A Simple Strategy to Provable Invariance via Orbit Mapping

Kanchana Vaishnavi Gandikota[1], Jonas Geiping[2], Zorah Lähner[1], Adam Czapliński[1], Michael Möller[1]

[1] University of Siegen,[2] University of Maryland

Abstract. Many applications require robustness, or ideally invariance, of neural networks to certain transformations of input data. Most commonly, this requirement is addressed by training data augmentation, using adversarial training, or defining network architectures that include the desired invariance by design. In this work, we propose a method to make network architectures provably invariant with respect to group actions by choosing one element from a (possibly continuous) orbit based on a fixed criterion. In a nutshell, we intend to 'undo' any possible transformation before feeding the data into the actual network. Further, we empirically analyze the properties of different approaches which incorporate invariance via training or architecture, and demonstrate the advantages of our method in terms of robustness and computational efficiency. In particular, we investigate the robustness with respect to rotations of images (which can hold up to discretization artifacts) as well as the provable orientation and scaling invariance of 3D point cloud classification.

1 Introduction

Deep neural networks have revolutionized the field of computer vision over the past decade. Yet, deep networks trained in a straight-forward way often lack desired robustness. In image classification, for instance, rotational, scale, and shift invariance are often highly desirable properties. While training deep networks with millions of realistic images in datasets like Imagenet [1] confers some degree of in/equi-variance [2,3,4], these properties however, cannot be guaranteed. On the contrary, networks are susceptible to adversarial attacks with respect to these transformations (see e.g. [5,6,7,8]), and small perturbations can significantly affect their predictions. To counteract this behavior, the two major directions of research are to either modify the training procedure or the network architecture. Modifications of the training procedure replace the common training of a network $\mathcal{G}$ with parameters θ on training examples (x^i, y^i) via a loss function $\mathcal{L}$,

$$\min_{\theta} \sum_{\text{examples i}} \mathcal{L}(\mathcal{G}(x^i;\theta); y^i), \tag{1}$$

with a loss function that considers all perturbations in a given set S of transformations to be invariant towards. The most common choices are taking the mean loss of all predictions $\{\mathcal{G}(g(x^i);\theta) \mid g \in S\}$ (training with *data augmentation*), or

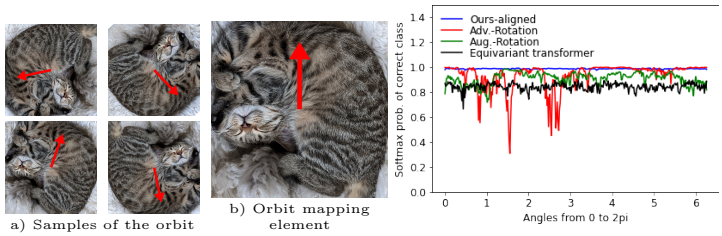

Fig. 1: (Left) Picture of a cat in 4 different rotation samples from the continuous orbit of rotations. Our orbit mapping selects the element with mean gradient direction (marked in red) along circle pointing upwards. (Right) Softmax probabilities of the true label when rotating an image by $0^\circ - 360^\circ$. Our method (in blue) is *robust for any angle*, which cannot be guaranteed through data augmentations (green) or adv. training (red).

the maximum loss among all predictions (*adversarial training*). However, such training schemes cannot guarantee provable invariance. In particular, training with data augmentation is far from being robust to transformations as illustrated in Fig. 1. The plot shows the softmax probabilities of the true label when feeding the exemplary image at rotations ranging from 0 to 2π into a network trained with rotational augmentation (green), adversarial training (red) and undoing rotations using a learned network (black). As we can see, rotational data augmentation is not sufficient to truly make a classification network robust towards rotations, and even the significantly more expensive adversarial training shows instabilities.

While modifications of the training scheme remain the best option for complex or hard-to-characterize transformations, more structured transformations, e.g., those arising from a group action, allow modifications to the network architecture to yield provable invariance. As opposed to previous works that largely rely on the ability to enlist all transformations of an input x (i.e., assume a finite *orbit*), we propose to make neural networks invariant by selecting a specific element from a (possibly infinite) orbit generated by a group action, through an application-specific *orbit mapping*. Simply put, we undo and fix the transformation or pose. Our proposed approach is significantly easier to train than adversarial training methods while being at least equally performant, robust, and computationally cheaper. We illustrate these findings on the rotation invariant classification of images (on which discretization artifacts from the interpolation after any rotation play a crucial role) as well as on the scale, rotation, and translation invariant classification of 3D point clouds. Our contributions can be summarized as follows:

- We present *orbit mapping*, a simple way to adapt neural networks to be in-(or equi)variant to transformations from sets S associated with a group action.
- We propose a gradient based orbit mapping strategy for image rotations, which can provably select unique orientation for continuous image models.
- Our proposed orbit mapping improves robustness of standard networks to transformations even *without* additional changes in training or architecture.
- Existing invariant approaches also demonstrate gain in robustness to discrete image rotations when combined with orbit mapping.
- We demonstrate orbit mappings to provable scale and orientation invariant 3D point cloud classification using well known scale normalization and PCA.

2 Related Work

Several approaches have been developed in the literature to encourage models to exhibit invariance or robustness to desired transformations of data. These include i) data augmentation using desired transformations, ii) regularization to encourage network output to be robust to transformations on the input [9], iii) adversarial training [10,11] and regularization [12], iv) unsupervised or self-supervised pre-training to learn transformation robust representations [13,14,15,16,17], v) parameterized learning of augmentations to learn invariances from training data[18,19], vi) use of hand-crafted invariant shallow [20,21,22,23,24] or deep [25,26,27] features for downstream classification tasks vii) incorporating desired invariance properties in to the network design [28,29,30,31,32], and viii) train time/test time data transformation. Recent works [33,34] have also explored certifying geometric robustness of networks. The approaches i)-v) can improve robustness but cannot yield provable invariance to transformations. Hand-crafting features can yield desired invariance, but is difficult and often sacrifices accuracy. Provable invariance to a finite number of transformations is achievable by applying all such transformations to the each input data point and pooling the corresponding features [35,36]. While this strategy can even be applied only during test time, it can not be extended to sets with infinitely many transformations. Recent approaches [28,37,30] incorporate in-/equivariances when the desired transformations of the data can be formulated as a group action, e.g. enforcing equivariance in each layer separately. Layer wise approaches for equivariance to finite groups such as [28] typically use all possible transformations at each layer.

Canonicalization Closely related to our approach are methods which align input to a normalized or canonical pose. The use of PCA or scale renormalization are well known approaches to normalizing point clouds. However, PCA-based pose canonicalization is known to suffer from ambiguities, and learning based approaches [38,32,39] have been proposed for disambiguation. Several recent works directly leverage deep learning for 3d pose canonicalization, for example training with ground truth poses [40,41] or self-supervised learning [42,43,44]. For 2D images, PCA-based canonicalization is possible only with binary images [45]; the use of Radon transformations [46] requires an expensive, fine discretization of continuous rotations. The use of spatial transformer networks [47] is an alternate learning based approach to 2D/3D pose normalization which can be used along with an application-dependent coordinate transformation [48,49]. Such learning-based approaches, however, require additional training with data augmentation and cannot guarantee invariance. Since our orbit mappings essentially select a canonical group orbit element, our work can be interpreted as a formalization of canonicalization for group transformations. In contrast to learning based approaches, we select a canonical element from the orbit using simple analytical solutions, which can improve robustness even without data augmentations.

Provable Rotational In-/equivariance in 2D Several works [26,27,28,50,51,52] have considered layer wise equivariance to discrete rotations using multiple rotated versions of filters at each layer, which was formalized using group convolutions in [28]. While [28,50,51,52] learn these filters by training, [26,27] make use of

rotated and scaled copies of fixed wavelet filters at each layer. For equivariance to continuous rotations, Worrall et al. [29] utilize circular harmonic filters at each layer. All these layer wise approaches for group equivariance in images were unified in a single framework in [30]. Instead of layer-wise approaches, [53,36,54] pool the features of multiple rotated copies of images input to the network.

Rotation Invariance in 3D Due to the different representations of 3D data (e.g. voxels, point clouds, meshes), many strategies exist. Some techniques for image invariances can be adapted to voxel representations, e.g. probing several rotations at test time [55,56], use of rotationally equivariant convolution kernels [57,58,59]. Spatial transformers have also been used to learn 3D pose normalization, e.g. in the classical PointNet architecture [60], and its extension PointNet++ [61] which additionally considers hierarchical and neighborhood information. While point clouds do not suffer from discretization artifacts after rotations, they struggle with less clear neighborhood information due to unordered coordinate lists. [62] solve this by adding hierarchical graph connections to point clouds and using graph convolutions. However, the features learned using graph convolutions still depend on the rotation of the input data. [63,64] propose graph convolution networks equivariant to isometric transformations. [65,66] project point clouds onto 2D sphere and employ spherical convolutions to achieve rotational equivariance. [67] and [68] achieve rotation invariance on point clouds by considering pairs of features in the tangent plane of each point. While local operations and convolutions on the surface of triangular meshes are invariant to global rotations by definition [69], they however do not capture global information. MeshCNN [70] addresses this by adding pooling operations through edge collapse. [71] defines a representation independent network structure based on heat diffusion which can balance between local and global information.

3 Proposed Approach

Our idea is straightforward. We make neural networks invariant by consistently selecting a fixed element from the orbit of group transformations, i.e, we modify the input pose such that every element from the orbit of transformations maps to the same canonical element. For example, different rotated versions of an image are mapped to have the same orientation as visualized in Fig. 2. In conjunction with such *orbit mapping*, any standard network architecture can achieve provable invariance. In the following, we formalize our approach to achieve invariance.

3.1 Invariant Networks w.r.t. Group Actions

We consider a network $\mathcal{G}$ to be a function $\mathcal{G} : \mathcal{X} \times \mathbb{R}^p \to \mathcal{Y}$ that maps data $x \in \mathcal{X}$ from some suitable input space $\mathcal{X}$ to some prediction $\mathcal{G}(x;\theta) \in \mathcal{Y}$ in an output space $\mathcal{Y}$ where the way this mapping is performed depends on parameters $\theta \in \mathbb{R}^p$. The question is how, for a given set $S \subset \{g : \mathcal{X} \to \mathcal{X}\}$ of transformations of the input data, we can achieve the *invariance* of $\mathcal{G}$ to S defined as

$$\mathcal{G}(g(x);\theta) = \mathcal{G}(x;\theta) \quad \forall x \in \mathcal{X},\ g \in S,\ \theta \in \mathbb{R}^p. \tag{2}$$

The invariance of a network with respect to transformations in S is of particular interest when S induces a *group action*[1] on $\mathcal{X}$, which is what we will assume about S for the remainder of this paper. Of particular importance for the construction of invariant networks, is the set of all possible transformations of input data x,

$$S \cdot x = \{g(x) \mid g \in S\}, \tag{3}$$

which is called the *orbit of* x. A basic observation for constructing invariant networks is that any network acting on the orbit of the input is automatically invariant to transformations in S:

Fact 1 ***Characterization of Invariant Functions via the Orbit:*** *Let S define a group action on $\mathcal{X}$. A network $\mathcal{G} : \mathcal{X} \times \mathbb{R}^p \to \mathcal{Y}$ is invariant under the group action of S if and only if it can be written as $\mathcal{G}(x;\theta) = \mathcal{G}_1(S \cdot x;\theta)$ for some other network $\mathcal{G}_1 : 2^{\mathcal{X}} \times \mathbb{R}^p \to \mathcal{Y}$.*

The above observation is based on the fact that $S \cdot x = S \cdot g(x)$ holds for any $g \in S$, provided that S is a group. Although not taking the general perspective of Fact 1, approaches, like [36], which integrate (or sum over finite elements of) the mappings of $\mathcal{G}$ over a (discrete) group can be interpreted as instances of Fact 1 where $\mathcal{G}_1$ corresponds to the summation. Similar strategies of applying all transformations in S to the input x can be pursued for the design of equivariant networks, see Appendix A.

3.2 Orbit Mappings

While Fact 1 is stated for general (even infinite) groups, realizations of such constructions from the literature often assume a finite orbit. In this work we would like to include an efficient solution even for cases in which the orbit is not finite, and utilize Fact 1 in the most straight-forward way: We propose to construct provably invariant networks $\mathcal{G}(x;\theta) = \mathcal{G}_1(S \cdot x;\theta)$ by simply using an

$$\textit{orbit mapping } h : \{S \cdot x \mid x \in \mathcal{X}\} \to \mathcal{X},$$

which uniquely selects a particular element from an orbit as a first layer in $\mathcal{G}_1$. Subsequently, we can proceed with any standard network architecture and Fact 1 still guarantees the desired invariance. A key in designing instances of orbit mappings is that they should not require enlisting all elements of $S \cdot x$ in order to evaluate $h(S \cdot x)$. Let us provide more concrete examples of orbit mappings.

Example 1 (Mean-subtraction). A common approach in data classification tasks is to first normalize the input by subtracting its mean. Considering $\mathcal{X} = \mathbb{R}^n$ and $S = \{g : \mathbb{R}^n \to \mathbb{R}^n \mid g(x) = x + a\mathbb{1}, \text{ for some } a \in \mathbb{R}\}$, with $\mathbb{1} \in \mathbb{R}^n$ being a vector of all ones, input-mean-subtraction is an orbit mapping that selects the unique element from any $S \cdot x$ which has zero mean.

[1] A (left) group action of a group S with the identity element e, on a set X is a map $\sigma : S \times X \to X$, that satisfies i) $\sigma(e, x) = x$ and ii) $\sigma(g, \sigma(h, x)) = \sigma(gh, x)$, $\forall g, h \in S$ and $\forall x \in X$. When the action being considered is clear from the context, we write $g(x)$ instead of $\sigma(g, x)$.

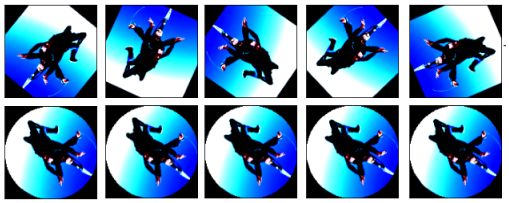

Fig. 2: Images of different orientations (top) are consistently aligned with the proposed gradient-based orbit mapping (bottom).

Example 2 (Permutation invariance via sorting). Consider $\mathcal{X} = \mathbb{R}^n$, and S to be all permutations of vectors in $\mathbb{R}^n$, i.e., $S = \{s \in \{0,1\}^{n\times n} \mid \sum_i s_{i,j} = 1\ \forall j,\ \ \sum_j s_{i,j} = 1\ \forall i\}$. We could define an orbit mapping that selects the element from an orbit whose entries are sorted by magnitude in an ascending order.

With the very natural condition that orbit mappings really select an element from the orbit, i.e., $h(S \cdot x) \in S \cdot x$, we can readily construct equivariant networks by applying the inverse mapping, see Appendix A. In our Example 2, undoing the sort operation at the end of the network allows to transfer from an invariant, to an equivariant network.

As a final note, our concept of orbit mappings can further be generalized by h not mapping to the input space $\mathcal{X}$, but to a different representation, which can be beneficial for particular, complex groups. In geometry processing, for instance, an important group action are isometric deformations of shapes. A common strategy to handle these (c.f. [72]) is to identify any shape with the eigenfunctions of its Laplace-Beltrami operator [73], which represents a natural (generalized) orbit mapping. We refer to [74,75,76] for exemplary deep learning applications.

4 Applications

We will now present two specific instances of orbit mappings for handling continuous rotations of images as well as for invariances in 3D point cloud classification.

4.1 Invariance to continous image rotations

Images as functions Let us consider the important example of invariance to continuous rotations of images. To do so, consider $\mathcal{X} \subset \{u : \Omega \subset \mathbb{R}^2 \to \mathbb{R}\}$ to represent images as functions. For the sake of simplicity, we consider grayscale images only, but this extends to color images in a straight-forward way. In our notation $z \in \mathbb{R}^2$ represents spatial coordinates of an image (to avoid an overlap with our previous $x \in \mathcal{X}$, which we used for the input of a network). We set

$$\begin{aligned} &S = \{g : \mathcal{X} \to \mathcal{X} \mid g \circ u(z) = u(r(\alpha)z),\ \text{for } \alpha \in \mathbb{R}\}, \\ &\text{and } r(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}. \end{aligned} \tag{4}$$

As S has infinitely many elements, approaches that worked well for rotations by 90 degrees like [28] are not applicable anymore. We instead propose to uniquely select an element from the continuous orbit of rotation $g \in S$ by choosing a rotation that makes the average gradient of the image $\int_Z \nabla(g \circ u)(z)\ dz$ over a suitable set Z, e.g. a circle around the image center point upwards. It holds that

$$\nabla(g \circ u)(z) = r^T(\alpha)\nabla u\left(r(\alpha)z\right) \text{ such that}$$
$$\int_Z \nabla(g \circ u)(z)dz = \int_Z r^T(\alpha)\nabla u\left(r(\alpha)z\right) dz.$$

Substituting $\varphi = r(\alpha)z$, we obtain

$$\int_Z r^T(\alpha)\nabla u\left(r(\alpha)z\right) dz = \int_{r^T(\alpha)Z} r^T(\alpha)\nabla u\left(\varphi\right) d\varphi = r^T(\alpha)\int_Z \nabla u\left(\varphi\right) d\varphi \quad (5)$$

where we used that Z is rotationally invariant. Thus, choosing a rotation that makes $\int_Z \nabla(g \circ u)(z)\ dz$ point upwards is equivalent to solving

$$r(\hat{\alpha}) = \arg\max_{r(\alpha)} \left\langle \begin{pmatrix} 1 \\ 0 \end{pmatrix}, r^T(\alpha) \int_Z \nabla u(\varphi)\ d\varphi \right\rangle \quad (6)$$

whose solution is given by $\hat{\alpha}$ such that

$$\begin{pmatrix} \cos\hat{\alpha} \\ \sin\hat{\alpha} \end{pmatrix} = \left(\frac{\int_Z \nabla u(z)\ dz}{\| \int_Z \nabla u(z)\ dz \|} \right). \quad (7)$$

Note that (7) yields unique solution to the maximization problem. Since a consistent pose is always selected[2], it is an invariant mapping. When $\int_Z \nabla u(z)\ dz = 0$, any $g \in S$ maximizes (6). However, numerically $\int_Z \nabla u(z)\ dz$ rarely evaluates to exact zero and its magnitude of determines the stability of orbit mapping.

Discretization For a discrete (grayscale) image given a matrix $\tilde{u} \in \mathbb{R}^{n_y \times n_x}$, we first apply Gaussian blur with a standard deviation of $\sigma = 1.5$ (to reduce the effect of noise and create a smooth image), and subsequently construct an underlying continuous function $u : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ by bilinear interpolation. For the set Z we choose two circles of radii 0.05 and 0.4 (for Ω being normalized to $[0, 1]^2$). We approximate the integral by a sum over finite evaluations of the derivative along each circle, using exact differentiation of the continuous image model. This strategy can stabilize arbitrary rotations successfully as illustrated in Fig. 2. However, in practice, the magnitude of $\int_Z \nabla u(z)\ dz$ and interpolation artifacts affect the stability of the orbit mapping. We analyze the stability of the proposed gradient based orbit-mapping for discrete images in Appendix C, where we observe that use of forward or central differences to approximate gradients further deteriorates the stability of orbit mapping. Since the orbit mapping for

[2] Note that $r^T(\alpha) = r(-\alpha)$, therefore if the predicted rotation for $u(z)$ is β, then for $u(r(\gamma)z)$, it is $\beta - \gamma$, i.e the same element is consistently selected.

Method	OM$^{(\text{Ours})}$	CIFAR10			HAM10000			CUB200		
		Clean	Avg.	Worst	Clean	Avg.	Worst	Clean	Avg.	Worst
Std.	✗	**93.98**	40.06	1.31	93.82	91.73	82.52	**77.41**	53.45	8.07
	✓ Train+Test	87.99	84.12	68.60	93.31	91.38	87.96	71.19	71.56	58.80
RA	✗	85.54	75.99	44.71	93.30	90.81	82.30	69.89	70.12	41.01
	✓ Train+Test	85.40	81.82	71.09	93.41	92.13	88.55	70.35	70.72	57.54
STN	✗	83.74	78.86	54.03	–	–	–	–	–	–
ETN	✗	84.39	80.30	64.08	92.47	90.85	84.32	64.14	66.95	52.85
Adv.	✗	69.32	68.54	50.21	92.28	91.87	85.04	64.54	64.07	42.82
Mixed	✗	91.15	68.37	17.15	93.71	92.13	84.53	68.56	65.91	42.87
Adv.-KL	✗	72.28	70.29	51.05	92.54	91.79	85.42	64.47	64.65	43.04
Adv.-ALP	✗	71.25	70.30	52.29	92.89	91.84	85.98	64.63	64.34	43.63
TIpool	✗	93.56	66.46	20.22	93.19	91.87	88.16	76.80	74.90	59.04
	✓ Train+Test	91.94	**88.77**	76.26	**93.83**	92.05	**89.81**	76.82	**77.18**	**69.19**
TIpool-RA	✗	91.40	84.65	67.28	93.39	91.87	88.12	73.47	74.71	62.82
	✓ Train+Test	90.47	87.92	**80.07**	93.68	**92.78**	89.30	74.78	75.89	67.78

Table 1: Comparison of orbit mapping *(OM)* with training and architecture based methods. Robustness to rotations is compared using the average and worst case accuracies over 5 runs with test images rotated in steps of 1° using bilinear interpolation.

discrete images has instabilities, exact invariance to rotations cannot be guaranteed. Even when the integral values are large leading to a stable orbit mapping, our approach does not need to give the same rotation angle for semantically similar content, for example, different cars are not necessarily rotated to have the same orientation. Due to these reasons, our approach can further benefit from augmentation.

Experiments To evaluate our approach, we use orbit mapping in conjunction with image classification networks on three datasets: On CIFAR10, we train a Resnet-18 [77] from scratch. On the HAM10000 skin image dataset [78], we finetune an NFNet-F0 network [79], and on CUB-200 [80] we finetune a Resnet-50 [77], both pretrained on ImageNet. While the datasets CIFAR10 and CUB-200 have an inherent variance in orientation, for the HAM10000 skin lesion classification, exact rotation invariance is desirable. Finally, we also perform experiments with RotMNIST using state of the art E2CNN network[30]. The details of the protocol used for training all our networks as well as some additional experiments are provided in the Appendix E. We compare with following approaches on CIFAR10, HAM10000, and CUB-200: *i) adversarial training:* $\min_\theta \sum_{\text{examples i}} \mathcal{L}(\mathcal{G}(\hat{x}^i;\theta);y^i)$, for $\hat{x}^i = \arg\max_{z\in S\cdot x^i} \mathcal{L}(\mathcal{G}(z);y^i)$. This is approximated by selecting the worst out of 10 different random rotations for each image in every iteration, following [10]. It is referred to as Adv. in Tab. 1. *ii) mixed mode training:* $\min_\theta \sum_{\text{examples i}} \mathcal{L}(\mathcal{G}(\hat{x}^i;\theta);y^i) + \mathcal{L}(\mathcal{G}(x^i;\theta);y^i)$ which uses both natural and adversarial examples $\hat{x}^i$. *iii) adversarial training with regularization:* Use of adversarial logit pairing and KL-divergence regularizers [12] along with adversarial training (indicated as Adv.-ALP and Adv.-KL in Tab. 1):

a) adversarial logit pairing (ALP): $R_{ALP}(\mathcal{G},x^i,y^i) = \|\mathcal{G}(x^i;\theta) - \mathcal{G}(\hat{x}^i;\theta)\|_2^2$,

b) KL-divergence:$R_{KL}(\mathcal{G},x^i,y^i) = D_{KL}(\mathcal{G}(x^i;\theta)||\mathcal{G}(\hat{x}^i;\theta))$.

iv) transformation invariant pooling (TIpool): which is a provably invariant approach for discrete rotations [36], where the features of multiple rotated copies of input image are pooled before the final classification. We use 4 rotated copies

Train	OM	Clean	Average			Worst-case		
			Nearest	Bilinear	Bicubic	Nearest	Bilinear	Bicubic
Std.	✗	**93.98±0.32**	35.12±0.81	40.06±0.44	42.81±0.50	0.79±0.38	1.31±0.13	2.22±0.17
	✓ Train+Test	87.99±0.43	72.40±0.33	84.12±0.55	86.61±0.49	34.57±0.94	68.60±0.81	74.49±0.84
RA	✗	85.54±0.72	80.47±0.74	75.99±0.72	79.47±0.65	45.50±0.83	44.71±0.74	50.50±0.78
	✓ Test	79.26±0.42	74.93±0.51	69.31±0.65	73.94±0.63	48.93±0.75	52.18±0.91	58.69±0.78
	✓ Train+Test	85.40±0.57	84.37±0.58	81.82±0.59	84.82±0.52	66.22±0.75	71.09±1.01	76.44±0.89
RA-combined	✗	92.42±0.21	80.90±0.64	82.23±0.74	82.71±0.69	36.98±1.27	48.07±1.66	49.51±1.47
	✓ Test	82.55±0.86	76.33±0.95	77.93±0.68	78.42±0.64	45.44±1.32	60.23 ±1.24	62.18±1.33
	✓ Train+Test	86.69±0.12	84.06±0.21	85.27±0.23	86.06±0.20	61.75±0.76	75.29±0.42	77.25±0.27
Adv.	✗	69.32±1.61	61.73±1.12	68.54±0.68	68.00±0.31	36.95±0.97	50.21±0.55	49.73±0.98
Mixed	✗	91.15±0.15	54.55±0.40	68.37±0.66	68.48±0.37	3.86±0.13	17.15±1.25	16.85±0.93
Adv.-KL	✗	72.28±2.05	62.60±1.72	70.29±1.42	69.84±1.29	32.60±0.74	51.05±2.47	51.11±1.03
Adv.-ALP	✗	71.25±0.97	62.36±2.19	70.30±1.50	69.71±1.22	33.98±1.44	52.29±1.76	52.57±1.57
STN	✗	83.74±0.50	81.94±0.51	78.86±0.73	82.21±0.55	51.23±1.01	54.03±1.36	59.65±1.31
ETN	✗	84.39±0.09	82.98±0.28	80.30±0.55	83.31±0.31	59.40±0.76	64.08±0.78	68.75±0.83
Augerino	✗	83.68±0.76	80.17±0.70	82.27±0.69	81.69±0.72	52.44±0.66	60.36±1.00	60.63±0.94
TIpool	✗	93.56±0.25	55.96±0.39	66.46±1.36	70.70±0.77	3.14±1.09	20.22±1.51	27.88±1.09
TIpool-RA	✗	91.40±0.17	87.50±0.24	84.65±0.51	87.31±0.29	66.52±1.31	67.28±1.03	72.35±0.83
TIpool	✓Train+Test	91.94±0.38	78.66±0.83	88.77±0.51	**90.76±0.40**	42.01±1.07	76.26±1.12	81.46±1.02
TIpool-RA	✓Train+Test	90.47±0.36	**89.37±0.36**	87.92±0.36	89.91±0.34	**74.51±0.79**	80.07±0.69	83.76±0.60
TIpool-RA combined	✓Train+Test	91.09±0.40	89.02±0.30	**90.13±0.34**	90.64±0.30	70.18±1.12	**82.71±0.62**	**84.26±0.41**

Table 2: Effect of augmentation on robustness to rotations with different interpolations. Shown are clean accuracy on standard CIFAR10 test set, average and worst-case accuracies on rotated test set with mean and standard deviations over 5 runs.

of images rotated in multiples of 90 degrees. *v) Spatial transformer networks (STN):* which learns to undo the transformation by training using appropriate data augmentation [47]. *vi) Equivariant transformer networks (ETN):* which additionally uses appropriate coordinate transformation along with a learned spatial transformer to undo the transformation [48]. We also compare with the simple baseline of augmenting with random rotations, referred to as RA in Tab. 1. Additionally, we also compare with [19], an approach which learns distribution of augmentations on the task of rotated CIFAR10 classification, referred to as Augerino in Tab. 2. We use 4 samples from the learned distribution of augmentations during both training and test. We would also like to point out that adversarial training using the worst of 10 samples roughly increases the training effort of the underlying model by a factor of 5.

Results We measure the accuracy on the original testset(*Clean*), as well as the average (*Avg.*) and worst-case (*Worst*) accuracies in the orbit of rotations discretized in steps of 1 degree, where '*Worst*' counts an image as misclassified as soon as there exists a rotation at which the network makes a wrong prediction.

As we can see in Tab. 1, networks trained without rotation augmentation perform poorly in terms of both, the average and worst-case accuracy if the data set contains an inherent orientation. While augmenting with rotations during training results in improvements, there is still a huge gap ($\sim 30\%$ for CIFAR10 and CUB200) between the average and worst-case accuracies. While adversarial training approaches [10,12] improve the performance in the worst case, there is a clear drop in the clean and average accuracies when compared to data augmentation. Learned approaches to correct orientation i.e. STN [47], ETN [48] show an improvement over adversarial training schemes in terms of average and

worst case accuracies, when training from scratch, with ETN demonstrating even higher robustness than plain STNs. While pooling over features of rotated versions of image provides provable invariance to discrete rotations, this approach is still susceptible to continuous image rotations. The robustness of this approach to continuous rotations is boosted by rotation augmentation, with improvements over even learned transformers. Note that using TI-pooling with 4 rotated copies increases the computation by 4 times. In contrast, our orbit mapping effortlessly leads to significant improvements in robustness even without augmenting with rotations, with performance better than adversarial training, learned transformers and discrete invariance based approaches. Since our orbit mapping for discrete images has some instabilities, our approach also benefits from augmentation with image rotations. Further, when combined with discrete invariant approach [36], we obtain the best accuracies for average and worst case rotations.

Even when finetuning networks, we observe that orbit mapping readily improves robustness to rotations over standard training, even without the use of augmentations. Furthermore, combination of orbit mapping with the discrete invariant approach of pooling over rotated features yields the best performance. For the birds dataset with inherent orientation, undoing rotations using ETN significantly improves robustness when compared to adversarial training schemes, which only marginally improve robustness over rotation augmentation. We found it difficult to train STN with higher accuracies (*Clean/Avg./Worst*) than plain augmentation with rotated images for CUB200 and HAM10000, despite extensive hyperparameter optimization, therefore we do not report the numbers here[3]. When the data itself does not contain a prominent orientation as in the HAM10000 data set, the general trend in accuracies still holds (*Clean*>*Avg.*>*Worst*), but the drops in accuracies are not drastic, and adversarial training schemes provide improvements over undoing transormations using ETN. Further, orbit mapping and pooling over rotated images provide comparable improvements in robustness, with their combination achieving the best results.

Discretization Artifacts: It is interesting to see that while consistently selecting a single element from the continuous orbit of rotations leads to provable rotational invariance when considering images as continuous functions, discretization artifacts and boundary effects still play a crucial role in practice, and rotations cannot be fully stabilized. As a result, there is still discrepancy between the average and worst case accuracies, and the performance is further improved when our approach also uses rotation augmentation. Motivated by the strong effect the discretization seems to have, we investigate different interpolation schemes used to rotate the image in more detail: Tab. 2 shows the results different training schemes with and without our orbit mapping (*OM*) obtained with a ResNet-18 architecture on CIFAR-10 when using different types of interpolation. Besides standard training (*Std.*), we use rotation augmentation (*RA*) using the Pytorch-default of nearest-neighbor interpolation, a combined augmentation scheme (*RA-combined*) that applies random rotation only to a fraction of images

[3] We use a single spatial transformer as opposed to multiple STNs used in [47] and train on randomly rotated images.

Train.	OM	D4/C4			D16/C16		
		Clean	Avg.	Worst	Clean	Avg.	Worst
Std.	✗	98.73±0.04	98.61±0.04	96.84±0.08	99.16±0.03	99.02±0.04	98.19±0.08
Std.	✓(Train+Test)	98.86±0.02	98.74±0.03	98.31±0.05	99.21±0.01	99.11±0.03	98.82±0.06
RA.	✗	99.19±0.02	99.11±0.01	98.39±0.05	99.31±0.02	99.27±0.02	98.89±0.03
RA.	✓(Train+Test)	98.99±0.03	98.90±0.01	98.60±0.02	99.28±0.02	99.23±0.01	99.04±0.02

Table 3: Effect of orbit mapping and rotation augmentation on RotMNIST classification using regular D4/C4 and D16/C16 E2CNN models. Shown are clean accuracy on standard test set and average and worst-case accuracies on test set rotated in steps of 1 degree, with mean and standard deviations over 5 runs.

in a batch using at least one nearest neighbor, one bilinear and one bicubic interpolation. The adversarial training and regularization from [10,12] are trained using bilinear interpolation (following the authors' implementation).

Results show that interpolation used in image rotation impacts accuracies in all the baselines. Most notably, the worst-case accuracies between different types of interpolation may differ by more than 20%, indicating a huge influence of the interpolation scheme. Adversarial training with bi-linear interpolation still leaves a large vulnerability to image rotations with nearest neighbor interpolation. Further, applying an orbit mapping at test time to a network trained with rotated images readily improves its worst case accuracy, however, there is a clear drop in clean and average case accuracies, possibly due to the network not having seen doubly interpolated images during training. While our approach without rotation augmentation is also vulnerable to interpolation effects, it is ameliorated when using orbit mapping along with rotation augmentation. We observe that including different augmentations (RA-combined) improves the robustness significantly. Combining the orbit mapping with the discrete invariant approach [36] boosts the robustness, with different augmentations further reducing the gap between clean, average case and worst case performance.

Experiments with RotMNIST We investigate the effect of orbit mapping on RotMNIST classificationwith the state of the art network from [30] employing regular steerable equivariant models[81]. This model uses 16 rotations and flips of the learned filters (with flips being restricted till layer3). We also compare with a variation of the same architecture with 4 rotations. We refer to these models as D16/C16 and D4/C4 respectively. We train and evaluate these models using their publicly available code[4]. Results in Tab. 3 indicate that even for these state of the art models, there is a discrepancy between the accuracy on the standard test set and the worst case accuracies, and their robustness can be further improved by orbit mapping. Notably, orbit mapping significantly improves worst case accuracy (by around 1.5%) for D4/C4 steerable model trained without augmenting using rotations, showing gains in robustness even over naively trained D16/C16 model of much higher complexity. Training with augmentation leads to improvement in robustness, with orbit mapping providing gains further in robustness. However, artifacts due to double interpolation affect performance of orbit mapping.

[4] code url `https://github.com/QUVA-Lab/e2cnn_experiments`

Augment.	Unscaling	with STN			without STN		
		Clean	Avg.	Worst	Clean	Avg.	Worst
$[0.8, 1.25]$	✗	86.15± 0.52	24.40±1.56	0.01±0.02	85.31±0.39	33.57±2.00	2.37±0.06
$[0.8, 1.25]$	✓(Train+Test)	**86.15± 0.28**	**86.15± 0.28**	**86.15± 0.28**	85.25±0.43	85.25±0.43	85.25±0.43
$[0.8, 1.25]$	✓(Test)	86.15± 0.52	85.59±0.79	85.59±0.79	85.31±0.39	83.76±0.35	83.76±0.35
$[0.1, 10]$	✗	85.40±0.46	47.25±1.36	0.04±0.05	75.34±0.84	47.58±1.69	1.06±0.87
$[0.1, 10]$	✓(Test)	85.40±0.46	85.85±0.73	85.85±0.73	75.34±0.84	81.45±0.56	81.45±0.56
$[0.001, 1000]$	✗	33.33± 7.58	42.38± 1.54	2.25±0.22	5.07±2.37	25.42±0.73	2.24±0.11
$[0.001, 1000]$	✓(Train+Test)	85.66± 0.39	85.66± 0.39	85.66± 0.39	85.05±0.43	85.05±0.43	85.05±0.43

Table 4: Scaling invariance in 3D pointcloud classification with PointNet trained on modelnet40, with and without data augmentation, with and without STNs or scale normalization. Mean and standard deviations over 10 runs are reported.

RA	STN	PCA	Clean	Rotation		Translation	
				Avg.	Worst	Avg.	Worst
✗	✓	✗	**86.15±0.52**	10.37±0.18	0.09±0.07	10.96±1.22	0.00±0.00
✗	✗	✗	85.31±0.39	10.59±0.25	0.26±0.10	6.53±0.12	0.00±0.00
✗	✓	✓(Train+Test)	74.12± 1.80	74.12± 1.80	74.12± 1.80	74.12± 1.80	74.12± 1.80
✗	✗	✓(Train+Test)	75.36±0.70	**75.36±0.70**	**75.36±0.70**	**75.36±0.70**	**75.36±0.70**
✓	✓	✗	72.13± 5.84	72.39± 5.60	35.91± 4.87	5.35±0.98	0.00±0.00
✓	✗	✗	63.93±0.65	64.75±0.57	45.53±0.29	3.90±0.71	0.00±0.00
✓	✓	✓(Test)	72.13± 5.84	72.96± 5.85	72.96± 5.85	72.96± 5.85	72.96± 5.85
✓	✗	✓(Test)	64.56±0.91	64.56±0.91	64.56±0.91	64.56±0.91	64.56±0.91
✓	✓	✓(Train+Test)	72.84±0.77	72.84±0.77	72.84±0.77	72.84±0.77	72.84±0.77
✓	✗	✓(Train+Test)	74.84±0.86	74.84±0.86	74.84±0.86	74.84±0.86	74.84±0.86

Table 5: Rotation and translation invariances in 3D pointcloud classification with PointNet trained on modelnet40, with and without rotation augmentation, with and without STNs or PCA. Mean and standard deviations over 10 runs are reported.

4.2 Invariances in 3D Point Cloud Classification

Invariance to orientation and scale is often desired in networks classifying objects given as 3D point clouds. Popular architectures, such as PointNet [60] and its extensions [61], rely on the ability of spatial transformer networks to learn such invariances by training on large datasets and extensive data augmentations. We analyze the robustness of these networks to transformations with experiments using Pointnet on *modelnet40* dataset [55]. We compare the class accuracy of the final iterate for the clean validation set *(Clean)*, and transformed validation sets in the average *(Avg.)* and worst-case *(Worst)*. We show that PointNet performs better with our orbit mappings than with augmentation alone.

In this setting, $\mathcal{X} = \mathbb{R}^{d\times N}$ are N many d-dimensional coordinates (usually with $d = 3$). The desired group actions for invariance are left-multiplication with a rotation matrix, and multiplication with any number $c \in \mathbb{R}^+$ to account for different scaling. We also consider translation by adding a fixed coordinate $c_t \in \mathbb{R}^3$ to each entry in $\mathcal{X}$. Desired invariances in point cloud classification range from class-dependent variances to geometric properties. For example, the classification of airplanes should be invariant to the specific wing shape, as well as the scale or translation of the model. While networks can learn some invariance from training data, our experiments show that even simple transformations like scaling and translation are not learned robustly outside the scope of what was provided in the training data, see Tabs. 4, 5, 6. This is surprising, considering that both can be undone by centering around the origin and re-scaling.

Augmentation			STN	OM	Clean	Scaling		Rotation		Translation	
Scale	RA	Translation		All		Avg.	Worst	Avg.	Worst	Avg.	Worst
$[0.8, 1.25]$	✓	$[-0.1, 0.1]$	✓	✗	72.13± 5.84	19.74± 4.01	0.16± 0.42	72.39± 5.60	35.91± 4.87	5.35±0.98	0.00±0.00
$[0.8, 1.25]$	✓	$[-0.1, 0.1]$	✓	✓ Test	67.38± 7.96	64.88± 12.16	64.88± 12.16	64.88± 12.16	64.88± 12.16	64.88± 12.16	64.88± 12.16
$[0.8, 1.25]$	✓	$[-0.1, 0.1]$	✓	✓ Train+Test	**77.52±1.03**	**77.52±1.03**	**77.52±1.03**	**77.52±1.03**	**77.52±1.03**	**77.52±1.03**	**77.52±1.03**
$[0.8, 1.25]$	✓	$[-0.1, 0.1]$	✗	✗	63.93±0.65	12.85±0.29	0.27±0.55	64.75±0.57	45.53±0.29	3.90±0.71	0.00±0.00
$[0.8, 1.25]$	✓	$[-0.1, 0.1]$	✗	✓Test	64.71±0.92	57.10±1.14	57.10±1.14	57.10±1.14	57.10±1.14	57.10±1.14	57.10±1.14
$[0.8, 1.25]$	✓	$[-0.1, 0.1]$	✗	✓Train+Test	74.41±0.58	74.41±0.58	74.41±0.58	74.41±0.58	74.41±0.58	74.41±0.58	74.41±0.58

Table 6: Combined Scale, rotation and translation invariances in 3D pointcloud classification with PointNet trained on modelnet40, with data augmentation and analytical inclusion of each invariance. Mean and standard deviations over 10 runs are reported.

Scaling Invariance to scaling can be achieved in the sense of Sec. 3 by scaling input point-clouds by the average distance of all points to the origin. Our experiments show that this leads to robustness against much more extreme transformation values without the need for expensive training, both for average as well as worst-case accuracy. We tested the worst-case accuracy on the following scales: $\{0.001, 0.01, 0.1, 0.5, 1.0, 5.0, 10, 100, 1000\}$. While our approach performs well on all cases, training PointNet on random data augmentation in the range of possible values actually reduces the accuracy on clean, not scaled test data. This indicates that the added complexity of the task cannot be well represented within the network although it includes spatial transformers. Even when restricting the training to a subset of the interval of scales, the spatial transformers cannot fully learn to undo the scaling, resulting in a significant drop in average and worst-case robustness, see Tab. 4. While training the original Pointnet including the desired invariance in the network achieves the best performance, dropping the spatial transformers from the architecture results in only a tiny drop in accuracy with significant gains in training and computation time[5]. This either indicates that in the absence of rigid deformation the spatial transformers do not add much knowledge and is strictly inferior to modeling invariance, at least on this dataset.

Rotation and Translation In this section, we show that 3D rotations and translations exhibit a similar behavior and can be more robustly treated via orbit mapping than through data augmentation. This is even more meaningful than scaling as both have three degrees of freedom and sampling their respective spaces requires a lot more examples. For rotations, we choose the unique element of the orbit to be the rotation of $\mathcal{X}$ that aligns its principle components with the coordinate axes. The optimal transformation involves subtracting the center of mass from all coordinates and then applying the singular value decomposition $X = U\Sigma V$ of the point cloud X up to the arbitrary orientation of the principle axes, a process also known as PCA. Rotation and translation can be treated together, as undoing the translation is a substep of PCA.To remove the sign ambiguity in the principle axes, we choose signs of the first row of U and encode them into a diagonal matrix D, such that the final transform is given by $\hat{X} = XV^\top D$. We apply this rotational alignment to PointNet with and without spatial transformers and evaluate its robustness to rotations in average-case and worst-case when rotating the validation dataset in 16×16 increments (i.e. with 16

[5] Model size of PointNet with STNs is 41.8 MB, and without STNs 9.8 MB

discrete angles along each of the two angular degrees of freedom of a 3D rotation). We test robustness to translations in average-case and worst-case for the following shifts in each of x, y and z directions: $\{-10.0, -1.0, -0.5, -0.1, 0.1, 0.5, 1.0, 10.0\}$. Tab. 5 shows that PointNet trained without augmentation is susceptible in worst-case and average-case rotations and even translations. The vulnerability to rotations can be ameliorated in the average-case by training with random rotations, but the worst-case accuracy is still significantly lower, even when spatial transformers are employed. Also notable is the high variance in performance of Pointnets with STNs trained using augmentations. On the other hand, explicitly training and testing with stabilized rotations using PCA does provide effortless invariance to rotations and translations, even without augmentation. Interestingly, the best accuracy here is reached when training PointNet entirely without spatial transformers, which offer no additional benefits when the rotations are stabilized. The process for invariance against translation is well-known and well-used due to its simplicity and robustness. We show that this approach arises naturally from our framework, and that its extension to rotational invariance inherits the same numerical behavior, i.e., provable invariance outperforms learning to undo the transformation via data augmentation.

Combined invariance to Scaling, Rotation, Translation. Our approach can be extended to make a model simultaneously invariant to scaling, rotations and translations. In this setup, we apply a PCA alignment before normalizing the scale of input point cloud. Tab. 6 shows that PointNet trained with such combined orbit mapping does achieve the desired invariances.

5 Discussion and Conclusions

We proposed a simple and general way of incorporating invariances to group actions in neural networks by uniquely selecting a specific element from the orbit of group transformations. This guarantees provable invariance to group transformations for 3D point clouds, and demonstrates significant improvements in robustness to continuous rotations of images with a limited computational overhead. However, for images, a large discrepancy between the theoretical provable invariance (in the perspective of images as continuous functions) and the practical discrete setting remains. We conjecture that this is related to discretization artifacts when applying rotations that change the gradient directions, especially at low resolutions. Notably, such artifacts appear more frequently in artificial settings, e.g. during data augmentation or when testing for worst-case accuracy, than in photographs of rotating objects that only get discretized once. While we found a consistent advantage of enforcing the desired invariance via orbit mapping rather than training alone, combination of data augmentation and orbit mappings yields additional advantages (in cases where discretization artifacts prevent a provable invariance of the latter). Moreover, our orbit mapping can be combined with existing invariant approaches for improved robustness.

A Extension of Orbit Mapping to Equivariant Networks

The *equivariance* of $\mathcal{G}$ preserves the structure of transformations $g \in S$ of input data in the elements $y \in \mathcal{Y}$ (including, but not limited to, the case where $\mathcal{X} \equiv \mathcal{Y}$). The *equivariance* of $\mathcal{G}$ to S is defined as

$$\mathcal{G}(g(x);\theta) = g(\mathcal{G}(x;\theta)) \quad \forall x \in \mathcal{X},\ g \in S,\ \theta \in \mathbb{R}^p. \tag{8}$$

We now show that equivariant networks can be designed by applying all transformations in S to the input x.

Proposition 1. *Let S define a group action on $\mathcal{X}$. A network $\mathcal{G}$ is equivariant under the group action of S if it can be written as*

$$\mathcal{G}(x;\theta) = \mathcal{G}_1(\{g(\mathcal{G}_2(g^{-1}(x);\theta_2)) \mid g \in S\};\theta_1) \tag{9}$$

for some other arbitrary network $\mathcal{G}_2 : \mathcal{X} \times \mathbb{R}^{p_2} \to \mathcal{X}$, and a network $\mathcal{G}_1 : 2^{\mathcal{X}} \times \mathbb{R}^{p_1} \to \mathcal{X}$ that commutes with any element $h \in S$, i.e., for $h \in S$, and $Z \subset \mathcal{X}$, it satisfies $\mathcal{G}_1(h(Z);\theta_2) = h(\mathcal{G}_1(Z;\theta_2))$, where $h(Z)$ denotes the set obtained by the applying h to every element of Z.

Proof. We want to show that a network satisfying the condition (5) is equivariant. Let $h \in S$ be arbitrary. Note that

$$\{g \mid g \in S\} = \{h^{-1}g \mid g \in S\} \tag{10}$$

such that a substitution of variables from $g \in S$ to $z = h^{-1}g \in S$ (i.e., $g = hz$ and $z^{-1} = g^{-1}h$) yields

$$\begin{aligned}&\{g(\mathcal{G}_2(g^{-1}(h(x));\theta_2)) \mid g \in S\}\\ =&\{h(z(\mathcal{G}_2(z^{-1}(x);\theta_2))) \mid z \in S\}.\end{aligned}$$

This means that we can also write

$$\begin{aligned}\mathcal{G}(h(x);\theta) &= \mathcal{G}_1(\{h(z(\mathcal{G}_2(z^{-1}(x);\theta_2))) \mid z \in S\};\theta_1)\\ &= \mathcal{G}_1(h(\{z(\mathcal{G}_2(z^{-1}(x);\theta_2)) \mid z \in S\});\theta_1)\\ &= h(\mathcal{G}_1(\{z(\mathcal{G}_2(z^{-1}(x);\theta_2)) \mid z \in S\});\theta_1)\\ &= h(\mathcal{G}(x;\theta))\end{aligned}$$

which yields the desired equivariance under the assumed commutative property.

The work [28] can be interpreted as an instance of the construction in Proposition 1, where equivariant linear layers w.r.t. rotations by 90 degrees are obtained by choosing $\mathcal{G}_2$ to be a simple convolution and $\mathcal{G}_1$ to be the summation over all (finitely many) elements of the set. Subsequently, they nest these layers with component-wise (and therefore inherently equivariant) non-linearities.

While Proposition 1 is stated for general groups, realizations of such constructions often rely on the ability to list an entire orbit of the group. In the following we show an efficient solution to obtain equivariant networks using orbit mapping.

Proposition 2 (Orbit mapping for equivariant networks). *Let h be an orbit mapping that satisfies $h(S \cdot x) \in S \cdot x$ for all x. Any network $\mathcal{G} : \mathcal{X} \times \mathbb{R}^p \to \mathcal{X}$ that can be written as*

$$\mathcal{G}(x; \theta) = \hat{g}^{-1}(\mathcal{G}_2(\hat{g}(x); \theta)) \tag{11}$$

for an arbitrary network $\mathcal{G}_2 : \mathcal{X} \times \mathbb{R}^p \to \mathcal{X}$ and $\hat{g} \in S$ denoting the element that satisfies $\hat{g}(x) = h(S \cdot x)$ is equivariant.

Proof. We want to show that a network satisfying the condition (11) is equivariant. Consider an input $a = r(x)$ to the network, where r denotes an arbitrary element of S. We first need to determine the element $\tilde{g} \in S$ such that $\tilde{g}(a) = h(S \cdot a)$. From the definition of the orbit, it follows that $S \cdot x = S \cdot r(x)$, such that our orbit mapping satisfies remains the same, i.e., $h(S \cdot x) = h(S \cdot a) = \hat{g}(x)$. Solving the equation $\tilde{g}(a) = \hat{g}(x)$ with $a = r(x)$, i.e., $x = r^{-1}(a)$ for $\tilde{g}$ yields $\tilde{g} = \hat{g}r^{-1}$. Now it follows that

$$\begin{aligned}
\mathcal{G}(r(x); \theta) = \mathcal{G}(a; \theta) &= \tilde{g}^{-1}(\mathcal{G}_2(\tilde{g}(a); \theta)) \\
&= r(\hat{g}^{-1}(\mathcal{G}_2(\tilde{g}(a); \theta))) \\
&= r(\hat{g}^{-1}(\mathcal{G}_2(\hat{g}(x); \theta))) \\
&= r(\mathcal{G}(x; \theta)),
\end{aligned}$$

which concludes the proof.

B A Discussion on Isometry Invariance

Here, we will elaborate on how the functional map framework [29] can be seen as an application of our orbit mapping for isometry invariance. Functional maps are a widely used method to find correspondences between isometric shapes, and we will show here that the framework fits within our proposed theory. Non-rigid correspondence is a notoriously hard problem, and joint optimization within a larger framework makes it even more complex. To resolve this the idea of functional maps is to change the representation of the correspondence from point-wise to function-wise. By choosing the eigenfunctions of the Laplace-Beltrami operator [31] as the basis for functions on the shapes, the problem becomes a least squares problem aligning suitable descriptor functions in the space of functions.

Here, $F \in \mathcal{F}(\mathcal{X})$ and $G \in \mathcal{F}(\mathcal{Y})$ are descriptor functions on the shapes $\mathcal{X}$ and $\mathcal{Y}$ respectively. They are assumed to take similar values on corresponding points on $\mathcal{X}, \mathcal{Y}$, and generate the designated orbit element within our framework. These descriptors are projected onto the eigenfunctions of $\mathcal{X}, \mathcal{Y}$, named Φ, Ψ respectively. These projections are the chosen elements of the orbit we will align, and, for isometries and sufficiently comparable descriptors, the projections can be aligned by an orthogonal transformation generating the group action which is exactly the functional map C. The vanilla functional map optimization looks like this:

$$\arg\min_{C \in O(k)} \|C\Phi^{-1}F - \Psi^{-1}G\|_2^2 \tag{12}$$

Functional maps are often used when shape correspondence is required within another framework, and has been used in many deep learning applications [7],[16],[22]. Due to its wide application, we will not provide extra experiments to show its efficacy but want to emphasize that this is a possible implementation of our theory.

C Stability of gradient based orbit mapping

In this section we analyze the stability of the proposed gradient based orbit mapping strategy for discrete images. While the proposed gradient based orbit mapping our approach leads to unique orientation as long as $\int_Z \nabla u(z)\ dz$ is non-zero, practically, the magnitude of $\int_Z \nabla u(z)\ dz$ and interpolation artifacts affect the stability of the orbit mapping. While one could possibly use forward or central differences to calculate gradients at pixels along approximate circles, this further deteriorates the stability of orbit mapping. This is seen in Tab. 7 a) which shows the mean standard deviation orientation of orbit-mapped images when input images rotated in steps of 1 degree using bilinear interpolation. We find that using forward differences to approximate the gradient has the most instability. In the following section, we derive a necessary condition for provable invariance using general convolution kernels (instead of gradients in x and y direction), where we show that forward differences does not satisfy these conditions for any rotation.

Tab. 7 b) shows the histogram of standard deviations in orientation for CIFAR10 images when calculating exact gradients along the circle. The standard deviations of predicted orientations of over 78% of the images is less than 10 degrees, and over 44% of images is less than 4 degrees, indicating a relatively stable orbit mapping for these images. However, a fraction of images also have a higher variance, in predicted orientation possibly due to small values of the integral. Tab. 7 c) shows that our gradient based orbit mapping is fairly robust to small additive Gaussian noise.

D Invariance to image rotations using convolution kernels

Following the notation from the paper, let $u(z)$ denote the continuous image function with $z \in \mathbb{R}^2$ representing the spatial coordinates of an image. The invariance set for the orbit of continuous image rotations is

$$S = \{g : \mathcal{X} \to \mathcal{X} \mid g(u)(z) = (u \circ r(\alpha))(z),\ \text{for } \alpha \in \mathbb{R}\},$$
$$\text{and } r(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \text{ is the rotation matrix.}$$

a)

Dataset	Exact	Central Diff.	Forward Diff
CIFAR10	10.46	12.47	23.89
CUB200	9.05	14.56	24.75

c)

Dataset	clean	σ^2=0.01	σ^2=0.05	σ^2=0.1
CIFAR10	10.46	11.36	14.08	16.69
CUB200	9.05	10.55	15.99	20.610

b)

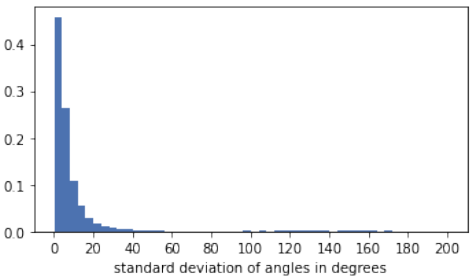

Table 7: Stability and robustness of proposed gradient based Orbit Mapping strategy. a) The mean standard deviation values of angles in degrees over the images in dataset are reported when rotating images based on exact gradients computed along circle using bilinear interpolation, and approximate gradients using finite differences along pixels closest to the circle. b) The histogram of standard deviations of the predicted orientation in degrees for CIFAR10. c) The mean standard deviation values of angles in degrees over the images in CIFAR10 dataset are reported, for different levels of additive Gaussian noise.

Let us consider two kernels $k_i : \mathbb{R}^2 \to \mathbb{R}$, $i = \{1, 2\}$. We now investigate the convolution of a kernel with a rotated image $(u \circ r(\alpha))(z)$

$$\begin{aligned}
(k_i * u \circ r(\alpha))(z) &= \int_{\mathbb{R}^2} k_i(x)(u \circ r(\alpha))(z - x)dx \\
&= \int_{\mathbb{R}^2} k_i(x)u(r(\alpha)z - r(\alpha)x)dx \\
&= \int_{\mathbb{R}^2} k_i(r^T\varphi)u(r(\alpha)z - \varphi)d\varphi \\
&\qquad \text{with } \varphi = r(\alpha)x
\end{aligned}$$

Now assume

$$\begin{pmatrix} k_1(r^T(\alpha)\varphi) \\ k_2(r^T(\alpha)\varphi) \end{pmatrix} = r^T(\alpha) \begin{pmatrix} k_1(\varphi) \\ k_2(\varphi) \end{pmatrix}. \tag{13}$$

Then

$$\begin{aligned}
\begin{pmatrix} (k_1 * (u \circ r(\alpha)))(z) \\ (k_2 * (u \circ r(\alpha)))(z) \end{pmatrix} &= \int_{\mathbb{R}^2} r^T(\alpha) \begin{pmatrix} k_1(\varphi) \\ k_2(\varphi) \end{pmatrix} u(r(\alpha)z - \varphi)d\varphi. \\
&= r^T(\alpha) \begin{pmatrix} (k_1 * u)(r(\alpha)z) \\ (k_2 * u)(r(\alpha)z) \end{pmatrix}
\end{aligned}$$

Then for a suitable set Z which makes the integral rotationally invariant, (e.g. circles around image center)

$$\int_Z \begin{pmatrix} (k_1 * (u \circ r(\alpha)))(z) \\ (k_2 * (u \circ r(\alpha)))(z) \end{pmatrix} dz = r^T(\alpha) \int_Z \begin{pmatrix} (k_1 * u)(\varphi) \\ (k_2 * u)(\varphi) \end{pmatrix} d\varphi \tag{14}$$

And we can determine the optimal rotation as solution to

$$\hat{g} = \arg\max_{g \in S} \left\langle \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \int_Z \begin{pmatrix} k_1 * u \\ k_2 * u \end{pmatrix}(z)\, dz \right\rangle \tag{15}$$

whose solution is given by $\hat{\alpha}$ such that

$$\begin{pmatrix} \cos\hat{\alpha} \\ \sin\hat{\alpha} \end{pmatrix} = \frac{\int_Z \begin{pmatrix} k_1 * u \\ k_2 * u \end{pmatrix}(z)\, dz}{\left\| \int_Z \begin{pmatrix} k_1 * u \\ k_2 * u \end{pmatrix}(z)\, dz \right\|} \tag{16}$$

We can see that (13) is a necessary condition to ensure invariance to image rotations using orbit mapping with (16) employing convolution kernels k_1 and k_2. For discrete convolution kernels, eq. (13) is not exactly satisfied for arbitrary rotations due to discretization problem. We can deduce necessary conditions on discrete kernels k_1 and k_2 to satisfy eq. (13) for rotations in multiples of 90^o. For square kernels k_1 and k_2 of size $N \times N$, we find that

$$k_1[i,j] = k_1[N-i+1, N-j+1] \text{ and} \tag{17}$$

$$k_2 = k_1 \circ r(-90^o) \tag{18}$$

are necessary to satisfy the condition (13) for $\alpha = 90^o$.
For $N = 2$, this gives kernels of the form

$$k_1 = \begin{pmatrix} a & b \\ -b & -a \end{pmatrix} \text{ and } k_2 = \begin{pmatrix} -b & a \\ -a & b \end{pmatrix}$$

For $N = 3$,

$$k_1 = \begin{pmatrix} a & b & c \\ d & 0 & -d \\ -c & -b & -a \end{pmatrix} \text{ and } k_2 = \begin{pmatrix} -c & d & a \\ -b & 0 & b \\ -a & -d & c \end{pmatrix}$$

Note that computing gradients using central differences satisfies (17) and (18), whereas using forward differences does not satisfy these conditions. Therefore, we observe more instabilities in orbit mapping when forward differences are used for gradient computation, see Tab. 7.

E Details about the Experimental Setting

In the following we provide the detailed training settings used in our experiments.

E.1 Rotation invariance for images

For our experiments with image rotational invariance, we used Pytorch(v.1.8.1), python(v.3.8.8), torchvision(v.0.9.1). The exact training protocol is provided

below.

CIFAR10 We trained a Resnet18 [77] on the CIFAR 10 dataset, using stochastic gradient descent with initial learning rate 0.1, momentum 0.9, and weight decay 5e-4. Additionally, we trained a small Convnet and a linear model which used an initial learning rate of 0.01. For all the models, the learning rate is decayed by a factor of 0.5 whenever the validation loss does not decrease for 5 epochs. Training data is augmented using random horizontal flips, random crops of size 32 after zero-padding by 4 pixels. We divide the training data into train (80%) and validation (20%) sets. Networks are trained for 150 epochs with a batch size of 128 and we report the results on the test set using the model with best validation accuracy. The experiments with CIFAR10 were performed partially on a machine with one Nvidia TITAN RTX, and partially on machine with 4 NVIDIA GeForce RTX 2080 GPUs.

HAM10000 We fine-tuned an imagenet pretrained[6] NFNet-F0 [79] on HAM10000 dataset [78]. The dataset is split into 8912 train and 1103 validation images using stratified split, ensuring there are no duplicates with the same lesion ids in the train and validation sets. Training data is augmented using random horizontal and vertical flips and color jitter, and randomly oversample the minority classes to mitigate class imbalance. The network is finetuned for 5 epochs, with a batch size of 128 and learning rate of 1e-4, weight decay of 5e-4 using Adam optimizer [82] with exponential learning rate decay, with factor 0.2. For training using TI-pool which uses 4 rotated copies of images, we reduce the batch size to 32 to fit the GPU memory. For experiments with STN we use a 3 layered CNN with convolution filers of size 3×3 followed by 2 fully connected layers for pose prediction. For experiment with ETN we use a CNN with 4 conv layers with 64 channels and 2 fully connected layers for pose prediction. We report results using final iterate on the validation set. The experiments with HAM10000 dataset were partially performed on a machine with one NVIDIA TITAN RTX card, and partially on machine with 4 NVIDIA GeForce RTX 2080 GPUs.

CUB200 This is a small dataset containing 11,788 images of birds, split into 5994 images for training and 5794 test images. Since training a network from scratch gives low accuracies (around 35% clean accuracy with Resnet-50), we instead perform finetuning using an imagenet pretrained Resnet-50 from pytorch torchvision (v.0.9.1) on CUB-200 dataset [80]. The training data is augmented using random horizontal flips, random resized crops of size 224. The network is finetuned for 60 epochs with batch size of 128 and initial learning rate of 1e-4, using Adam optimizer [82] , weight decay of 5e-4, with exponential learning rate decay, with factor 0.9. . For training using TI-pool which uses4 rotated copies of images, we reduce the batch size to 64 to fit in the GPU memory. For experiment with ETN we use a CNN with 4 conv layers with 64 channels and 2 fully connected layers for pose prediction. We report the accuracies using the final iterate on the test set. The experiments on CUB-200 dataset were performed on machine with 4 NVIDIA GeForce RTX 2080 GPUs.

[6] pretrained model from `https://github.com/rwightman/pytorch-image-models` licensed Apache 2.0

All the three image datasets including HAM10000 dataset [78] used in our experiments are publicly available and widely used in machine learning literature. To the best of our knowledge these do not contain offensive content or personally identifiable information.

E.2 Rotation and Scale invariance for 3D point clouds

We investigate invariance to rotations and scale for 3D point clouds with the task of point cloud classification on the *modelnet40* dataset [83]. For this dataset note the asset descriptions at `https://modelnet.cs.princeton.edu/`: "All CAD models are downloaded from the Internet and the original authors hold the copyright of the CAD models. The label of the data was obtained by us via Amazon Mechanical Turk service and it is provided freely. This dataset is provided for the convenience of academic research only." We use the resampled version of `shapenet.cs.stanford.edu/media/modelnet40_normal_resampled.zip`. We follow the hyperparameters of [60,61] with improvements from the implementation of [84] on which we base our experiments. We train a standard PointNet for 200 epochs with a batch size of 24 with Adam [82] with base learning rate of 0.001, weight decay of 0.0001. During training we sample 1024 3D points from every example in *modelnet40*, randomly scale with a scale from the interval $[0.8, 1.25]$, and randomly translate by an offset of up to 0.1 - if not otherwise mentioned in our experiments. This is the training procedure proposed in [84]. However, we always train the model for the the full 200 epochs and report final *class* accuracy based on the final result - we do not report instance accuracy. We further report invariance tests based on the final model.

As described in the main body, we evaluate rotational invariance by testing on 16×16 regularly spaced angles from $[0, 2\pi]$, rotating along xy and yz axes. We evaluate scaling invariance by testing the scales $\{0.001, 0.01, 0.1, 0.5, 1.0, 5.0, 10, 100, 1000\}$. All experiments for this dataset were run on three single GPU office machines, containing an NVIDIA TITAN Xp, and two GTX 2080ti, respectively.

F Additional Numerical Results

F.1 Invariance to continous image rotations

Discretization effects in CUB200 We further investigate the effect of discretization using different interpolation schemes for rotation on higher resolution on the CUB-200 dataset (trained at 224x224 resolution) fine-tuned using Resnet-50. Tab. 8 shows the results of different training schemes with and without our orbit mapping (OM) obtained when using different interpolation schemes for rotation. Besides standard training (*Std.*), we use rotation augmentation (RA), and the adversarial training and regularization from [10,12]. Even for this higher resolution dataset, the worst-case accuracies between different types of interpolation may differ by more than 15%.

In particular, adversarial training with bi-linear interpolation is still more vulnerable to image rotations with nearest neighbor interpolation. Even the

Train	OM	Clean.	Average			Worst-case		
			Nearest	Bilinear	Bicubic	Nearest	Bilinear	Bicubic
Std.	✗	**77.41±0.33**	37.67±0.35	52.45±0.29	51.87±0.31	3.19±0.49	8.07±0.35	8.16±0.33
	✓Train+Test	71.19±0.34	63.35±0.30	71.56±0.34	70.93±0.35	40.63±0.48	58.80±0.39	59.02±0.41
RA.	✗	69.89±0.28	67.61±0.33	70.12±0.34	68.83±0.37	34.88±0.47	41.01±0.41	40.50±0.43
	✓Test	69.41±0.31	69.19±0.32	69.27±0.29	68.53±0.38	48.63±0.43	56.28±0.39	55.86±0.40
	✓Train+Test	70.35±0.46	69.41±0.23	70.72±0.18	70.37±0.34	47.92±0.26	57.54±0.39	57.62±0.14
Advers.	✗	64.54±0.17	53.74±0.65	64.07±0.25	63.22±0.54	26.63±0.79	42.82±0.60	42.44±0.55
Mixed	✗	68.56±0.46	57.17±0.60	65.91±0.42	65.76±0.51	28.06±0.58	42.87±0.32	42.92±0.38
Advers.-KL	✗	64.47±0.35	53.93±0.35	64.65±0.26	64.02±0.34	26.94±0.46	43.04±0.63	42.61±0.37
Advers.-ALP	✗	64.63±0.31	55.56±0.67	64.34±0.17	63.21±0.24	29.55±0.69	43.63±0.21	43.48±0.32
ETN	✗	64.14±0.24	64.26±0.65	66.95±0.42	64.32±0.62	43.33±1.01	52.85±1.12	49.72±1.31
TIpool	✗	76.80±0.25	60.67±0.79	74.90±0.15	74.82±0.24	36.06±1.12	59.04±0.37	59.50±0.41
TIpool-RA	✗	73.47±0.48	72.30±0.51	74.71±0.29	73.65±0.36	57.22±0.64	62.82±0.56	62.31±0.42
TIpool	✓Train+Test	76.82±0.15	68.50±0.58	**77.18±0.18**	**77.04±0.16**	49.85±0.65	**69.19±0.36**	**69.64±0.33**
TIpool-RA	✓Train+Test	74.78±0.20	**73.79±0.48**	75.89±0.17	75.07±0.16	**59.57±0.57**	67.78±0.20	67.64±0.18

Table 8: Effect of augmentation and including gradient based orbit mapping *(OM)* on robustness to rotations with different interpolations for CUB200 classification using Resnet50. Shown are clean accuracy on standard test set and average and worst-case accuracies on rotated test set. Mean and standard deviations over 5 runs are reported.

learned ETN also exhibits similar behavior. While our approach is also affected by the interpolation effects, the vulnerability to nearest neighbor interpolation is ameliorated when using rotation augmentation. We obtain best results using orbit mapping in conjunction with the discrete invariant approach [36]

Effect of Network architecture for CIFAR10 To investigate the effectiveness of our approach, we experiment three different network architectures: *i) a linear network, ii) a 5-layer convnet ii) a Resnet18.* We compare the performance of our orbit mapping approach with training schemes, i.e. augmentation and adversarial training for rotational invariance in Tab. 9. For all the three architectures considered, our orbit mapping together with rotation augmentation consistently results in the most accurate predictions in the worst case.

Comparing Computation Complexity for CIFAR10 In Tab. 10, the training times using different approaches are compared for rotation-invariant CIFAR10 classification. It can be noted that the proposed gradient based orbit mapping is significantly easier and computationally cheaper to train in comparison with other approaches for incorporating invariance. In contrast, adversarial training is the most computationally expensive approach.

Comparing Computational Complexity of ROTMNIST Tab. 11 compares the computational complexity of the D4/C4 and D16/C16 models. The D16/C16 model has significantly higher computational complexity than the D4/C4 model, though the number of learnable parameters is nearly same. The network size of D16/C16 network s higher due to more rotated copies of the filters, resulting in larger training and inference times. Orbit mapping adds no learnable parameters and increases training time very marginally ($\sim$0.3 seconds/epoch). Training times correspond to runs on a machine with single Titan-RTX GPU.

Network	Train	OM	Std.	Average			Worst-case		
				Nearest	Bilinear	Bicubic	Nearest	Bilinear	Bicubic
Linear	Std.	✗	**38.89±0.17**	25.31±0.21	25.57±0.22	25.48±0.24	2.50±0.11	3.56±0.17	3.26±0.11
		✓Train+Test	31.87±0.10	**31.25±0.04**	**31.58±0.05**	**31.33±0.04**	13.08±0.23	18.85±0.21	18.21±0.21
	RA	✗	29.73±0.18	30.66±0.03	30.77±0.03	30.72±0.03	14.30±0.42	18.31±0.29	16.94±0.37
		✓Test	30.60±0.13	30.52±0.07	30.65±0.08	30.54±0.09	16.83±0.47	21.17±0.28	20.37±0.26
		✓Train+Test	31.06±0.26	31.07±0.11	31.27±0.10	31.13±0.09	**19.19±0.28**	**24.25±0.31**	**23.68±0.31**
	Advers.	✗	28.82±0.77	29.46±0.60	29.62±0.56	29.36±0.56	11.45±0.81	14.20±0.93	13.65±0.55
Convnet	Std.	✗	**86.12±0.33**	32.01±0.32	35.97±0.26	38.15±0.36	0.85±0.09	0.57±0.06	0.89±0.14
		✓Train+Test	76.13±0.96	64.34±0.35	71.21±0.96	**74.61±0.84**	25.78±0.49	49.60±0.79	55.57±0.81
	RA	✗	75.03±0.99	71.77±0.84	65.45±0.66	70.22±0.66	27.96±0.50	27.06±0.61	32.51±0.53
		✓Test	70.12±0.64	67.64±0.55	61.03±0.67	66.09±0.71	39.01±0.57	42.88±0.90	49.39±0.68
		✓Train+Test	74.30±0.77	**73.24±0.58**	69.52±0.53	73.38±0.59	**46.25±0.54**	**53.36±0.57**	**59.04±0.53**
	Advers.	✗	72.96±0.95	62.08±0.59	**74.29±0.88**	73.86±0.76	26.24±0.43	50.99±0.54	52.46±0.51
Resnet18	Std.	✗	**93.98±0.32**	35.12±0.81	40.06±0.44	42.81±0.50	0.79±0.38	1.31±0.13	2.22±0.17
		✓ Train+Test	87.99±0.43	72.40±0.33	**84.12±0.55**	**86.61±0.49**	34.57±0.94	68.60±0.81	74.49±0.84
	RA	✗	85.54±0.72	80.47±0.74	75.99±0.72	79.47±0.65	45.50±0.83	44.71±0.74	50.50±0.78
		✓ Test	79.26±0.42	74.93±0.51	69.31±0.65	73.94±0.63	48.93±0.75	52.18±0.91	58.69±0.78
		✓ Train+Test	85.40±0.57	**84.37±0.58**	81.82±0.59	84.82±0.52	**66.22±0.75**	**71.09±1.01**	**76.44±0.89**
	Advers.	✗	69.32±1.61	61.73±1.12	68.54±0.68	68.00±0.31	36.95±0.97	50.21±0.55	49.73±0.98

Table 9: Comparing rotational invariance using training schemes vs. orbit mapping for CIFAR10 classification using *i) Linear network ii) 5-layer Convnet iii) Resnet18*. Shown are the mean clean accuracy and the average and worst case accuracies when test images are rotated in steps of 1 degree. The mean and standard deviation values over 5 runs are reported.

Method	Std.	STN	ETN	Adv.	OM
Train-time/epoch	18.05±0.05	18.90±0.05	18.89±0.07	72.09±0.18	18.59±0.04

Table 10: Average training time per epoch in seconds for different approaches to incorporate rotation invariance, with Resnet18 as base architecture for CIFAR10 classification. Training time correspond to runs on a machine with single Titan-RTX GPU.

OM	D4/C4 Train-time/epoch	D16/C16 Train-time/epoch
✗	4.47 s	41.89 s
✓	4.78 s	42.08 s

Table 11: Comparing computational complexity of D4/C4 and D16/C16 models. Orbit mapping adds no learnable parameters and increases training time very marginally (∼0.3 seconds/epoch). Training times correspond to runs on a machine with single Titan-RTX GPU.

References

1. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115** (2015) 211–252
2. Tensmeyer, C., Martinez, T.: Improving invariance and equivariance properties of convolutional neural networks. (2016)
3. Olah, C., Cammarata, N., Voss, C., Schubert, L., Goh, G.: Naturally occurring equivariance in neural networks. Distill **5** (2020) e00024–004
4. Lenc, K., Vedaldi, A.: Understanding image representations by measuring their equivariance and equivalence. International Journal of Computer Vision **127** (2018) 456–476
5. Engstrom, L., Tsipras, D., Schmidt, L., Madry, A.: A rotation and a translation suffice: Fooling cnns with simple transformations. arXiv preprint arXiv:1712.02779 **1** (2017) 3
6. Finlayson, S.G., Bowers, J.D., Ito, J., Zittrain, J.L., Beam, A.L., Kohane, I.S.: Adversarial attacks on medical machine learning. Science **363** (2019) 1287–1289
7. Zhao, Y., Wu, Y., Chen, C., Lim, A.: On isometry robustness of deep 3d point cloud models under adversarial attacks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020)
8. Lang, I., Kotlicki, U., Avidan, S.: Geometric adversarial attacks and defenses on 3d point clouds. In: 2021 International Conference on 3D Vision (3DV). (2021)
9. Simard, P., Victorri, B., LeCun, Y., Denker, J.: Tangent prop-a formalism for specifying selected invariances in an adaptive network. In: Advances in neural information processing systems. Volume 4. (1991)
10. Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., Madry, A.: Exploring the landscape of spatial robustness. In: International Conference on Machine Learning. (2019) 1802–1811
11. Wang, R., Yang, Y., Tao, D.: Art-point: Improving rotation robustness of point cloud classifiers via adversarial rotation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (2022) 14371–14380
12. Yang, F., Wang, Z., Heinze-Deml, C.: Invariance-inducing regularization using worst-case transformations suffices to boost accuracy and spatial robustness. Advances in Neural information processing systems (2019) 14757–14768
13. Anselmi, F., Leibo, J.Z., Rosasco, L., Mutch, J., Tacchetti, A., Poggio, T.: Unsupervised learning of invariant representations. Theoretical Computer Science **633** (2016) 112–121
14. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: European conference on computer vision, Springer (2016) 69–84
15. Komodakis, N., Gidaris, S.: Unsupervised representation learning by predicting image rotations. In: International Conference on Learning Representations (ICLR). (2018)
16. Zhang, L., Qi, G.J., Wang, L., Luo, J.: Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 2547–2555
17. Gu, J., Yeung, S.: Staying in shape: learning invariant shape representations using contrastive learning. In de Campos, C., Maathuis, M.H., eds.: Conference on Uncertainty in Artificial Intelligence. Volume 161., PMLR (2021) 1852–1862

18. Wilk, M.v.d., Bauer, M., John, S., Hensman, J.: Learning invariances using the marginal likelihood. In: Advances in Neural information processing systems. (2018) 9960–9970
19. Benton, G.W., Finzi, M., Izmailov, P., Wilson, A.G.: Learning invariances in neural networks from training data. In: Advances in Neural information processing systems. (2020)
20. Sheng, Y., Shen, L.: Orthogonal fourier–mellin moments for invariant pattern recognition. Journal of the Optical Society of America **11** (1994) 1748–1757
21. Yap, P.T., Jiang, X., Chichung Kot, A.: Two-dimensional polar harmonic transforms for invariant image representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **32** (2010) 1259–1270
22. Tan, T.: Rotation invariant texture features and their use in automatic script identification. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 751–756
23. Lazebnik, S., Schmid, C., Ponce, J.: A sparse texture representation using local affine regions. IEEE Transactions on Pattern Analysis and Machine Intelligence **27** (2005) 1265–1278
24. Manthalkar, R., Biswas, P.K., Chatterji, B.N.: Rotation and scale invariant texture features using discrete wavelet packet transform. Pattern Recognition Letters **24** (2003) 2455–2462
25. Bruna, J., Mallat, S.: Invariant scattering convolution networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013) 1872–1886
26. Sifre, L., Mallat, S.: Rotation, scaling and deformation invariant scattering for texture discrimination. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2013)
27. Oyallon, E., Mallat, S.: Deep roto-translation scattering for object classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2015)
28. Cohen, T., Welling, M.: Group equivariant convolutional networks. In: International conference on machine learning. (2016) 2990–2999
29. Worrall, D.E., Garbin, S.J., Turmukhambetov, D., Brostow, G.J.: Harmonic networks: Deep translation and rotation equivariance. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
30. Weiler, M., Cesa, G.: General E(2)-Equivariant Steerable CNNs. In: Advances in Neural information processing systems. (2019)
31. Zhang, J., Yu, M.Y., Vasudevan, R., Johnson-Roberson, M.: Learning rotation-invariant representations of point clouds using aligned edge convolutional neural networks. In: 2020 International Conference on 3D Vision (3DV), IEEE (2020) 200–209
32. Yu, R., Wei, X., Tombari, F., Sun, J.: Deep positional and relational feature learning for rotation-invariant point cloud analysis. In: European Conference on Computer Vision. (2020)
33. Balunovic, M., Baader, M., Singh, G., Gehr, T., Vechev, M.: Certifying geometric robustness of neural networks. Advances in Neural information processing systems **32** (2019)
34. Fischer, M., Baader, M., Vechev, M.: Certified defense to image transformations via randomized smoothing. In: Advances in Neural information processing systems. Volume 33. (2020)
35. Manay, S., Cremers, D., Hong, B.W., Yezzi, A.J., Soatto, S.: Integral invariants for shape matching. IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006) 1602–1618

36. Laptev, D., Savinov, N., Buhmann, J.M., Pollefeys, M.: Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In: IEEE conference on computer vision and pattern recognition. (2016) 289–297
37. Ravanbakhsh, S., Schneider, J., Poczos, B.: Equivariance through parameter-sharing. In: International Conference on Machine Learning, PMLR (2017) 2892–2901
38. Xiao, Z., Lin, H., Li, R., Geng, L., Chao, H., Ding, S.: Endowing deep 3d models with rotation invariance based on principal component analysis. In: IEEE International Conference on Multimedia and Expo (ICME), IEEE (2020)
39. Li, F., Fujiwara, K., Okura, F., Matsushita, Y.: A closer look at rotation-invariant deep point cloud analysis. In: International Conference on Computer Vision (ICCV). (2021) 16218–16227
40. Rempe, D., Birdal, T., Zhao, Y., Gojcic, Z., Sridhar, S., Guibas, L.J.: Caspr: Learning canonical spatiotemporal point cloud representations. Advances in Neural information processing systems **33** (2020) 13688–13701
41. Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 2642–2651
42. Sun, W., Tagliasacchi, A., Deng, B., Sabour, S., Yazdani, S., Hinton, G.E., Yi, K.M.: Canonical capsules: Self-supervised capsules in canonical pose. Advances in Neural information processing systems **34** (2021)
43. Spezialetti, R., Stella, F., Marcon, M., Silva, L., Salti, S., Di Stefano, L.: Learning to orient surfaces by self-supervised spherical cnns. Advances in Neural information processing systems **33** (2020)
44. Sajnani, R., Poulenard, A., Jain, J., Dua, R., Guibas, L.J., Sridhar, S.: Condor: Self-supervised canonicalization of 3d pose for partial shapes. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2022) 16969–16979
45. Rehman, H.Z.U., Lee, S.: Automatic image alignment using principal component analysis. IEEE Access **6** (2018) 72063–72072
46. Jafari-Khouzani, K., Soltanian-Zadeh, H.: Radon transform orientation estimation for rotation invariant texture analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **27** (2005) 1004–1008
47. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: Advances in Neural information processing systems. (2015)
48. Tai, K.S., Bailis, P., Valiant, G.: Equivariant transformer networks. In: International Conference on Machine Learning, PMLR (2019) 6086–6095
49. Esteves, C., Allen-Blanchette, C., Zhou, X., Daniilidis, K.: Polar transformer networks. In: International Conference on Learning Representations. (2018)
50. Marcos, D., Volpi, M., Komodakis, N., Tuia, D.: Rotation equivariant vector field networks. In: IEEE International Conference on Computer Vision. (2017) 5048–5057
51. Veeling, B.S., Linmans, J., Winkens, J., Cohen, T., Welling, M.: Rotation equivariant cnns for digital pathology. In: International Conference on Medical image computing and computer-assisted intervention, Springer (2018) 210–218
52. Marcos, D., Volpi, M., Tuia, D.: Learning rotation invariant convolutional filters for texture classification. In: International Conference on Pattern Recognition (ICPR), IEEE (2016) 2012–2017
53. Fasel, B., Gatica-Perez, D.: Rotation-invariant neoperceptron. In: International Conference on Pattern Recognition (ICPR). Volume 3., IEEE (2006) 336–339
54. Henriques, J.F., Vedaldi, A.: Warped convolutions: Efficient invariance to spatial transformations. In: International Conference on Machine Learning, PMLR (2017) 1461–1469

55. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shape modeling. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2015)
56. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. ACM Transactions on Graphics (SIGGRAPH) **36** (2017)
57. Weiler, M., Geiger, M., Welling, M., Boomsma, W., Cohen, T.: 3d steerable cnns: learning rotationally equivariant features in volumetric data. In: Advances in Neural information processing systems. (2018) 10402–10413
58. Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., Riley, P.: Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. arXiv preprint arXiv:1802.08219 (2018)
59. Fuchs, F., Worrall, D., Fischer, V., Welling, M.: Se (3)-transformers: 3d roto-translation equivariant attention networks. Advances in Neural information processing systems **33** (2020)
60. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
61. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural information processing systems. (2017)
62. Zhang, Y., Rabbat, M.: A graph-cnn for 3d point cloud classification. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP). (2018)
63. Horie, M., Morita, N., Hishinuma, T., Ihara, Y., Mitsume, N.: Isometric transformation invariant and equivariant graph convolutional networks. In: International Conference on Learning Representations. (2020)
64. Satorras, V.G., Hoogeboom, E., Welling, M.: E(n) equivariant graph neural networks. In Meila, M., Zhang, T., eds.: International Conference on Machine Learning. Volume 139., PMLR (2021) 9323–9332
65. Esteves, C., Allen-Blanchette, C., Makadia, A., Daniilidis, K.: Learning so (3) equivariant representations with spherical cnns. In: European Conference on Computer Vision (ECCV). (2018) 52–68
66. Rao, Y., Lu, J., Zhou, J.: Spherical fractal convolutional neural networks for point cloud recognition. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 452–460
67. Deng, H., Birdal, T., , Ilic, S.: Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In: European Conference on Computer Vision (ECCV). (2018)
68. Zhao, Y., Birdal, T., Deng, H., Tombari, F.: 3d point capsule networks. In: Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
69. Monti, F., Boscaini, D., Masci, J., Rodolá, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
70. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: Meshcnn: A network with an edge. ACM Transactions on Graphics (TOG) **38** (2019) 90:1–90:12
71. Sharp, N., Attaiki, S., Crane, K., Ovsjanikov, M.: Diffusion is all you need for learning on surfaces. CoRR **abs/2012.00888** (2020)

72. Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., Guibas, L.: Functional maps: a flexible representation of maps between shapes. ACM Transactions on Graphics **31** (2012)
73. Pinkall, U., Polthier, K.: Computing discrete minimal surfaces and their conjugates. Experimental Mathematics (1993)
74. Litany, O., Remez, T., Rodolà, E., Bronstein, A., Bronstein, M.: Deep functional maps: Structured prediction for dense shape correspondences. In: International Conference on Computer Vision (ICCV). (2017)
75. Eisenberger, M., Toker, A., Leal-Taixé, L., Cremers, D.: Deep shells: Unsupervised shape correspondence with optimal transport. In: Advances in Neural information processing systems. (2020)
76. Huang, R., Rakotosaona, M.J., Achlioptas, P., Guibas, L., Ovsjanikov, M.: Operatornet: Recovering 3d shapes from difference operators. In: International Conference on Computer Vision (ICCV). (2019)
77. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition. (2016) 770–778
78. Tschandl, P., Rosendahl, C., Kittler, H.: The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Scientific data **5** (2018) 1–9
79. Brock, A., De, S., Smith, S.L., Simonyan, K.: High-performance large-scale image recognition without normalization. arXiv preprint arXiv:2102.06171 (2021)
80. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset. (2011)
81. Weiler, M., Geiger, M., Welling, M., Boomsma, W., Cohen, T.: 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In: Advances in Neural information processing systems. (2018)
82. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
83. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: IEEE conference on computer vision and pattern recognition. (2015) 1912–1920
84. Yan, X.: Pointnet/pointnet++ pytorch. https://github.com/yanx27/Pointnet_Pointnet2_pytorch (2019)