CubeNet: Equivariance to 3D Rotation and Translation

Daniel Worrall[1][0000−0002−9810−0709] and Gabriel Brostow[1][0000−0001−8472−3828]

Computer Science Department, University College London, UK
{d.worrall,g.brostow}@cs.ucl.ac.uk

Abstract. 3D Convolutional Neural Networks are sensitive to transformations applied to their input. This is a problem because a voxelized version of a 3D object, and its rotated clone, will look unrelated to each other after passing through to the last layer of a network. Instead, an idealized model would preserve a meaningful representation of the voxelized object, while explaining the pose-difference between the two inputs. An equivariant representation vector has two components: the invariant identity part, and a discernable encoding of the transformation. Models that can't explain pose-differences risk "diluting" the representation, in pursuit of optimizing a classification or regression loss function.

We introduce a Group Convolutional Neural Network with linear equivariance to translations *and* right angle rotations in three dimensions. We call this network *CubeNet*, reflecting its cube-like symmetry. By construction, this network helps preserve a 3D shape's global and local signature, as it is transformed through successive layers. We apply this network to a variety of 3D inference problems, achieving state-of-the-art on the ModelNet10 classification challenge, and comparable performance on the ISBI 2012 Connectome Segmentation Benchmark. To the best of our knowledge, this is the first 3D rotation equivariant CNN for voxel representations.

Keywords: Deep Learning, Equivariance, 3D Representations

1 Introduction

Convolutional neural networks (CNNs) are the go-to model for most prediction-based computer vision problems. However, most popularized CNNs are treated as black-boxes, lacking interpretability and simple properties concerning the data domains they act on. For instance, in 3D object recognition, we know that object categories are *invariant* to object pose, but convolutional neural network filters are orientation, scale, reflection, and parity (point reflection) selective. This means that every activation in any intermediate layer is sensitive to local pose, and ultimately the global output of the network is too. A simple solution to obtain this sought-after invariance is to augment the input data with transformed copies, spanning all possible variations, to which we seek to be invariant [2]. This method is simple and effective, but relies on an efficient and realistic data augmentation pipeline. There is also the argument,

why should we bother learning these invariances, if we can enforce them *a priori*? If successful, we would not need as much training data [8, 50]. Indeed, convolutional neural networks already have i) filter locality and ii) translational weight-tying built directly into their architectures, which arguably could be learned using a multilayer perceptron with a enough computational budget and training data.

We introduce a CNN architecture, which is *linearly equivariant* (a generalization of invariance defined in the next section) to 3D rotations about patch centers. To the best of our knowledge, this paper provides the first example of a group-CNN [8] with linear equivariance to 3D rotations and 3D translations of voxelized data. By exploiting the symmetries of the classification task, we are able to reduce the number of trainable parameters using judicious weight tying. We also need less training and test time data augmentation, since some aspects of 3D geometry are already 'hard-baked' into the network. We demonstrate state-of-the-art and comparable performance on i) the ModelNet10 classification challenge, which is a standard 3D classification benchmark task, and ii) the ISBI 2012 connectome segmentation benchmark, which is a 3D anisotropic boundary segmentation problem. We have released our code at `https://deworrall92.github.com`.

2 Background

For completeness, we set out our terminology and definitions. We outline definitions of linear equivariance, invariance, groups, and convolution, and then combine these ideas into the group convolution, which is the workhorse of the paper. These definitions are not our contribution and can be found in textbooks such as [7], but we have tried to standardize them and simplify notation.

Definition 1 (Equivariance) *Consider a set of transformations G, where individual transformations are indexed as $g \in G$. Consider also a function or feature map $\boldsymbol{\Phi}: \mathcal{X} \to \mathcal{Y}$ mapping inputs $\boldsymbol{x} \in \mathcal{X}$ to outputs $\boldsymbol{y} \in \mathcal{Y}$. Transformations can be applied to any $\boldsymbol{x} \in \mathcal{X}$ using the operator $\mathcal{T}_g^{\mathcal{X}}: \mathcal{X} \to \mathcal{X}$, so that $\boldsymbol{x} \mapsto \mathcal{T}_g^{\mathcal{X}}[\boldsymbol{x}]$. The same can be done for the outputs with $\boldsymbol{y} \mapsto \mathcal{T}_g^{\mathcal{Y}}[\boldsymbol{y}]$. We say that $\boldsymbol{\Phi}$ is* equivariant *to G if*

$$\boldsymbol{\Phi}(\mathcal{T}_g^{\mathcal{X}}[\boldsymbol{x}]) = \mathcal{T}_g^{\mathcal{Y}}[\boldsymbol{\Phi}(\boldsymbol{x})], \qquad \forall g \in G. \tag{1}$$

Since $\mathcal{T}_g^{\mathcal{X}}$ and $\mathcal{T}_g^{\mathcal{Y}}$ are related via (1), they are essentially different *representations* of the same transformation. Due to this connection, it is customary to drop the $\mathcal{T}_g^{\bullet}$ notation and write

$$\boldsymbol{\Phi}(g\mathbf{x}) = g\boldsymbol{\Phi}(\mathbf{x}). \tag{2}$$

Equivariance is important, because it highlights an explicit relationship between input transformations and feature-space transformations, which in the context of deep learning is not well-understood. An example of an equivariant task is pose-detection, where g represents the sought-after pose. The kind of equivariant feature maps, we are

interested in, are those where $\mathcal{T}^{\mathcal{X}}$ and $\mathcal{T}^{\mathcal{Y}}$ are linear. Such feature maps are known as *linearly equivariant*. A special case of equivariance is *invariance*, where we have

$$\boldsymbol{\Phi}(\mathbf{x}) = \boldsymbol{\Phi}(g\mathbf{x}), \tag{3}$$

that is, the feature-space transformation is just the identity. An example of an invariant task is object classification. Note when we use the term equivariant in the rest of the paper, we will generally refer to non-invariance.

Groups Invertible transformations are members of a class of mathematical objects called *groups*. Groups are a mathematical abstraction, which are used to describe the compositional structure of mathematical operators, such as transformations. Groups have four main properties: for group elements $f,g,h \in G$

1. **closure**: chained transformations are transformations, e.g. $fg \in G$
2. **associativity**: f(gh) = (fg)h = fgh
3. **identity**: there exists a transformation $e \in G$ (sometimes written $\mathbf{0}$) such that $eg = ge = g, \forall g \in G$
4. **invertibility**: every transformation g has an inverse g^{-1}, so $gg^{-1} = g^{-1}g = e$.

Rotations and translations are both examples of groups.

Convolution The fundamental operation in convolutional neural networks is the convolution—technically CNNs perform cross-correlation, but we stick with the term 'convolution' to remain in sync with the literature. $\star$. In 3D, convolution is the inner product of a *filter* $\mathbf{W} \in \mathbb{R}^{h \times w \times d}$ with patches extracted from an *activation tensor* or *feature map* $\mathbf{F} \in \mathbb{R}^{H \times W \times D}$ where h,w,d,H,W,D are the **h**eight, **w**idth, and **d**epth of the filter/activations respectively. The method of patch extraction is usually a translationally sliding window. So given a filter $\mathbf{W}$, the translated version is $g\mathbf{W}$, such that

$$[\mathbf{F} \star \mathbf{W}]_g = \sum_{\mathbf{x} \in \mathbb{Z}^3} [g\mathbf{W}]_{\mathbf{x}} \mathbf{F}_{\mathbf{x}} = \sum_{\mathbf{x} \in \mathbb{Z}^3} \mathbf{W}_{g^{-1}\mathbf{x}} \mathbf{F}_{\mathbf{x}}; \tag{4}$$

where to index elements of the filters/activations we have used the multi-index notation $\mathbf{W}_{\mathbf{x}} := \mathbf{W}_{x,y,z}$ for $\mathbf{x} = [x,y,z]^\top \in \mathbb{Z}^3$, and so in this example $\mathbf{W}_{g^{-1}\mathbf{x}} = \mathbf{W}_{x-g_x, y-g_y, z-g_z}$ for voxel-wise translation in 3D by $g = [g_x, g_y, g_z]^\top$. This sliding-window interpretation of convolution can be viewed as applying the same filter to different local regions of the inputs. Note that in reality, since the feature map is zero outside of a a certain neighborhood, we need not sum over all $\mathbb{Z}^3$. Note also how the output of the convolution is indexed by the transformation parameter g; that is, the g^{th} activation corresponds to the response of a g-shifted filter $g\mathbf{W}$. We have used the notation $[\mathbf{F} \star \mathbf{W}]_g$ to emphasize that $[\mathbf{F} \star \mathbf{W}]$ is an indexable object like $\mathbf{W}$ or $\mathbf{F}$, and it can be viewed as a vector (see Figure 1). CNNs usually have multiple *channels* k per

activation tensor, so in general we really have

$$[\mathbf{F}\star\mathbf{W}]_g^k = \sum_{i=1}^{I}\sum_{\mathbf{x}\in\mathbb{Z}^3}[g\mathbf{W}]_{\mathbf{x}}^{ik}\mathbf{F}_{\mathbf{x}}^i, \tag{5}$$

where the dummy index i is over input channels with output channel k.

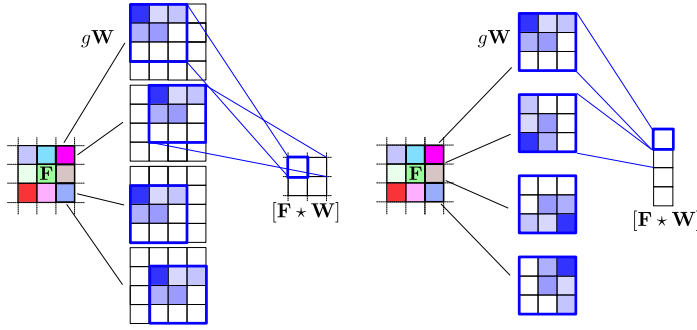

Fig. 1. (Best viewed in color) On the left we show the standard 2D convolution of Equation 4 between a sliding filter $\mathbf{W}$ and an input patch $\mathbf{F}$. On the right we show the 2D right-angle rotation convolution (called Z_4-convolution) acting on an input where $G=\mathbb{Z}^2$.

One can show (c.f. [8, 11] and Equations 8 & 9) that the standard translational convolution is equivariant to translations; that is, translations of the input to the convolution result in translations in the feature space representation $[\mathbf{F}\star\mathbf{W}]$. The extension of this translational equivariance to other groups of transformation is embodied in the *group convolution* [8], which we show next. This has been proven [29] to be the only operator which is equivariant to (compact) group-structured transformations.

Definition 2 (Group Convolution) *A group convolution between a filter* $\boldsymbol{W}$ *and a single-channel feature map* $\boldsymbol{F}$ *over a group of transformations* G *is*

$$[\boldsymbol{F}\star\boldsymbol{W}]_g = \sum_{h\in G}[g\,\boldsymbol{W}]_h\boldsymbol{F}_h = \sum_{h\in G}\boldsymbol{W}_{g^{-1}h}\boldsymbol{F}_h. \tag{6}$$

The extension to multichannel activations parallels Equation (5).

We see that the main difference between the standard convolution of Equation 4 and the group convolution of Equation 6 is that we have replaced the domain of summation from $\mathbb{Z}^3$ to the group G. So the sliding inner product could generalize to a sliding-and-rotating inner-product, or sliding-and-flipping inner product, or even sliding-and-scaling inner product depending on the choice of group G. A simple example is shown in Figure 1, where we show a 2D translational convolution and a first layer 2D right-angle rotational convolution (called Z_4-convolution). In this example, the domain of the Z_4-convolution is $G=\mathbb{Z}^2$, the standard 2D image domain,

but the output is over the group of four 2D rotations, Z_4. This amounts to taking an inner product of the kernel $\mathbf{W}$ rotated four times, with each individual response being stacked into a vector. If we were to then convolve a kernel over the response of this first Z_4-convolution, the domain of that convolution would be $G = Z_4$. Stacks of group convolutions turn out to be equivariant as well.

Note that the dimensionality of the convolutional responses is linear in the number of elements of the group G. At each layer, it is common to choose the size of the group to be the same, or smaller if we include pooling. To maintain a transformation invariant output, we using average over the the group at the final layer of the network, which is an extension of global average pooling to groups.

In this paper, we are interested in the group of 3D roto-translations. The group convolution for this group will involve us convolving an activation tensor with rotated and shifted copies of a filter $[g\mathbf{W}]_{\mathbf{x}} = \mathbf{W}_{g^{-1}\mathbf{x}} = \mathbf{W}_{\mathbf{R}_g^{-1}\mathbf{x}-\mathbf{z}_g}$, where $\mathbf{R}_g$ is a 3D rotation matrix and $\mathbf{z}_g$ is a translational offset.

3 Related Work

Recently there has been an explosion of interest into CNNs with predefined transformation equivariances, beyond translation [8, 50, 11, 29, 16, 42, 36, 14, 19, 31, 18, 22, 15, 55, 49, 26, 25, 48, 33, 28, 9]. However, with the exception of Cohen and Welling [9] (projections on sphere), Kondor [28] (point clouds), and Thomas et al. [48] (point clouds), these have mainly focused on the 2D scenario. There are also examples of CNNs, which have explicit regularization to learn equivariance [43, 30, 51, 40]. To the best of our knowledge, *we are the first to develop a 3D rotation equivariant CNN architecture for voxelized data.*

Handcrafted equivariance There are many computer vision models that exhibit equivariance properties. Perhaps the first notable instance is the scale-space [13], which specifically displays equivariance to isotropic scale, later extended to affine equivariance by Lindeberg [34]. In the presence of continuous transformations, Freeman and Adelson famously [17] (and less famously Lenz [32]), shored up the theory of *steerable filters*, which are a set of bandlimited linear filters $\mathbf{w}_\theta \in \mathbb{R}^{H\times W}$, which can be synthesized *exactly* at any rotation θ as a *finite* linear combination of basis filters

$$w_\theta(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n(\theta)\phi_n(\mathbf{x}). \tag{7}$$

These are attractive because their expressiveness is controlled by the number of coefficients N, rather than the spatial size of the filter. These have been applied to scale-spaces/pyramids in Simoncelli et al. [44], and have been placed on firm theoretical ground by Teo [47] in his PhD thesis. It has also been shown that for certain transformations, such as scalings (or more generally non-compact groups),

exact steering is only possible if $N = \infty$. In this case, Perona [37] showed that he could approximate Equation 7 using an SVD formulation. Like our method, all these works display handcrafted linear equivariance to a predefined set of transformations.

2D Rotation Invariant Neural Networks For CNNs, as mentioned, most works have focussed on 2D rotations. Fasel & Gatica-Perez [16], Laptev et al. [31], and Gonzalez et al. [19] average classifier predictions on multiple rotated copies of an input. Sifre & Mallat [42] and Oyallon & Mallat [36] use a scattering network [5] for roto-translation invariant classification. Every layer of these networks is locally (patch-wise) rotation invariant, performing a pre-determined wavelet transform averaging responses over rotation. Cotter and Kingsbury [12] recently suggested, however, that these networks lack discriminativeness, partially from the phase removal and partially from the fact that the wavelet transforms are not optimized per-task, which our method can handle.

2D Rotation Equivariant Neural Networks Henriques & Vedaldi [22] and Esteves et al. [15] perform a log-polar transform of the input, which converts scalings and rotations about a single point into a translation. Applying a standard translation equivariant CNN to this representation is then equivariant to rotations and scalings about the image center. This is only equivariant to global rotations, and does not generalize to 3D. For locally equivariant methods Dieleman et al. [14] maintain multiple rotated feature maps at every layer of a network; whereas, Cohen & Welling [8] rotate the filters. In the same paper, Cohen and Welling also extended this method to finite groups and later generalized this to arbitrary compact groups in [11]. Worrall et al. [50] generalized the filter rotation method to continuous rotations, using circular Fourier transforms to compute continuous rotation responses with a finite number of filters. At the same time Zhou et al. [55] extended the filter rotation method to non-90° rotations using bilinear interpolation. Gonzalez et al. [18] do similar, but also pool over rotations and use a representation similar to [50]. Weiler et al. [49] so far have the best solution to rotate filters, using steerable filters to solve the interpolation problem. Our method can be seen as an instance of Cohen & Welling [8] adapted to 3D rotation and translation.

Deeply Learned Equivariance There are many papers which also focus on learning equivariance. Tangent Prop by Simard et al. [43] is a classic example of an invariance inducing regularizer. Hinton et al. [23] introduced the transforming autoencoder to build latent spaces with equivariant structure. More recently, Worrall et al. [51] extended this method by imposing explicit transformation rules on the latent space. Papers such as InfoGAN by Chen et al. [6] and the Deep Convolutional Inverse Graphics Network of Kulkarni et al. [30] seek to learn equivariant structure in unsupervised fashion unsupervised. Most recently Sabour et al. [40] and Hinton et al. [24] achieved highly impressive results on the MNIST dataset with capsule networks by learning approximations to affine equivariance. While these methods are very flexible, they require lots of training data

3D Methods For classification, the most straightforward CNNs operating on 3D voxel data use 3D convolutions as of Equation 4 such as Maturana & Scherer [35] or 3D

Convolutional Deep Belief Network as in Wu et al. [53]. Brock et al. [4] take this to the extreme, designing an ensemble of six 45-layer deep inception- and resnet-style networks trained with a lot of data-augmentation and rotation averaging. Sedaghat et al. [41] rely less on brute force, augmenting the prediction task with orientation estimation. For 3D rotation equivariant methods, Cohen & Welling introduce the Spherical CNN [10], which operates on images projected onto the sphere, while Kondor [28] and Thomas et al. [48] operate on point clouds. All three methods use variants of a 3D extension of Worrall et al. [50], which introduced continuous rotation equivariance into CNNs, by use of the shifting property of Fourier transforms.

4 Method

We have introduced the concept of groups as a way to model transformations, and as a way to extend standard convolution to these transformations. Here, we chart out three different discrete 3D rotation groups; namely, Klein's four-group, the tetrahedral group and the cube group. We then show how to apply these groups in a group equivariant CNN using Cayley tables to build three different 3D rotation equivariant CNNs. We do not consider equivariant to continuous 3D rotations in this paper, leaving it for future work.

Cube Group The set of all right-angle rotations of a cubic filter $\mathbf{F}_\mathbf{x} \in \mathbb{R}^{N\times N\times N}$ forms a group. There are 24 such rotations, going by the name of the *cube group*[1] S_4. Each of the 24 rotations applied to a cube is shown in Figure 2. The group is non-commutative, so $\mathbf{F}_{(g_1 g_7)^{-1}\mathbf{x}} \neq \mathbf{F}_{(g_7 g_1)^{-1}\mathbf{x}}$ for rotations g_1 and g_7, for example.

Tetrahedral Group Using 24 copies of the same filter increases the computational overhead 24 times. A cheaper subsampling is the rotations of the tetrahedron. This has 12 states, and goes by the name of the *rotational tetrahedral group* T_4. T_4 is formally a subgroup[2] of the cube group, comprised of all even rotations (i.e. all rotations which can be made by two 90°-rotations). It is shown as the 12 cube rotations wrapped in thin blue in Figure 2.

Klein's Four-group The smallest subsampling of rotations, which can be seen as rotations about 3 independent axes is Klein's *Vierergruppe* V or *four-group*. It has four rotations as can be seen in Figure 3. This group is a subgroup of the rotational tetrahedral group and the cube group. Interestingly, it is commutative and also the smallest non-cyclic group. It is shown as the 4 rotations wrapped in dashed red in Figure 2.

[1] Other names are the subgroup O of the octohedral group; symmetric group S_4; and full tetrahedral group T_d.

[2] A subgroup H is any subset of G, which satisfies the four group axioms, which we introduced in the background section

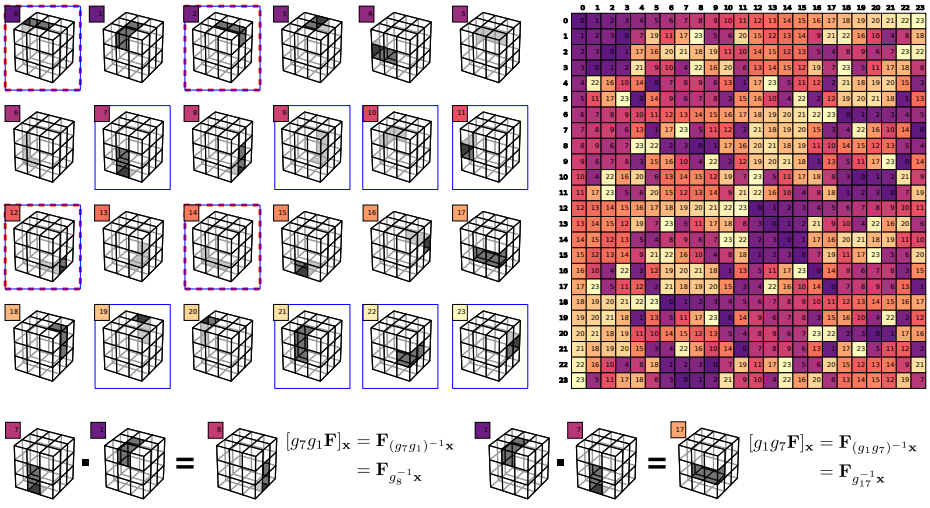

Fig. 2. (Best viewed in color) LEFT: The 24 rotations of the cube group S_4, applied to the a cube $\mathbf{F_x}$ are shown. For instance, rotation g_{22} applied to the cube returns $\mathbf{F}_{g_{22}^{-1}\mathbf{x}}$, shown by the #22 in the bottom row. The 12 cubes wrapped in thin blue boxes are the rotational tetrahedral group T_4. The 4 cubes wrapped in thick dashed red lines are the Klein four-group V. RIGHT: The Cayley table of the cube group, representing how rotations are composed. For instance, on the BOTTOM LEFT, we have the example of composing rotation g_7 with rotation g_1. The composition is performed by i) first applying g_7 to the cube to yield $\mathbf{F}_{g_7^{-1}\mathbf{x}}$ then ii) applying g_1 to $\mathbf{F}_{g_7^{-1}\mathbf{x}}$, returning $\mathbf{F}_{g_1^{-1}g_7^{-1}\mathbf{x}}$. The first transformation is easy to visualize - it is by #7 in the grid of cubes. The transformation g_1 is a rotation by 90° counter-clockwise about the vertical axis, thus for the composition we rotate $\mathbf{F}_{g_7^{-1}\mathbf{x}}$ 90° counter-clockwise about the z-axis. This results in $F_{g_8^{-1}\mathbf{x}}$. This result is stored in the Cayley table by placing the first rotation down the left column and the second rotation along the top row. The intersection of row **7** with column **1** is the rotation **8**. On the BOTTOM RIGHT, we show the composition $g_7 g_1 = g_{17} \neq g_8 = g_1 g_7$, demonstrating the non-commutativity property of the cube group and 3D rotations in general.

4.1 Cayley tables

Knowing how a rotation of the input will permute the convolutional response can be figured out from the group *Cayley table*. This is a multiplication table enumerating every composition of transformations. For Klein's four-group, we label the rotations as g_0 (the identity), g_1, g_2, & g_3. The Cayley table with instructions of how to read it are given in Table 1. The Cayley table is useful for determining how to perform the group convolution in deeper layers. We can see why this is the case because looking to the expression for the group convolution $\sum_{h \in G} \mathbf{W}_{g^{-1}h} \mathbf{F}_h$, we see a product $g^{-1}h$ in the indices of $\mathbf{W}$. We can use the Cayley table to ascertain the single transformation that is the result of the product. Looking closely at a Cayley table we see that all the rows/columns are permutations of one another, this will be important for understanding how input rotations affect the group-convolutional response.

Table 1. The Cayley table for Klein's four-group. The product $g_2 g_3$ (a g_2-rotation followed by a g_3-rotation) can be found by looking down the left column for the first transformation g_2, then finding the second transformation g_3 in the top row. The cell at the intersection of row-g_2 and column-g_3 (shaded in yellow) is g_1, so $g_2 g_3 = g_1$.

$\bullet$	g_0	g_1	g_2	g_3
g_0	g_0	g_1	g_2	g_3
g_1	g_1	g_0	g_3	g_2
g_2	g_2	g_3	g_0	g_1
g_3	g_3	g_2	g_1	g_0

4.2 Discrete Group Equivariance and Permutations

Rotating an input to a group convolution will lead to a transformation of its output. Specifically a rotation will lead to a permutation of the output, where we view the output as a vector of responses, with each dimension corresponding to a different group element/transformation $g \in G$. An example of this vectorized output can be seen in Figure 1. For translations the permutation is a voxel-wise shift, but for the aforementioned 3D rotations the permutations are much more complicated. If we apply a transformation p to the input features $\mathbf{F}$, then

$$[[p\mathbf{F}] \star \mathbf{W}]_g = \sum_{h \in G} [g\mathbf{W}]_h [p\mathbf{F}]_h = \sum_{h \in G} \mathbf{W}_{g^{-1}h} \mathbf{F}_{p^{-1}h} \tag{8}$$

$$= \sum_{h' \in G} \mathbf{W}_{g^{-1}ph'} \mathbf{F}_{h'} = [\mathbf{F} \star \mathbf{W}]_{p^{-1}g} = [p[\mathbf{F} \star \mathbf{W}]]_g. \tag{9}$$

Here we have made the substitution $h' = p^{-1}h$ and noted that $p^{-1}G = G$ for $p \in G$, where $p^{-1}G := \{p^{-1}g \mid g \in G\}$. What lines 8 and 9 say is that the output of the group convolution is permuted whenever the input $\mathbf{F}$ is transformed by an element of the group G. The specific permutation of the output depends on the specific transformation and transformation group. Thinking of $\mathbf{F} \star \mathbf{W}$ and $[p\mathbf{F}] \star \mathbf{W}$ as vectors separated by a permutation, we can write

$$[p\mathbf{F}] \star \mathbf{W} = p[\mathbf{F} \star \mathbf{W}] = \mathbf{P}_p [\mathbf{F} \star \mathbf{W}], \tag{10}$$

where the first equality is from Equations 8 and 9 and in the second equality we have rewritten the permutation as multiplication with the *permutation matrix* $\mathbf{P}_p$. In fact $\mathbf{P}_p$ is the permutation matrix corresponding to the p^{th} column of the Cayley table. Thus we see that group convolutions are *linearly* equivariant to transformations $p \in G$, as defined in Equation 1. We see an example of this for Klein's four-group in Figure 3, where we have labeled the four rotations as g_0 (the identity), g_1, g_2, & g_3.

4.3 Implementation: Roto-translational group-convolution

Now we show how to implement a group-convolution for a 3D roto-translation. In this example, we focus on the four-group to model rotations. A roto-translation can be

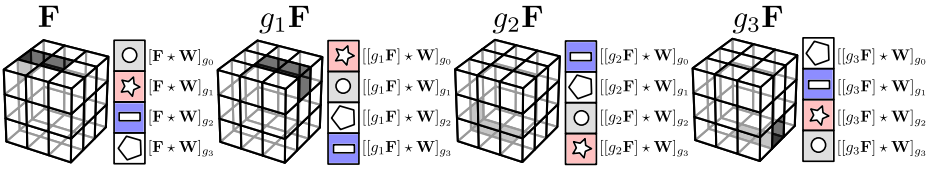

Fig. 3. Example of how the group convolution output permutes as a function of the input rotation. This example is for Klein's four-group V. Each cube represents a rotation from V and a corresponding example feature vector is given with each cube.

synthesized from a rotation, followed by a translation. Roto-translations form a group, which can be seen as the product[3] of V and $\mathbb{Z}^3$. For our purposes, it is safe to assume that we can write the elements of this producted group as tr for $t\in\mathbb{Z}^3$ and $r\in V$. So,

$$[\mathbf{F}\star\mathbf{W}]_{tr}=\sum_{\tau\in\mathbb{Z}^3}\sum_{\rho\in V}[tr\mathbf{W}]_{\tau\rho}\mathbf{F}_{\tau\rho}=\sum_{\tau\in\mathbb{Z}^3}\sum_{\rho\in V}\Big[t[r\mathbf{W}]_{\rho}\Big]_{\tau}\mathbf{F}_{\tau\rho}. \tag{11}$$

The interpretation behind this equation is as follows. First we start with a filter $\mathbf{W}$. $\mathbf{W}$ has a different value for each voxel in its receptive field, indexed by the translation variable τ, and also for every input rotation ρ—it may be easier just to think of four 3D filters, $\mathbf{W}_{\rho_0},\mathbf{W}_{\rho_1},\mathbf{W}_{\rho_2},\mathbf{W}_{\rho_3}$, one for each rotation in V. To convolve, we first rotate the kernel as $r\mathbf{W}_{\rho_\bullet}$, then we perform a translational shift $t[r\mathbf{W}_{\rho_\bullet}]$—this second part ends up as the standard convolution of Equation 4, which is efficient on GPUs. The initial rotation of the filter $r\mathbf{W}_{\rho_\bullet}$ can be found from composing r and $\rho_\bullet$ using our Cayley tables. It is the rotation needed to rotate r into $\rho_\bullet$. When the input is a raw image, the input domain is just $\mathbb{Z}^3$, so the rotation of $\mathbf{W}$ is just r.

To compute gradient for backpropagation we leverage the power of automatic differention, which is available in most modern neural network libraries.

5 Experiments and results

Here we describe two simple experiments we performed to demonstrate the effectiveness of group-convolutions on 3D voxelized data. We tested on the ModelNet10 classification challenge, which is a small 3D voxel dataset, and on the ISBI 2012 connectome segmentation challenge. In both examples, we found Klein's four-group to be the most effective group for the rotation-equivariant group-convolutions.

5.1 ModelNet10

The ModelNet 10 dataset [53] contains 4905 CAD models from 10 categories with a train:test split of 3991:914. Each model is aligned to a canonical frame and then

[3] Formally, this is a semi-direct product.

rotated at 12 evenly-sampled orientations about the z-axis. These rotated models are then voxelized to a 32x32x32 grid. We use the voxelized version of Maturana and Scherer [35]. While the dataset consists of vertically aligned models, rotated only about the z-axis, we posit that local features occur at all 3D rotations, and so a Cubenet is well positioned to operate on such as dataset. We use the four-group of rotations and the rotational tetrahedral group T_4, since we found the cube-group too large and slow to be trained practically multiple times during a model search.

We use a simple VGG-like [45] network architecture shown in Figure 4. It consists of 10 group-convolutional layers followed by a 2-layer fully-connected network. Before every convolution, we combine multiplicative dropout with 0.1 standard deviation on the filter tensors, and after every convolution we add batch normalization. We use ReLU nonlinearities and global average pooling before two fully-connected layers at the end of the network. The loss function is the multi-class cross-entropy. We initialize all weights using the He method [20] and train the network with ADAM stochastic gradient descent [27], with a learning rate of 1e-3, which steps down by 1/5 every 5 epochs for 25 epochs.

The data augmentation is performed similar to the implementation found in Brock [4] with 12 stratified rotations about the z-axis, reflections in the x- and y-axis with uniform probability and uniformly random translations of up to ± 4 voxels along all three axes. We use this data augmentation to maintain a direct comparison with prior works. It should also be noted that rotational data augmentation cannot be avoided entirely, since our networks are only equivariant to subgroups of the full roto-translation group $SE(3)$, so we still need to augment for all angles in the quotient $SE(3)/G$, where G is the subgroup of interest. We also rescale the voxel values to $\{-1{,}5\}$ instead of $\{0{,}1\}$ as in [4], who showed it helps with sparse voxel volumnes. We show our results in Table 2. We compare the rotational tetrahedral group and the four-group models. For the four-group model, we compare the average single-view accuracy across 5 models for robustness, with rotation averaged accuracy and single-view accuracy for the best model. The single view accuracy is computed as the accuracy averaged over each of the 12 rotated test views; whereas, the rotation averaged accuracy is computed as the accuracy of the average of all 12 predictions.

For the single-model category, our four-group, rotation-averaged network attains state-of-the-art performance. Interestingly, our single-view result we obtain is very similar to ORION [41], which introduces an orientation estimation task along with the classification. We posit that the T_4-model does not perform as well as the V-model, because increasing the number of filter copies reduces the diversity of filters, when the number of total filters (number of learnable filters times number of copies) is constrained. Essentially there is a tradeoff between filter diversity and the extent of equivariance due to weight-tying. The Klein-group appears to achieve best in this situation. It is also interesting to see that rotation averaging improves performance slightly, compared to our single-view model. We suggest this is because we are averaging over rotations not covers by the four-group. Looking across the model sizes, we see that the group-convolutional models sit somewhere in the middle in terms of number of parameters. Speed-wise, we found that during development the four-group network only trained about $2\times$ slower than non-group CNNs.

Table 2. Results for the ModelNet 10 benchmark. We compare against other methods which operate on a voxel-representation of the data. The only model to beat us is Brock et al.'s ensemble of 6 models. If we just restrict to a single model, then we hold state-of-the-art accuracy.

Method	ModelNet10 # params ($\times 10^6$, 2 s.f.)	
3D ShapeNets [53]	0.8354	12
Xu & Todovoric [54]	0.8800	0.080
3D-GAN [52]	0.9100	11
VRN [4]	0.9133	18
VoxNet [35]	0.9200	0.92
Fusion-Net [21]	0.9311	120
ORION [41]	0.9380	0.91
Ours T_4	0.9127	4.5
Our V (average)	0.9372	4.5
Ours V (best model single-view)	0.9420	4.5
Ours V (best model rotation averaged)	**0.9460**	4.5
VRN Ensemble [4]	0.9714	108

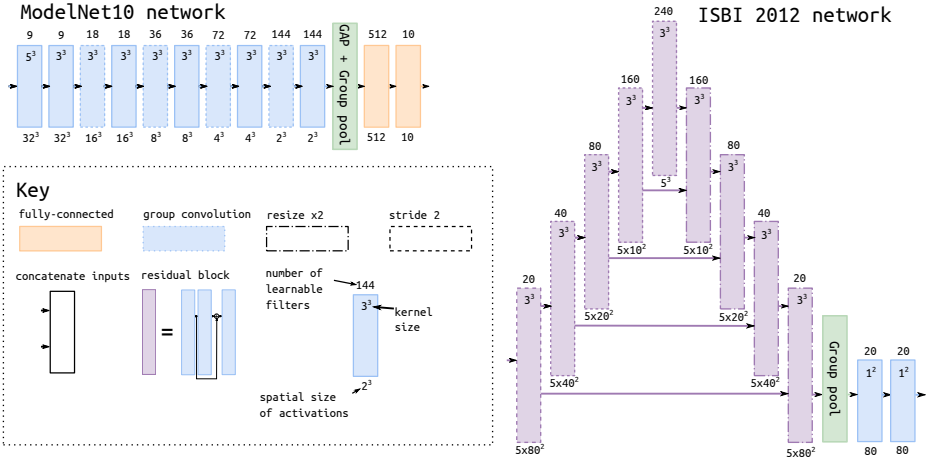

Fig. 4. (Best viewed in color) The architectures used in our experiments. We use a simple VGG-like architecture for the ModelNet10 classification challenge, and a UNet/FusionNet-like architecture for the ISBI2012 boundary segmentation benchmark.

5.2 ISBI 2012 Challenge: Connectome Segmentation

The ISBI 2012 Challenge is a volumetric boundary segmentation benchmark. The task is to segment Drosophilia ventral nerve cords from a serial-section transmission electron microscopy (EM) image [1]. The training set is a single $2 \times 2 \times 1.5$ μm^3 volume of anisotropic imaging resolution (high x-y resolution, low z resolution). Each voxel is $4 \times 4 \times 50$ nm^3 so the full training image is $512 \times 512 \times 30$ voxels in shape. The

test image is $512 \times 512 \times 30$ voxel, with withheld labels. Scoring is performed using the metrics V_{rand} and V_{info} described in [1]. Larger is better.

Here we are faced with two major issues, a) small dataset, b) high imaging anisotropy. We counter a) with heavy data augmentation as per [38] and by noting that group convolutions reduce the number of trainable parameters through significant weight-tying. To counter the imaging anisotropy, we use Klein's four-group, which is not affected by stretching of one of the axes.

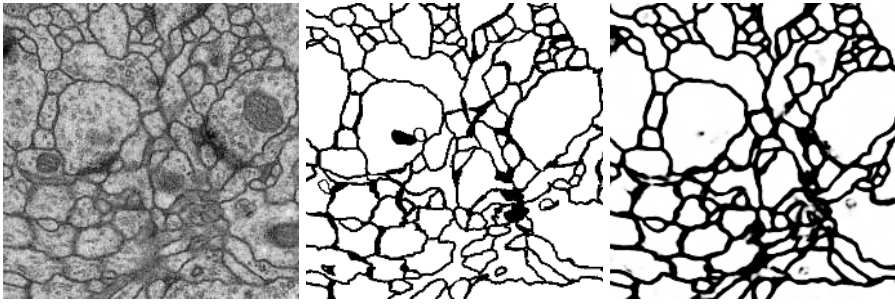

Fig. 5. Examples of 2D slices from the training volume, the associated label mask, and the prediction made by our network. The original volume contains small amounts of noise and certain structures within the volume are ambiguous in nature.

Competing methods segment on a single 2D high-resolution slice at a time, but as a proof of concept we try segmentation as a 3D problem, feeding 3D image chunks into a 3D network. We use an architecture as shown in Figure 4, based on Weiler et al.'s steerable version [49] of the FusionNet [38]. It is a UNet [39] with added skip connections within the encoder and decoder paths to encourage better gradient flow. We place Gaussian multiplicative dropout [46] with standard deviation 0.1 before every convolution. By this we mean if x is an activation and $n \sim \text{Normal}(n;1,0.1^2)$ then the result of dropout is $x \cdot n$. We also place batch normalization after every convolution and use ReLU nonlinearities directly before each convolution, except on the input.

For the training set we extract random $100 \times 100 \times 5$ voxel patches from the training volume and predict the center slice. We reflection pad 10 voxels in the x-y plane, and constant pad up to 5 voxels in the the z-direction if we sample at the upper or lower image boundaries. We then apply a random elastic distortion in the x-y-plane, and pass the patches through our group-equivariant FusionNet. We keep our implementation close to the design of Weiler et al. to maintain a close comparison, and do not perform extensive model search. The results are shown in Table 3.

Our results are comparable with other leading methods. Our V_{rand} metric is slightly improved over UNet and Quan et al., but not as good as Weiler et al., who use a 2D group convolutional neural network approach, with 17 rotations about the z-axis and lifting multicut post-process. The leading method uses the lifting multicut method

Table 3. Results for the ISBI 2012 challenge. We have tried to keep our implementation as close as possible to Weiler et al.. Unlike other methods, we perform no post-processing at all unlike Weiler et al. who use a lifting multi-cut [3] post-process, or UNet and Quan et al. who use rotation averaging. Quan also adds an optional median filtering to boost scores. This shows that we can adapt state-of-the-art models to process 3D volumetric data with little change in the competitiveness of the results.

Method	V_{rand}	V_{info}
UNet [39]	0.97276	0.98662
Quan et al. [38]	0.97804	0.98995
Ours	0.98018	0.98202
Weiler et al. [49]	0.98680	0.99144
IAL MC/LMC	0.98792	0.99183

too. Our V_{info} metric is not as good as the other methods, but we believe with sufficient model search, and extensive post-processing we could increase this number further. The main point of this experiment, as with the ModelNet10 experiment, was to demonstrate that we could get relatively good performance, without the need for extensive test-time rotation averaging.

6 Conclusion

We have presented a 3D convolutional neural network architecture, which is equivariant to right-angle rotations in three dimensions. This relies on an extension of the standard convolution to 3D rotations. On the ModelNet10 classification challenge, we have achieved state-of-the-art for a single model, beating some much larger models, which rely on heavy data augmentation. Since our models are rotation in/equivariant by design, our CNNs need not learn to *overcome* rotations, the way a standard CNN does. In 3D, this is an especially important gain. As a result, our model is positioned to get better generalization with less data, while avoiding the need to perform time-costly rotation averaging at test-time.

Another perspective on our approach is to think of it as global average pooling over rotations, where we expose a new 'rotation-dimension.' Without adhering to a defined group, it would be challenging to disentangle or orient a feature space (at any one layer, or across multiple layers) with respect to such a rotation dimension. The trade-off is that we commit to a group and its corresponding CubeNet architecture, to avoid the considerable effort of learning to disentangle pose.

We leave it to future work to examine whether these models can be generalized to continuous rotations and other challenging transformations, such as scale. There is also the untouched challenge of finding 3D rotation groups, which are not aligned to the Cartesian voxel-grid.

References

1. Arganda-Carreras, I., Turaga, S.C., Berger, D.R., Cirean, D., Giusti, A., Gambardella, L.M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J.M., Liu, T., Seyedhosseini, M., Tasdizen, T., Kamentsky, L., Burget, R., Uher, V., Tan, X., Sun, C., Pham, T.D., Bas, E., Uzunbas, M.G., Cardona, A., Schindelin, J., Seung, H.S.: Crowdsourcing the creation of image segmentation algorithms for connectomics. Frontiers in Neuroanatomy **9**, 142 (2015). https://doi.org/10.3389/fnana.2015.00142, https://www.frontiersin.org/article/10.3389/fnana.2015.00142
2. Barnard, E., Casasent, D.: Invariance and neural nets. IEEE Trans. Neural Networks **2**(5), 498–508 (1991). https://doi.org/10.1109/72.134287, https://doi.org/10.1109/72.134287
3. Beier, T., Andres, B., Köthe, U., Hamprecht, F.A.: An efficient fusion move algorithm for the minimum cost lifted multicut problem. In: Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II. pp. 715–730 (2016). https://doi.org/10.1007/978-3-319-46475-6_44,*https://doi.org*/10.1007/978−3−319−46475−6_44
4. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Generative and discriminative voxel modeling with convolutional neural networks (2016)
5. Bruna, J., Mallat, S.: Invariant scattering convolution networks. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1872–1886 (2013). https://doi.org/10.1109/TPAMI.2012.230, https://doi.org/10.1109/TPAMI.2012.230
6. Chen, X., Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 2172–2180 (2016), http://papers.nips.cc/paper/6399-infogan-interpretable-representation-learning-by-information-maximizing-generative-adversarial-nets
7. Chirikjian, G.S.: Engineering Applications of Noncommutative Harmonic Analysis: With Emphasis on Rotation and Motion Groups. CRC Press, Abingdon (2000), https://cds.cern.ch/record/1086012
8. Cohen, T., Welling, M.: Group equivariant convolutional networks. In: Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016. pp. 2990–2999 (2016), http://jmlr.org/proceedings/papers/v48/cohenc16.html
9. Cohen, T.S., Geiger, M., Koehler, J., Welling, M.: Spherical cnns (2018)
10. Cohen, T.S., Geiger, M., Köhler, J., Welling, M.: Spherical cnns. CoRR (2018), http://arxiv.org/abs/1801.10130
11. Cohen, T.S., Welling, M.: Steerable cnns. CoRR (2016), http://arxiv.org/abs/1612.08498
12. Cotter, F., Kingsbury, N.G.: Visualizing and improving scattering networks. In: 27th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2017, Tokyo, Japan, September 25-28, 2017. pp. 1–6 (2017). https://doi.org/10.1109/MLSP.2017.8168136, https://doi.org/10.1109/MLSP.2017.8168136
13. Crowley, J.L., Parker, A.C.: A representation for shape based on peaks and ridges in the difference of low-pass transform. IEEE Trans. Pattern Anal. Mach. Intell. **6**(2), 156–170 (1984). https://doi.org/10.1109/TPAMI.1984.4767500, https://doi.org/10.1109/TPAMI.1984.4767500
14. Dieleman, S., Fauw, J.D., Kavukcuoglu, K.: Exploiting cyclic symmetry in convolutional neural networks. In: Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016. pp. 1889–1898 (2016), http://jmlr.org/proceedings/papers/v48/dieleman16.html

15. Esteves, C., Allen-Blanchette, C., Zhou, X., Daniilidis, K.: Polar transformer networks. CoRR (2017), http://arxiv.org/abs/1709.01889
16. Fasel, B., Gatica-Perez, D.: Rotation-invariant neoperceptron. In: 18th International Conference on Pattern Recognition (ICPR 2006), 20-24 August 2006, Hong Kong, China. pp. 336–339 (2006). https://doi.org/10.1109/ICPR.2006.1020, https://doi.org/10.1109/ICPR.2006.1020
17. Freeman, W.T., Adelson, E.H.: The design and use of steerable filters. IEEE Trans. Pattern Anal. Mach. Intell. **13**(9), 891–906 (1991). https://doi.org/10.1109/34.93808, http://dx.doi.org/10.1109/34.93808
18. Gonzalez, D.M., Volpi, M., Komodakis, N., Tuia, D.: Rotation equivariant vector field networks. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. pp. 5058–5067 (2017). https://doi.org/10.1109/ICCV.2017.540, https://doi.org/10.1109/ICCV.2017.540
19. Gonzalez, D.M., Volpi, M., Tuia, D.: Learning rotation invariant convolutional filters for texture classification. In: 23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016. pp. 2012–2017 (2016). https://doi.org/10.1109/ICPR.2016.7899932, https://doi.org/10.1109/ICPR.2016.7899932
20. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015. pp. 1026–1034 (2015). https://doi.org/10.1109/ICCV.2015.123, https://doi.org/10.1109/ICCV.2015.123
21. Hegde, V., Zadeh, R.: Fusionnet: 3d object classification using multiple data representations. CoRR (2016), http://arxiv.org/abs/1607.05695
22. Henriques, J.F., Vedaldi, A.: Warped convolutions: Efficient invariance to spatial transformations. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. pp. 1461–1469 (2017), http://proceedings.mlr.press/v70/henriques17a.html
23. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I. pp. 44–51 (2011). https://doi.org/10.1007/978-3-642-21735-7_6,*https*://*doi.org*/10.1007/978−3−642−21735−7_6
24. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with EM routing. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=HJWLfGWRb
25. Jacobsen, J.H., Oyallon, E., Mallat, S., Smeulders, A.W.M.: Hierarchical attribute cnns. In: ICML Workshop on Principled Approaches to Deep Learning (2017), https://ivi.fnwi.uva.nl/isis/publications/2017/JacobsenPADL2017
26. Jacobsen, J., Brabandere, B.D., Smeulders, A.W.M.: Dynamic steerable blocks in deep residual networks. CoRR (2017), http://arxiv.org/abs/1706.00598
27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR (2014), http://arxiv.org/abs/1412.6980
28. Kondor, R.: N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials (2018)
29. Kondor, R., Trivedi, S.: On the generalization of equivariance and convolution in neural networks to the action of compact groups (2018)
30. Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.B.: Deep convolutional inverse graphics network. In: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada. pp. 2539–2547 (2015), http://papers.nips.cc/paper/5851-deep-convolutional-inverse-graphics-network

31. Laptev, D., Savinov, N., Buhmann, J.M., Pollefeys, M.: TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 289–297 (2016). https://doi.org/10.1109/CVPR.2016.38, https://doi.org/10.1109/CVPR.2016.38
32. Lenz, R.: Group Theoretical Methods in Image Processing, Lecture Notes in Computer Science, vol. 413. Springer (1990). https://doi.org/10.1007/3-540-52290-5, http://dx.doi.org/10.1007/3-540-52290-5
33. Li, J., Yang, Z., Liu, H., Cai, D.: Deep rotation equivariant network (2017)
34. Lindeberg, T.: Generalized gaussian scale-space axiomatics comprising linear scale-space, affine scale-space and spatio-temporal scale-space. Journal of Mathematical Imaging and Vision **40**(1), 36–81 (2011). https://doi.org/10.1007/s10851-010-0242-2, https://doi.org/10.1007/s10851-010-0242-2
35. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015. pp. 922–928 (2015). https://doi.org/10.1109/IROS.2015.7353481, https://doi.org/10.1109/IROS.2015.7353481
36. Oyallon, E., Mallat, S.: Deep roto-translation scattering for object classification. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 2865–2873 (2015). https://doi.org/10.1109/CVPR.2015.7298904, https://doi.org/10.1109/CVPR.2015.7298904
37. Perona, P.: Deformable kernels for early vision. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 1991, 3-6 June, 1991, Lahaina, Maui, Hawaii, USA. pp. 222–227 (1991). https://doi.org/10.1109/CVPR.1991.139691, http://dx.doi.org/10.1109/CVPR.1991.139691
38. Quan, T.M., Hildebrand, D.G.C., Jeong, W.: Fusionnet: A deep fully residual convolutional neural network for image segmentation in connectomics. CoRR (2016), http://arxiv.org/abs/1612.05360
39. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III. pp. 234–241 (2015). https://doi.org/10.1007/978-3-319-24574-4_28,*https://doi.org/*10.1007/978−3−319−24574−4_28
40. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA. pp. 3859–3869 (2017), http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules
41. Sedaghat, N., Zolfaghari, M., Brox, T.: Orientation-boosted voxel nets for 3d object recognition. CoRR (2016), http://arxiv.org/abs/1604.03351
42. Sifre, L., Mallat, S.: Rotation, scaling and deformation invariant scattering for texture discrimination. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013. pp. 1233–1240 (2013). https://doi.org/10.1109/CVPR.2013.163, https://doi.org/10.1109/CVPR.2013.163
43. Simard, P.Y., Victorri, B., LeCun, Y., Denker, J.S.: Tangent prop - A formalism for specifying selected invariances in an adaptive network. In: Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]. pp. 895–903 (1991), http://papers.nips.cc/paper/536-tangent-prop-a-formalism-for-specifying-selected-invariances-in-an-adaptive-network
44. Simoncelli, E.P., Freeman, W.T., Adelson, E.H., Heeger, D.J.: Shiftable multiscale transforms. IEEE Trans. Information Theory **38**(2), 587–607 (1992). https://doi.org/10.1109/18.119725, http://dx.doi.org/10.1109/18.119725

45. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR (2014), http://arxiv.org/abs/1409.1556
46. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(1), 1929–1958 (2014), http://dl.acm.org/citation.cfm?id=2670313
47. Teo, P.C.: Theory and Applications of Steerable Functions. Ph.D. thesis, Dept. Computer Science, Stanford University (3 1998)
48. 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 (2018)
49. Weiler, M., Hamprecht, F.A., Storath, M.: Learning steerable filters for rotation equivariant cnns. CoRR (2017), http://arxiv.org/abs/1711.07289
50. Worrall, D.E., Garbin, S.J., Turmukhambetov, D., Brostow, G.J.: Harmonic networks: Deep translation and rotation equivariance. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 7168–7177 (2017). https://doi.org/10.1109/CVPR.2017.758, https://doi.org/10.1109/CVPR.2017.758
51. Worrall, D.E., Garbin, S.J., Turmukhambetov, D., Brostow, G.J.: Interpretable transformations with encoder-decoder networks. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. pp. 5737–5746 (2017). https://doi.org/10.1109/ICCV.2017.611, https://doi.org/10.1109/ICCV.2017.611
52. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 82–90 (2016), http://papers.nips.cc/paper/6096-learning-a-probabilistic-latent-space-of-object-shapes-via-3d-generative-adversarial-modeling
53. 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, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 1912–1920 (2015). https://doi.org/10.1109/CVPR.2015.7298801, https://doi.org/10.1109/CVPR.2015.7298801
54. Xu, X., Todorovic, S.: Beam search for learning a deep convolutional neural network of 3d shapes. In: 23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016. pp. 3506–3511 (2016). https://doi.org/10.1109/ICPR.2016.7900177, https://doi.org/10.1109/ICPR.2016.7900177
55. Zhou, Y., Ye, Q., Qiu, Q., Jiao, J.: Oriented response networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 4961–4970 (2017). https://doi.org/10.1109/CVPR.2017.527, https://doi.org/10.1109/CVPR.2017.527