbiased, using the Moore-Penrose pseudo-inverse $\boldsymbol{Y}\tilde{\boldsymbol{Y}}^{-1T} = I_{c\times c} = u$ the identity matrix. Assume $\boldsymbol{y}_i \in \mathbb{Z}_2^c$ are one-hot vectors as in a classification problem. With discrete labels, $\boldsymbol{A}$ will be projecting a subset of features as captured by $\tilde{\boldsymbol{X}}^{-1}$ to each label class. Denoting the samples belonging to label class a as $\boldsymbol{y}_i = \{a\}$ and y_a a one-hot vector nonzero at entry a, we find the mean of all inputs $\overline{\boldsymbol{X}}_a$ for a given class a appears in

$$\overline{\boldsymbol{X}}_a \equiv \frac{1}{n}\sum_{\boldsymbol{y}_i=\{a\}} \boldsymbol{X}, \qquad \boldsymbol{Y}\boldsymbol{X}^T = n\sum_{a=1}^{c} \boldsymbol{y}_a \overline{\boldsymbol{X}}_a^T \qquad \boldsymbol{A}_a = \overline{\boldsymbol{X}}_a^T \boldsymbol{H}^{-1}. \tag{28}$$

$\boldsymbol{A}$ is a $c \times d$ matrix and is the row $\boldsymbol{A}_{a,:}$ corresponding to class a. With $u_c = I$, equation 21 states that a symmetry should preserve each row as $\boldsymbol{A}_{a,:} = \boldsymbol{A}_{a,:}u$. The second condition in equation 23 becomes

$$u\overline{\boldsymbol{X}}_a = \overline{\boldsymbol{X}}_a \tag{29}$$

equation 29 is much more restrictive than in equation 23 and requires u to also completely preserve mean class input $\overline{\boldsymbol{X}}_a$. Note that, unlike $u_c\boldsymbol{Y} = \boldsymbol{Y}$, which resulted in $u_c = I$ for balanced classes, $u\overline{\boldsymbol{X}}_a = \overline{\boldsymbol{X}}_a$ does not make u trivial because $\overline{\boldsymbol{X}}_c$ generally spans only a subspace of features and is not full-rank.

Finding u which satisfy $u\boldsymbol{H}u^T$ is a well-defined problem and we will show below how to find some u using properly regularized optimization. While the symmetry group $SO_H(d)$ is a continuous Lie group, in most cases it will be finitely generated. Next, we review the conditions on the generators of the group.

E EXTRACTING SYMMETRIES IN LINEAR REGRESSION

Using the $L_a\overline{\boldsymbol{X}}_c = 0$ condition in equation 25, the symmetries shared among all classes can be found by minimizing the loss function

$$\mathcal{L}_{sym}(\epsilon) = \sum_a \|\epsilon \cdot L\overline{\boldsymbol{X}}_a\|^2 + \lambda_2\left(\|\epsilon\|^2 - 1\right)^2 \tag{30}$$

where the second term is the L_2 regularization to enforce $\|\epsilon\| = 1$ and avoid vanishing ϵ. Every $n_L = d(d-1)/2$ dimensional vector ϵ for which $\mathcal{L}_{sym}(\epsilon) = 0$ is a symmetry generator for all label classes. The maximum number of possible symmetry generator $\epsilon \cdot L$ is n_L, but in practice, we expect to find much fewer linearly independent symmetries. To make sure different generators are independent, we use the Cartan-Killing (CK) form, which defines a metric on the Lie algebra. For $so(d)$, the CK form is simply $B(L, K) = (D-2)\mathrm{Tr}\,[LK]$. Using equation 27, and the orthoganality $\mathrm{Tr}\left[L'_i L'_j\right] = \delta_{ij}$ we have

$$\begin{aligned} L_{(i)} &\equiv \sum_j \epsilon_i^j L_j \\ B(L_{(i)}, L_{(j)}) &= \sum_{k,l} \epsilon_i^k \epsilon_j^l \mathrm{Tr}\left[H^{1/2} L'_k L'_l H^{-1/2}\right] = \epsilon_i^T \epsilon_j \end{aligned} \tag{31}$$

We will look for $n \leq n_L$ generators by finding n vectors ϵ_i which minimize the total loss

$$\mathcal{L}\left(\epsilon_1, \cdots, \epsilon_n\right) = \sum_{i=1}^{n} \mathcal{L}_{sym}(\epsilon_i) + \lambda_{CK} \sum_{i,j=1}^{n} \|\epsilon_i^T \epsilon_j\|^2. \tag{32}$$

Here $n \leq n_L$ is a hyperparameter because only a subset of the symmetries may preserve all label classes.

Implementation In practice, L_i can be encoded using skew-symmetric matrices. Similar to $GL(d)$ equation 8, the generators of $SO(d)$ can be labeled with two anti-symmetric indices ($A_{[ij]k} \equiv A_{ijk} - A_{jik}$)

$$[L'_i]^\nu_\mu = \boldsymbol{E}_{[\alpha\beta]}{}^\nu_\mu = \delta_{\alpha\mu}\delta^\nu_\beta - \delta_{\beta\mu}\delta^\nu_\alpha \tag{33}$$

We can also express ϵ using two anti-symmetric indices $\epsilon_i = \hat{\epsilon}_{\alpha\beta}$, where $\hat{\epsilon}$ is a skew-symmetric matrix, and get

$$[\epsilon \cdot L]_{\mu\nu} = \sum_{\alpha\beta\lambda\rho} \hat{\epsilon}_{\alpha\beta} H^{1/2}_{\mu\lambda} \boldsymbol{E}_{[\alpha\beta]}{}^\rho_\lambda H^{-1/2}_{\rho\nu} = \sum_{\alpha\beta} \hat{\epsilon}^{\alpha\beta} H^{1/2}_{\mu[\alpha} H^{-1/2}_{\beta]\nu} = \left[H^{1/2}\hat{\epsilon}H^{-1/2}\right]_{\mu\nu}. \tag{34}$$

Hence, the generators L_i have a structure similar to L_i except that the canonical $SO(d)$ generators L'_a are replaced with a general skew-symmetric matrix $\hat{\epsilon}$.

E.1 Regularizers

The direction in the parameter space which are related to each other via symmetries will have the same value of loss and are continuously connected to each other. This means that gradient vanishes along the direction of all generators of the symmetry group, making the optimization process find different u with each initialization. To make this problem slightly more convex, we introduce additional constraints on the form u using regularization terms. The full regularization has three terms. The first term is regularization on $L_{(i)}X_c = 0$. The second term is regularization on the orthogonality of different $\hat{\epsilon}_i$. The third term is regularization on the sparsity of $\hat{\epsilon}_i$, namely that each row and column has two non-zero elements with absolute value equal to 1, and other elements equal to 0.

F Finding Symmetries Using Optimization

We use optimization based on equation 32 and equation 34 to find generators L of the symmetries of a dataset directly. As in $so(D)$ the simplest generators only rotate a 2D subspace, i.e. may only mix two pixels in the image. To ensure that we find generators which act more globally on all pixels, we add a third regularization term to equation 32, as below.

Global Sparsity Regularizer This term ensures both having a global effect as well as the sparsity of $\hat{\epsilon}$, in the form of having only two non-zero element in each row and column with one equal to 1

and the other equal to -1 (since $\hat{\epsilon}$ is skew-symmetric). The regularizer is given by

$$\mathcal{L}_{sparse}(\hat{\epsilon}) = \sqrt{\sum_{i=0}^{n}\left[\left(\sum_{j=0}^{n}|\hat{\epsilon}_{ij}|\right)-2\right]^2} + \sqrt{\sum_{j=0}^{n}\left[\left(\sum_{i=0}^{n}|\hat{\epsilon}_{ij}|\right)-2\right]^2} + \sqrt{\sum_{i=0}^{n}\left[\max_j\left(|\hat{\epsilon}_{ij}|\right)-1\right]^2} + \sqrt{\sum_{j=0}^{n}\left[\max_i\left(|\hat{\epsilon}_{ij}|\right)-1\right]^2} \tag{35}$$

In addition to the Rescaled and Rotated MNIST datasets shown in the paper, we tested this methodology on other synthetic datasets, described in sec. G.

Finding Common Generators In order to find the larest set of orthogonal $\hat{\epsilon}$, we use the Bron-Kerbosch algorithm, an algorithm designed to find all cliques (fully connected components) in a network. We first build a network of all $\hat{\epsilon}$ the optimization finds, to which an edge between $\hat{\epsilon}_i$ and $\hat{\epsilon}_j$ is added if $\mathrm{Tr}\left[\hat{\epsilon}_i\hat{\epsilon}_j\right]$ is less than the threshold 0.01. We then use the Bron-Kerbosch algorithm to find all cliques in this network, and choose the one with the largest size. The $\hat{\epsilon}$ in this largest clique are the orthogonal $\hat{\epsilon}$.

F.1 Alternative Optimization

To find symmetries in linear regression, we can look for Q satisfying equation 23 by minimizing $\|uHu^T - H\|^2$. However, this is not a convex optimization, as a continuum of minima exist, corresponding to different group elements u. Unlike $SO_H(d)$ which is an infinite group, the number of linearly independent generators in $so_H(d)$ are finite, $d(d-1)/2$ to be precise. Therefore, we will be looking for L_a satisfying equation 25 by minimizing the following loss

$$\mathcal{L}_0(L_a) = \|L_aH + HL_a^T\|^2 + \lambda_{tr}\mathrm{Tr}\left[L_a\right]^2. \tag{36}$$

where $\|A\|^2 = \mathrm{Tr}\left[AA^T\right]$ is the Frobenius norm and λ_{tr} is the Lagrange multiplier for the $\mathrm{Tr}\left[L_a\right] = 0$ regularizer. We have $n_L = d(d-1)/2$ generators, and we want the procedure to find all of them. Yet simply minimizng $\mathcal{L}(\{L\}) = \sum_a \mathcal{L}(L_a)$ won't be enough as it can lead to many similar generators. To resolve this, we use a metric defined on the Lie algebra called the Cartan-Killing form.

F.2 Orthogonality and Cartan-Killing Form

The generators L_a of $SO_H(d)$ form a Lie algebra $[L_a, L_b] = f_{ab}{}^cL_c$. The Cartan-Killing (CK) form $B : so_H(d) \times so_H(d) \to R$ defines a metric on the Lie algebra via

$$B(L_a, L_b) = \sum_\mu [L_a, [L_b, e_\mu]]^T e_\mu \tag{37}$$

where e_μ are a set of basis matrices for $so_H(d)$. We want to find independent generators by enforcing the orthogonality condition $B(L_a, L_b) = k\delta_{ab}$. For compact Lie groups, such as the orthogonal groups, B is negative definite meaning $k < 0$. But the exact value of k depends on the group. For $so(d)$, $B(X, Y) = (d-2)\mathrm{Tr}\left[XY\right]$, but this may change slightly for $so_H(d)$. To enforce orthogonality we add the following term to our loss function when

$$\mathcal{L}_{CK}(L_a, L_b) = \lambda_{CK}\sum_R\left(\tanh\left(r\left\|[[L_a, [L_b, R]]R^T\right\|^2\right) - \delta_{ab}\right)^2 \tag{38}$$

where R are random matrices with $\mathrm{Tr}\left[R^TR\right] = 1$ and $r \gg 1$. The intuition for the $\tanh(r\|B(L_a, L_b)\|^2)$ is that, since we don't know the normalization constant for the CK form, we only want to make sure that $B(L_a, L_b) \propto \delta_{ab}$, while allowing the magnitude of $B(L_a, L_b)$ to be arbitrary. Minimizing $\mathcal{L}_{CK}$ over sufficiently many R it should enforce the orthogonality condition. The full loss function to optimize is then

$$\mathcal{L}(\{L\}) = \sum_a \mathcal{L}_0(L_a) + \sum_R\sum_{a,b}\mathcal{L}_{CK}(L_a, L_b; R) \tag{39}$$

where the sum over a and b runs to $n_L = d(d-1)/2$.

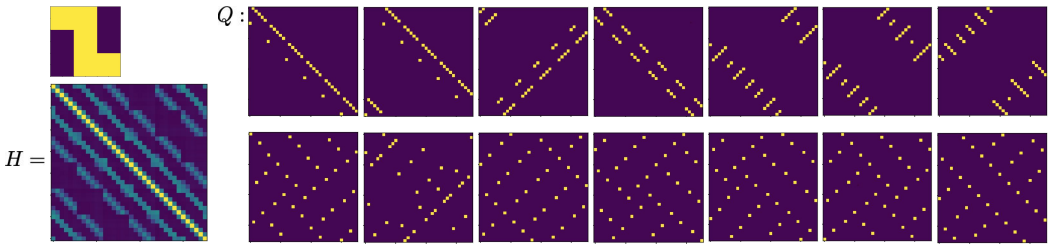

Figure 10: We choose a simple 3x3 pattern and put it in a 6x6 empty image, and then shift the pattern randomly with periodic boundary conditions. Then we attempt to find symmetry in 1000 samples generated in this fashion by finding the symmetry of the correlation coefficient matrix H of these images. We found that our method of symmetry extraction is very effective for this dataset, and we are able to find many operators Q such that $QHQ^T = H$. Upon observing these Q operators, we found that some of them have straightforward interpretations, for example, the first Q in the first row represents shifting an 6x6 image in the horizontal axis by one pixel, and the second Q in the first row represents shifting an 6x6 image in the horizontal axis by two pixels. Some of the Q operators we found have less obvious interpretations, because they are a combination of multiple "simple" operations such as shifting by horizontal or vertical axis.

G Additional Synthetic Datasets

We have tested this optimization algorithm on several different datasets. The first one is $1,000$ 6×6 images of one simple 3×3 pattern (Fig. 11a) shifted randomly with periodic boundary conditions. We chose the pattern deliberately to avoid rotational or reflectional symmetry in the pattern itself. Fig. 11b shows the correlation matrix of the $1,000$ images, H. Fig. 11c shows the number of orthogonal $\hat{\epsilon}$ found, as a function of the input parameter num_L that governs the total number of $\hat{\epsilon}$ the optimization is trying to find. In practice the orthogonality of $\hat{\epsilon}$ is determined by $\mathrm{Tr}\left[\hat{\epsilon}_i \hat{\epsilon}_j\right] < 0.01$. We can see that the number of orthogonal $\hat{\epsilon}$ increases to 3 quickly and then stay the same even as num_L increases to 10. Therefore, we believe that for this dataset, there are 3 independent symmetries, with 3 orthogonal generators. The 3 independent $\hat{\epsilon}$, $L = H^{\frac{1}{2}} \hat{\epsilon} H^{-\frac{1}{2}}$, and $u = \exp[L]$ are shown in Fig. 11d. The left image in Fig. 11e) shows one of the 1000 images in this dataset. We applied the three independent us onto this image and the resulting images are shown in the rest of Fig. 11e). Fig. 11f) shows the effect of us on a gradient image. In each of Fig. 11e) and Fig. 11f), the images are transformed into other images that are similar to the original, but not exactly the same. It is not very clear what the transformations do.

The second dataset we considered is $2,000$ 6×6 images of two simple 3×3 patterns, 1000 images for each (Fig. 12a), shifted randomly with periodic boundary conditions. Similar to the one pattern dataset, we show our results in Fig 12. We can see that the $\hat{\epsilon}$ all have stripe structure alone the diagonal direction. From Fig 12g we can see that when the three independent symmetry operators u act on a gradient matrix, the resulting matrices have wave-like structure, suggesting that the u we found are meaningful.

H Intersection with $SO(d)$

Some symmetries may be in the intersection $\hat{u} \in SO_H(d) \cap SO(d)$, meaning $\hat{u}^T = \hat{u}^{-1}$. We can exploit these to find generators for $so_H(d)$. It follows

$$\hat{L} \equiv \hat{u} - \hat{u}^T \qquad \hat{L}^T = -\hat{L} \qquad \mathrm{Tr}\left[\hat{L}\right] = 0 \qquad \hat{L}H = H\hat{L} = -H\hat{L}^T. \tag{40}$$

Therefore, if $\hat{u} \in SO_H(d) \cap SO(d)$, then $\hat{L} \in so_H(d) \cap so(d)$. Following 40, we first find $\hat{u}$ first through optimization, with loss function ensuring that $\hat{u}$ commutes with H, $\hat{u}$ has determinant 1, and a sparsity condition which is that $\hat{u}$ has one non-zero element in each row and column, and it is equal to one, and finally all $\hat{u}$ are orthogonal with each other, namely $\mathrm{Tr}\left[\hat{u}_1 \hat{u}_2\right] = 0$. Examples of $\hat{u}$ can be found in Fig. 10. Then we construct the generators $\hat{L}$ from $\hat{u}$.

H.1 Experiment Datasets

We use three datasets to test our architecture. The first dataset is rescaled MNIST, where we take the original images in the MNIST dataset (which are 28×28 images) and rescale it to 6×6. The second

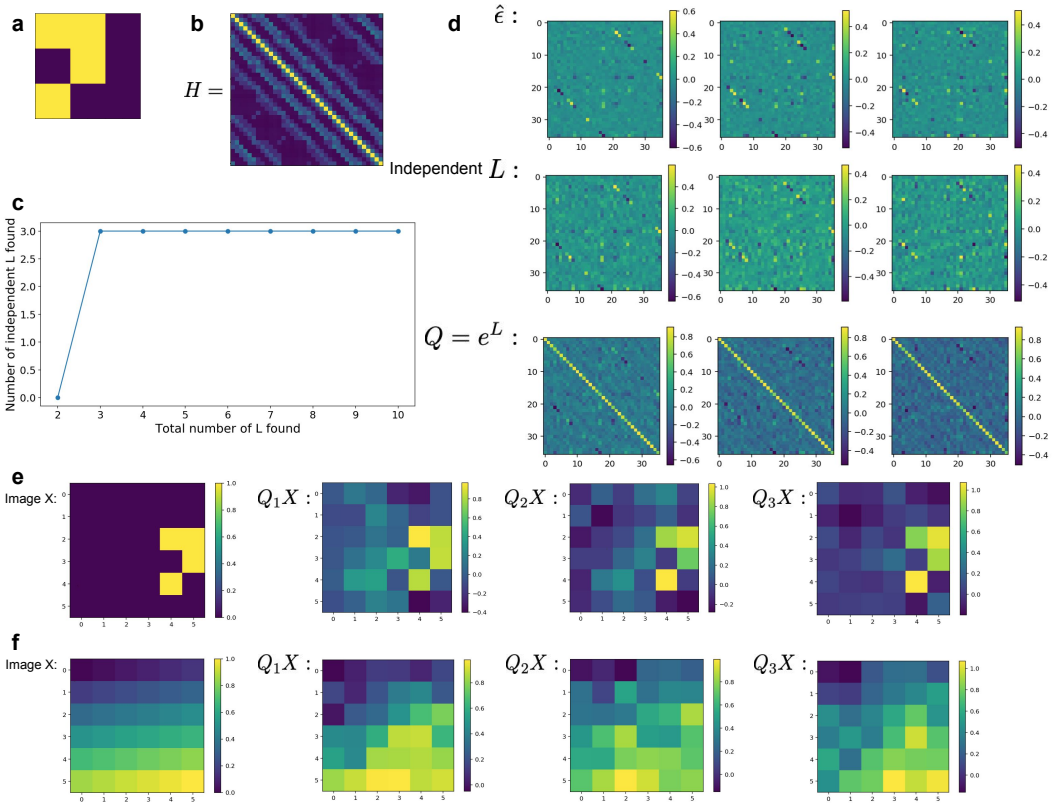

Figure 11: **Single class of pattern.** **a.** The pattern that is being shifted in different images in the dataset. **b.** The correlation matrix H for the 1000 images with shifted pattern in **a**. **c.** The number of independent generators L as a function of the total number of L, one of the input parameters. **d.** The independent $\hat{\epsilon}$, the L, and $Q = e^L$. **e.** One image X from the dataset, and the resulting images after symmetry operators Q act on it. **f.** One image X of a gradient, and the resulting images after symmetry operators Q act on it.

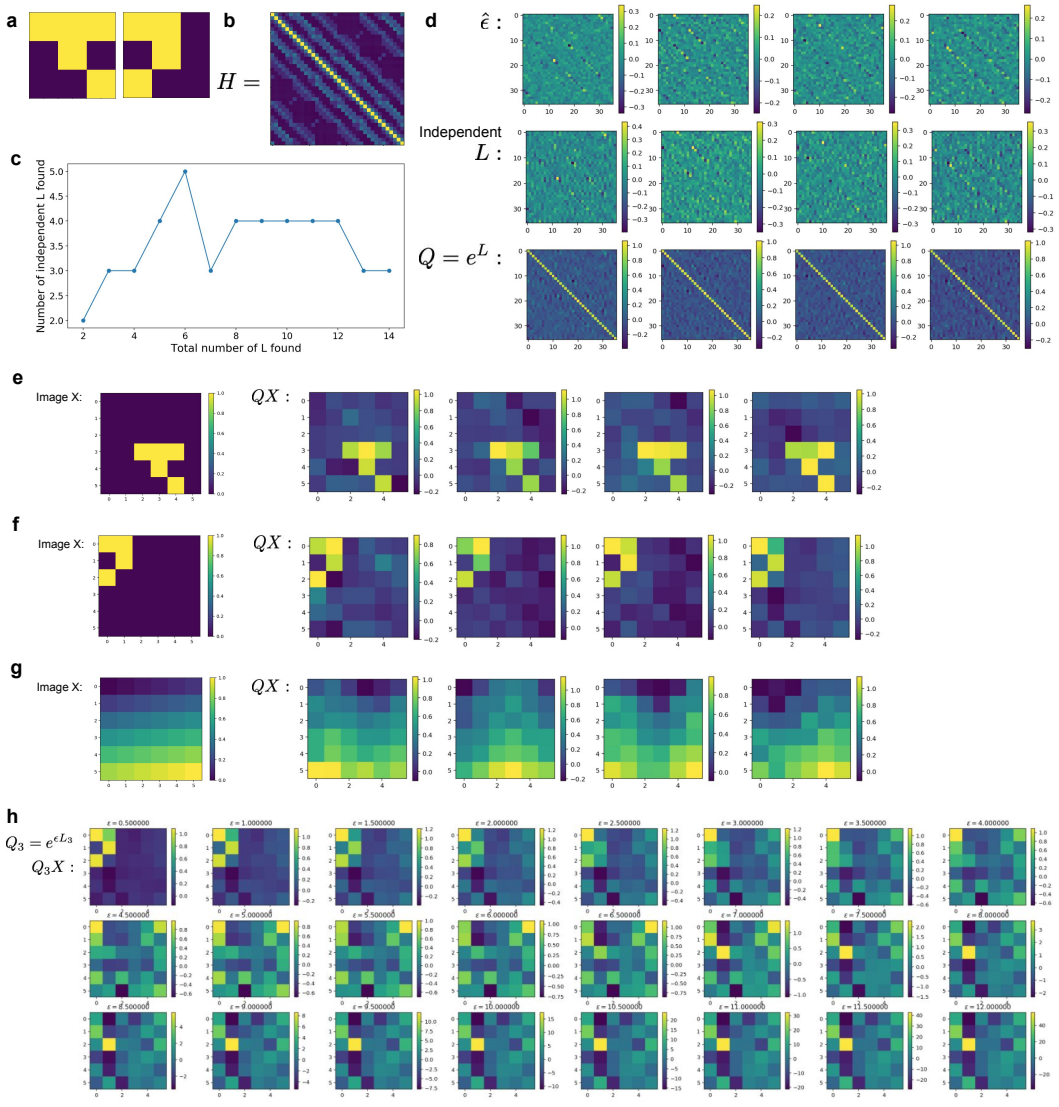

Figure 12: **Two classes of patterns.** **a.** The two pattern that are being shifted in different images in the dataset. **b.** The correlation matrix H for the $2,000$ images with shifted patterns in **a**, $1,000$ images for each pattern. **c.** The number of independent generators L as a function of the total number of L, one of the input parameters. **d.** The independent $\hat{\epsilon}$, the L, and $Q = e^L$. **e.** One image X from the dataset with the first pattern in **a**, and the resulting images after symmetry operators Q act on it. **f.** One image X from the dataset with the second pattern in **a**, and the resulting images after symmetry operators Q act on it. **g.** One image X of a gradient, and the resulting images after symmetry operators Q act on it. **h.** Q_3 with the third generator L_3 with different ϵ values acting on image X in **f**.

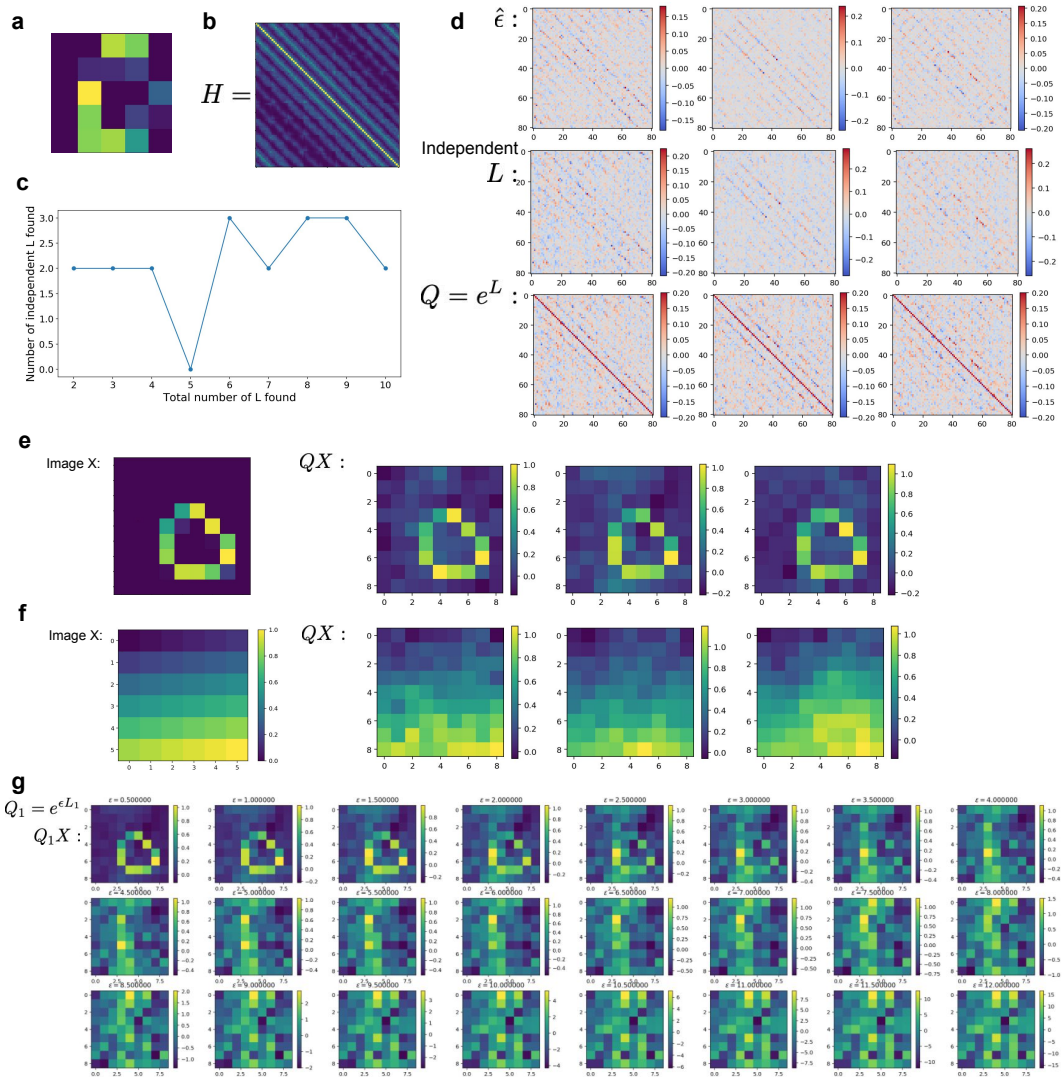

Figure 13: **Two classes of patterns.** **a.** One example of a zero in the MNIST dataset. **b.** The correlation matrix H for the $1,000$ images with shifted zeros in MNIST dataset. **c.** The number of independent generators L as a function of the total number of L, one of the input parameters. **d.** The independent $\hat{\epsilon}$, the L, and $Q = e^L$. **e.** One image X from the dataset, and the resulting images after symmetry operators Q act on it. **f.** One image X of a gradient, and the resulting images after symmetry operators Q act on it. **g.** Q_1 with the third generator L_1 with different ϵ values acting on image X in **e**.

dataset is rotated MNIST, where we take the original MNIST images, select the images in class zero, one, two, three, four, six, seven, eight. We excluded classes five and nine because five looks like two rotated by 180 degrees and nine looks like six rotated by 180 degrees. We don't want the symmetry between those digits to interfere with the symmetry of rotated images themselves. Then we take the images in these classes and rotate them with angle $\theta \in [-\pi/2, \pi/2]$. We take a randomly sampled θ in that range for each image. The third is scrambled MNIST, where we first rescale images to 14×14, and then rearrange the pixels to a fixed random order for every image.