Department of Physics and Astronomy

University of Heidelberg

Master thesis

in Physics

submitted by

Maurice Weiler

born in Wiesbaden

2017

# Learning Steerable Filters for Rotation Equivariant Convolutional Neural Networks

This Master thesis has been carried out by Maurice Weiler

at the

Heidelberg Collaboratory for Image Processing

under the supervision of

Prof. Dr. Fred A. Hamprecht

## Abstract

In many machine learning tasks it is desirable that a model's prediction transforms in an equivariant way under transformations of its input. Convolutional neural networks (CNNs) implement translational equivariance by construction; for other transformations, however, they are compelled to learn the proper mapping. In this work, we develop *Steerable Filter CNNs* which achieve joint equivariance under translations and rotations. The proposed architecture employs steerable filters to efficiently compute orientation dependent responses for many orientations without suffering interpolation artifacts from filter rotation. We utilize group convolutions which guarantee an equivariant mapping while allowing to preserve the full information on the extracted features and their relative orientations. A commonly used weight initialization scheme is generalized from pixel based filters to filters which are defined as a linear combinations of a system of atomic filters. The proposed approach significantly improves upon the state-of-the-art on the rotated MNIST benchmark as well as on the ISBI 2012 2D EM segmentation challenge.

## Zusammenfassung

In vielen Anwendungen des maschinellen Lernens ist es von Vorteil, wenn sich die Vorhersagen des verwendeten Modells in äquivarianter Weise unter Transformationen der Eingabe transformieren. Convolutional Neural Networks (CNNs) sind per Konstruktion translations-äquivariant, müssen eine adäquate Abbildung für andere Transformationen allerdings explizit erlernen. In dieser Arbeit entwickeln wir *Steerable Filter CNNs*, welche sich durch Äquivarianz unter gleichzeitigen Translationen und Rotationen auszeichnen. Die vorgeschlagene Netzwerkarchitektur verwendet Steerable Filter, um effizient orientierungs-abhängige Filterantworten für eine Vielzahl von Orientierungen zu berechnen, ohne unter Interpolationsartefakten zu leiden. Um eine äquivariante Abbildung zu garantieren, wenden wir Faltungen auf Symmetriegruppen an. Diese können potentiell die volle Information der extrahierten Features und ihrer relativen Orientierung erhalten. Wir verallgemeinern ein standardmäßig genutztes Schema zur Initialisierung der Netzwerkparameter von pixelbasierten Filtern auf Filter, welche als Linearkombination atomarer Filter definiert sind. Der vorgeschlagene Ansatz erzielt auf dem *rotated MNIST* Datensatz sowie auf der *ISBI 2012 EM segmentation challenge* signifikant bessere Ergebnisse als andere dem aktuellen Stand der Technik entsprechende Netzwerke.

## Acknowledgements

I would first like to thank my thesis advisor Prof. Fred A. Hamprecht for always having an open ear for discussions and for fostering a vivid research environment. Special thanks are due to Dr. Martin Storath who co-supervised this work and constantly supported me during all phases of its development. I would also like to acknowledge Nasim Rahaman for facilitating the implementation of the neuron segmentation experiment with his Inferno and Neurofire libraries as well as Thorsten Beier and Constantin Pape for supporting me with the integration of their Multicut pipeline. I further want to emphasize the inspiring and insightful discussions with Taco Cohen and Ullrich Köthe. Many thanks also to Ignacio Arganda-Carreras for the rapid evaluation of our uploaded predictions on the ISBI 2012 EM segmentation challenge.

# Contents

# 1 Introduction

The visual system of human beings shows a remarkable performance in wide-ranging cognitive tasks like object categorization and identification, foreground segmentation, motion analysis, depth estimation or grouping of distinct entities. A primary goal of computer vision research is to emulate, or better to exceed upon, such capabilities in artificial systems. While recent advances in machine learning, especially in the field of artificial neural networks, led to a vast progress in performance of such systems, they are still far from being competitive compared to our visual cognitive abilities. This raises the question why computer vision is such a difficult problem. The input to a computer vision pipeline, corresponding to the sensation on our retina, is a high-dimensional, spatially or spatiotemporally structured signal. Focusing on object recognition first, the task is to reliably detect objects in this signal and to assign them to a category. Given that objects typically show high variabilities in appearance, originating in, e.g. changes in position, orientation, size, pose, illumination or clutter, it is, however, hard to define features which allow to discriminate among different categories. Evidently, the desired mapping from low level sensory stimuli to object categories should be *invariant* under class-preserving transformations of the input. Indeed, neuroscientific studies support the hypothesis that late stages of the visual system of primates represent object identity via a sparse neuron-population coding which meets the aforementioned invariance properties. For instance, activation patterns in test subjects' brains allow to discriminate between face images of different individuals while staying nearly invariant to factors like position, viewing angle or illumination [DiCarlo et al., 2012, Quiroga et al., 2005, 2008, Barlow, 2009]. In more general tasks where not only the object category is of interest, the representations of perceived abstract concepts should rather *covary* with relevant transformations of the input while being invariant to nuisance transformations. Again, there are neuroscientific studies which show that this seems to be the case in primate brains: In the work of Young and Yamane [1992] the axes of a low-dimensional embedding of measured activation patterns in the inferotemporal cortex could be identified with continuous changes in the appearance of the presented stimuli.

These findings suggest that a key factor in solving computer vision is to find an appropriate representation of the data which allows to disentangle the latent sources of variability. While in general suitable representations can be *learned* without making assumptions on internal structure in the data, the task can be greatly facilitated by incorporating such a priori knowledge into learning algorithms [Bengio et al., 2013]. One of the arguably most successful approaches

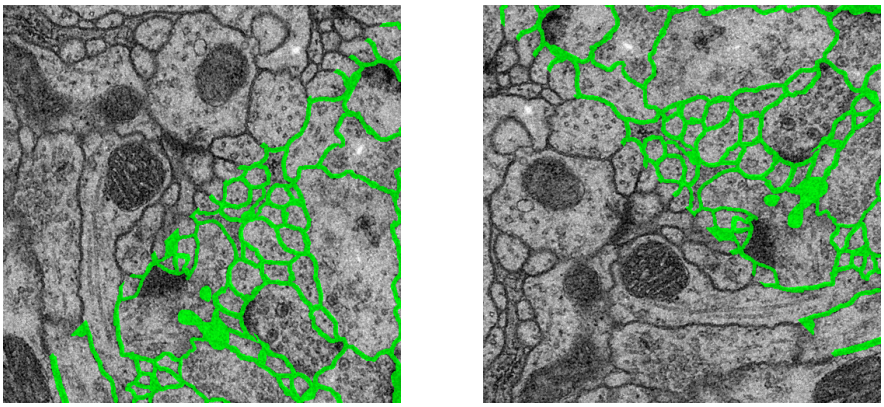


Figure 1.1: *Left*: Electron microscopy sample of neural tissue from the ISBI 2012 EM segmentation challenge, partly overlaid with neuron boundary labels. Characteristic patterns appear in all orientations. *Right*: A rotated version of the same sample. Since the labels are rotated together with the EM image, the pair of them is said to transform in an equivariant way. A computer vision algorithm learning the mapping between the raw data and labels should optimally incorporate such a-priori knowledge by design.

leveraging additional structure are convolutional neural networks (CNNs) [LeCun et al., 1998, Krizhevsky et al., 2012]. These architectures take advantage of the topological arrangement of pixels on a grid by (*i*) restricting neuronal connectivity to a small spatial adjacency called local receptive field and (*ii*) sharing connectivity weights over spatial positions. Immediate consequences of this design principle are a substantially reduced number of model parameters and an enriched spatial structuring of intermediate representations in form of feature maps which transform in an equivariant way under translations of the input. The latter guarantees a generalization of learned patterns over spatial positions. From the viewpoint of object recognition this can be interpreted as reducing the amount of intra-class variability to be learned explicitly. An immediate question arising under these considerations is how to adapt the network architecture such that generalization is extended further to other transformations than just translations.

In this work a focus is laid on the broad class of images which, besides translational invariances, exhibit additional rotational invariances. Typical examples are biomedical microscopy images of tissue, astronomical data or satellite imagery which do not show a prevailing global orientation. For an example see Figure 1.1. Ideally, the output of a network processing such data should be equivariant with respect to the orientation of its input – that is, if the input is rotated, the output should transform accordingly. This is not the case for conventional CNNs which are compelled to learn several rotated versions of the same filter, introducing redundant degrees of freedom and increasing the risk of overfitting.

## 1.1 Contribution

We propose a rotation-equivariant CNN architecture which shares weights over filter orientations to improve generalization and to reduce sample complexity. A key property of our network is that its filters are learned such that they are steerable which enables an exact and efficient rotation of the filters. We accomplish this by representing the filters as linear combinations of a fixed system of atomic steerable filters.

In all intermediate layers of the network, we utilize group convolutions to ensure an equivariant mapping of feature maps. Group-convolutional networks were proposed by Cohen and Welling [2016] who considered four filter orientations. An advantage of our construction based on steerable filters is that we can achieve an arbitrary angular resolution w.r.t. the sampled filter orientations. Indeed, our experiments show that results improve significantly when using more than four orientations.

An important practical aspect of CNNs is a proper weight initialization. Since the weights to be learned serve as expansion coefficients for the steerable system, common weight initialization schemes need to be adapted. Here, we generalize the results found in Glorot and Bengio [2010] and He et al. [2015] to networks which learn filters as a composition of (not necessarily steerable) atomic filters.

The proposed approach significantly improves upon the state-of-the-art on rotation-invariant recognition tasks: (*i*) Our approach is the first one to obtain an accuracy higher than 99% on the rotated MNIST dataset, which is the standard benchmark for rotation-invariant classification. (*ii*) A processing pipeline based on the proposed Steerable Filter CNN ranks first in the ISBI 2012 EM segmentation challenge.

## 1.2 Outline

The remainder of this thesis is organized as follows: First, we will introduce the basic concepts of group theory which are relevant for the formal development of group-equivariant CNNs. The following chapter first discusses how symmetries being present in images can be exploited by computer vision algorithms in general. We then review conventional CNNs from the perspective of group theory as a specific example and point out their equivariance properties under translations. Equipped with these insights, we introduce two possible network architectures which are simultaneously equivariant under translations and rotations and discuss their advantages and disadvantages over each other. Next, we propose a system of atomic steerable filters which is suitable for learning composed filters that can be rotated efficiently and exactly. The rotation-equivariant CNNs are then reformulated in terms of these steerable

filters. As the weights parameterizing the filterbanks are expansion coefficients of the atomic filters rather than pixel values we generalize a common weight initialization scheme to this case in the following chapter. In chapter 6 we give an overview over related approaches and highlight similarities and differences to our approach. Finally, the transformation properties of the proposed architecture are empirically validated and the network performance is experimentally evaluated on two rotation-invariant recognition tasks.

# 2 Elements of group theory

This chapter reviews some basic concepts of group theory which are relevant for a formal investigation of transformation properties of CNNs and in particular for the development of Steerable Filter CNNs. In order to keep the discussion practical, many examples are chosen in foresight of the application to convolutional networks in later chapters. Since the statements made in this chapter are well known results of group theory we will often omit their proofs; the interested reader is referred to Serre [2012], Carter [2009], Milne [2013] and Berndt [2007] on which this chapter is mainly based. All figures in this chapter are taken over and slightly adapted from Carter [2009].

## 2.1 Basic definitions

**Definition 2.1.1.** A ***group*** is a tuple $(G, \cdot)$ consisting of a set $G$ and a binary operation

$$\cdot : G \times G \to G,\ (g,h) \mapsto g \cdot h$$

when the following *group-axioms* are satisfied:

- *Associativity:* for all $g,h,k \in G$, $(g \cdot h) \cdot k = g \cdot (h \cdot k)$
- *Identity element:* $\exists e \in G$ such that $\forall g \in G$, $g \cdot e = g = e \cdot g$
- *Inverse element:* $\forall g \in G$, $\exists g^{-1} \in G$ such that $g \cdot g^{-1} = e = g^{-1} \cdot g$

In the following we will often write $gh$ for $g \cdot h$ and abbreviate $(G, \cdot)$ with $G$ in cases where the meaning is unambiguous. It can be shown that the neutral element and the inverse of each group element are unique.

**Example 2.1.1.** The set $\mathbb{R}^n$ together with addition $+$ as binary operation forms the *translation group* $T_n$ in $n$ dimensions: Addition is associative, the identity element is given by the zero vector 0 and for each $u \in T_n$ there is an inverse element, namely $-u$.

**Definition 2.1.2.** The ***order*** $|G|$ of a group $G$ is defined as the cardinality of its set.

**Example 2.1.2.** The set of all $N$th complex roots of unity $\left\{e^{2\pi i\, k/N} \mid k = 0, \ldots, N-1\right\}$ together with complex multiplication form a cyclic group of *order* N.

Two different groups can be resembling or even coincide in their algebraic structure. To formalize the relationship between groups one introduces the notion of homomorphisms which are functions that map groups in a structure preserving way:

**Definition 2.1.3.** A ***homomorphism*** is a map $\alpha : G \to G'$ from a group $(G, \cdot)$ to another group $(G', \star)$ such that one has for all $g, h \in G$ that

$$\alpha(g \cdot h) = \alpha(g) \star \alpha(h).$$

That is, the product of two elements in the domain $G$ needs to be reflected in the codomain $G'$. It follows directly that $\alpha(e_G) = e_{G'}$ and that for all $g \in G,\ \alpha(g^{-1}) = \alpha(g)^{-1}$. A special case are ***isomorphisms*** which additionally demand the map $\alpha$ to be bijective. Isomorphisms between two groups $G$ and $G'$ exist whenever these are structurally identical. In this case we write $G \cong G'$. When an isomorphism maps a mathematical object to itself, i.e. when its domain and codomain coincide, it is called an ***automorphism***.

**Example 2.1.3.** Again, let $T_n$ be the translation group in $n$ dimensions and let $u, v \in T_2$. The projection $\alpha : T_2 \to T_1,\ (x_1, x_2)^T \mapsto x_1$ is a group homomorphism from $T_2$ to $T_1$ since for all $u, v \in T_2$ it holds that $\alpha(u) + \alpha(v) = u_1 + v_1 = \alpha(u + v)$. Since projections are, however, not bijective, $\alpha$ is no isomorphism between $T_1$ and $T_2$.

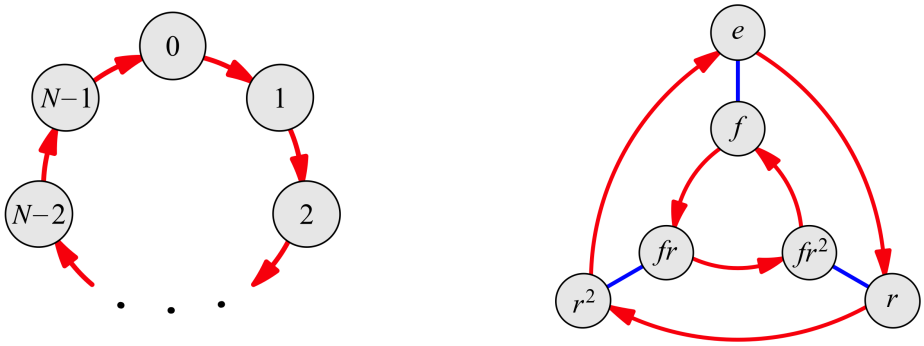
**Example 2.1.4.** Consider the group $\mathrm{U}(1) := \left\{ e^{i\phi} \,\middle|\, \phi \in [0, 2\pi) \right\}$ of unitary matrices in $\mathbb{C}^{1\times 1}$. It can easily be shown that this group is isomorphic to the special orthogonal group in two dimensions

$$\mathrm{SO}(2) := \left\{ \begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix} \,\middle|\, \phi = [0, 2\pi) \right\},$$

that is, $\mathrm{U}(1) \cong \mathrm{SO}(2)$. Indeed, both groups can be interpreted as rotations when acting on $\mathbb{C}$ or $\mathbb{R}^2$, respectively.

A common technique to make the structure of finite order groups visually accessible are so called Cayley diagrams which illustrate groups as graphs with nodes for each group element. Before introducing these diagrams we first need to define the concept of *generating sets* of a group since these sets determine the connectivity of the Cayley graph.

**Definition 2.1.4.** Let $G$ be a group. A subset $S \subseteq G$ is called ***generating set*** of the group if every element $g \in G$ can be expressed by a finite product of elements of $S$ and inverses of those. That is, for every $g \in G$ there exists a $n \in \mathbb{N}$ as well as $a_i \in S$ or $a_i^{-1} \in S$ for $i \in 1, \ldots, n$ such that $g = \prod_{i=1}^{n} a_i$. The neutral element is typically not included in $S$ since it is generated for $n = 0$. We write $G = \langle S \rangle$ when $G$ is generated by $S$.




(a) The Cayley diagram of the cyclic group of order $N$. There is only one generator connecting the elements cyclically.

(b) Cayley diagram of the dihedral group $D_3$. There is one generator $r$ for rotations and another generator $f$ for mirroring.

Figure 2.1: Examples of Cayley diagrams for simple, finitely generated groups.

In general, this decomposition is not unique. A group is called *finitely generated* when $S$ has finite cardinality.

**Example 2.1.5.** The group consisting of all $N$th complex roots of unity introduced in example 2.1.2 is generated by $\{r_N\}$ where we identify $r_N$ with $e^{2\pi i/N}$. Cyclic groups are actually *defined* as groups being generated by one element.

**Definition 2.1.5.** Let $G = \langle S \rangle$ be a group which is generated by a subset $S \not\ni e$. Its ***Cayley graph*** $\Gamma(G,S)$ is a directed and colored graph with

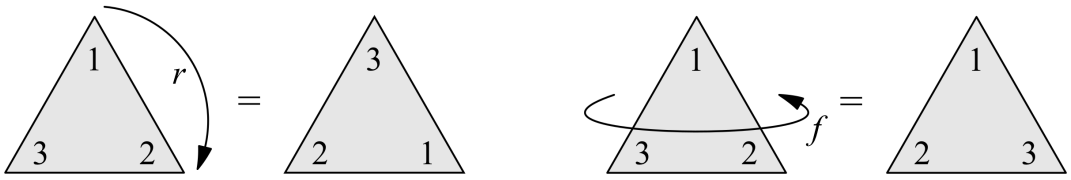
- the nodes are associated with the groups elements $g \in G$
- a color $c_s$ is assigned to each generator $s \in S$
- the edges $\{(g,gs) \,|\, g \in G, s \in S\}$ of the graph connect each group element $g$ with the group elements that result by multiplying it with generator elements $s$ and are of color $c_s$.

**Example 2.1.6.** Figure 2.1a shows the Cayley diagram for a cyclic group of order $N$. The node labels chosen here are the number of times the single generator $r_N$ is multiplied to the identity to reach this node.

In all groups mentioned so far the order in which its elements are composed is irrelevant. This is in general not the case.

**Definition 2.1.6.** A group $G$ is called ***abelian*** if and only if its elements commute, that is iff for all $g,h \in G, \quad gh = hg$

**Example 2.1.7.** All aforementioned groups are abelian. As an example for a non-abelian




(a) Rotational symmetry.

(b) Bilateral symmetry.

Figure 2.2: Action of the rotational and bilateral symmetries generators on a 3-gon.

group let us consider dihedral groups $D_n$ which can be thought of as the $n$ rotational and $n$ bilateral symmetries of $n$-gons (regular $n$-sided polygons). These symmetries are visualized for $n = 3$ in Figure 2.2. The dihedral groups can be generated by the set $\{r, f\}$ consisting of an elementary rotation and a mirroring. Figure 2.1b shows the corresponding Cayley diagram from which one can easily see that the group is not abelian. For example, commuting the two generators leads to different results: $rf = fr^2 \neq fr$

## 2.2 Group actions and equivariance

In practice groups often appear as *transformation groups* acting on some set or space. An application of particular interest in this work is the transformation of signals, e.g. translations or rotations of an image. Let $G$ be a group and $X$ be a set.

**Definition 2.2.1.** A function

$$\phi : G \times X \to X,\ (g, x) \mapsto \phi(g, x) =: g.x$$

is called a ***left group action*** iff it satisfies the following properties:

- *Identity:* $e.x = x \quad \forall x \in X$
- *Compatibility:* $(gh).x = g.(h.x)\ \forall g, h \in G,\ x \in X$

$G$ is then said to ***act*** on $X$ from the left and $X$ is called a *left G-set*. A common abbreviation is to write the group element as index, that is $\phi_g(x) := \phi(g, x)$.

The action of $G$ on $X$ permutes the elements of $X$. Given a specific element from $X$ one can ask where it can possibly be moved by the action of $G$. If all elements in the set can be mapped to any other element, i.e. if for all $x, x' \in X$ there exists a $g \in G$ such that $x' = g.x$, the group action is called *transitive*. This is in general not the case. The set of all points which can be reached when acting on a specific element in $X$ is known as its orbit:

**Definition 2.2.2.** For $x_0 \in X$ we call the subset

$$G.x_0 \;:=\; \left\{ g x_0 \;\middle|\; g \in G \right\}$$

of $X$ an orbit of $G$ through $x_0$.

Being an element of the same orbit is an *equivalence relation*: It is *reflective* since each group has a identity element and hence $x = e.x$, i.e. $x$ is an element of its own orbit. Further, symmetry holds since $y \in G.x$ if and only if $x \in G.y$. This is because $y \in G.x$ implies the existence of a $g \in G$ such that $y = g.x$ which, by the existence of the inverse, implies that $x = g^{-1}.y$. Transitivity (in the sense of equivalence relations) here means that $z = G.x$ follows from $y = G.x$ and $z = G.y$. This property is ensured by the closure of the group under its binary operation: The statements $y = G.x$ and $z = G.y$ imply the existence of $g, h \in G$ such that $y = g.x$ and $z = h.y$. Then $z = h.(g.x) = k.x$ for $k := hg$. The *equivalence classes* under this equivalence relation are exactly the *group orbits*. Being equivalence classes of $X$, the orbits form a *partitioning* of the set $X$ :

**Definition 2.2.3.** The set of all G-orbits

$$X/G \;=\; \left\{ G.x \;\middle|\; x \in X \right\}$$

is called the ***quotient set*** of $X$ by $G$.

**Example 2.2.1.** Consider the special orthogonal group SO(2) in two dimensions and the continuous plane $\mathbb{R}^2$. One possible group action of SO(2) on $\mathbb{R}^2$ is given by multiplying $x \in \mathbb{R}^2$ with the matrices in SO(2), i.e.

$$\rho : \mathrm{SO}(2) \times \mathbb{R}^2 \to \mathbb{R}^2,\ (R,\, x) \mapsto Rx\,,$$

which rotates the points in $x$ around the origin. The orbit of $x_0 \in \mathbb{R}^2$ is given by the set of all points with the same distance to the origin as $x_0$, that is

$$\mathrm{SO}(2).x_0 = \left\{ x \in \mathbb{R}^2 \;\middle|\; |x| = |x_0| \right\}.$$

Consequently, the quotient

$$\mathbb{R}^2/\mathrm{SO}(2) = \left\{ \left\{ x \in \mathbb{R}^2 \;\middle|\; |x| = r \right\} \;\middle|\; r \in \mathbb{R}^+ \right\}$$

is a set consisting of all circles centered at the origin, its elements differing in the radii of the rings.

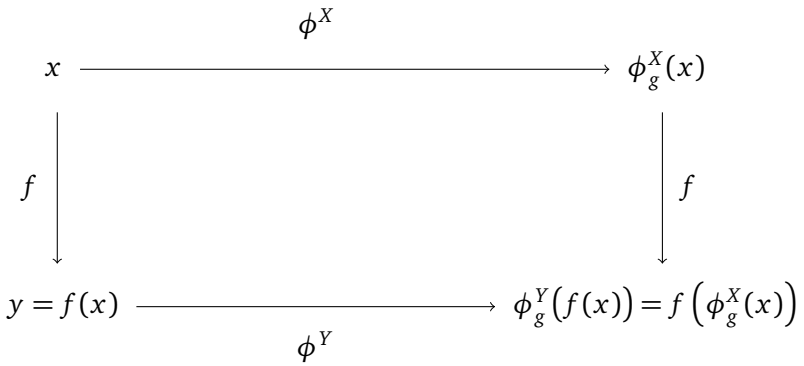
Consider a space which is acted on by a transformation group and a function with this space as

its domain. One can then ask in which way the group action on the domain is reflected in the functions image. When the permutation of elements in the functions image induced by acting on its domain is itself a group action the function is called *equivariant* under this group.

**Definition 2.2.4.** Let $G$ be a group and $X$ and $Y$ be two *G-sets* under group actions $\phi^X : G \times X \to X$ and $\phi^Y : G \times Y \to Y$ respectively. Assume further a map $f : X \to Y$ between these sets to be given. This map is called ***G-equivariant*** iff one has

$$f\left(\phi_g^X(x)\right) = \phi_g^Y\big(f(x)\big) \quad \forall g \in G,\ x \in X.$$

The function can be interpreted as commuting with the group action as visualized in the following commutative diagram:




**Example 2.2.2.** An equivariant mapping of great importance for convolutional networks are ***group-convolutions*** which are a natural generalization of spatial convolutions from translations to more general transformation groups. Let $G$ be such a transformation group and, aiming at later practical applications, let $\zeta : G \to \mathbb{R}$ and $\Psi : G \to \mathbb{R}$ be integrable functions of finite support on this group. Then, their group convolution is defined by

$$(\zeta \circledast \Psi)(g) = \int_G \zeta(h)\,\Psi(h^{-1}g)\,\mathrm{d}\lambda(h), \tag{2.1}$$

where we use the symbol $\circledast$ to distinguish group convolutions from the spatial convolution operator $*$, and $\lambda$ denotes a Haar measure. The result of this operation is again a function on the group. Group convolutions are *equivariant* under the action

$$\phi : G \times \{\zeta : G \to \mathbb{R}\} \to \{\zeta : G \to \mathbb{R}\} : \ \big(h, \zeta(g)\big) \mapsto \big(\phi_h(\zeta)\big)(g) := \zeta\left(h^{-1}g\right)$$

which transforms the functions by acting on their domain. This can be easily shown by

writing out

$$\begin{aligned}\big(\phi_h(\zeta)\circledast\Psi\big)(g) &= \int_G \phi_h\big(\zeta(k)\big)\,\Psi(k^{-1}g)\,\mathrm{d}\lambda(k)\\ &= \int_G \zeta(h^{-1}k)\,\Psi(k^{-1}g)\,\mathrm{d}\lambda(k)\end{aligned}$$

and substituting $\tilde{k} := h^{-1}k$, hence $k^{-1} = \big(h\tilde{k}\big)^{-1} = \tilde{k}^{-1}h^{-1}$ which further gives

$$\begin{aligned}\big(\phi_h(\zeta)\circledast\Psi\big)(g) &= \int_G \zeta\big(\tilde{k}\big)\,\Psi\big(\tilde{k}^{-1}(h^{-1}g)\big)\,\mathrm{d}\lambda\big(\tilde{k}\big)\\ &= \big(\zeta\circledast\Psi\big)(h^{-1}g)\\ &= \big(\phi_h\big(\zeta\circledast\Psi\big)\big)(g).\end{aligned} \tag{2.2}$$

The following diagram gives an overview over the functions transformation under group convolutions and group actions:

$$\begin{array}{ccc}\zeta(g) & \xrightarrow{\quad\phi_h\quad} & \phi_h\big(\zeta(g)\big)\\ \Big\downarrow{\scriptstyle \big(\cdot\circledast\Psi\big)} & & \Big\downarrow{\scriptstyle \big(\cdot\circledast\Psi\big)}\\ (\zeta\circledast\Psi)(g) & \xrightarrow[\quad\phi_h\quad]{} & \phi_h\big(\zeta\circledast\Psi\big) = \big(\phi_h(\zeta)\circledast\Psi\big)\end{array}$$

Note that while in this example the actions before and after the equivariant mapping are the same this is in general not necessary for equivariance.

In some cases the image of a function is not affected by transformations of its domain at all. This special case of equivariance is called *invariance*:

**Definition 2.2.5.** A $G$-equivariant function $f : X \to Y$ from $X$ to $Y$ is called ***G-invariant*** iff the group action in the image is the identity, i.e. when one has

$$f\left(\phi_g^X(x)\right) = f(x) \quad \forall g \in G,\ x \in X.$$

**Example 2.2.3.** Another mapping which appears in convolutional networks is ***global max-***

***pooling*** which on groups is given by

$$\max_G : \{\zeta : G \to \mathbb{R}\} \to \mathbb{R}, \;\; \zeta(g) \mapsto \max_{g \in G} \zeta(g). \tag{2.3}$$

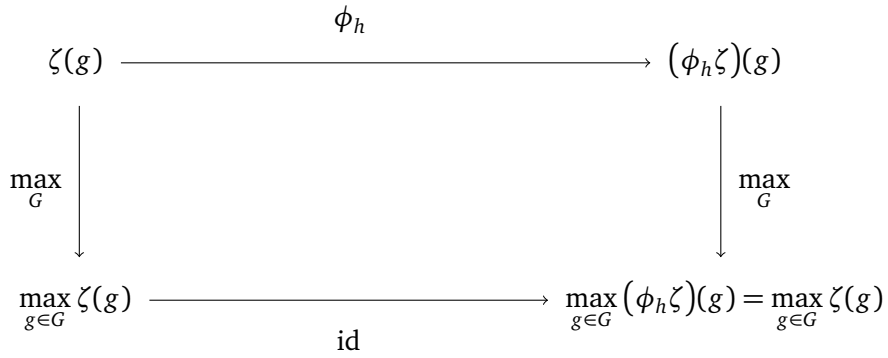
Consider again the group action

$$\phi : G \times \{\zeta : G \to \mathbb{R}\} \to \{\zeta : G \to \mathbb{R}\}, \; \big(h, \zeta(g)\big) \mapsto \big(\phi_h \zeta\big)(g) := \zeta\big(h^{-1} g\big)$$

from the last example. The global pooling operation is *invariant* under this group action since the action simply moves the maximal value to another group element but does not alter its value:

$$\max_{g \in G} \big(\phi_h \zeta\big)(g) \;=\; \max_{g \in G} \zeta\big(h^{-1} g\big) \;=\; \max_{g \in G} \zeta(g) \tag{2.4}$$

The commutative diagram for this example is given below.

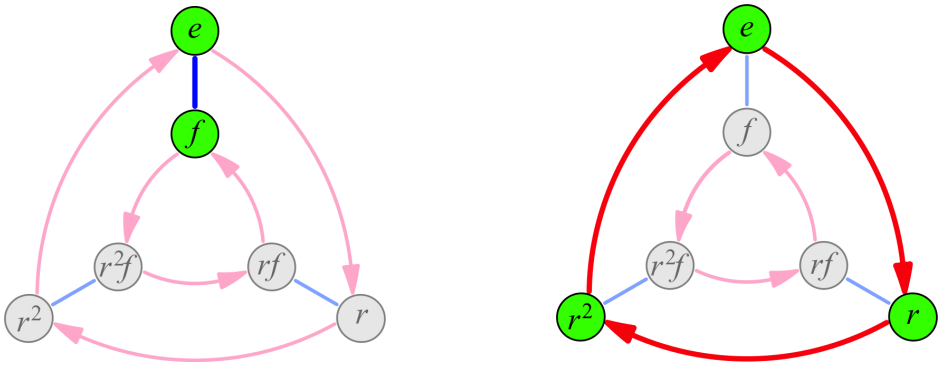



## 2.3 Subgroups and products of groups

The properties of more complex groups can often be understood better by investigating their internal structure, especially whether they are composed of simpler building blocks. An important concept in this context are subgroups which are nonempty subsets of a group that themselves form a group.

**Definition 2.3.1.** Let $(G, \cdot)$ be a group and $H \subseteq G$ be a subset of $G$. If $H$ together with the group operation (restricted to $H$) satisfies

- *Closure:* for all $a, b \in H$ it follows that $a \cdot b \in H$
- *Inverse:* for all $a \in H$ the inverse is contained in the subset, i.e. $a^{-1} \in H$,




(a) Subgroup $\langle f \rangle \cong C_2$ contained in $D_3$.

(b) Subgroup $\langle r \rangle \cong C_3$ contained in $D_3$.

Figure 2.3: Visualizations of two subgroups of $D_3$.

then $H$ is itself a group and is called a ***subgroup*** of $G$. We denote subgroups symbolically by writing $H \leq G$.

Every group contains at least the *trivial subgroup* $\{e\}$ and itself as subgroup. The neutral element $e$ is necessarily included in every subgroup. Further, the intersection of subgroups of $G$ can be shown to form a subgroup.

Each subgroup of $G$ comes with related subsets of $G$ which are known as *cosets* and form a partitioning of the group.

**Definition 2.3.2.** Let $G$ be a group, $H \leq G$ be a subgroup of it and consider an element $g \in G$. Then we define

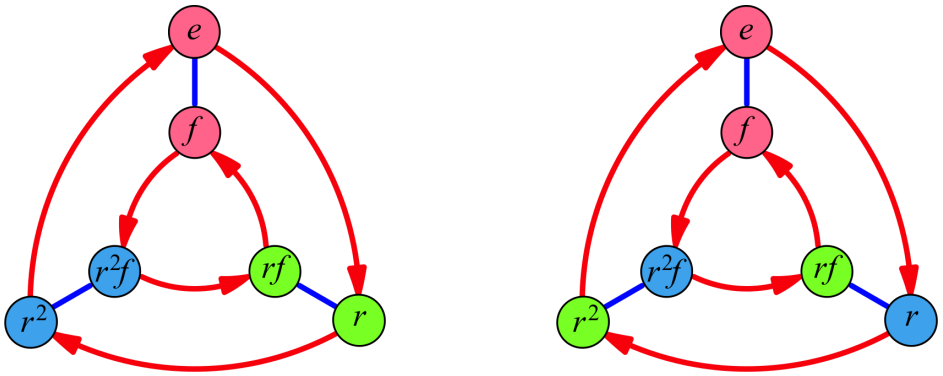
$$gH := \{gh \,|\, h \in H\} \text{ as } \textbf{\textit{left coset}} \text{ of } H \text{ in } G \text{ w.r.t. } g \text{ and}$$

$$Hg := \{hg \,|\, h \in H\} \text{ as } \textbf{\textit{right coset}} \text{ of } H \text{ in } G \text{ w.r.t. } g.$$

A subgroup is always a coset of itself, that is $aH = H$ for $a \in H$ since always $e \in H$. As already stated above, the cosets are either disjoint or equal and hence form a partitioning of the group. Therefore, all cosets but the subgroup from which they are derived do not contain the identity element and thus do *not* form groups themselves. Further, it can be shown that they all contain the same number of elements. Another quantity of interest is the number of cosets of a subgroup.

**Definition 2.3.3.** The ***index*** $(G : H)$ of $H$ in $G$ number of left cosets of $H$ in $G$. It is formally defined as the cardinality of the set $\{gH \,|\, g \in G\}$ of left cosets.

An important property of finite groups which is known as *Lagrange's theorem* is that the index




(a) Left cosets of $\langle f \rangle$ in $D_3$.

(b) Right cosets of $\langle f \rangle$ in $D_3$.

Figure 2.4: Left and right cosets of $\langle f \rangle$ in $D_3$ highlighted by different node colors. Left cosets can be understood in terms of Cayley diagrams by first moving to some group element $g$ and from thereon exploring the diagram by following the generators of the subgroup under consideration. In the case of right cosets the order is interchanged: One first moves within the subgroup and then goes further by taking the same steps as from $e$ to $g$.

of a subgroup $H$ in $G$ is related to the groups orders via

$$(G : H) = \frac{|G|}{|H|}$$

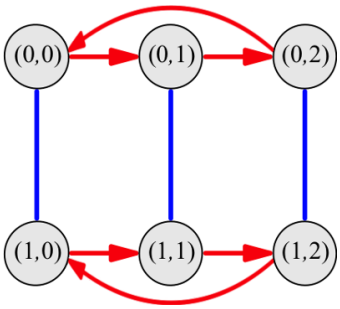
and thus the order of each subgroup necessarily divides the order of the group.

**Example 2.3.1.** Consider again the dihedral group $D_3$ from example 2.1.7. The subsets $\{e, f\}$ and $\{e, r, r^2\}$ of $D_3$ are both closed under the group operation and contain the inverses of each of their elements. They hence form two different *subgroups* $\langle f \rangle$ and $\langle r \rangle$ which are shown embedded in $D_3$ in Figure 2.3. Since $D_3$ and therefore its subsets are finite, we can apply Lagrange's theorem to calculate the subgroups *indices*. For example, there are $(D_3 : \langle f \rangle) = \frac{|D_3|}{|\langle f \rangle|} = \frac{6}{2} = 3$ cosets of $\langle f \rangle$ in $D_3$. The *left cosets* of $\langle f \rangle$ in $D_3$ are $e\langle f \rangle = \{e, f\}$, $r\langle f \rangle = \{r, rf\}$ and $r^2\langle f \rangle = \{r^2, r^2 f\}$ which are highlighted by different node-colors in Figure 2.4a. Similarly, Figure 2.4b shows the *right cosets* of $\langle f \rangle$ in $D_3$ which are given by $\langle f \rangle e = \{e, f\}$, $\langle f \rangle r = \{r, r^2 f\}$ and $\langle f \rangle r^2 = \{r^2, rf\}$.

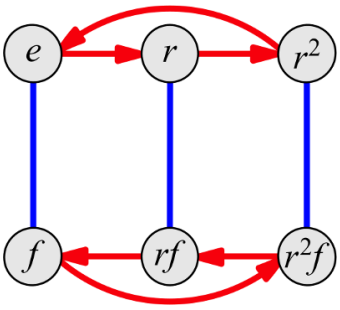
Given a group with several subgroups one can ask whether this group can be understood as being assembled as a combination of its subgroups. One way to construct a large group from smaller ones is by means of the direct product:

**Definition 2.3.4.** Assume two groups $(H, \cdot)$ and $(K, \star)$ to be given. On the Cartesian product of the groups underlying sets $H \times K$ one defines the binary operation

$$\big((h_1, k_1), (h_2, k_2)\big) \mapsto (h_1 \cdot h_2,\, k_1 \star k_2)$$




(a) Direct product $C_3 \times C_2 \cong C_6$ which is isomorphic to the cyclic group of order 6.




(b) Example of a semidirect product $C_3 \rtimes C_2 \cong D_3$ which is isomorphic to the dihedral group with three orientations.

Figure 2.5: Two possible product groups constructed from $C_3$ and $C_2$.

which multiplies elements of $H$ and $K$ independently of each other. This construction forms a group itself and is known as (outer) ***direct product***. It is common to abbreviate the direct product by its set $H \times K$.

It is possible to generalize the direct product to more than two groups by analogously taking the Cartesian product over all sets and and defining the binary operation element wise.

**Example 2.3.2.** Figure 2.5a shows the direct product $C_3 \times C_2$ of finite cyclic groups with a periodicity of 3 and 2 respectively. The independence of the subgroups manifests in the diagram in the fact that the order in which one moves in the subgroups commutes.

A direct product $H \times K$ has subgroups $H' = \{(h, e_K) \,|\, h \in H\}$ and $K' = \{(e_H, k) \,|\, k \in K\}$ which are by construction isomorphic to $H$ and $K$ and thus can be identified with these. The subgroups obey the following algebraic properties:

- the intersection $H' \cap K' = \{(e_H, e_K)\}$ is trivial
- every element in $H \times K$ can be uniquely decomposed in the product of one element in $H'$ and one element in $K'$
- all elements in $H'$ commute with all elements in $K'$

When a given group contains subgroups with the above properties it is called *inner direct product* and is guaranteed to be isomorphic to the *outer direct product* of these subgroups. Both inner and outer products are thus equivalent but emphasize a different viewpoint: The inner product decomposes a group in its subgroups while the outer product constructs a larger group from smaller ones. The last algebraic property stated above is often stated equivalently by saying that $H'$ and $K'$ are both *normal* in $H \times K$.

**Definition 2.3.5.** A subgroup $N$ of $G$ is called ***normal*** if its left and right cosets coincide.

Mathematically this can be formulated as

$$gNg^{-1} = N \quad \forall g \in G.$$

In this case we write $N \lhd G$.

In addition to the requirements of conventional subgroups, normal subgroups $N \lhd G$ hence need to be closed under conjugation with arbitrary group elements, that is $\forall n \in N, g \in G\ \exists n' \in N$ such that $gng^{-1} = n'$. Conjugation with $g \in G$ is an *automorphism* of $N$ which means that $N$ is closed under conjugation and the permutation of its elements preserves the group structure of $N$.

**Example 2.3.3.** Take another look at the dihedral group $D_3$ and its subgroups $\langle f \rangle$ and $\langle r \rangle$, both visualized in Figure 2.3. It is clear that $\langle f \rangle$ is not normal since for example $rfr^{-1} = r^2 f \notin \langle f \rangle$ gives an element in a coset of $\langle f \rangle$ but not in $\langle f \rangle$ itself. The subgroup $\langle r \rangle$, however, is normal, which can easily be seen in the Cayley diagram: Conjugating an arbitrary element of $\langle r \rangle$ with any element of $\langle f \rangle$ always leads back to $\langle r \rangle$. To be more specific, we have $fr^n f^{-1} = r^{-n} \in \langle r \rangle$.

A generalization of the direct product is the so called *semidirect product*. It distinguishes itself from the direct product in that the product group is not necessarily factorizable into two normal subgroups but into two subgroups from which only one needs to be normal. Lets first give the outer, constructive definition:

**Definition 2.3.6.** Assume we are given two groups $(N, \cdot)$ and $(H, \star)$ as well as a homomorphism $\theta : H \to \mathrm{Aut}(N),\ h \mapsto \theta_h$ from $H$ into the group of automorphism of $N$. We can then construct the (outer) ***semidirect product*** group $N \rtimes_\theta H$ which is defined by

- the underlying set is the Cartesian product $N \times H$
- a binary operation given by $\big((n_1, h_1), (n_2, h_2)\big) \mapsto \big(n_1 \cdot \theta_{h_1}(n_2),\ h_1 \star h_2\big)$.

Analogously to the direct product, the semidirect product $N \rtimes_\theta H$ can be decomposed into subgroups $N' = \big\{(n, e_H) \,\big|\, n \in N\big\}$ and $H' = \big\{(e_N, h) \,\big|\, h \in H\big\}$ isomorphic to $N$ and $H$. In general only $N$ is a normal subgroup of $N \rtimes_\theta H$. A further asymmetry in the two factors is that the elements of $H$ are multiplied independently from $N$ while the elements of $N$ are operated on by $H$ via $\theta$ during multiplication. Different homomorphisms $\theta$ lead to different product groups. For example, the trivial case $\theta_h = \mathrm{id}\ \forall h \in H$ recovers the direct product $N \rtimes_{\mathrm{id}} H = N \times H$.

**Example 2.3.4.** In example 2.3.2 we constructed the direct product of $C_3$ and $C_2$. Let us this time combine these groups differently by building a semidirect product. As homomor-

phism from $C_2$ into the automorphism group of $C_3$ we define

$$\theta : C_2 \to \mathrm{Aut}(C_3),\ h \mapsto \theta_h \quad \text{with} \quad \theta_h : C_3 \to C_3,\ n \mapsto \begin{cases} n, & \text{for } h = e_{C_2}, \\ n^{-1}, & \text{for } h = f\ . \end{cases}$$

Then $C_3 \rtimes_\theta C_2 \cong D_3$ is isomorphic to the dihedral group with three orientations and we can identify the factors with the subgroups of the dihedral group found before in example 2.3.1. Specifically, we have $C_3 \cong \langle r \rangle$ and $C_2 \cong \langle f \rangle$. Intuitively, the automorphism $\theta_f$ inverts the direction of cycling through $C_3$. This is visualized in the Cayley diagram in Figure 2.5b whose nodes are for clarity organized in the same pattern as for the diagram of the direct product in Figure 2.5a.

One can again define an *inner* version of the semidirect product. For this purpose, let $G$ be a group with a normal subgroup $N' \lhd G$ and a conventional subgroup $H' \leq G$ such that each $g \in G$ can be uniquely factorized into $g = nh$ with $n \in N'$ and $h \in H'$. Further, let $\theta : H' \to \mathrm{Aut}(N') : h \mapsto \theta_h$ be a homomorphism with $\theta_h : N' \to N',\ n \mapsto hnh^{-1}$ for all $n \in N',\ h \in H'$. By definition of the normality of $N'$ the conjugation with $g \in G$, and therefore with $h \in H'$, is an automorphism of $N'$. Then $N' \rtimes_\theta H'$ is isomorphic to $G$, that is, both are structurally identical and $G$ can be interpreted as being an *inner semidirect product* of $N'$ and $H'$. The definition of the inner semidirect product sheds light on the origin of the structure of the outer semidirect product's binary operation: Let $g_i = n_i h_i \in G$ with $i \in 1, 2$ and $n \in N',\ h \in H'$, then we can write the product of the $g_i$ in factorized form by the following manipulations:

$$g_1 g_2 = n_1 h_1 n_2 h_2 = n_1 h_1 n_2 \left(h_1^{-1} h_1\right) h_2 = \left(n_1 \theta_{h_1}(n_2)\right)(h_1 h_2)$$

This shows that the isomorphic transformation of $n_2$ to $\theta_{h_1}(n_2) = h_1 n_2 h_1^{-1}$ is a result of moving $h_1$ to the right when the elements of $N'$ and $H'$ do not commute.

## 2.4 Group representations

In order to perform calculations with abstract algebraic structures like groups it is often convenient to switch to so called *representations* of these structures. Here we will focus on *linear representations* of groups which are linear maps on some vector space which mimic the group structure under function composition. This approach opens the possibility to investigate the groups properties with the tools of linear algebra which is well understood. Since a linear representation is formally defined as a homomorphism from the group under consideration into the general linear group we will next define the latter.

**Definition 2.4.1.** The ***general linear group*** $\mathrm{GL}_n(\mathbb{K})$ of degree $n$ over a field $\mathbb{K}$ is the group of

all invertible $n \times n$ matrices together with matrix multiplication as group operation.

The general linear group $\mathrm{GL}_n(\mathbb{K})$ consists of all automorphisms in a fixed basis of a $n$-dimensional vector space $V$ over $\mathbb{K}$. It is often written $\mathrm{GL}(V)$ or $\mathrm{Aut}(V)$.

**Definition 2.4.2.** A ***linear representation*** of a group $G$ in a vector space $V$ is a homomorphism

$$\rho : G \to \mathrm{GL}(V)$$

from the group to the general linear group of the vector space.

There are in general infinitely many possible representations of a group. A representation is called *faithful* if it is injective. In this case the representation fully preserves the group structure since its surjective restriction is an isomorphism.

**Example 2.4.1.** Every group $G$ is represented by the *trivial representation*

$$\rho : G \mapsto \mathrm{GL}_1(\mathbb{K}),\ g \mapsto 1.$$

This representation is an homomorphism which looses all information of the group structure and is hence not faithful.

**Example 2.4.2.** Consider the translation group $T_2$ introduced in example 2.1.1 which translates vectors in $\mathbb{R}^2$ by adding an offset to them. It is not possible to find a representation $\rho : T_2 \to \mathbb{R}^2$. This is because the origin is a fixed point under linear maps and hence translations starting from the origin can not be represented. We can, however, write the vector $(x, y)^T$ in homogeneous coordinates $(x, y, 1)^T$ and operate on it by multiplication with translation matrices

$$\Gamma_u := \begin{pmatrix} 1 & 0 & u_1 \\ 0 & 1 & u_2 \\ 0 & 0 & 1 \end{pmatrix}.$$

The map $\rho : T_2 \to \mathrm{GL}_3(\mathbb{R}),\ u \mapsto \Gamma_u$ is a homomorphism since $\Gamma_u \Gamma_v = \Gamma_{u+v}$ and is thus a representation of the translation group. In matrix form the inverse and identity are represented by $\Gamma_u^{-1} = \Gamma_{-u}$ and $\Gamma_0 = \mathbb{1}_{3\times 3}$. The representation is injective and hence faithful.

We will later on utilize a representation of the special Euclidean group SE(2) used in the group-convolutional network to work out its group operations in parameterized form from geometrical considerations.

# 3 A group theoretic view on computer vision

The signals under consideration in this thesis are images which we understand as a mapping from an underlying two-dimensional domain to response vectors representing e.g. brightness or RGB channels. Since the data is spatially organized, a natural class of image transformations is induced by group actions transforming the underlying domain. Examples include spatial translations or rotations of an image. In the following section we will first show on an abstract level how and in which cases image classification can be facilitated by designing a classifier which is invariant under such transformations. Subsequently, we analyze the equivariance properties of conventional CNNs from a group theoretic perspective and investigate how this leads to invariant classification.

## 3.1 Invariant image classification

We formally define an image as a function $f : \mathcal{X} \to \mathbb{R}^K$ which maps a two-dimensional domain $\mathcal{X}$ to $K$ response channels. In practice images are usually sampled on a pixel grid $\mathcal{X} = \{1, \dots, m\} \times \{1, \dots, n\}$. For convenience we will, however, consider them to be defined as functions with finite support on either $\mathbb{Z}^2$ or $\mathbb{R}^2$. Let

$$\mathcal{F} = \left\{ f \,\middle|\, f : \mathcal{X} \to \mathbb{R}^K \right\}$$

be the vector-space of all images which we denote as *image-space*. In classification settings one is interested in a partitioning of this space into regions associated with different classes. Next, let $\mathcal{L} = \{0, 1, \dots, \mathrm{C}\}$ be an exhaustive set of $\mathrm{C} \in \mathbb{N}$ class-labels with an additional non-class, labeled as 0, for all images not falling in one of the other classes. This allows to formally define a ground-truth function of class-ids

$$\mathcal{C} : \mathcal{F} \to \mathcal{L}$$

assigning a class to each image. Given a dataset $\{(f_i, l_i)\}_{i=1}^N$ of $N$ labeled training samples where $f_i \in \mathcal{F}$ and $l_i = \mathcal{C}(f_i) \in \mathcal{L} \;\; \forall \; i \leq N$, a classifier is expected to learn decision boundaries which reflect the partitioning of the class-id function. Due to the often large intra-class variability the different regions in image space are usually highly entangled such that the classifier

has to learn convoluted boundaries.

We next show how the complexity of learning decision boundaries can be reduced when certain symmetries exist in the data. For that purpose consider a transformation group $G$ acting on the domain $\mathcal{X}$ via a left group action

$$\phi^{\mathcal{X}} : G \times \mathcal{X} \to \mathcal{X}, \;\; (g, x) \mapsto \phi_g^{\mathcal{X}}(x) = g.x\,.$$

This group action induces another group action on images:

$$\phi^{\mathcal{F}} : G \times \mathcal{F} \to \mathcal{F}, \;\; (g, f) \mapsto \phi_g^{\mathcal{F}}(f) := f\left(g^{-1}.x\right)$$

Assume that these transformations are label-preserving, i.e. all images in the orbit $G.f$ are assigned the same label:

$$\mathcal{C}\big(\phi_g^{\mathcal{F}}(f)\big) \;=\; \mathcal{C}(f) \quad \forall\, g \in G,\; f \in \mathcal{F}.$$

In such cases it is reasonable to *enforce* the invariance on the learned classifier by design. Doing so constricts the hypothesis space

$$\mathcal{H}_{\text{free}} = \big\{h \,\big|\, h : \mathcal{F} \to \mathcal{L}\big\} \tag{3.1}$$

of an unrestricted classifier to the hypothesis space

$$\mathcal{H}_{\text{inv}} = \big\{h \,\big|\, h : \mathcal{F}/G \to \mathcal{L}\big\} \tag{3.2}$$

of all hypotheses assigning a class label to each orbit. As a consequence the sample complexity is reduced since the quotient space $\mathcal{F}/G$ is smaller than the full image-space $\mathcal{F}$. Further, samples $f \in \mathcal{F}$ seen by the classifier are generalized over their whole orbit $G.f \in \mathcal{F}/G$ such that the classifier can detect class instances in related poses $f' \in G.f$ even if never seen before. The next section will substantiate these abstract ideas for the translation group on the example of convolutional neural networks.

## 3.2 Equivariance properties of CNNs

Image classification is nowadays often done by convolutional neural networks. In principle it is, however, possible to classify images with conventional multilayer perceptrons (MLPs) which are simple fully connected feedforward networks. An image is fed to a MLP as a vector of pixel values which is in each layer multiplied by a weight matrix, added by a bias and then element-wise transformed by a nonlinearity. Such a network's output is invariant under a joint

permutation of the input vector entries and columns of the first layer's weight matrix. In image classification this corresponds an arbitrary permutation of the image pixels. This shows that the model completely ignores the pixels' spatial arrangement. This problem can be alleviated by restricting the weight matrices entries to be nonzero only in a small adjacency of every pixel in the input image. Then the network senses the image in form of small receptive field for each neuron which capture local patterns that are assembled to the final prediction in a hierarchy of layers. Besides imprinting the spatial arrangement of pixels into the network this approach leads to a drastic reduction in network parameters: assuming that the number of output neurons of each layer is proportional to the number of input pixels $N$, a conventional MLP has of the order $\mathcal{O}\left(N^2\right)$ weights while the parameter number of a MLP with local receptive fields scales with $\mathcal{O}(N)$.

A second issue of MLPs is that they do not capture any symmetry present in the spatial arrangement of the data. For example, image classification should often be insensitive to spatial shifts, that is, the network prediction should not vary with the position of the object to be classified. In more general settings like image segmentation the resulting segmentation mask should move in an equivariant way under translations of the input image. Convolutional neural networks are designed to solve this problem for the specific case of translational symmetries. CNNs for image analysis tasks map an input image to a desired output by repeatedly convolving it with filters, applying nonlinearities and pooling the feature maps. The convolution operation can be seen as a further constraint on the networks parameters: since the kernels are shifted over the image, the local receptive fields share their weights. As a consequence, the number of parameters is of order $\mathcal{O}(1)$, i.e. it does not scale with the size of the image at all. Furthermore, patterns learned at one position will be detected again at every other position. Together, these effects lead to a drastically increased sample complexity and generalization in comparison to MLPs.

Mathematically, the convolution operation results in an equivariant mapping from one feature map to the next. Specifically, let

$$\zeta_h^{(l)} : \mathbb{Z}^2 \to \mathbb{R}$$

be a feature map channel in layer $l$ where $h = 1, \ldots, H$ counts the channels. To produce activation channel $i$, where $i = 1, \ldots, I$, each input channel is first convolved with a corresponding filter

$$\Psi_{ih}^{(l)} : \mathbb{Z}^2 \to \mathbb{R}$$

over the spatial dimensions. Subsequently, the input channel responses are summed together

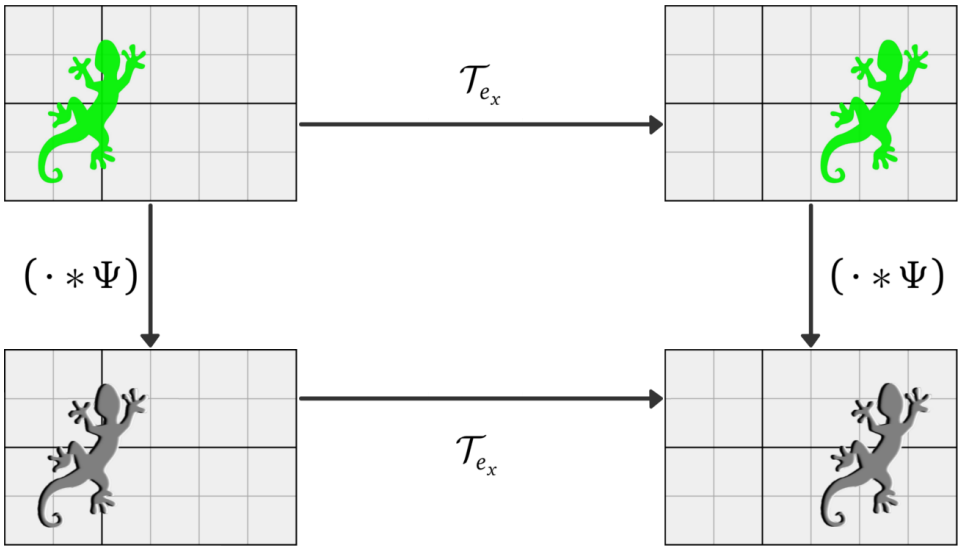



Figure 3.1: Translation-equivariance of the convolution operation. The filter response of a translated pattern is equivalent to the translated filter response of the original pattern.

to give the pre-nonlinearity activations

$$y_i^{(l)} = \left( \sum_{h=1}^{H} \zeta_h^{(l-1)} * \Psi_{ih}^{(l)} \right)(x).$$

The spatial convolution is a special case of the group convolution (2.1) for the translation group $T_2$ in two dimensions which was introduced in example 2.1.1. Therefore the derivation in equation (2.2) shows that the convolution operation is equivariant with respect to the group action

$$\mathcal{T} : \; T_2 \times \left\{ \zeta_h : \mathbb{Z}^2 \to \mathbb{R} \right\} \to \left\{ \zeta_h : \mathbb{Z}^2 \to \mathbb{R} \right\}, \;\; \left( u, \zeta_h(x) \right) \mapsto \zeta_h(x-u) \tag{3.3}$$

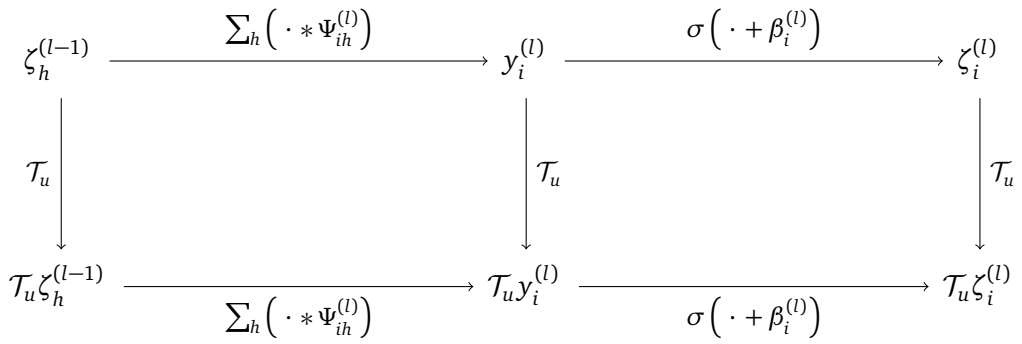
which acts on a feature map by translating it by an offset $u$. This means that translations commute with the convolution, i.e.

$$\left( \sum_{h=1}^{H} \mathcal{T}_u(\zeta_h) * \Psi_{ih} \right)(x) = \left( \mathcal{T}_u \left( \sum_{h=1}^{H} \zeta_h * \Psi_{ih} \right) \right)(x),$$

which is visualized in Figure 3.1 The pre-nonlinearity activations are further transformed by adding a bias $\beta_i$ per channel and applying a nonlinearity $\sigma : \mathbb{R} \to \mathbb{R}$ element-wise:

$$\zeta_i^{(l)}(x) = \sigma \left( y_i^{(l)}(x) + \beta_i^{(l)} \right)$$

Since these operations are independent of spatial positions they are trivially equivariant with respect to translations. The equivariance properties of one convolutional layer are visualized in the following commutative diagram:




Blocks of convolutional layers as described above are typically applied alternatingly with pooling layers which are used to make the network invariant to small local deformations and to reduce the spatial extent of the feature map. The latter has the effect that the next layers' filters have a larger field of view and are therefore able to extract more abstract features on a larger scale. Motivated by the discussion in Cohen and Welling [2016] we analyze the transformation properties of pooling layers by splitting them into a non-strided pooling step and subsequent subsampling. Without loss of generality we focus on maximum pooling operations; the derivations for average pooling operations are analogous. We formalize the non-strided pooling operation as the operator $\mathcal{P}$ acting on the feature map via

$$\big(\mathcal{P}_P \zeta\big)(x) \;:=\; \max_{\tilde{x} \in x.P} \zeta(\tilde{x}).$$

where $P \subseteq \mathbb{Z}^2$ is the kernel of the pooling operation. This kernel is typically a small domain $\left\{-\lfloor \frac{k-1}{2} \rfloor, \ldots, \lfloor \frac{k}{2} \rfloor\right\}^2$ covering $k \times k$ pixels around the origin. Further, $x.P = \{x.p = x + p \,|\, p \in P\}$ is the group transformation of the pooling region which, here for the translation group, shifts the whole kernel by an offset of $x$ over the feature map. This way, the non-strided pooling operation selects the maximum response inside the kernel region around each pixel. A translation of the input feature map leads to the same translation of the pooling output, i.e.

$$\begin{aligned}
(\mathcal{P}_P \mathcal{T}_u \zeta)(x) &= \max_{\tilde{x} \in x.P} (\mathcal{T}_u \zeta)(\tilde{x}) \\
&= \max_{\tilde{x} \in x.P} \zeta(\tilde{x} - u) \\
&= \max_{\tilde{x} \in (x-u).P} \zeta(\tilde{x}) \\
&= (\mathcal{T}_u \mathcal{P}_P \zeta)(x)\,,
\end{aligned}$$

hence the non-strided pooling is translation-equivariant.

In typical applications the pooling operation is performed with a stride larger than one. Given the result of the non-strides pooling operation, its strided counterparts can be calculated from the former by downsampling. Specifically, let $s \in \mathbb{N}$ be the stride, that is, the integer factor by which the signal is sampled down. Then we define a downsampling operator $\mathcal{D}_s$ acting on a signal $f : \mathbb{Z}^2 \to \mathbb{R}$ via

$$\mathcal{D}_s f(x) \;:=\; f(sx)\,.$$

In order to see in which way the downsampling and translation operations interact, lets switch to a representation of the corresponding groups. When acting on homogeneous coordinate vectors $x = (x, y, 1)^T$ we can identify

$$\mathcal{D}_s\, x = \begin{pmatrix} \frac{1}{s} & 0 & 0 \\ 0 & \frac{1}{s} & 0 \\ 0 & 0 & 1 \end{pmatrix} x \qquad \text{and} \qquad \mathcal{T}_u\, x = \begin{pmatrix} 1 & 0 & u_1 \\ 0 & 1 & u_2 \\ 0 & 0 & 1 \end{pmatrix} x\,,$$

where the inverse in $\frac{1}{s}$ comes from the definition of $\mathcal{D}_s$ as shrinking operation rather than dilation. The two operations do not commute since

$$\mathcal{D}_s\mathcal{T}_u\, x = \begin{pmatrix} \frac{1}{s} & 0 & 0 \\ 0 & \frac{1}{s} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & u_1 \\ 0 & 1 & u_2 \\ 0 & 0 & 1 \end{pmatrix} x \;=\; \begin{pmatrix} \frac{1}{s} & 0 & \frac{u_1}{s} \\ 0 & \frac{1}{s} & \frac{u_2}{s} \\ 0 & 0 & 1 \end{pmatrix} x \;=\; \begin{pmatrix} \frac{x_1+u_1}{s} \\ \frac{x_2+u_2}{s} \\ 1 \end{pmatrix}$$

while

$$\mathcal{T}_u\mathcal{D}_s\, x = \begin{pmatrix} 1 & 0 & u_1 \\ 0 & 1 & u_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{s} & 0 & 0 \\ 0 & \frac{1}{s} & 0 \\ 0 & 0 & 1 \end{pmatrix} x \;=\; \begin{pmatrix} \frac{1}{s} & 0 & u_1 \\ 0 & \frac{1}{s} & u_2 \\ 0 & 0 & 1 \end{pmatrix} x \;=\; \begin{pmatrix} \frac{x_1}{s} + u_1 \\ \frac{x_2}{s} + u_2 \\ 1 \end{pmatrix}.$$

Instead they are rather related by

$$\mathcal{D}_s\mathcal{T}_{su} \;=\; \mathcal{T}_u\mathcal{D}_s\,. \tag{3.4}$$

In continuous space the dilation operation is equivariant under arbitrary translations since for every translation by $u \in \mathbb{R}^2$ before downsampling we find a translation by $\frac{u}{s}$ after downsampling. For digitized images, however, translations by $u \in \mathbb{Z}^2$ after downsampling correspond to translations by $su \in s.\mathbb{Z}^2 = \left\{su \,\middle|\, u \in \mathbb{Z}^2\right\} \leq \mathbb{Z}^2$ before downsampling. All other translations before downsampling lead to an undefined behavior. The downsampling operation is hence not equivariant under arbitrary translations but only under the subgroup $sT_2 \leq T_2$ with different group actions in the original and the strided domain[1]. Note that the loss of full translation-

[1] Of course both groups are isomorphic, $sT_2 \cong T_2$, hence one could equivalently redefine the group action (3.3) to

equivariance in inevitable when demanding the network to be invariant to small local deformations. Again, we give a commutative diagram to give an overview over the equivariance properties of pooling layers:

$$\begin{array}{ccccc}
\zeta^{(l)}(x) & \xrightarrow{\mathcal{P}_P} & \big(\mathcal{P}_P\zeta^{(l)}\big)(x) & \xrightarrow{\mathcal{D}_s} & \zeta^{(l+1)}(x) = \big(\mathcal{P}_P\zeta^{(l)}\big)(sx) \\
\Big\downarrow{\scriptstyle \mathcal{T}_{su}} & & \Big\downarrow{\scriptstyle \mathcal{T}_{su}} & & \Big\downarrow{\scriptstyle \mathcal{T}_{u}} \\
\zeta^{(l)}(x-su) & \xrightarrow[\mathcal{P}_P]{} & \big(\mathcal{P}_P\zeta^{(l)}\big)(x-su) & \xrightarrow[\mathcal{D}_s]{} & \zeta^{(l+1)}(x-u) = \big(\mathcal{P}_P\zeta^{(l)}\big)(s(x-u))
\end{array}$$

After the last convolutional layer there are different options how to deal with the remaining spatial extent of the feature maps. One possibility is to flatten the tensor over the spatial coordinates, interpreting the responses at different locations as independent features. This allows to explicitly keep all information and to make the network sensitive to the coarse spatial layout of the features in the sense discussed in the first paragraph of this section. It does, however, break the network's translation-equivariance. Another option which is often used is global pooling over the remaining spatial responses which we have already shown to be translation-invariant in equation (2.4) with $G = T_2$.

The (approximate) translation-equivariance of each layer together with the observation that the group action in one layer's codomain coincides with the group action of the next layer's domain makes the convolutional part of a CNN (approximately) equivariant under translations as a whole. When global pooling is used after the last convolutional layer the whole network becomes invariant to translations of the input signal. This way convolutional networks restrict the full hypothesis space (3.1) to the hypothesis space (3.2) of an invariant classifier, here given by

$$\mathcal{H}_{\text{CNN}} = \left\{ h \;\middle|\; h : \mathcal{F}/T_2 \to \mathcal{L} \right\}.$$

The orbits $T_2.f \in \mathcal{F}/T_2$ of signals $f$ in $\mathcal{F}$ comprise all images which are translated versions of each other. When the CNN has learned to classify one representative element of an orbit correctly, it is guaranteed to generalize this knowledge over all other images in its orbit.

stride $s$ translations to get an equivariant mapping under $T_2$. The claims made are to be understood with fixed group action (3.3) in the non-pooled domain and an adapted action of $sT_2$ with unit strides in the downsampled domain.

# 4 Rotation equivariant CNNs

Figure 4.1: *Left*: Group of interacting galaxies Arp 273 in the constellation Andromeda, recorded by the Hubble space telescope. *Right*: A rotated version of the same image which would also be a plausible recording since galaxies appear in all orientations.
Image credit: NASA, ESA and the Hubble Heritage Team (STScI/AURA). Accessed on Sep. 6 2017 from http://www.spacetelescope.org/images/heic1107a/

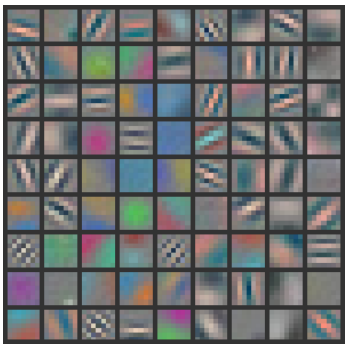


Figure 4.2: Filterbank of the first layer of a conventional CNN trained on natural images [Zeiler and Fergus, 2014]. Many filters are learned such that they occur in several orientations.

Motivated by the observations made in the last chapter we aim to generalize the set of transformations under which the network is equivariant to further enhance its sample complexity and generalization capabilities. A class of transformations which are of great interest in this regard are similarity transformations which, in addition to translations, include rotations, reflections and dilations. Here we focus on exploiting rotational symmetries which are present in a broad class of images. Typical examples are biomedical microscopy images of tissue, astronomical data or satellite imagery, e.g. the recording in Figure 4.1. Since these images do not show a prevailing global orientation, the output of a network processing such data should be equivariant with respect to the orientation of its input – that is, if the input is rotated, the output should transform accordingly. Even when there is a predominant direction in an image as a whole, the low level features in the first layers such as edges usually appear in all orientations; see for example the learned filterbanks visualized in Figure 4.2. In both cases of globally or locally rotation-invariant images, conventional CNNs are compelled to learn several rotated versions of the same filter, introducing redundant degrees of freedom and increasing the risk of overfitting.

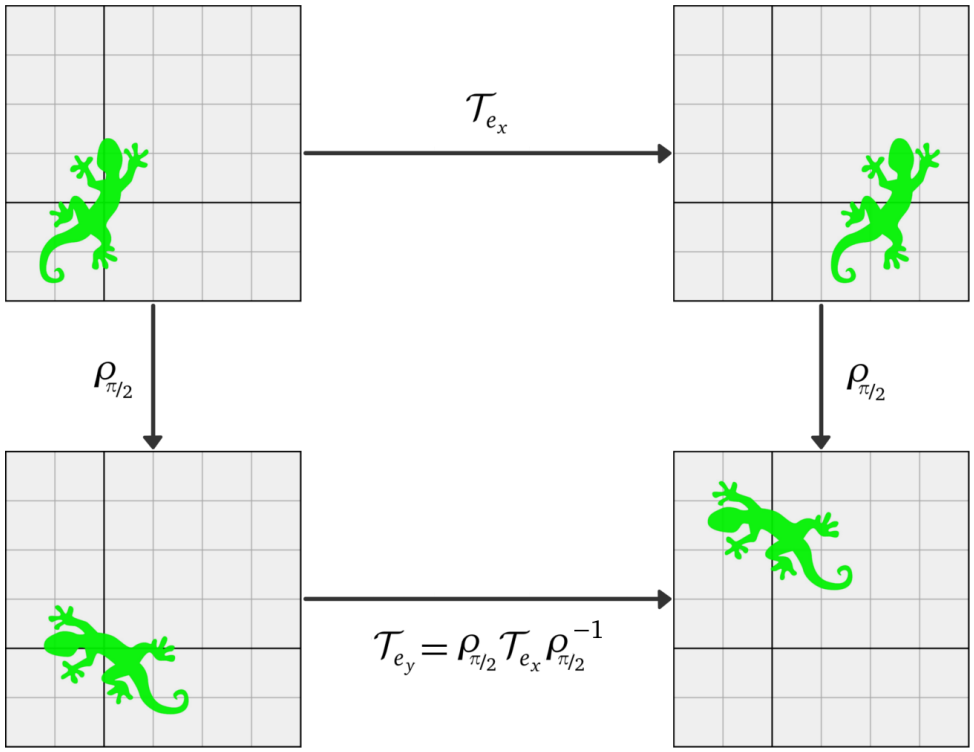



Figure 4.3: Translations $\mathcal{T}$ and rotations $\rho$ on the plane do not commute. Instead, a translation, followed by a rotation is equivalent to first applying the same rotation and then a different translation. This is because the corresponding group structure, the special Euclidean motion group, is a semidirect product of translations and rotations with the former being the normal subgroup acted on by conjugation with elements of the latter.

For a formal mathematical treatment it is necessary to investigate the group structure formed by combined rotational and translational transformations which are known as rigid motions. Figure 4.3 shows the effect of successively acting on a space by translating and rotating it. Here $\mathcal{T}$ denotes the action of translations as before and $\rho$ is the group action of the rotation group on the plane. From this visual construction one can see that translations and rotations do not commute but that instead the conjugation of a translation by some rotation gives a different translation. This suggests that the group of rigid motions can be understood as a semidirect product of the translation group $T_2$ and the rotation group SO(2), the latter consisting of all orthogonal matrices of unit determinant in $\mathbb{R}^{2\times 2}$. Indeed, the special Euclidean group[1] SE(2) which describes rigid motions is isomorphic to the semidirect product $T_2 \rtimes \mathrm{SO}(2)$ with the translations being a normal subgroup acted on by rotations by conjugation. We denote the

[1]The premodifier *special* is inherited from the special orthogonal group SO(2). The full Euclidean group is constructed from $T_2$ and the full orthogonal group O(2) which, in addition to translations and rotations, contains flips.

group elements of the special Euclidean group by $(x, \phi)$ where $x \in \mathbb{R}^2$ and $\phi \in [0, 2\pi)$. In order to work out the binary operation of SE(2) as well as the inverse of group elements in this parametrization we switch to a representation in homogeneous coordinates where we associate

$$(x, \phi) \mapsto \begin{pmatrix} \cos(\phi) & -\sin(\phi) & x_1 \\ \sin(\phi) & \cos(\phi) & x_2 \\ 0 & 0 & 1 \end{pmatrix}.$$

From the multiplication of such matrices

$$\begin{aligned} &\begin{pmatrix} \cos(\phi) & -\sin(\phi) & x_1 \\ \sin(\phi) & \cos(\phi) & x_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) & u_1 \\ \sin(\theta) & \cos(\theta) & u_2 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos(\phi+\theta) & -\sin(\phi+\theta) & x_1 + u_1\cos(\phi) - u_2\sin(\phi) \\ \sin(\phi+\theta) & \cos(\phi+\theta) & x_2 + u_1\sin(\phi) + u_2\cos(\phi) \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

we can read off the binary operation to be given by

$$(x, \phi)(u, \theta) = (x + \rho_\phi u,\ \phi + \theta), \tag{4.1}$$

where the group action $\rho$ of rotations on the plane is defined as

$$\rho : \mathrm{SO}(2) \times \mathbb{R}^2 \to \mathbb{R}^2,\ (R_\phi, x) \mapsto R_\phi x\,. \tag{4.2}$$

Furthermore, we identify the inverse of a group element to be given by

$$(x, \phi)^{-1} = (-\rho_{-\phi}, -\phi)\,. \tag{4.3}$$

The key concept leading to the translation-equivariance of conventional CNNs is translational weight sharing, implemented by convolution operations. A convolution effectively translates the filter over the image or feature map and evaluates the Frobenius inner product at each position, that is

$$\begin{aligned} y(x) &:= \big(f * \Psi\big)(x) \\ &= \int_{\mathbb{R}^2} f(u)\Psi(x-u)\,\mathrm{d}u \\ &= \int_{\mathbb{R}^2} f(u)\big(\mathcal{T}_u\Psi\big)(x)\,\mathrm{d}u\,. \end{aligned} \tag{4.4}$$

This suggests that the transformation group under which the network's layers are equivariant can be extended by generalizing the translation of filters over feature maps to the action of a larger transformation group. Focusing on rigid motions in SE(2) we have the group action

$$\mathcal{E} : \mathrm{SE}(2) \times \mathbb{R}^2 \to \mathbb{R}^2, \ \big((u,\theta),x\big) \mapsto \mathcal{E}_{(u,\theta)}\, x := \mathcal{T}_u \rho_\theta \ x$$

which implies to extract pre-nonlinearity features in analogy to (4.4) via

$$\begin{aligned} y(x,\theta) &:= \int_G f(u) \left(\mathcal{E}_{(u,\theta)}\Psi\right)(x) \, \mathrm{d}\lambda((u,\theta)) \\ &= \int_G f(u) \, (\rho_\theta \Psi)(x-u) \, \mathrm{d}\lambda((u,\theta)) \\ &= \big(f * \rho_\theta \Psi\big)(x) \, . \end{aligned} \tag{4.5}$$

This convolution with several rotated versions of the same filter implements a rotational weight sharing which leads to an improved sample complexity and to a generalization over orientations in the same way as the weight sharing over spatial positions in convolutional networks does for translations. Compared to a conventional CNN which independently learns filters for features in $\Lambda$ orientations in a rotation-invariant recognition task, a corresponding rotation-equivariant CNN which shares weights over filter orientations consumes $\Lambda$ times less parameters to extract the same representation.

The activations (4.5) resulting from the convolution with rotated filters do not only depend on the spatial position of the filter but also on its orientation and are thus functions on the transformation group applied, that is

$$y : \mathrm{SE}(2) \to \mathbb{R}, \ (x,\theta) \mapsto y(x,\theta). \tag{4.6}$$

In principle it would be possible to interpret the additional orientation component as conventional filter channels such that one ends up with $\tilde{I} = \Lambda I$ channels. This would, however, discard the internal structure of the activations such that a subsequent equivariant mapping under the full transformation group SE(2) becomes impossible. In the following sections we will investigate two different network designs which take the full group structure of the activations (4.6) into account to ensure an equivariant mapping.

## 4.1 Orientation pooling architecture

In convolutional networks the feature maps consist of several channels $h \in \{1,\dots,H\}$ which are mapped to the next layer's channels $i \in \{1,\dots,I\}$. The convolution operation (4.5) is adapted to the case of an image containing channels $f_h : \mathbb{R}^2 \to \mathbb{R}$ by

$$y_i^{(1)}(x,\theta) := \sum_{h=1}^{H} \big(f_h * \rho_\theta \Psi_{ih}^{(1)}\big)(x)\,. \tag{4.7}$$

Here $\Psi_{ih} : \mathbb{R}^2 \to \mathbb{R}$ are the filter channels which are now also defined on continuous domain. The group action $\rho : \mathrm{SO}(2) \times \mathcal{F} \to \mathcal{F}$ on images, filters or feature maps is induced by the action (4.2) on the underlying domain via

$$\rho_\theta f(x) := f(\rho_{-\theta} x).$$

As usual, after the convolution step a bias $\beta$ is added and a nonlinearity $\sigma$ is applied, so that we end up with feature maps given by

$$\zeta_i^{(1)}(x,\theta) = \sigma\Big(y_i^{(1)}(x,\theta) + \beta_i^{(1)}\Big)\,. \tag{4.8}$$

One possibility to process the feature maps on the group further is to perform a pooling operation over orientations, that is

$$\hat{\zeta}_i^{(1)}(x) = \max_\theta \zeta_i^{(1)}(x,\theta).$$

This eliminates the dependence on $\theta$ such that one can process the pooled feature maps in subsequent layers $l > 1$ in the same way as the input image:

$$\hat{\zeta}_i^{(l)}(x) = \max_\theta \sigma\left(\sum_{h=1}^{H} \Big(\hat{\zeta}_h^{(l-1)} * \rho_\theta \Psi_{ih}^{(l)}\Big)(x) + \beta_i^{(l)}\right)$$

The feature maps resulting from such an operation are invariant under local rotations of patches in the input: by rotating a small patch at an arbitrary position in the input, the features at the corresponding location would retain their values since only the maximum responses over orientations are kept. A global rotation of the input image leads to a spatial displacement of its patches on the one hand and to a rotation of their orientation on the other hand. By the invariance w.r.t. the latter, the overall output will hence simply be rotated spatially and therefore transforms in an equivariant way with the input. This intuition can be shown rigorously

by transforming the input $\hat{\zeta}^{(l-1)}$ to $\rho_\phi \hat{\zeta}^{(l-1)}$ which in the convolution step leads to

$$\begin{aligned}
&\sum_{h=1}^{H} \left(\rho_\phi \hat{\zeta}_h^{(l-1)} * \rho_\theta \Psi_{ih}^{(l)}\right)(x) \\
&= \sum_{h=1}^{H} \int_{\mathbb{R}^2} \rho_\phi \hat{\zeta}_h^{(l-1)}(u)\, \rho_\theta \Psi_{ih}^{(l)}(x-u)\, \mathrm{d}u \\
&= \sum_{h=1}^{H} \int_{\mathbb{R}^2} \hat{\zeta}_h^{(l-1)}(\rho_{-\phi} u)\, \rho_\theta \Psi_{ih}^{(l)}(x-u)\, \mathrm{d}u\,.
\end{aligned}$$

Substituting $\tilde{u} := \rho_{-\phi} u$ with $\det\left(\frac{\partial \tilde{u}}{\partial u}\right) = \det\left(R_{-\phi}\right) = 1$ since $R_{-\phi} \in \mathrm{SO}(2)$ implies $x - u = x - \rho_\phi \tilde{u} = \rho_\phi(\rho_{-\phi} x - \tilde{u})$. The convolution can hence be manipulated further to

$$\begin{aligned}
&\sum_{h=1}^{H} \left(\rho_\phi \hat{\zeta}_h^{(l-1)} * \rho_\theta \Psi_{ih}^{(l)}\right)(x) \\
&= \sum_{h=1}^{H} \int_{\mathbb{R}^2} \hat{\zeta}_h^{(l-1)}(\tilde{u})\, \rho_{\theta-\phi} \Psi_{ih}^{(l)}(\rho_{-\phi} x - \tilde{u})\, \mathrm{d}\tilde{u} \\
&= \sum_{h=1}^{H} \left(\hat{\zeta}_h^{(l-1)} * \rho_{\theta-\phi} \Psi_{ih}^{(l)}\right)(\rho_{-\phi} x) \\
&= \rho_\phi y_i^{(l)}(x, \theta - \phi)
\end{aligned}$$

which reveals that the rotation of the input leads to spatially rotated pre-nonlinearity activations which are in addition cyclically shifted over the orientation index. This transformation behavior, acting on both the spatial position as well as on the orientation, is generic for oriented patterns; see Figure 4.4 for the example of rotating vector fields.

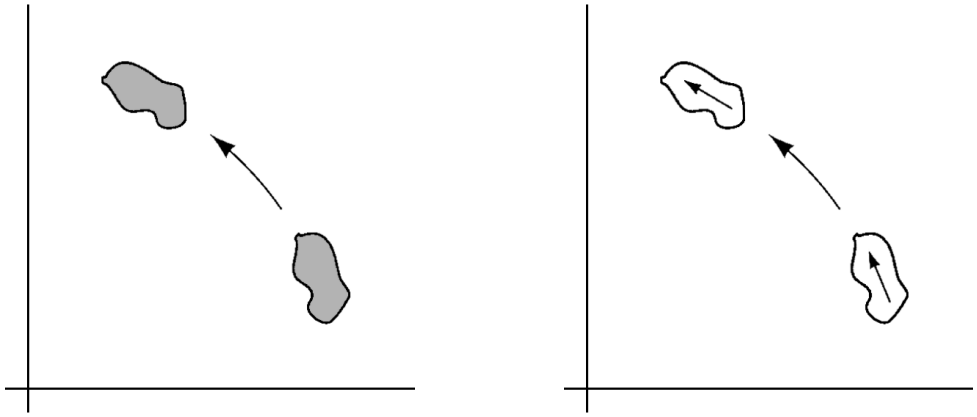
The permutation of responses under rotation of the input is captured by the group action

$$\mathcal{R} : \mathrm{SO}(2) \times \{y_i : \mathrm{SE}(2) \to \mathbb{R}\} \to \{y_i : \mathrm{SE}(2) \to \mathbb{R}\},\ (\phi, y_i(x,\theta)) \mapsto \rho_\phi y_i(x, \theta - \phi)$$

which operates on the space of activations before pooling over orientations. This allows to write the equivariance of the convolution step as

$$y_i^{(l)}(x,\theta) \xrightarrow{\hat{\zeta}^{(l-1)} \to \rho_\phi \hat{\zeta}^{(l-1)}} \sum_{h=1}^{H} \left(\rho_\phi \hat{\zeta}_h^{(l-1)} * \rho_\theta \Psi_{ih}^{(l)}\right)(x) \;=\; \mathcal{R}_\phi y_i^{(l)}(x,\theta)\,.$$

The pooling over orientations after addition of a bias and application of a nonlinearity is also



(a) Rotation of a scalar field.

(b) Rotation of a vector field.

Figure 4.4: Action of rotations on non-oriented and oriented patterns, here a scalar field and a vector field. A rotation of a scalar field is mathematically expressed by the field evaluated at the inversely rotated spatial position. That is, given a scalar field $S : \mathbb{R}^2 \to \mathbb{R}$ and a rotation matrix $R_\phi \in \mathrm{SO}(2)$, the action of the rotation transforms the scalar field according to $S(x) \to S(R_\phi^{-1}x)$. A rotation of a vector field, however, manifests twofold. On the one hand it displaces the spatial position of each vector, on the other hand it rotates the orientation of the displaced vectors themselves. Given a vector field $V : \mathbb{R}^2 \to \mathbb{R}^2$, the action of the rotation transforms it according to $V(x) \to R_\phi V(R_\phi^{-1}x)$.
Image credit: M. E. Peskin and D. V. Schröder. An Introduction to Quantum Field Theory. Westview Press. 1995.

equivariant:

$$\begin{aligned}
&\max_{\theta\in\Theta} \sigma\left(\mathcal{R}_\phi y_i^{(l)}(x,\theta) \,+\, \beta_i\right) \\
&= \max_{\theta\in\Theta} \sigma\left(\rho_\phi y_i^{(l)}(x,\theta-\phi) \,+\, \beta_i\right) \\
&= \max_{\theta\in\Theta} \rho_\phi\, \zeta_i^{(l)}(x,\theta-\phi) \\
&= \rho_\phi \max_{\theta\in\Theta} \zeta_i^{(l)}(x,\theta-\phi) \\
&= \rho_\phi \hat{\zeta}_i^{(l)}(x).
\end{aligned} \tag{4.9}$$

In the second and third step we used that the bias, the nonlinearity and the maximum-operation commute with the spatial rotation since they are independent of the spatial position. The last step can be interpreted as pooling over the subgroup $\mathrm{SO}(2) \leq \mathrm{SE}(2)$ which was argued to be invariant in (2.4). Note that the orientation-pooled feature maps transform by a mere spatial rotation, similar to the scalar field in Figure 4.4. This is because the pooled features do not carry any orientation information anymore.

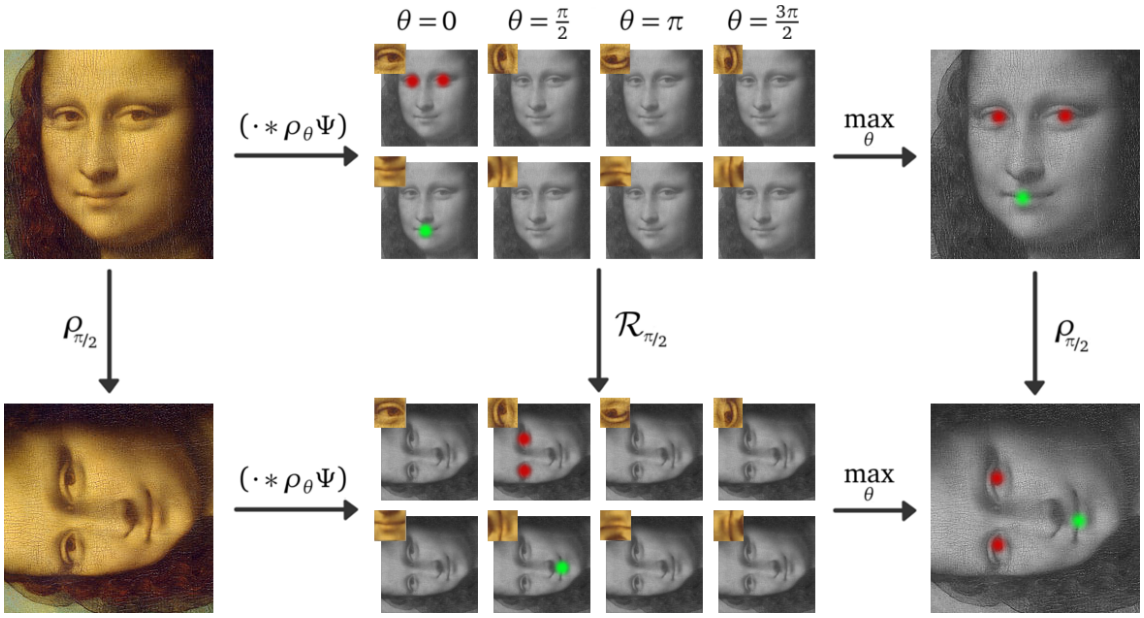



Figure 4.5: Transformation behavior of the responses of a hypothetical face detector which extracts eye- and mouth-features in four orientations. A rotated face evokes the same responses at a different spatial position in cyclically shifted orientation channels. Pooling over the orientation channels therefore gives the same responses, spatially rotated with respect to each other.

Together, the last calculations prove the equivariance of the proposed layer under rotations. Equivariance with respect to translations follows directly from the convolution operation as before. Again, we give an overview over the layers full transformation behavior under actions of SE(2) in a commutative diagram:

$$\begin{array}{ccccc}
\hat{\zeta}_h^{(l-1)} & \xrightarrow{\sum_h \left( \cdot * \rho_\theta \Psi_{ih}^{(l)} \right)} & y_i^{(l)} & \xrightarrow{\max_\theta \sigma \left( \cdot + \beta_i^{(l)} \right)} & \hat{\zeta}_i^{(l)} \\
\downarrow \mathcal{T}_u \rho_\phi & & \downarrow \mathcal{T}_u \mathcal{R}_\phi & & \downarrow \mathcal{T}_u \rho_\phi \\
\mathcal{T}_u \rho_\phi \hat{\zeta}_h^{(l-1)} & \xrightarrow[\sum_h \left( \cdot * \rho_\theta \Psi_{ih}^{(l)} \right)]{} & \mathcal{T}_u \mathcal{R}_\phi y_i^{(l)} & \xrightarrow[\max_\theta \sigma \left( \cdot + \beta_i^{(l)} \right)]{} & \mathcal{T}_u \rho_\phi \hat{\zeta}_i^{(l)}
\end{array}$$

To enhance the intuition on the origin of the different group actions $\rho_\phi$ and $\mathcal{R}_\phi$ in the space of orientation-pooled feature maps and pre-nonlinearity activations we give a graphical realization of the same commutative diagram in Figure 4.5. In this figure we consider a hypothetical

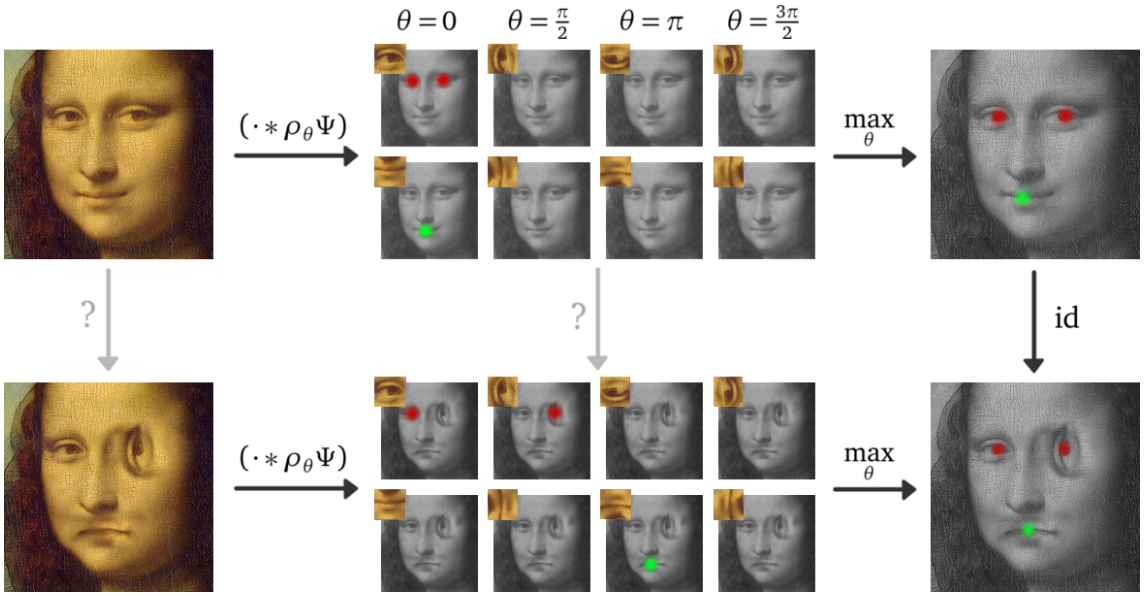



Figure 4.6: Ambiguous detections of the hypothetical face detector from Figure 4.5 under local rotation of small patches of the input image, in this example one eye and the mouth. The intermediate representation on the group captures the relative orientation of the features with e.g. the two eyes occurring in different orientation channels for the deformed face. After pooling over orientations this information is lost and both images result in the same feature map. The network can thus not discriminate between the two input images. A group-convolutional network learns filters directly on the group before pooling. The filters on the group learn the relative orientation of the features: in this example a valid face would have its eye-features in the same orientation channel as in the upper part of this figure or in the bottom part of Figure 4.5. The responses of the deformed face in the bottom of this figure do not reside in the same orientation channel such that a filter on the group would not respond.

face detector which extracts eye- and mouth-features in four orientations and highlight the transformation of the extracted features under rotation of the input.

While the proposed architecture is indeed equivariant under the Euclidean motion group there is, however, a huge loss of information accompanied with the step of pooling over orientations. In particular, the relative orientation between detected features is lost which may lead to sub-optimal results. The problematic nature of the orientation pooling architecture is highlighted in Figure 4.6 which again considers the hypothetical face detector introduced in Figure 4.5. Here we feed in a deformed face with an rotated eye and mouth which should obviously not be classified as a valid image of a face. While the pre-nonlinearity activations still have the full information of the relative orientation of the features there is no chance to discriminate between both inputs after pooling over orientations.

## 4.2 Group-convolutional architecture

In the last section we found that pooling over orientations is inherently accompanied by a loss of information on the relative alignment of features. We therefore investigate an alternative layer design which is capable to capture the features' relative orientations in an equivariant way. This design utilizes group convolutions to process feature maps which are functions of a group. It was proposed by Cohen and Welling [2016] for general transformation groups and implemented for the group consisting of translations, rotations by multiples of $\frac{\pi}{2}$ and reflections, that is for the subgroup $\mathbb{Z}^2 \rtimes D_4 \leq \mathrm{E}(2)$ of the Euclidean group. We consider again translations and an arbitrary number of orientations.

The first layer here is very similar to the layers of the orientation-pooling architecture but omits the pooling step. Specifically, we have pre-nonliniarity activations

$$y_i^{(1)}(x,\theta) \;:=\; \sum_{h=1}^{H} \big(f_h * \rho_\theta \Psi_{ih}^{(1)}\big)(x) \tag{4.10}$$

which lead to feature maps

$$\zeta_i^{(1)}(x,\theta) \;:=\; \sigma\left(y_i^{(1)}(x,\theta) + \beta_i^{(1)}\right).$$

Note that with this definition the feature maps are functions on the special Euclidean motion group, i.e. we have $\zeta_i : \mathrm{SE}(2) \to \mathbb{R}$. The presence of this group structure allows to perform group convolutions (2.1) in the following layers $l > 1$ with filters $\Psi_{ih} : \mathrm{SE}(2) \to \mathbb{R}$ now also being functions on the group. The advantage of group convolutions is that they allow to propagate the full information contained in their input feature maps, depending on the learned filterbanks.[2] In particular, the filters on the group are able to detect the difference between the two faces shown in Figure 4.6 since they can learn that eyes should occur in the same orientation and hence in the same orientation-channel. Keeping the parameterization by $(x,\theta)$, the group convolutions can be explicitly instantiated as

$$\begin{aligned} y_i^{(l)}(x,\theta) \;&=\; \sum_{h=1}^{H} \big(\zeta_h^{(l-1)} \circledast \Psi_{ih}^{(l)}\big)(x,\theta) \\ &= \sum_{h=1}^{H} \int_{\mathrm{SE}(2)} \zeta_h^{(l-1)}(u,\phi)\; \Psi_{ih}^{(l)}\big((u,\phi)^{-1}(x,\theta)\big)\, \mathrm{d}\lambda(u,\phi) \\ &= \sum_{h=1}^{H} \int_0^{2\pi} \int_{\mathbb{R}^2} \zeta_h^{(l-1)}(u,\phi)\; \Psi_{ih}^{(l)}\big(\rho_{-\phi}(x-u), \theta-\phi\big)\, \mathrm{d}u\, \mathrm{d}\phi \end{aligned} \tag{4.11}$$

[2] For example, the filter could coincide with the delta distribution which acts as identity under convolutions and therefore preserves the feature maps.

$$
\begin{aligned}
&= \sum_{h=1}^{H} \int_0^{2\pi} \left( \zeta_h^{(l-1)}(\cdot, \phi) \ast \rho_\phi \Psi_{ih}^{(l)}(\cdot, \theta - \phi) \right)(x) \, \mathrm{d}\phi \\
&= \sum_{h=1}^{H} \int_0^{2\pi} \left( \zeta_h^{(l-1)}(\cdot, \phi) \ast \left( \mathcal{R}_\phi \Psi_{ih}^{(l)} \right)(\cdot, \theta) \right)(x) \, \mathrm{d}\phi \,,
\end{aligned}
$$

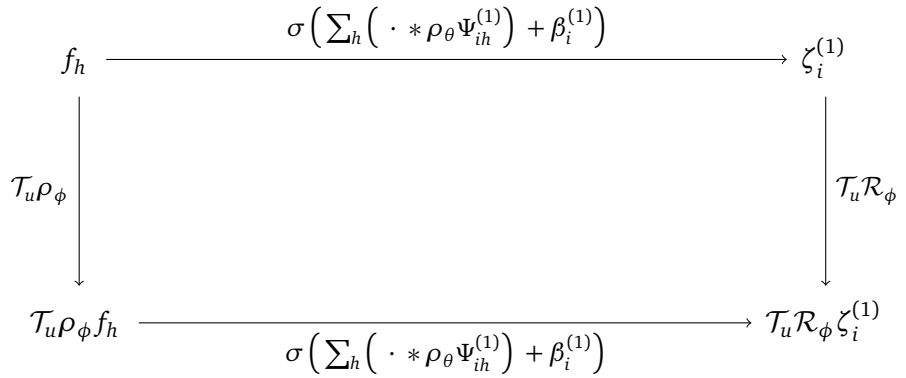
where the multiplication with the inverse group element was evaluated by making use of the previously derived relations (4.1) and (4.3). The above equation reveals that the group convolution can be decomposed into spatial convolutions of feature maps sliced over orientation channels $\zeta_h(\cdot, \phi)$ with filter slices $\Psi_{ih}(\cdot, \theta)$ acted on by $\mathcal{R}_\phi$. Note that this is the same group action under which the feature maps transform when the input image is rotated. In order to make the mapping non-linear, we apply an activation function after adding a bias as before and obtain

$$
\zeta_i^{(l)}(x, \theta) = \sigma \left( y_i^{(l)}(x, \theta) + \beta_i^{(l)} \right) .
$$

Since these feature maps are again functions on the group they can be mapped further by applying subsequent group-convolutional layers.

After the last group-convolutional layer we extract the desired information. For rotation-invariant classification we pool over both spatial and orientation dimensions. Pooling over orientations after the last layer is also done for rotation-equivariant segmentation where spatial dimensions remain and the output rotates according to the rotation of the network's input, i.e. under the action $\rho$. If the orientation itself is of interest it can be kept as extra feature.

Each individual layer of the group-convolutional architecture is equivariant under joint translations and rotations. This is proven for the first layer in the last section and for the group-convolutional layers in example 2.2.2. For the first layer the commutative diagram is given by:



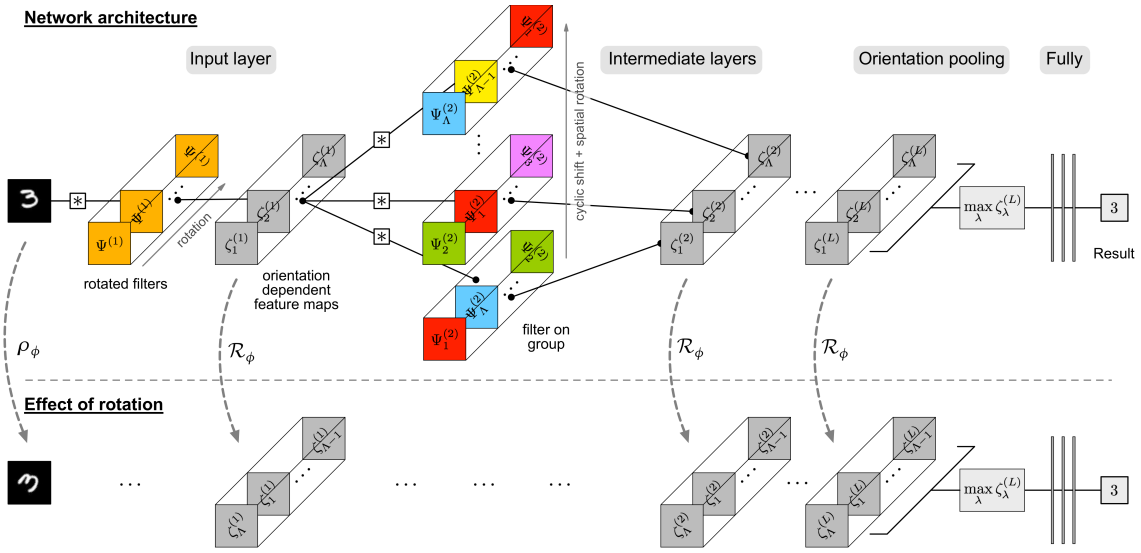



Figure 4.7: Visualization of the basic structure of the proposed rotation-equivariant network. The upper part shows how the data is processed by the architecture while the bottom part concentrates on the transformation of feature maps under rotation of the input. A detailed explanation of the whole figure is given in the main text.

The subsequent group-convolutional layers transform according to:

$$\begin{array}{ccc} \zeta_h^{(l-1)} & \xrightarrow{\quad \sigma\left(\sum_h\left(\,\cdot\, \circledast\, \rho_\theta \Psi_{ih}^{(l)}\right) + \beta_i^{(l)}\right) \quad} & \zeta_i^{(l)} \\ \Big\downarrow \mathcal{T}_u\mathcal{R}_\phi & & \Big\downarrow \mathcal{T}_u\mathcal{R}_\phi \\ \mathcal{T}_u\mathcal{R}_\phi\, \zeta_h^{(l-1)} & \xrightarrow[\quad \sigma\left(\sum_h\left(\,\cdot\, \circledast\, \rho_\theta \Psi_{ih}^{(l)}\right) + \beta_i^{(l)}\right) \quad]{} & \mathcal{T}_u\mathcal{R}_\phi\, \zeta_i^{(l)} \end{array}$$

We give a visual intuition on the transformation properties of the feature maps in the network in Figure 4.7. The upper part of the figure gives an overview over the basic structure of the proposed network. For clarity, we display only a single group-convolutional layer and a single feature channel. Filters which are related by rotations are highlighted in the same color, rotated Greek letters represent the spatial orientations of the filters and the feature maps. Note that we consider a subgroup of $T_2 \rtimes C_\Lambda \leq \mathrm{SE}(2)$ with discrete orientations $\theta_\lambda$, $\lambda \in \{1, \dots, \Lambda\}$ and abbreviate the orientation component as subscript, i.e. $\Psi_\lambda = \Psi(\,\cdot\,, \theta_\lambda)$. In the first layer

the input image on $\mathbb{R}^2$ is convolved with rotated versions of the same filter on $\mathbb{R}^2$, leading to feature maps on $T_2 \rtimes C_\Lambda$. The orientation dimension is visualized with an additional axis reaching into the image plane. Group convolutions in the intermediate layers map these features further by convolving them with filters on the group. Each orientation component of the resulting feature map results from spatially convolving all orientation channels of the input features with the orientation channels of one filter on the group and summing their responses together. The filters on the group leading to the different orientation channels of the output feature map share weights and are related by the group action $\mathcal{R}$. For example, the filter for calculating $\zeta_1^{(2)}$ contains the slice $\Psi_1^{(2)}$, highlighted in red, in its first orientation component while the filter for calculating $\zeta_2^{(2)}$ contains the same filter slice in a rotated version in its second orientation component. One can concatenate an arbitrary number of these group-convolutional layers. In a classification setting, we finally pool over orientations and spatial dimensions in the last equivariant layer, followed by fully connected layers.

The bottom part of Figure 4.7 gives a visualization of the layerwise rotation-equivariance. Applying a rotation $\rho_\phi$ to the input image results in a joint cyclic shift and rotation operation $\mathcal{R}_\phi$ on the feature maps $\zeta^{(l)}$. This becomes intuitively clear when paying attention to the relative orientation of the input and the first layer's filters: The original input has the same relative orientation to the first filter rotation as the rotated input with the second filter rotation. Therefore, both give the same response $\zeta_1^{(1)}$ but with a different spatial rotation and in a different orientation channel. When these feature maps are mapped further by group-convolutional layers, this structure is preserved: To see that this is the case, consider the relative orientation between the output features of the first layer and the bottom filter on the group. This relative orientation again coincides with the relative orientation between the transformed output features of the first layer and the second filter from the bottom. Consequently, both give the same response but rotated with respect to each other and in a different orientation channel. The orientation pooling after the last group-convolutional layer therefore gets the same input in a rotated and cyclically shifted version. Since the shift over orientations is removed, the resulting outputs will be connected by a spatial rotation.

As before we can analyze how the network's equivariance is reflected in its hypothesis space. In comparison to a conventional CNN the proposed architecture is additionally invariant under rotations such that the hypothesis space is further constrained to

$$\mathcal{H}_{\mathrm{SE}(2)} \;=\; \left\{ h \;\middle|\; h : \mathcal{F}/\mathrm{SE}(2) \to \mathcal{L} \right\} .$$

This implies that the network learns to classify orbits $\mathrm{SE}(2).f \in \mathcal{F}/\mathrm{SE}(2)$ consisting of all images $f \in \mathcal{F}$ which are related by rotations and shifts. As a consequence, a pattern learned in one specific orientation and position will be detected again correctly in any orientation at arbitrary positions.

# 5 Steerable Filter CNNs

An important point which we left open so far is the design of the network's filters. Since convolutional neural networks build on the concept of learning filters we need to find a suitable parametrization. Our construction demands for filters whose responses can be computed both accurately and economically for several filter orientations. Simultaneously the filters should not be restricted in their expressive power, i.e. in the patterns they are able to learn. All of these requirements are naturally met by learning linear combinations of a system of steerable filters.

Another aspect to be addressed is the discrete nature of digitized signals. The rotationally equivariant network architectures introduced in the last chapter made the assumptions that the input images live on a continuous domain and considered the transformation group SE(2) which contains continuous rotations. In practical applications, however, we need to work with signals sampled on a pixel grid $\mathbb{Z}^2$ and with a finite number $\Lambda \in \mathbb{N}$ of sampled filter orientations. This means, we are interested in the subgroups $(\mathbb{Z}^2, +) \leq T_2$ of translations and discrete rotations in $C_\Lambda \leq \mathrm{SO}(2)$.[1]Unfortunately, the restriction of SE(2) to the pixel grid and discrete rotations does in general not form a group structure itself since rotations by angles not being multiples of $\frac{\pi}{2}$ do not map the pixel grid onto itself such that the resulting algebraic structure is not closed under the binary operation. This is one reason why related approaches, proposed in Cohen and Welling [2016], Dieleman et al. [2016] and Cohen and Welling [2017], consider only four filter orientations. In order to circumvent these issues we only restrict to $\Lambda$ discretely sampled orientations $\theta_\lambda \in \Theta := \left\{0, \dots, 2\pi\frac{\Lambda-1}{\Lambda}\right\}$ but keep the assumption of spatially continuous translations. Together they form the subgroup $T_2 \rtimes C_\Lambda \leq \mathrm{SE}(2)$. Numerical experiments in chapter 8 show that the artifacts resulting from a spatially discrete implementation are measurable but that the gains of the equivariant construction are nonetheless significant.

In the next section we will first describe a reasonable construction of steerable filters for learning in CNNs. Subsequently we formulate the rotation-equivariant CNN introduced in the last chapter in terms of these steerable filters.

[1]To be precise, $C_\Lambda$ is isomorphic to a subgroup of SO(2).

## 5.1 Parametrization of steerable filters

A filter $\Psi$ is called steerable in the sense of Freeman and Adelson [1991], when its rotation by an arbitrary angle $\theta$ can be expressed as linear combination of versions of itself which are rotated by a fixed set of typically equidistant basis angles $\alpha_0, \ldots, \alpha_{R-1}$. More formally, a steerable filter $\Psi : \mathbb{R}^2 \to \mathbb{R}$ satisfies

$$\rho_\theta \Psi(x) = \sum_{r=0}^{R-1} \kappa_r(\theta) \rho_{\alpha_r} \Psi(x), \tag{5.1}$$

for all angles $\theta \in (-\pi, \pi]$ and for angular coefficient functions $\kappa_r$. An important practical consequence of the linearity of steerable filters is that steering and convolution commute. Therefore, the response of each orientation can be synthesized from the responses with respect to the basic directions $f * \rho_{\alpha_r} \Psi$; that is,

$$(f * \rho_\theta \Psi)(x) = \sum_{r=0}^{R-1} \kappa_r(\theta) \big(f * \rho_{\alpha_r} \Psi\big)(x).$$

A family of steerable filters which is particularly easy to handle are circular harmonics; see e.g. Hsu and Arsenault [1982] or Rosen and Shamir [1988]. They are defined by a sinusoidal angular part multiplied with a radial function $\tau : \mathbb{R}^+ \mapsto \mathbb{R}$, i.e.

$$\psi_k(r, \phi) = \tau(r) e^{ik\phi},$$

where $(r, \phi)$ denote polar coordinates of $x = (x_1, x_2)$ and $k \in \mathbb{Z}$ is the angular frequency. By construction, $\psi_k$ can be rotated by multiplication with a complex exponential,

$$\rho_\theta \psi_k(x) = e^{-ik\theta} \psi_k(x),$$

and is thus steerable in the sense of Freeman and Adelson [1991] [2] with $R = 1$ and interpolation functions $\kappa_0(\theta) = e^{-i\,k\theta}$.

In our network, we utilize a system of circular harmonics $\psi_{jk}$ with $j = 1, \ldots, J$, and $k = 0, \ldots, K_j$ where the additional index $j$ controls the radial part of $\psi_{jk} = \tau_j(r) e^{ik\phi}$. The left-hand side of Figure 5.1 shows the real and imaginary parts of the atoms used in the experiments

[2] Circular harmonics can equivalently be interpreted as a pair $\psi_k^{\mathrm{Re}}(r, \phi) = \tau(r)\cos(k\phi)$ and $\rho_{\frac{\pi}{2}} \psi^{\mathrm{Re}}(r, \phi) = \psi_k^{\mathrm{Im}}(r, \phi) = -\tau(r)\sin(k\phi)$ with $R = 2$ and $\kappa_0(\theta) = \cos(k\theta)$ and $\kappa_1(\theta) = -\sin(k\theta)$.

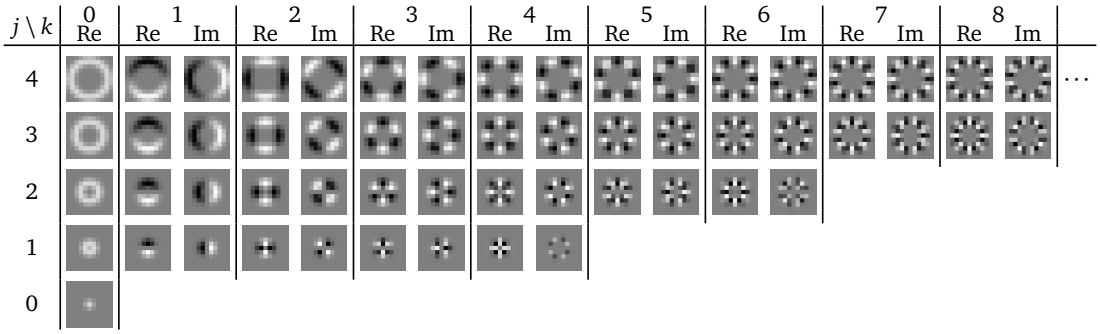



Figure 5.1: Illustration of the circular harmonics $\psi_{jk}(r,\phi) = \tau_j(r)e^{ik\phi}$ sampled on a $9 \times 9$ grid. The steerable system is given by real and imaginary parts of a complex valued filter. Each row shows a different radial part $j$, the angular frequencies are arranged in the columns. For larger scales there are higher frequency filters not shown here.

where we chose Gaussian radial parts

$$\tau_j(r) = \exp\left(-\frac{(r-\mu_j)^2}{2\sigma^2}\right).$$

The maximum angular frequencies $K_j$ are limited to the point where aliasing effects occur. As a consequence, we let them grow linearly with the rings' circumferences. We found this system to be convenient for learning as the filters are approximately orthogonal and radially localized.

The learned filters are then defined as linear combinations of the elementary filters, that is,

$$\tilde{\Psi}(x) = \sum_{j=1}^{J}\sum_{k=0}^{K_j} w_{jk}\psi_{jk}(x), \tag{5.2}$$

with weights $w_{jk} \in \mathbb{C}$. The complex phase of the weights allows to rotate the atomic filters *with respect to each other*. Such a composed filter can subsequently be steered *as a whole* by the steerability of the atoms via

$$\rho_\theta\tilde{\Psi}(x) = \sum_{j=1}^{J}\sum_{k=0}^{K_j} w_{jk}e^{-i\,k\theta}\psi_{jk}(x). \tag{5.3}$$

While it is not immediately obvious that this formulation is steerable in the sense of Freeman and Adelson [1991] as defined in (5.1), the combined filters are still rotated by taking a linear combination of filters. Indeed, both approaches can be shown to be equivalent as they are related by a unitary transformation of the expansion coefficients; see appendix B. The form

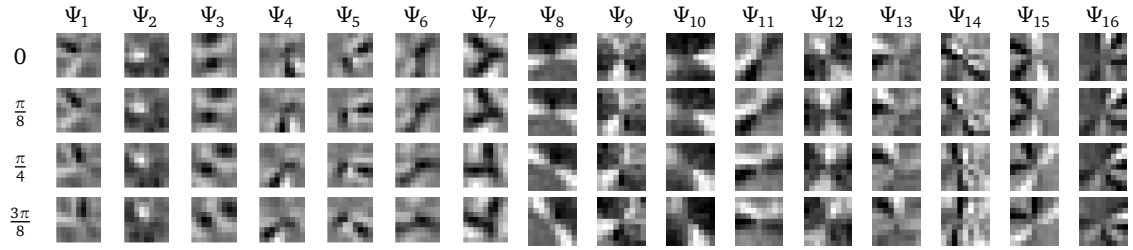



Figure 5.2: Some of the filters learned in the first layer of a group-convolutional CNN. The first row depicts the composed filters $\Psi_i$, the subsequent rows show their steered versions by the angle indicated.

(5.3) is convenient for our network since it forms a natural basis for arbitrary angularly band-limited steerable filters on the one hand and since it allows to reuse the atomic responses for each combined filter on the other hand.

In practice we select a single orientation of the combined filters by taking their real part

$$\Psi(x) = \operatorname{Re}\tilde{\Psi}(x)$$

and let

$$\rho_\theta \Psi = \operatorname{Re}\rho_\theta \tilde{\Psi}\,.$$

Figure 5.2 shows an example of such filters learned in the first layer of a group-convolutional CNN together with some of their steered versions.

In practice we sample the atomic filters $\psi_{jk} : \mathbb{Z}^2 \to \mathbb{C}$ on a grid. For conventional filters the sampled version does not suffice to exactly rotate them by angles which do not map the pixel grid onto itself. Instead some interpolation needs to be chosen for which there is, however, no unique choice. Furthermore, any interpolation comes with its own artifacts which can be severe on the length scale of pixels. Given that filters in convolutional networks typically have very small support these artifacts may become a big issue. Steerable filters on the other hand are rotated exactly by steering, even when sampled on a grid. This is, because steerability (5.1) is defined locally: the filter values of the basis orientations at each single point fully suffice to reconstruct the filter at this point in all orientations. Figure 5.3 shows the artifacts resulting from rotating an image with either interpolation methods or by fitting a system of steerable filters to it and rotating it analytically.

(a) Original image (b) *Nearest* interpolation (c) Linear interpolation (d) Steerable filters

Figure 5.3: Comparison of different methods to rotate images or filters. In all cases the original image is rotated by 30 degrees and then rotated back. This is what happens in CNNs which apply rotated the filters in the forward pass and to rotate the gradients back during backpropagation. One can see that the *nearest* interpolation suffers severe interpolation artifacts on the length scale of pixels. The linear interpolation gives a visually more appealing result which, however, tend to be smoother than the original image. Higher order interpolations tend to give even smoother results. A different approach to rotate the image is shown in the rightmost figure where we fitted a system of our steerable atomic filters to the original image and rotated it analytically. This method suffers less artifacts and the rotated image looks sharper.

## 5.2 Steerable filter formulation of the CNN

Given a system $\psi_{jk}$ of atomic steerable filters as defined in the last section we now formulate the proposed rotation-equivariant network architectures in terms of this basis. We will concentrate on the group-convolutional design since it comprises the layers utilized in the orientation-pooling architecture as its first layer.

Plugging steerable filters

$$\Psi_{ih}^{(1)} = \mathrm{Re}\sum_{j=1}^{J}\sum_{k=0}^{K_j} w_{ihjk}^{(1)}\psi_{jk}$$

into the formula for the first layers pre-nonlinearity activations (4.10) we obtain

$$\begin{aligned} y_i^{(1)}(x,\theta_\lambda) &= \sum_{h=1}^{H}\big(f_h * \rho_{\theta_\lambda}\Psi_{ih}^{(1)}\big)(x) \\ &= \sum_{h=1}^{H}\left(f_h * \mathrm{Re}\sum_{j=1}^{J}\sum_{k=0}^{K_j} w_{ihjk}^{(1)} e^{-ik\theta_\lambda}\psi_{jk}\right)(x) \qquad (5.4) \\ &= \sum_{h=1}^{H}\sum_{j=1}^{J}\sum_{k=0}^{K_j}\mathrm{Re}\Big(w_{ihjk}^{(1)} e^{-ik\theta_\lambda}\big(f_h * \psi_{jk}\big)\Big)(x) \qquad (5.5) \end{aligned}$$

with complex parameters $w_{ihjk}^{(1)} \in \mathbb{C}$. In this setting the rotational weight sharing is reflected by the independence of the weights from the angle $\theta_\lambda$ which comes into play only by the

complex exponentials modulating the weights' phases. A higher resolution in orientations can be achieved by expanding the tensor containing these phase-factors. In analogy to the first layer we make use of the steerable filters in the following layers $l > 1$ which on the group are defined by

$$\Psi_{ih}^{(l)}(x, \theta_\lambda) = \mathrm{Re} \sum_{j=1}^{J} \sum_{k=0}^{K_j} w_{ihjk\theta_\lambda}^{(l)} \psi_{jk}(x).$$

Note that the additional orientation dimension is reflected by an additional index of the weight tensor. Inserting these steerable filters in the pre-nonlinearity activations (4.11) of the group-convolutional layers gives:

$$\begin{aligned}
y_i^{(l)}(x, \theta_\lambda) &= \sum_{h=1}^{H} \left( \zeta_h^{(l-1)} \circledast \Psi_{ih}^{(l)} \right)(x, \theta_\lambda) \\
&= \sum_{h=1}^{H} \sum_{\phi \in \Theta} \left( \zeta_h^{(l-1)}(\cdot, \phi) \ast \rho_\phi \Psi_{ih}^{(l)}(\cdot, \theta_\lambda - \phi) \right)(x) \\
&= \sum_{h=1}^{H} \sum_{\phi \in \Theta} \left( \zeta_h^{(l-1)}(\cdot, \phi) * \mathrm{Re} \sum_{j=1}^{J} \sum_{k=0}^{K_j} w_{ihjk,\theta_\lambda - \phi}^{(l)} e^{-i k \phi} \psi_{jk} \right)(x) \qquad (5.6) \\
&= \sum_{h=1}^{H} \sum_{\phi \in \Theta} \sum_{j=1}^{J} \sum_{k=0}^{K_j} \mathrm{Re} \left( w_{ihjk,\theta_\lambda - \phi}^{(l)} e^{-i k \phi} \left( \zeta_h^{(l-1)}(\cdot, \phi) * \psi_{jk} \right) \right)(x) \qquad (5.7)
\end{aligned}$$

The last two lines (5.4) and (5.5) as well as (5.6) and (5.7) of the steerable filter formulations of the network's layers are mathematically equivalent but differ in the order of the computation of the linear combination and the convolution. While in the second last lines the bank of rotated filters is explicitly built and afterwards convolved with the input channels, the last lines suggest to first compute the responses of the input with respect to the atomic filters and then linearly combine these. The single steps performed in these different implementations are visualized in Figures 5.4 and 5.5. The advantage of the approach which first computes the atomic responses before linearly combining them is that the number of convolutions to be performed does not depend on the number of output channels or sampled filter orientations. Since all combined filters are based on the same system of steerable filters each atomic response is reused multiple times for each learned filter and orientation. A drawback is that the linear combination of the atomic responses is more costly than a linear combination of filters since the feature maps are typically larger.

As in conventional CNNs we apply pooling operations with local pooling kernels after a few convolutional layers. In addition to spatial pooling we can here also pool over the orientation coordinate. The pooling over translations is independent from the orientation index and is thus

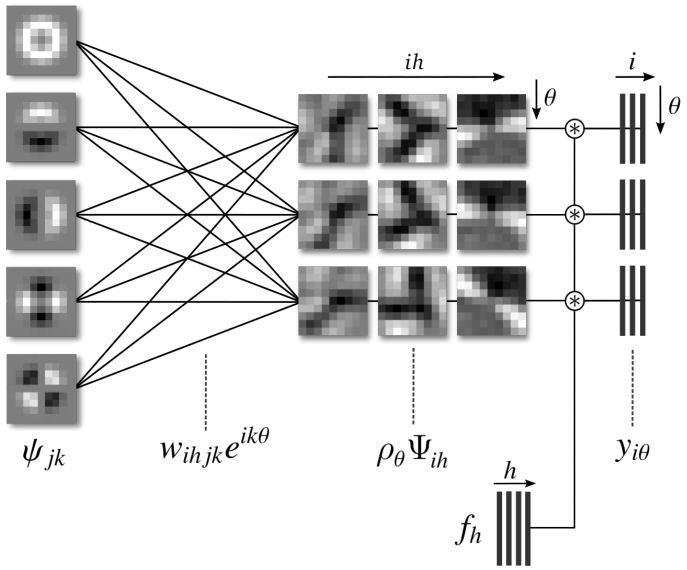



Figure 5.4: Visualization of an implementation of the first layer which explicitly builds the filter bank by linearly combining atomic steerable filters before convolving them with the feature maps as usual. This implementation follows the order of computation implied by the bracketing in equation (5.4). The number of convolution operations to be performed grows linearly with the number of output feature channels and sampled orientations.

equivalent to the case of conventional CNNs as discussed in section 3.2. Conversely, pooling over orientation coordinates does not depend on the spatial position and is hence very similar to translational pooling. A non-strided pooling step, modeled by a pooling operator $\mathcal{P}^{\text{ang}}_{\widetilde{\Theta}}$ over a pooling region $\widetilde{\Theta} := \left\{\frac{2\pi\lambda}{\Lambda}\right\}_{\lambda=0}^{\tilde{\Lambda}-1}$ with $\tilde{\Lambda} \leq \Lambda$ is rotationally equivariant since the spatial part of the rotational group action commutes with the maximum operation:

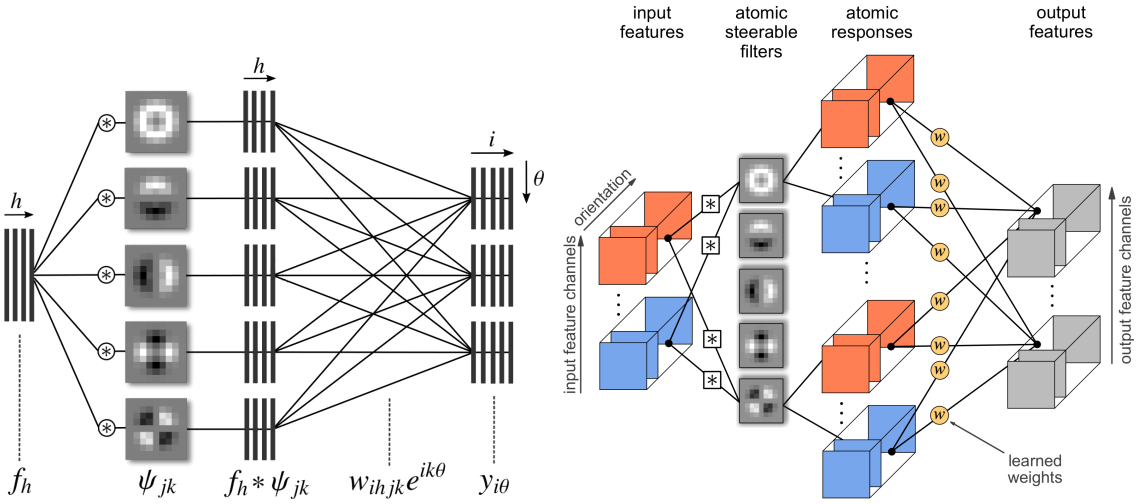
$$\begin{aligned}
\left(\mathcal{P}^{\text{ang}}_{\widetilde{\Theta}} \mathcal{R}_\phi \zeta\right)(x,\theta) &= \max_{\tilde{\theta} \in \theta\widetilde{\Theta}} \left(\mathcal{R}_\phi \zeta\right)(x,\tilde{\theta}) \\
&= \max_{\tilde{\theta} \in \theta\widetilde{\Theta}} \zeta(\rho_{-\phi} x, \tilde{\theta} - \phi) \\
&= \rho_\phi \max_{\tilde{\theta} \in (\theta-\phi)\widetilde{\Theta}} \zeta(x,\tilde{\theta}) \\
&= \left(\mathcal{R}_\phi \mathcal{P}^{\text{ang}}_{\widetilde{\Theta}} \zeta\right)(x,\theta)
\end{aligned}$$

Given an $s \in \mathbb{N}$ dividing $\Lambda$ we can then define a downsampling over orientations via

$$\mathcal{D}^{\text{ang}}_s \zeta(x,\theta_\lambda) := \zeta(x,\theta_{s\lambda})$$

for $\lambda = 0,\ldots,\frac{\Lambda}{s}-1$. In analogy to relation 3.4 it is easy to show that

$$\mathcal{D}^{\text{ang}}_s \mathcal{R}_{\theta_{s\lambda}} = \mathcal{R}_{\theta_\lambda} \mathcal{D}^{\text{ang}}_s .$$




(a) Input layer

(b) Group-convolutional layers

Figure 5.5: Visualization of possible implementations of the first or of group-convolutional layers. In both cases one can first convolve each channel of the input with every atomic filters to obtain atomic responses. These can subsequently be linearly combined to obtain the output feature maps. The implementations shown here follow the order of computation as implied by the bracketing in equations (5.5) and (5.7) respectively. This design has the advantage that only a fixed number of convolution operations has to be performed, independent of the number of output feature maps and filter orientations. A downside of the approach is that the linear combination of the atomic responses is more expensive in common operating domains where the spatial extent of feature maps is way bigger than the one of the filters.

A strided pooling composed of these two steps is thus equivariant under arbitrary translations and rotations in the subgroup $C_{\Lambda/s} \leq C_\Lambda$. Which number of sampled orientations is sensible depends on the size of the filter kernel. As the circumference of the largest ring fitting in the kernel grows linearly with its size we recommend to choose the number of sampled orientations proportional to the filter size. This implies to sample down over orientations only in cases where the kernel size of subsequent layers is reduced by a factor larger than two. For example, when the first layer utilizes $11 \times 11$ filters while the group-convolutional layers use $5 \times 5$ filters it makes sense to reduce the number of sampled orientations by half. When the kernel size of all layers is approximately constant we do not do intermediate pooling over orientations at all. As discussed before, the global pooling over orientations after the last rotation-equivariant layer is rotationally invariant; see equation (4.9). Together with the equivariance of the input layer and the group-convolutional layer which were proven in chapter 4 this makes all building blocks of the proposed network equivariant under $T_2 \rtimes C_\Lambda$ when operating on continuous signals.

In the following subsections we will discuss the computational complexity of the two different approaches for computing the responses as well as the reduced parameter cost by rotational weight sharing. In appendix A we add a formulation of the group-convolutional layer in Fourier space which makes use of the fact that the filter basis is fixed and that they are rotated by phase manipulation to achieve an efficient implementation.

### 5.2.1 Computational complexity

In order to assess the computational expense of the two implementations quantitatively we analyze their scaling with the relevant variables on the example of the input layer. Therefore, we assume the convolution operations to be performed with "same" boundary conditions, that is, with a padding such that the spatial extent of the output feature maps equals the size of the input. We further assume square images with $X^2$ pixels and square convolution kernels with $s^2$ pixels and denote the absolute number of atomic filters by $Q$. Then, the complexity of the convolutional part of the implementation combining atomic responses is given by $\mathcal{O}(HQs^2X^2)$ while the complexity of the subsequent linear combination reads $\mathcal{O}(IHQ\Lambda X^2)$. By contrast, the implementation first building the filter bank consisting of rotated filters has a runtime of $\mathcal{O}(IHQ\Lambda s^2)$ for the linear combination and a complexity of $\mathcal{O}(HI\Lambda s^2X^2)$ for convolving with the explicitly rotated filters. From that one can see that the convolutional parts differ in their scaling by a factor of $Q$ and $I\Lambda$. Since we usually have $Q \leq s^2$ in settings with non over-parameterized filters and small kernel sizes in comparison to the number of output channels and rotations, the approach of convolving with atomic filters is often cheaper than convolving with explicitly rotated filters. On the contrary, the linear combinations differ in their scaling by a factor of $X^2$ and $s^2$ respectively. Here the approach of first combining the filters is less expensive since one typically has $X^2 \gg s^2$. We implemented both approaches and found that it is almost always more efficient to build the filter bank explicitly before convolving it, especially for large images. The approach based on computing atomic responses may be more efficient in settings with small image sizes together with smooth atomic filter bases, i.e. with small $Q$ as investigated by Jacobsen et al. [2016].

A further problem which can arise when implementing the approach based on computing atomic responses in a naive way are memory issues. These arise when using the automatic differentiation of the current generation of deep learning frameworks[3] which stores all atomic responses for backpropagating the gradients to the weights such that the memory consumption grows by the factor $Q$ of atomic filters. A more memory efficient way is to implement the layers as custom operations which recompute the atomic responses from the feature maps when they are needed.

[3] Tested in Theano, Tensorflow and Pytorch.

### 5.2.2 Reduced parameter cost

By the weight sharing the parameter cost of the input layer is independent of the number of sampled orientations $\Lambda$. This independence translates to the whole orientation-pooling architecture since it consists only of input-type layers. Put in comparison to a conventional CNN which explicitly learns rotated filters in an rotation-invariant vision task this means that the proposed layer utilizes by this factor less parameters and therefore has a lower sample complexity. This argumentation is of course only valid in the range of sampled filter orientations where the conventional network is not able to generalize over different rotations by, e.g. the invariance to small deformations induced by spatial pooling.

In the group-convolutional layers we need to learn filters on the group which implies that the parameter cost of these filters grows linearly with $\Lambda$. Again, the weight sharing makes the equivariant network more parameter efficient since a conventional CNN needs to learn filters on the (implicitly given) group in multiple orientations such that the parameter cost grows with $\Lambda^2$. All in all we see that a group-convolutional steerable filter CNN consumes by a factor of $\Lambda$ less parameters than a conventional CNN such that its parameter cost only grows linearly with the number of sampled orientations.

Lets put this in analogy to the gain in parameter utilization by introducing translational weight sharing when transitioning from MLPs to CNNs as discussed in section 3.2. There we found that an MLP consumes $\mathcal{O}(N^2)$ parameters when fully connecting each of $\mathcal{O}(N)$ output neurons to $N$ input pixels. When sharing weights over spatial position one reduces the number of parameters in a similar way as for rotational weight sharing to linear scaling $\mathcal{O}(N)$ in spatial positions. For translations it was, however, possible to reduce the number of parameters further to $\mathcal{O}(1)$ by connecting each output neuron only to a local receptive field. A justification for using local receptive fields is that large objects can be disassembled into a hierarchy of local patterns. Unfortunately, this argument does not apply to rotations since there is no natural hierarchy in orientations. The filters on the group should rather span over all $\Lambda$ orientations because two patterns may appear in arbitrary relative orientations.

# 6 Generalized weight initialization

The training process of neural networks requires an appropriate initialization of the weights in order to converge to a reasonable local minimum. Motivated by convergence issues of deep architectures, Glorot and Bengio [2010] examined how the flow of signals through their layers depends on the initial distribution of the weights. Assuming sigmoidal activation functions, they proposed to choose the parameters' scales such that the sigmoids neither saturate, nor prevalently act in their linear regime. This can be ensured by demanding the variance of the pre-nonlinearity activations and the backpropagated gradients to be constant over the layers. The resulting *Xavier initialization scheme* indeed shows convincing convergence results and has established as a standard for the chosen nonlinearities. A similar derivation, which was added later by He et al. [2015], adapts these ideas to the nowadays more popular ReLU activations. Our approach picks up these ideas but generalizes the results to CNNs learning filters as linear combinations of a fixed basis of atomic filters.

For the derivation consider the activation of a single neuron in layer $l$,

$$\zeta_{ix}^{(l)} \;=\; \max\Big(0, y_{ix}^{(l)}\Big), \tag{6.1}$$

where we chose ReLU nonlinearities and switched to a full index notation of all variables. For convenience we define the pre-nonlinearity activations with the bias already added:

$$y_{ix}^{(l)} \;=\; \sum_{h}\sum_{x'} \Psi_{ihx'}^{(l)}\, \zeta_{h,x-x'}^{(l-1)} \;+\; \beta_i^{(l)} \tag{6.2}$$

As above we consider composed filters

$$\Psi_{ihx}^{(l)} \;=\; \sum_{q=1}^{Q} w_{ihq}^{(l)}\, \psi_{qx}$$

which are built from $Q$ real valued elementary filters and map $H$ input-feature maps to $I$ output-feature maps. We keep the discussion general by not restricting the atoms to be steerable. Note that this setup includes the case He et al. considered for the choice of a Dirac basis where $\psi_q$ is zero for all but one pixel. In analogy to Glorot and Bengio [2010] and He et al. [2015] we assume the activations and gradients to be i.i.d. and to be independent from the weights. We let the weights themselves be mutually independent and have zero mean but

do not constrict them to be identically distributed because of the inherent asymmetry coming from the different atomic filters. Furthermore we initialize all biases to be zero.

In order to prevent vanishing or exploding gradients of the loss $\mathcal{E}$ we examine their variances' dependence on the higher layers' gradients and weights. The gradient with respect to the activation of a particular neuron $\zeta^{(l)}_{h_0 x_0}$ in layer $l$ follows from (6.1) and (6.2) to be given by

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \zeta^{(l)}_{h_0 x_0}} &= \sum_i \sum_x \frac{\partial \mathcal{E}}{y^{(l+1)}_{ix}} \frac{y^{(l+1)}_{ix}}{\partial \zeta^{(l)}_{h_0 x_0}} \\ &= \sum_i \sum_x \frac{\partial \mathcal{E}}{\zeta^{(l+1)}_{ix}} \mathbb{I}_{y^{(l+1)}_{ix} > 0} \sum_q w^{(l+1)}_{i h_0 q} \psi_{q, x - x_0},\end{aligned} \tag{6.3}$$

where the indicator function $\mathbb{I}$ stems from the derivative of the rectified linear unit. Like He et al. [2015] we assume the factors occurring in (6.3) to be statistically independent. Observing that $\mathbb{E}\left[w^{(l)}\right]$ and $\mathbb{E}\left[\frac{\partial \mathcal{E}}{\partial \zeta^{(l)}}\right]$ vanish, and w.l.o.g. setting $x_0 = 0$ this leads to

$$\begin{aligned}&\operatorname{Var}\left[\frac{\partial \mathcal{E}}{\partial \zeta^{(l)}_{h_0 x_0}}\right] \\ &= \mathbb{E}\left[\left(\frac{\partial \mathcal{E}}{\partial \zeta^{(l)}_{h_0 x_0}}\right)^2\right] \\ &= \sum_{i,i'} \sum_{x,x'} \sum_{q,q'} \mathbb{E}\left[\frac{\partial \mathcal{E}}{\partial \zeta^{(l+1)}_{ix}} \frac{\partial \mathcal{E}}{\partial \zeta^{(l+1)}_{i'x'}}\right] \mathbb{E}\left[\mathbb{I}_{y^{(l+1)}_{ix} > 0} \mathbb{I}_{y^{(l+1)}_{i'x'} > 0}\right] \cdot \mathbb{E}\left[w^{(l+1)}_{i h_0 q} w^{(l+1)}_{i' h_0 q'}\right] \psi_{q,x} \psi_{q',x'} \\ &= \sum_i \sum_x \sum_q \mathbb{E}\left[\left(\frac{\partial \mathcal{E}}{\partial \zeta^{(l+1)}_{ix}}\right)^2\right] \mathbb{E}\left[\mathbb{I}_{y^{(l+1)}_{ix} > 0}\right] \mathbb{E}\left[\left(w^{(l+1)}_{i h_0 q}\right)^2\right] \psi^2_{q,x} \\ &= \sum_i \sum_x \sum_q \frac{1}{2} \operatorname{Var}\left[\frac{\partial \mathcal{E}}{\partial \zeta^{(l+1)}_{ix}}\right] \operatorname{Var}\left[w^{(l+1)}_{i h_0 q}\right] \psi^2_{q,x}.\end{aligned}$$

The factor $\frac{1}{2}$ in the last line goes back to the symmetric distribution of $y^{(l)}$ together with the indicator function. Using the fact that the weights' variances are initialized to only depend on $q$ and the assumption on identically distributed gradients both can be pulled out of the sums:

$$\operatorname{Var}\left[\frac{\partial \mathcal{E}}{\partial \zeta^{(l)}}\right] = \operatorname{Var}\left[\frac{\partial \mathcal{E}}{\partial \zeta^{(l+1)}}\right] \frac{I}{2} \sum_q \operatorname{Var}\left[w^{(l+1)}_q\right] \left\|\psi_q\right\|_2^2.$$

It seems reasonable to allocate the contribution to the overall variance equally over the $Q$ summands. Demanding the gradients' variance not to be amplified while propagating through the network then leads to the initialization condition

$$\mathrm{Var}\left[w_q\right] = \frac{2}{IQ\left\|\psi_q\right\|_2^2}. \tag{6.4}$$

The calculation for the forward pass is similar to the case of backpropagation but considers the variance $\mathrm{Var}\left[y^{(l)}\right]$ of pre-nonlinearity activations instead of gradients. An exact calculation depends on the expectation value $\mathbb{E}\left[\zeta^{(l-1)}\right]$, which is certainly different from zero because of the ReLU nonlinearity but otherwise unknown. We can, however, give an approximate result by exploiting the central limit theorem. To this end, we note that the pre-nonlinearity activations (6.2) are summed up from $H\sum_q |\operatorname{supp}\psi_q|$ independent terms of finite variance which is a relatively large number in typical networks. This allows to approximate the variance by the asymptotic result implied by the central limit theorem:

$$\begin{aligned}
\mathrm{Var}\left[y_{ix}^{(l)}\right] &= \mathrm{Var}\left[\sum_h \sum_{x'} \sum_q \zeta_{h,x-x'}^{(l-1)} w_{ihq}^{(l)} \psi_{qx'}\right] \\
&\overset{\text{(CLT)}}{\approx} \sum_h \sum_{x'} \sum_q \mathrm{Var}\left[\zeta_{h,x-x'}^{(l-1)} w_{ihq}^{(l)} \psi_{qx'}\right] \\
&= \sum_h \sum_{x'} \sum_q \mathbb{E}\left[\left(\zeta_{h,x-x'}^{(l-1)}\right)^2\right] \mathbb{E}\left[\left(w_{ihq}^{(l)}\right)^2\right] \psi_{qx'}^2.
\end{aligned}$$

In the last step we made use of the independence of the weights from the previous layers' feature maps and $\mathbb{E}[w] = 0$. The symmetric distribution of weights leads to a symmetric distribution of pre-nonlinearity activations which in conjunction with ReLU nonlinearities implies $\mathbb{E}[\zeta^2] = \frac{1}{2}\mathrm{Var}[y]$. To see this, note that the symmetry of the distribution of pre-nonlinearity activation leads on the one hand to

$$\mathrm{Var}[y] \;=\; \mathbb{E}[y^2] \;=\; \int_{\mathbb{R}} \tilde{y}^2\, p_y(\tilde{y})\,\mathrm{d}\tilde{y} \;=\; 2\int_{\mathbb{R}^+} \tilde{y}^2\, p_y(\tilde{y})\,\mathrm{d}\tilde{y}$$

and on the other hand to

$$\begin{aligned}
\mathbb{E}[\zeta^2] &= \int_{\mathbb{R}} \tilde{\zeta}^2\, p_\zeta(\tilde{\zeta})\,\mathrm{d}\tilde{\zeta} \\
&= \int_{\mathbb{R}} \tilde{\zeta}^2 \left(\frac{1}{2}\delta(0) + \Theta(\tilde{\zeta}) p_y(\tilde{\zeta})\right) \mathrm{d}\tilde{\zeta} \\
&= \int_{\mathbb{R}^+} \tilde{\zeta}^2\, p_y(\tilde{\zeta})\,\mathrm{d}\tilde{\zeta}\,,
\end{aligned}$$

where $\delta$ denotes the delta distribution and $\Theta$ is the Heavyside step function. As before, we drop all indices which the random variables are independent from to compute the sums. This leads to

$$\mathrm{Var}\left[y^{(l)}\right] \approx \mathrm{Var}\left[y^{(l-1)}\right] \frac{H}{2} \sum_q \mathrm{Var}\left[w_q^{(l)}\right] \left\|\psi_q\right\|_2^2,$$

which in turn suggests a weight initialization according to

$$\mathrm{Var}\left[w_q\right] = \frac{2}{HQ \left\|\psi_q\right\|_2^2} \tag{6.5}$$

to ensure that the activations' variances are not amplified.

The results (6.4) and (6.5) differ by the factor of the number of output- or input-feature maps of a layer. As discussed by He et al. [2015] this is not a big problem because these factors cancel out for all intermediate layers. Our initialization scheme recovers the results obtained earlier by He et al. [2015], which can be seen by inserting a Dirac filter basis which is effectively used in conventional CNNs.

We next note that the filters are combined of products $w_q^{(l)} \psi_q^{(l)}$ which implies that the factor $\left\|\psi_q\right\|_2^2$ in (6.4) and (6.5) counterbalances different energies of the basis filters. A convenient way to initialize the network is hence to normalize all filters to unit norm and subsequently initialize the weights uniformly by either $\mathrm{Var}\left[w_q\right] = \frac{2}{IQ}$ or $\mathrm{Var}\left[w_q\right] = \frac{2}{HQ}$.

An additional complication arises in our network construction where steerability is only preserved when the relative amplitude of the circular harmonics' real and imaginary parts is not changed. While these steerable filter pairs have equal norms in continuous space this is not necessarily true for their sampled counterparts which rules out an independent normalization. As a steerability consistent way of normalizing conjugate filter pairs we propose to adequately normalize the corresponding complex filters from which they originate. The proper scale follows from $\|\psi\|_2 = \sqrt{\|\mathrm{Re}[\psi]\|^2 + \|\mathrm{Im}[\psi]\|^2}$ for $\psi \in \mathbb{C}$ to be $\|\psi\|_2 = 1$ for DC filters whose imaginary part vanishes and $\|\psi\|_2 = \sqrt{2}$ for non-DC filters.

We emphasize that using normalization layers like batch normalization does not obviate the need for a proper weight initialization. This is because such layers only scale the activations as a whole while equations (6.4) and (6.5) indicate that the relative scale of the summands contributing to each activation matter.

# 7 Prior and related work

A priori knowledge about transformation-invariance of images can be exploited in manifold ways. In this chapter we review and analyze different approaches leveraging such knowledge and put them in comparison to the proposed Steerable Filter networks.

**Data augmentation:** A commonly utilized technique is data augmentation. The basic idea of this approach is to enrich the training set by its transformed samples. For example, Krizhevsky et al. [2012] augment by adding horizontal flips, Dieleman et al. [2015] add rotated versions of the image, and in Laptev et al. [2016] data is augmented by affine transformations. To analyze data augmentation from a group theoretic standpoint consider again a dataset $\{(f_i, l_i)\}_{i=1}^N$ consisting of $N$ images $f_i \in \mathcal{F}$ and corresponding labels $l_i$ as introduced in section 3.1. We further assume the transformation group $G$ acting on the images to augment the dataset to be given. Each image $f_i$ is then transformed into random samples from its orbit $G.f_i$ which enlarges the dataset by a factor depending on the specific group. This allows to train larger models and is easily applicable without modifying the network architectures. On the flipside the equivariance to the transformation is not guaranteed but needs to be learned explicitly by the network. In contrast to the proposed equivariant/invariant network the hypothesis space is not restricted: instead of learning a mapping from the quotient space $\mathcal{F}/G$ which assigns labels to whole orbits, the decision boundaries need to explicitly model the structure of the orbits. This demands a high learning capacity which makes the network prone to overfitting.

**General approaches and transformations:** More recent work thus focuses on incorporating equivariances to various transformations directly into the network's architecture. One way to do this is to make each of the network's layers individually invariant to local transformations in a similar way as in our orientation-pooling architecture. For example, Kanazawa et al. [2014] expand the input of each layer over a Gauss pyramid, convolve each level with the same filters and pool the resulting activation over the scales. This way they end up with a locally scale invariance network. An example from the field of speech representation is the work of Zhang et al. [2015] who propose an architecture that is invariant to vocal tract length perturbations. Sohn and Lee [2012] build a transformation invariant restricted Boltzmann machine by applying several transformations on the inputs and restricting the hidden units to choose the one minimizing the energy. Similarly to the orientation-pooling architecture,

these approaches loose the information of the relative pose of the features associated with the transformation which may lead to suboptimal results.

Instead of a full pooling over the whole transformation group after each layer it is possible to learn to which part of the group a layer should be invariant. An example is the work of Gens and Domingos [2014] who deal with general transformations leading to feature maps defined in symmetry spaces of arbitrary dimensionality. By endowing the pooling process in symmetry space with additional structure they manage to restrict to these transformations only which are sensible in regard to the task at hand.

For the case of invariance w.r.t. Abelian symmetry groups Henriques and Vedaldi [2016] devise warped convolutions which resample the input according to the given symmetry to apply standard convolution operations on the warped images. This approach is bound to abelian symmetry groups since the standard convolutions on the warped space correspond to the group convolution of the $N$-dimensional translation group which is abelian by construction. An example application is the resampling on a log-polar grid which leads to joint equivariance under rotations and dilations. It is, however, not possible to extend the symmetry group by translations since these do not commute with rotations. The Spatial Transformer networks of Jaderberg et al. [2015] introduce trainable modules estimating transformation parameters determining a sampling grid with which their input is resampled. The set of transformations accessible this way includes global rotations of the resampled area. In this setting there is no guarantee to obtain an equivariant representation.

**Rotation-equivariant CNNs:** In particular, there has recently been a considerable interest in rotation-equivariant CNNs. The work of Dieleman et al. [2016] introduces four operations which are easily included into existing networks and enrich both the batch- and feature dimension with transformed versions of their content. In [Cohen and Welling, 2016], the feature maps resulting from transformed filters are treated as functions of the corresponding symmetry-group which allows to use group-convolutional layers. Both approaches are mathematically equivalent[1] but the transformation of filters is computationally and memory wise cheaper than transforming feature maps. As the computational cost of group convolutions is coupled to the size of the group, Cohen and Welling [2017] propose an alternative way to obtain an equivariant mapping based on steerable representations. The filterbanks applied to these representations are expanded in a basis of intertwiners which are equivariant linear maps between representations. This setup includes group convolutions as a special case for "regular representations" but is capable to work with other, more resource-economic representations. Besides translations and rotations, the aforementioned works also incorporate reflections, i.e. they operate on the dihedral group. All of them are currently only implemented for rotations

[1] Personal correspondence with Taco Cohen.

by multiples of $\pi/2$, i.e. for four orientations.

In [Laptev et al., 2016], several rotated versions of the same image are sent through a conventional CNN. The resulting features are subsequently pooled over the orientation dimension before being classified by a fully connected network. The approach can easily be extended to other transformations. On the downside, the equivariance is only w.r.t. global transformations. This demands the authors to do segmentation of images with locally rotated patterns in a patch-wise manner. By contrast, our network applies filters which are locally rotated in several orientations and is therefore able to deal with locally rotated patterns.

Marcos et al. [2016b] also perform convolutions with rotated versions of a each filter followed by a global pooling over orientations which leads to an invariant architecture. Their architecture consists, however, only of one convolutional layer. In [Marcos et al., 2016a] these ideas were extended to deep networks which additionally propagate the orientation of the maximum response. The magnitude and orientation of these responses are interpreted as vectors in polar coordinates such that the feature maps can be thought of as vector fields. Subsequent layers therefore apply filters which themselves are defined as vector fields. In contrast to our approach, the parameter cost does not grow with the sampled orientations. One can view this approach as being intermediate between our orientation pooling architecture which pools over orientations and looses the full information on the maximum responses orientation and our group-convolutional architecture which keeps all information on any orientation. In both approaches the authors base the filter rotation on bicubic interpolation, which allows for fine resolutions with respect to the orientation but causes artifacts.

Worrall et al. [2016] achieve continuous resolution in orientations by working with complex valued steerable filters and feature maps. However, this requires the angular frequencies of the feature maps to be disentangled which results in severe constraints on the network topology and filters.

Rotation-equivariant feature extraction can also be achieved by using group-convolutional scattering transforms [Sifre and Mallat, 2013]. A fundamental difference to our work is that the filter banks are fixed rather than learned and are not steerable.

# 8 Experiments

We evaluate the proposed network architectures on two datasets exhibiting rotational symmetries: The rotated MNIST dataset and the ISBI 2012 EM segmentation challenge. On the rotated MNIST dataset we first investigate specific network properties like its accuracy for different numbers of sampled orientations and sizes of the training dataset as well as the quality of the approximate equivariance for discretely sampled images. We further compare the rotational generalization capabilities of conventional CNNs and the proposed group-convolutional architecture for different data augmentation techniques. The gains of the proposed weight initialization scheme are evaluated in comparison to the standard He-initialization. With the insights gained in these experiments we train final orientation-pooling and group-convolutional networks on mnist-rot. For the ISBI 2012 challenge we build a U-Net model to investigate the segmentation capabilities of the proposed layer design. On both datasets we improve significantly upon previous state-of-the-art results.

## 8.1 Rotated MNIST

A standard benchmark for rotation-equivariant/invariant models which is evaluated in most papers in the field is the rotated MNIST dataset[1] (mnist-rot). The dataset contains the handwritten digits of the classical MNIST dataset, rotated to random orientations in $[0, 2\pi)$. It is split in 12000 training and 50000 test images; model selection is done by training on 10000 images and validating on the 2000 remaining samples in the training set. Figure 8.1 shows a few images from the the dataset. Note that the intra-class variability of mnist-rot is higher than in the original MNIST dataset while simultaneously the size of the training set is substantially smaller than the 60000 training samples provided by MNIST.

For our initial experiments on the dependence on sampled orientations and the networks' generalization capabilities we utilize the group-convolutional architecture given in Table 8.1 as baseline. It consists of one steerable input layer which maps the input images to the group, five following group-convolutional layers and three fully connected layers. After every two steerable filter layers we perform a spatial $2 \times 2$ max-pooling. The orientation dimension and

[1] http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/MnistVariations (accessed Sep. 27, 2017).

Figure 8.1: Samples of the rotated MNIST dataset with labels 6, 5, 2, 3, 9, 0, 6, 1, 8. Note the visual ambiguity between sixes and nines.

| Operation | Filter size | Feature channels $I$ |
|---|---|---|
| Steerable input layer | $7 \times 7$ | 16 |
| Steerable group convolution | $5 \times 5$ | 24 |
| Spatial max pooling | $2 \times 2$ | |
| Steerable group convolution | $5 \times 5$ | 32 |
| Steerable group convolution | $5 \times 5$ | 32 |
| Spatial max pooling | $2 \times 2$ | |
| Steerable group convolution | $5 \times 5$ | 48 |
| Steerable group convolution | $5 \times 5$ | 64 |
| Global spatial pooling | | |
| Global orientation pooling | | |
| Fully connected | | 64 |
| Fully connected | | 64 |
| Fully connected + Softmax | | 10 |

Table 8.1: Architecture of the group-convolutional Steerable Filter CNN used in the experiments on the resolution of sampled orientations and on the rotational generalization capabilities. The number of channels denotes the number $I$ of independent filters. Each of these feature channels carries $\Lambda$ orientation dependent responses.

the remaining spatial dimensions are pooled out globally after the last convolutional layer. The number of feature channels stated in the tables refers to the number of learned filters $I$ of the corresponding layer. As these filters are themselves applied with respect to $\Lambda$ orientations we end up with $I\Lambda$ responses, e.g. $16 \cdot 16 = 256$ effective responses in the first layer. Note that the extraction of this comparatively large number of responses is possible without overfitting because the rotational weight sharing leads to an increased parameter utilization (in the sense of [Cohen and Welling, 2017]) by a factor of $\Lambda$. We trained the networks using standard techniques: Adam optimizer [Kingma and Ba, 2015], batch normalization [Ioffe and Szegedy, 2015], dropout with probability 0.3 in the fully connected layers [Srivastava et al., 2014] and a mild elastic net regularization. The batch normalization on the group does not interfere with the equivariance when the responses are normalized by averaging over both spatial and orientation dimensions. We set the initial learning rate to 0.015 and decayed it exponentially with a rate of 0.8 per epoch starting from epoch 15.

The number of sampled orientations $\Lambda$ is a parameter specific to our network, so we explore its influence on the accuracy. We are further interested in the network's sample complexity,

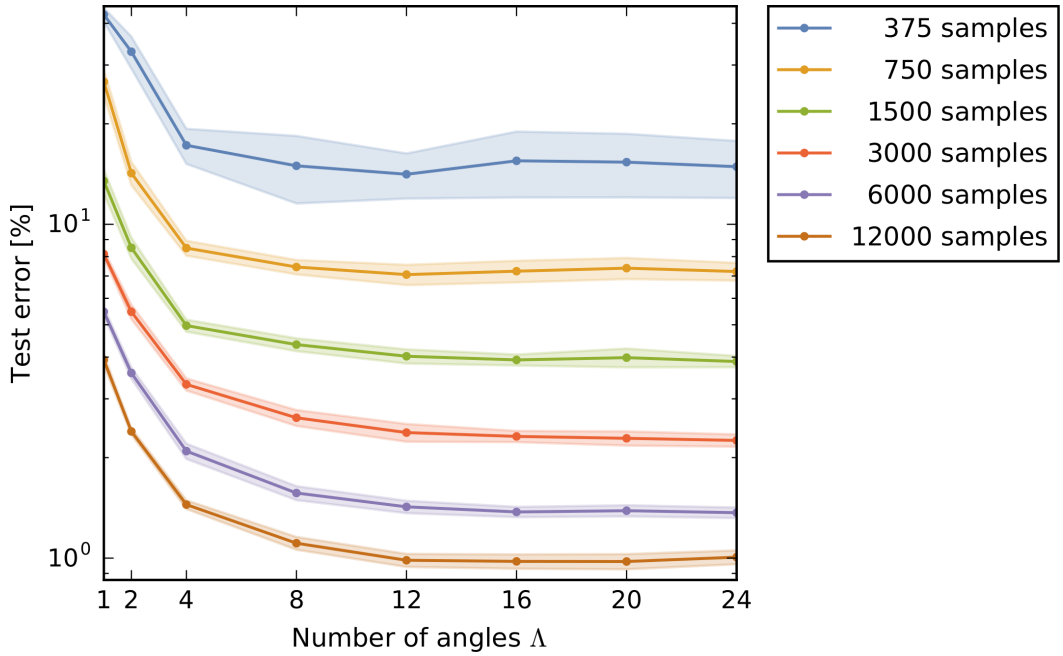
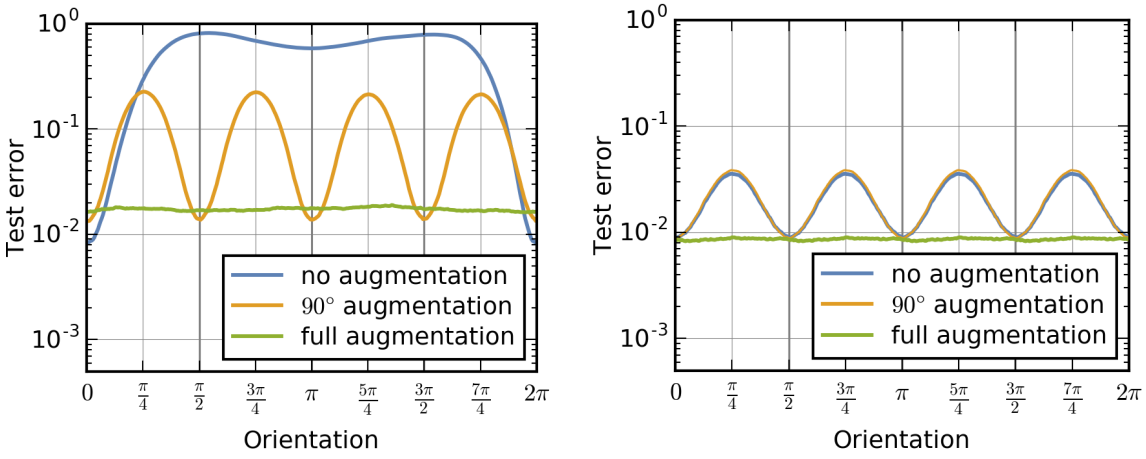



Figure 8.2: Test error versus number of sampled filter orientations for different training subsets from mnist-rot. Shaded regions highlight the standard deviations over several runs. The accuracy improves significantly with increasing angular resolution until it saturates at around 12 to 16 orientations for all training set sizes.

i.e. the dependence on the size of the training set. The results are reported in Figure 8.2. All data points are averaged over 12 runs with different seeds. As expected, the test error and its standard deviation decrease with the size of the training data set (note the logarithmic ordinate). We observe that the accuracy improves significantly when increasing the number of orientations until it saturates at around 12 to 16 angles, independently of the size of the training set. This saturation can be explained by the fact that the higher angular resolution counterbalances with an increased demand for parameters for larger groups. Up to this point, the gain of adding more sampled orientations is considerable. For example, in almost all cases, increasing the angular resolution from 2 to 4 sampled orientations provides a higher gain in accuracy than sticking with 2 orientations and doubling the number of training samples. We want to emphasize that the possibility of going beyond the four sampled orientations of [Dieleman et al., 2016, Cohen and Welling, 2016, 2017] leads to a significant gain in accuracy.

In order to test how well the networks generalize learned patterns over orientations we conduct an experiment where we train the networks on unrotated digits and record the accuracy over the orientation of rotated digits. Specifically, we take the the first 12000 samples of the conventional MNIST dataset as training samples to train a conventional CNN as well as a group-convolutional CNN with $\Lambda = 16$ using either no augmentation, augmentation by rota-



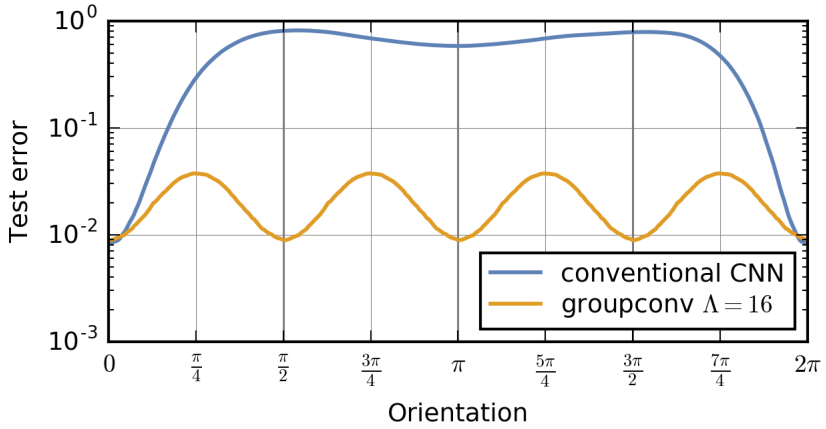

(a) Conventional CNN

(b) Group-convolutional network

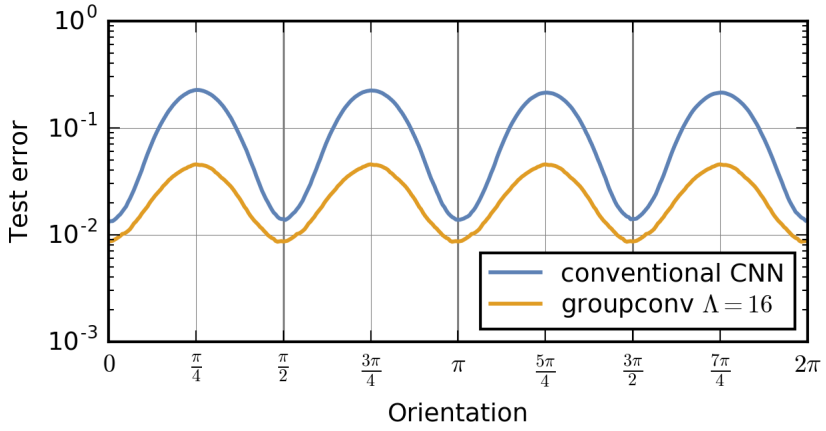
Figure 8.3: Rotational generalization capabilities of a conventional CNN and a group-convolutional network using different data augmentation strategies. In this experiment we train on unrotated MNIST digits and evaluate the test error depending on the orientation of the digits in a test dataset.

tions which are multiples of $\frac{\pi}{2}$ or augmentation by continuous rotations[2]. As test set we take the remaining 58000 samples and record the test errors' dependence on the orientation of this dataset. To obtain a fair comparison between the networks we experiment with CNNs with the same number of parameters or the same number of channels like the group-convolutional network and tune their hyperparameters independently from the group-convolutional network. Since both conventional CNNs show the same behavior we only report the accuracies of the network with the same number of channels which performs slightly better. The results of the experiment are plotted in Figure 8.3. One can see that, lacking rotational equivariance, the conventional CNN does not generalize well over orientations. While obtaining a high accuracy for small rotations of the test set, the accuracy drops rapidly to almost random guessing for larger angles. The accuracy increases slightly for rotations of 180 degrees since some of the numbers, for example the digits 0 or 8, are approximately invariant under such rotations. When augmenting the training samples by rotations by multiples of $\frac{\pi}{2}$ the average accuracy over orientations increases significantly. As to be expected by the symmetrization induced by the augmentation the test error becomes $\frac{\pi}{2}$-periodic. However, the accuracy still oscillates strongly between 98.5% and about 80%. When augmenting with orientations which are densely sampled from $[0, 2\pi)$, the accuracy becomes approximately constant over all orientations. The downside of both augmentation techniques is that the detection performance drops

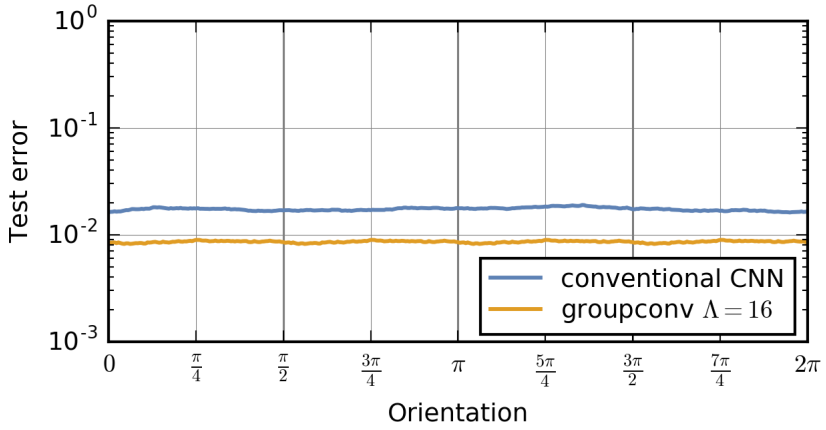
[2]The augmentation by continuous orientations as well as the rotation of the test dataset are done by bicubic interpolation.




(a) No data augmentation.




(b) Augmentation with rotations by multiples of $\frac{\pi}{2}$.




(c) Augmentation with densely sampled orientations.

Figure 8.4: Direct comparison of the rotational generalization capabilities of a CNN and a group-convolutional network. While augmentation diminishes the gap between both approaches the enhanced sample complexity of the equivariant network still leads to a significantly better result.

| Operation | Filter Size | Feature Channels $I$ |
|---|---|---|
| Steerable input layer | $9 \times 9$ | 24 |
| Steerable group convolution | $7 \times 7$ | 32 |
| Spatial max pooling | $2 \times 2$ | |
| Steerable group convolution | $7 \times 7$ | 36 |
| Steerable group convolution | $7 \times 7$ | 36 |
| Spatial max pooling | $2 \times 2$ | |
| Steerable group convolution | $7 \times 7$ | 64 |
| Steerable group convolution | $5 \times 5$ | 96 |
| Global spatial pooling | | |
| Global orientation pooling | | |
| Fully connected | | 96 |
| Fully connected | | 96 |
| Fully connected + Softmax | | 10 |

Table 8.2: Architecture of the group-convolutional Steerable Filter CNN used with $\Lambda = 16$ sampled orientations in the final experiments on mnist-rot.

for small angles. This is the case because the network needs to learn to detect the augmented samples additionally which demands an increased learning capacity. The group-convolutional network on the other hand generalizes way better over orientations. Even without augmentation the accuracy does not drop below 96%. In continuous space we would expect the test error curve to be $\frac{2\pi}{\Lambda}$-periodic because of the rotational equivariance. The deviations from this behavior can be attributed to the sampling effects of using digitized images. Augmentation with $\frac{\pi}{2}$-rotations does not influence the accuracy of the group-convolutional network since we chose $\Lambda = 16$ being a multiple of the 4 augmented orientations such that the augmented samples lie on the orbit on which the network is invariant anyway. When choosing the number of sampled orientations not to be a multiple of 4 this is not longer the case and the network accuracy should increase. Augmentation by continuous rotation angles again renders the test error approximately constant over orientations. Note that we do not observe a drop in accuracy when using augmentation in a group-convolutional network. This implies that the cost of learning rotated versions of each digit is negligible for the approximately rotation-equivariant architecture. The augmentation by continuous rotations mainly acts as a regularization preventing the filters to overfit on the pixel grid. Figure 8.4 shows a different arrangement of the plots which emphasizes the difference of the two networks' accuracies for the same augmentation technique. In all cases the rotation-equivariant networks perform significantly better than the conventional CNNs because of the weight sharing and reduced parameter cost.

Based on the results of these experiments we keep the number of sampled orientations fixed to $\Lambda = 16$ and tune the network architecture further. We achieved the best results using

| Network | Test Error (%) |
|---|---|
| Larochelle et al. [2007] | 10.4 ± 0.27 |
| Sohn and Lee [2012] | 4.2 |
| Schmidt and Roth [2012] | 4.0 |
| Cohen and Welling [2016] – (O-pool) | 3.21 ± 0.0012 |
| Cohen and Welling [2016] – (G-conv) | 2.28 ± 0.0004 |
| Worrall et al. [2016] | 1.69 |
| **Ours** – (O-pool + CoeffInit) | 1.293 ± 0.028 |
| Laptev et al. [2016] | 1.2 |
| Marcos et al. [2016a] | 1.09 |
| Marcos et al. [2016a] – (test time augmentation) | 1.01 |
| **Ours** – (G-conv + HeInit) | 0.957 ± 0.025 |
| **Ours** – (G-conv + CoeffInit) | 0.880 ± 0.029 |
| **Ours** – (G-conv + CoeffInit + data augmentation) | **0.714** ± **0.022** |

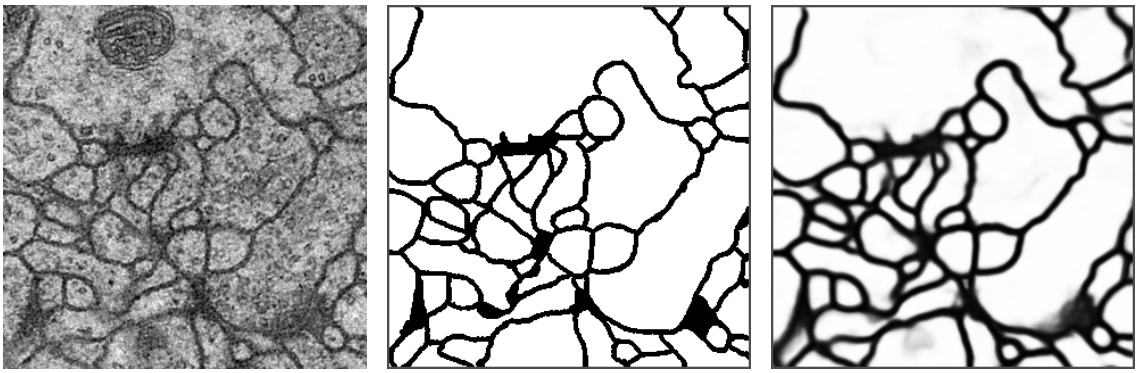
Table 8.3: Test errors on the rotated MNIST dataset.

the slightly larger network given in Table 8.2. In particular, we find that increasing the size of the filter masks improves the results which are reported in Table 8.3. Using He's weight initialization and no data augmentation, we obtain a test error of 0.957% which already exceeds the previous state-of-the-art. The adapted initialization scheme for the filter coefficients significantly improves the test error to 0.880%. When additionally augmenting the dataset with continuous rotations during training time we are able to decrease the error further to 0.714%. To summarize, our approach reduces the best previously published error by a factor of 29%.

In order to evaluate the benefit of using the group-convolutional network rather than the orientation-pooling architecture we ran a further experiment where we pooled over the orientation dimension after every layer. As discussed in section 4.1 this leads to layer-wise local rotation-invariance which is inherently accompanied with the loss of information on the relative orientation of features. The resulting test error of 1.293% reflects this issue and substantiates the gain of utilizing group convolutions.

## 8.2 ISBI 2012 2D EM segmentation challenge

In a second experiment we evaluate the performance of our model on the ISBI 2012 electron microscopy segmentation challenge [Arganda-Carreras et al., 2015]. Goal of the challenge is to predict the locations of the cell boundaries in the Drosophila ventral nerve cord from EM images which is a key step for investigating the connectome of the brain. The dataset consists



(a) Raw EM image.

(b) Groud truth membrane segmentation.

(c) Predicted membrane probability map.

Figure 8.5: Training sample of the ISBI 2012 EM segmentation challenge consisting of a raw EM image and the corresponding ground truth membrane segmentation mask. The shown patches are cropped from slice 30 of the training data set which we used for validation. Our network's prediction is shown on the right. It can be seen that the network learned to confidently detect and ignore vesicles (small dark bubbles) and mitochondria (large dark blob on the top) which is only possible with a large field of view extracting high level semantic information.

of 30 training and test slices of size $512 \times 512$ px with a binary segmentation ground truth provided for the training set. Figure 8.5 shows an exemplary raw EM image from the training set with the corresponding ground truth segmentation mask. An important property of the dataset is that the images have no preferred orientation which makes it suitable for evaluating rotation-equivariant networks.

We build on an established pipeline introduced in [Beier et al., 2017] where a crucial step is the boundary prediction via a conventional CNN. In the present experiment, we replace their network by a Steerable Filter CNN consisting of group convolution layers arranged in a U-net fashion [Ronneberger et al., 2015]. The network architecture is visualized in Figure 8.6. It is build as a symmetric encoder-decoder network with additional skip-connections between stages of the same resolution. This allows to extract semantic information from a large field of view while at the same time preserving precise spatial localization. We further adopt two modifications from [Quan et al., 2016]: we do not concatenate the skipped feature maps but add them to the decoder features upsampled from the previous stage, and we use intermediate residual blocks, here of depth 1. The images fed into the network are cropped regions of $256 \times 256$ pixels which are padded to $320 \times 320$ pixels by reflecting a region of 32 pixels around the border to alleviate boundary artifacts. After padding, the images are augmented by random elastic deformations, reflections and rotations by multiples of $\frac{\pi}{2}$. As discussed before, rotational augmentation is beneficial only when the network is not equivariant w.r.t. the additional orientations. We therefore sampled $\Lambda = 17$ orientations which is mutually prime with the four augmented orientations. This way the augmented images do not fall into a subgroup w.r.t.

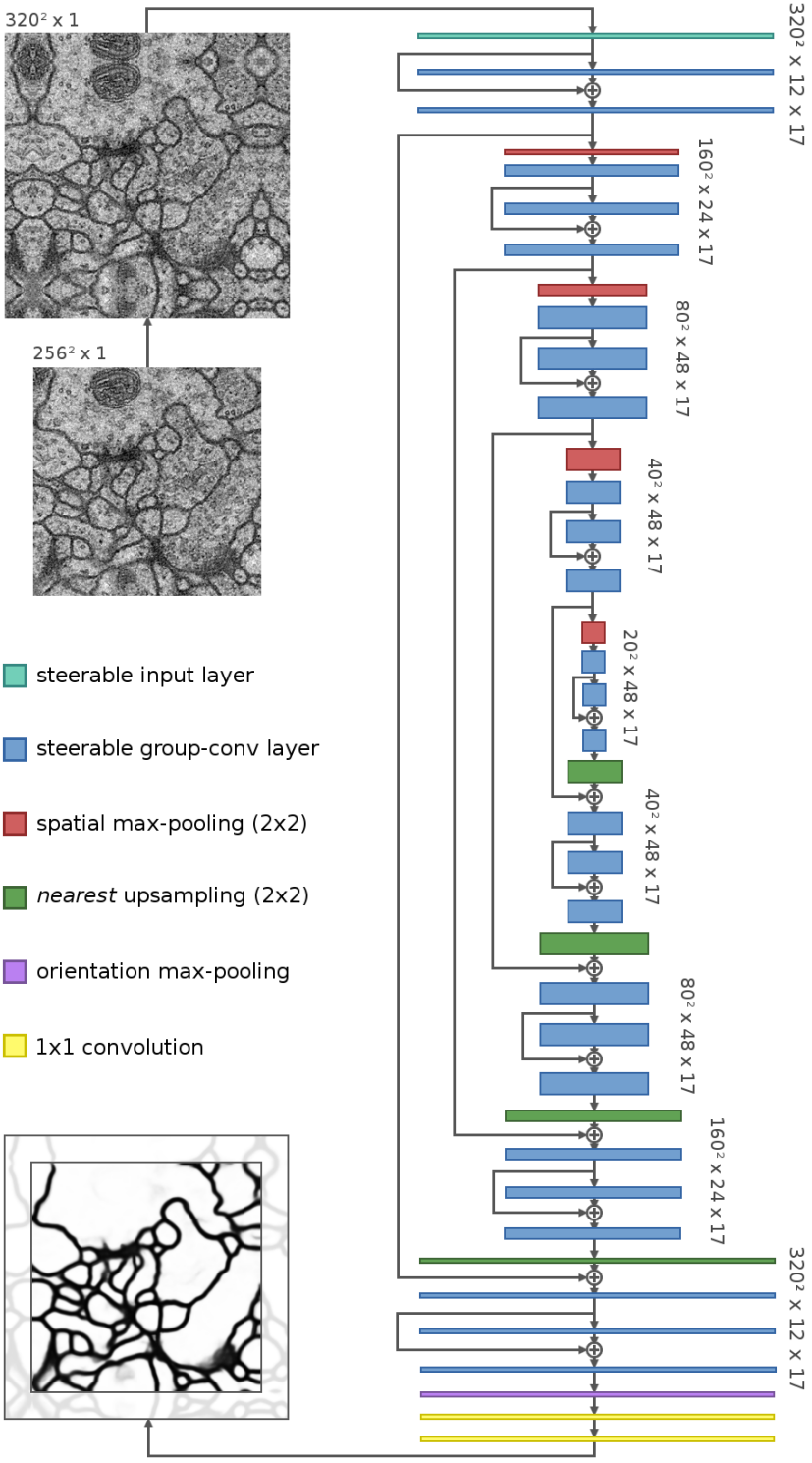



Figure 8.6: Network architecture used to predict the membrane probability map for the ISBI 2012 EM segmentation challenge. The topology is inspired by the U-Net [Ronneberger et al., 2015] and FusionNet [Quan et al., 2016] but uses the proposed steerable group convolution layers with $\Lambda = 17$ orientations. To mitigate boundary artifacts we feed reflect-padded images into the network.

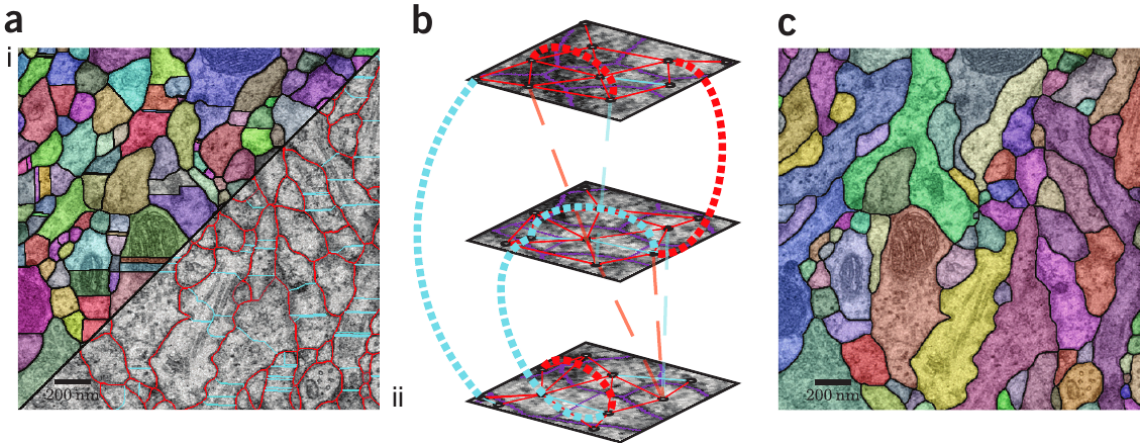



Figure 8.7: Visualization of the postprocessing pipeline introduced in Beier et al. [2017]. An oversegmentation is generated by running a watershed algorithm on the thresholded and distance-transformed probability maps generated by our Steerable Filter Network (**a**,i). The edges of the segments' region adjacency graph are weighted by attractive or repulsive potentials (**a**,ii). In the lifted version of the Multicut not only next neighbors but also longer range interactions are considered. We further run the 3d version of the algorithm which assembles information from different slices (**b**). Finally, the superpixels are merged by optimizing the lifted Multicut objective (**c**).
Image credit: Beier et al. Multicut brings automated neurite segmentation closer to human performance. Nature Methods, 14(2):101–102, 2017

which the network is equivariant/invariant and therefore give a different training signal in comparison to that of an unrotated image. No test time augmentation or model averaging is used. On the highest resolution level we learn $I = 12$ filters. Together with the orientation channels this corresponds to $I\Lambda = 204$ effective channels. The number of filters is doubled when going to the second and third level and is afterwards kept constant since we did not observe further gains in performance when adding more channels. After the decoder we max-pool over orientation channels to obtain locally rotation-invariant features and crop out a patch of $256 \times 256$ pixels centrally which correspond to the unpadded input. Two subsequent $1 \times 1$ convolution layers map these features pixel-wise to the desired probability map. All group-convolutional layers utilize kernels of size $7 \times 7$ pixel. The input layer applies $11 \times 11$ pixel kernels. The network is optimized by minimizing the spatially averaged binary cross-entropy loss between predictions and the ground truth segmentation masks. As on the rotated MNIST dataset we regularize the convolutional weights with an elastic net penalty with hyperparameters $\lambda_{L1} = \lambda_{L2}$ set to $10^{-7}$ for the steerable layers and to $10^{-8}$ for the $1 \times 1$ convolution layers respectively. Here we chose a higher dropout probability of $p = 0.4$ both in the steerable as well as in the $1 \times 1$ convolution layers. The learning rate is decayed exponentially by a factor of 0.85 per epoch starting from an initial rate of $5 \cdot 10^{-2}$.

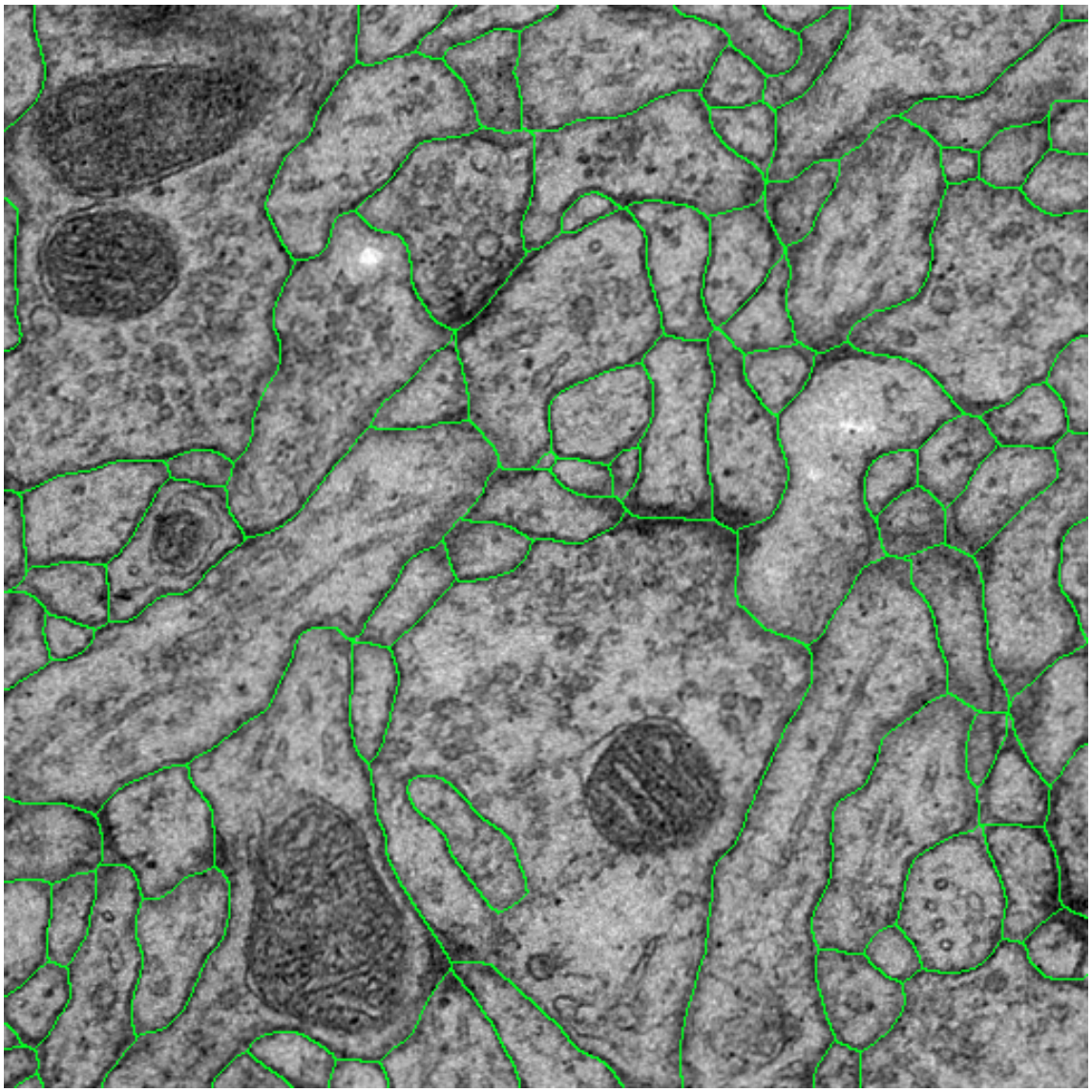


Figure 8.8: Final boundary predictions after the lifted Multicut postprocessing, overlaid over the raw data of test slice 27.

After predicting the membrane boundaries we run the postprocessing[3] introduced by Beier et al. [2017] whose steps are visualized in Figure 8.7. In a first step it generates an over-segmentation of both the samples in the training and the test set by applying a watershed transformation on the distance-transformed predictions. The resulting superpixels are then merged by a lifted Multicut algorithm. To do so, a region adjacency graph of the superpixels is build and a random forest predicts attractive or repulsive potentials on the edges of this

[3]The postprocessing of Beier et al. [2017] is used in all top-scoring competition entries.

| Group | $V^{\text{Rand}}$ | $V^{\text{Info}}$ |
|---|---|---|
| human values | 0.99785 | 0.99900 |
| **Ours** | **0.98680** | **0.99144** |
| HVCL@UNIST – Quan et al. [2016] | 0.98365 | 0.99130 |
| CASIA_MIRA | 0.98356 | 0.99063 |
| IAL MC/LMC – Beier et al. [2017] | 0.98261 | 0.98946 |
| PolyMtl – Drozdzal et al. [2016] | 0.98058 | 0.98816 |

Table 8.4: Top entries of the leaderboard of the ISBI 2012 EM segmentation challenge, accessed on October 5, 2017. Higher values mean better accuracy.

graph based on the local appearance. An ILP-solver then cuts edges of the graph to obtain minimal energy under the constraint that the cuts are globally consistent. This constraint ensures for example that two superpixels can not be cut directly while being connected over a third one. An advantage of using the Multicut algorithm is that it incorporates knowledge of the 3d structure instead of inferring isolated predictions for each EM slice alone. The resulting instance segmentation can then be transformed back to membrane labels which are our final predictions. An overlay of the final predictions on test slice 27 over the raw data is shown in Figure 8.8.

Segmentation predictions are evaluated by the challenge hosters and ranked according to the foreground-restricted Rand score $V^{\text{Rand}}$. The Rand score is a measure operating on instance labels which are extracted from the predicted boundary probability map by a thresholding followed by finding connected components. It is a statistics summarizing how often a pair of two pixels falls into the same or different segments in the ground truth and prediction. This measure is suitable for connectomics since it is highly sensible for holes in the predicted membranes which erroneously connect two different neurons while it tolerates small shifts of the boundaries. As secondary evaluation metric the information score $V^{\text{Info}}$ is used. For a detailed explanation of these metrics see [Arganda-Carreras et al., 2015]. The current leaderboard [4] in Table 8.4 shows that our approach significantly improves upon the state-of-the-art results, and in particular upon those of Beier et al. [2017].

[4] The full leaderboard is available at http://brainiac2.mit.edu/isbi_challenge/leaders-board-new.

# 9 Conclusions and Outlook

We have developed a convolutional network architecture which is equivariant under joint translations and rotations. It is suitable for isotropic datasets in which patterns do not occur in a predominant orientation. By sharing weights over filter rotations the network becomes more parameter-efficient than a conventional CNN while at the same time generalizing learned patterns over orientations. The learned filters in the network are designed such that they are steerable. This approach guarantees an exact rotation of the learned kernels which prevents interpolation artifacts and can as linear operation easily be implemented in existing deep learning frameworks. Layerwise equivariance is obtained either by pooling over orientations or by using group convolutions. In principle, the proposed network allows for an arbitrarily fine-grained angular resolution. The experimental results reveal that sampling more than four orientations improves the accuracy considerably. We have further empirically shown that the network indeed generalizes over orientations even for sampled signals where the equivariance is not exact. In comparison with a conventional CNN the proposed network performs significantly better, independently of the utilized data augmentation. A weight initialization scheme which is adapted to networks which learn expansion coefficients of general filter bases led to a further boost in accuracy. All in all, the proposed CNN significantly improves upon the latest published results on both mnist-rot and the ISBI 2012 2D EM segmentation challenge.

The proposed architecture can be extended in several respects. An interesting direction of research is to investigate how the choice of the atomic filter basis affects the results. It was shown by Jacobsen et al. [2016] that a smooth filter basis can have a regulatory effect and can improve the network accuracy, especially for small datasets. One could automatize the process of finding a suitable basis by parameterizing the atomic filter themselves and pruning over their expansion coefficients such that the network is biased to utilize fewer but smoother filters. A second extension which could readily be implemented is the generalization to the full Euclidean group E(2) which additionally incorporates filter reflections. Doing so would reduce the parameter cost further by a factor of two. Another symmetry which will certainly become an important tool for isotropic volumetric data is E(3) which rotates the filters in three dimensional space. The reduced parameter cost of equivariant networks allows to train larger models which consequently consume more memory than conventional CNNs with the same number of parameters. It is therefore necessary to investigate approaches to reduce the networks' memory footprint. Cohen and Welling [2017] proposed an equivariant architec-

ture which alleviates the memory issues. This network operates with four filter orientations but could be combined with our steerable filter approach to achieve a higher resolution w.r.t. sampled orientations.

# Appendix

# A Fourier space implementation

Here we will give a formulation of the proposed group-convolutional steerable filter layers in Fourier space. The mathematical structure of the layers allows to take advantage of some tricks which reduce the computational complexity of the calculation. Here we denote a Fourier transform over index $x$ by $\mathcal{F}_{(x)}$ and switch to a notation where spatial and orientation coordinates are explicitly written as tensor indices. The pre-nonlinearity responses can then be written as

$$
\begin{aligned}
& y_{i\lambda x_1 x_2} \\
=& \mathrm{Re} \sum_h \sum_q \sum_{\hat{\lambda}} \sum_{\hat{x}_1, \hat{x}_2} \zeta_{h\hat{\lambda}\hat{x}_1\hat{x}_2} \, e^{-ik_q\hat{\theta}_\lambda} \, w_{ihq,\lambda-\hat{\lambda}} \, \psi_{q,x_1-\hat{x}_1,x_2-\hat{x}_2} \\
=& \mathrm{Re}\left( \mathcal{F}^{-1}_{(\lambda x_1 x_2)} \left[ \sum_h \sum_q \mathcal{F}_{(\lambda x_1 x_2)} \left[ \zeta_{h\lambda x_1 x_2} e^{-ik_q\theta_\lambda} \right] \odot \mathcal{F}_{(\lambda x_1 x_2)} \left[ w_{ihq\lambda} \, \psi_{q x_1 x_2} \right] \right] \right) \\
=& \mathrm{Re}\left( \mathcal{F}^{-1}_{(\lambda x_1 x_2)} \left[ \sum_h \sum_q \mathcal{F}_{(\lambda x_1 x_2)} \left[\zeta\right]_{h,(\omega_\lambda + k_q)\%\Lambda, \omega_x \omega_y} \odot \mathcal{F}_{(\lambda)} \left[w\right]_{ihq\omega_\lambda} \odot \mathcal{F}_{(xy)} \left[\psi\right]_{q\omega_x\omega_y} \right] \right),
\end{aligned}
$$

where $\odot$ stands for an element-wise multiplication and % is the modulo operator. In the last step we made use of the DFT shift theorem

$$
\mathcal{F}_{(\lambda)} \left[ \zeta_\lambda e^{-ik\theta_\lambda} \right] \;=\; \mathcal{F}_{(\lambda)} \left[\zeta\right]_{(\omega_\lambda + k)\%\Lambda} \quad \text{for} \quad \theta_\lambda = \frac{2\pi\lambda}{\Lambda}
$$

which allows to reduce the cost of the forward convolution of the input feature map by a factor of $Q$. This way the Fourier transform of the phase-modulated feature maps can be extracted from the Fourier transform of the non-modulated feature maps by a cyclic shift over the orientation index. Further, the Fourier transform of the fixed atomic filter basis can be precomputed. In principle it is possible to learn the weights directly in Fourier space which would save another transform. By Parseval's theorem

$$
\sum_{\lambda=0}^{\Lambda-1} |w_\lambda|^2 \;=\; \frac{1}{\Lambda} \sum_{\omega_\lambda=0}^{\Lambda-1} \left| \mathcal{F}_{(\lambda)} \left[w\right]_{\omega_\lambda} \right|^2
$$

one can see that a weight decay ($\ell_2$ regularization) of such weights in Fourier space is equivalent to a weight decay on the untransformed parameters. The complexity of this formulation scales like $\mathcal{O}\left((I+H)\Lambda X^2 \log(\Lambda X^2)\right)$ for the Fourier transform part and $\mathcal{O}\left(IHQ\Lambda X^2\right)$ for the linear operation in Fourier space. For smooth filter bases with $Q \leq s^2$ this computation may be more efficient than the two implementations in real space.

# B Steerability of the composed filters

The proposed architecture applies filters of the form

$$\tilde{\Psi}(x) = \sum_{j=1}^{J} \sum_{k=0}^{K_j} w_{jk} \psi_{jk}(x)$$

which are rotated by the steerability of the atoms:

$$\rho_\theta \tilde{\Psi}(x) = \sum_{j=1}^{J} \sum_{k=0}^{K_j} w_{jk} e^{-ik\theta} \psi_{jk}(x).$$

We show that $\tilde{\Psi}$ is equivalent to a steerable filter in the sense of Freeman and Adelson [1991], i.e. it satisfies the condition

$$\rho_\theta \tilde{\Psi}(x) = \sum_{r=0}^{R-1} \kappa_r(\theta) \rho_{\alpha_r} \tilde{\Psi}(x)$$

for appropriately chosen expansion coefficients $\kappa_r(\theta)$ and $R \geq K + 1$. To this end we demand the two expansions to be equal, i.e.

$$\begin{aligned}
\rho_\theta \tilde{\Psi}(x) &= \sum_{j=1}^{J} \sum_{k=0}^{K_j} w_{jk} \rho_\theta \psi_{jk}(x) \\
&= \sum_{j=1}^{J} \sum_{k=0}^{K_j} w_{jk} e^{-ik\theta} \psi_{jk}(x) \\
&\overset{!}{=} \sum_{r=0}^{R-1} \kappa_r(\theta) \sum_{j=1}^{J} \sum_{k=0}^{K_j} w_{jk} \rho_{\alpha_r} \psi_{jk}(x) \\
&= \sum_{j=1}^{J} \sum_{k=0}^{K_j} w_{jk} \sum_{r=0}^{R-1} \kappa_r(\theta) e^{-ik\alpha_r} \psi_{jk}(x).
\end{aligned}$$

The condition for steerability in the sense of Freeman and Adelson [1991] is thus that there is a solution of the system of linear equations

$$e^{-ik\theta} = \sum_{r=0}^{R-1} \kappa_r(\theta) e^{-ik\alpha_r}$$

for all $k = 0, \ldots, K$ with $K = \max_j K_j$ and arbitrary rotation angles $\theta$, or, equivalently,

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ e^{-i\alpha_1} & e^{-i\alpha_2} & \cdots & e^{-i\alpha_R} \\ \vdots & \vdots & & \vdots \\ e^{-iK\alpha_1} & e^{-iK\alpha_2} & \cdots & e^{-iK\alpha_R} \end{pmatrix} \begin{pmatrix} \kappa_1(\theta) \\ \kappa_2(\theta) \\ \\ \vdots \\ \\ \kappa_{R-1}(\theta) \end{pmatrix} = \begin{pmatrix} 1 \\ e^{-i\theta} \\ \vdots \\ e^{-iK\theta} \end{pmatrix}.$$

Denoting the matrix by $A$ and the right hand side by $b(\theta)$ and using the unitarity $A^\dagger A = I$ we get

$$\kappa(\theta) = A^\dagger b(\theta).$$

Here we need $R \geq K+1$ distinct sampled angles $\alpha_r$ because otherwise the rank of $A^\dagger A$ is smaller than $K+1$.

# List of Figures

# List of Tables

# Bibliography


Ignacio Arganda-Carreras, Srinivas C. Turaga, Daniel R. Berger, Dan Cireşan, Alessandro Giusti, Luca M. Gambardella, Jürgen Schmidhuber, Dmitry Laptev, Sarvesh Dwivedi, Joachim M. Buhmann, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, 9, 2015.

Horace Barlow. Grandmother cells, symmetry, and invariance: how the term arose and what the facts suggest. *The Cognitive Neurosciences*, pages 309–320, 2009.

Thorsten Beier, Constantin Pape, Nasim Rahaman, Timo Prange, Stuart Berg, Davi D. Bock, Albert Cardona, Graham W. Knott, Stephen M. Plaza, Louis K. Scheffer, Ullrich Koethe, Anna Kreshuk, and Fred A. Hamprecht. Multicut brings automated neurite segmentation closer to human performance. *Nature Methods*, 14(2):101–102, 2017.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

Rolf Berndt. *Representations of linear groups: an introduction based on examples from physics and number theory*. Springer Science & Business Media, 2007.

Nathan Carter. *Visual group theory*. MAA, 2009.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, 2016.

Taco Cohen and Max Welling. Steerable CNNs. In *International Conference on Learning Representations (ICLR)*, 2017.

James J. DiCarlo, Davide Zoccolan, and Nicole C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

Sander Dieleman, Kyle Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2):1441–1459, 2015.

Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2016.

Michal Drozdzal, Eugene Vorontsov, Gabriel Chartrand, Samuel Kadoury, and Chris Pal. The importance of skip connections in biomedical image segmentation. In *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pages 179–187. Springer, 2016.

William Freeman and Edward Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

Robert Gens and Pedro Domingos. Deep symmetry networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2537–2545. Curran Associates, Inc., 2014.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

João Henriques and Andrea Vedaldi. Warped convolutions: Efficient invariance to spatial transformations. *Preprint arXiv:1609.04382*, 2016.

Yuan-Neng Hsu and H. Arsenault. Optical pattern recognition using circular harmonic expansion. *Applied Optics*, 21(22):4016–4019, 1982.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

Jorn-Henrik Jacobsen, Jan van Gemert, Zhongyu Lou, and Arnold W. M. Smeulders. Structured receptive fields in cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2610–2619, 2016.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2017–2025, 2015.

Angjoo Kanazawa, Abhishek Sharma, and David Jacobs. Locally scale-invariant convolutional neural networks. *Preprint arXiv:1412.5104*, 2014.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

Dmitry Laptev, Nikolay Savinov, Joachim Buhmann, and Marc Pollefeys. TI-POOLING: transformation-invariant pooling for feature learning in Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 289–297, 2016.

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning (ICML)*, pages 473–480, 2007.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. *arXiv:1612.09346*, 2016a.

Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. In *International Conference on Pattern Recognition (ICPR)*, 2016b.

James S. Milne. Group theory (v3.13), 2013. Available at www.jmilne.org/math/.

Tran Minh Quan, David G. C. Hilderbrand, and Won-Ki Jeong. Fusionnet: A deep fully residual convolutional neural network for image segmentation in connectomics. *arXiv preprint arXiv:1612.05360*, 2016.

R. Quian Quiroga, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005.

R. Quian Quiroga, Gabriel Kreiman, Christof Koch, and Itzhak Fried. Sparse but not 'grandmother-cell' coding in the medial temporal lobe. *Trends in cognitive sciences*, 12(3): 87–91, 2008.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

Joseph Rosen and Joseph Shamir. Circular harmonic phase filters for efficient rotation-invariant pattern recognition. *Applied Optics*, 27(14):2895–2899, 1988.

Uwe Schmidt and Stefan Roth. Learning rotation-aware features: From invariant priors to equivariant descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2050–2057, 2012.

Jean-Pierre Serre. *Linear representations of finite groups*, volume 42. Springer Science & Business Media, 2012.

Laurent Sifre and Stéphane Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1233–1240, 2013.

Kihyuk Sohn and Honglak Lee. Learning invariant representations with local transformations. In *International Conference on Machine Learning (ICML)*, pages 1311–1318, 2012.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Daniel Worrall, Stephan Garbin, Daniyar Turmukhambetov, and Gabriel Brostow. Harmonic networks: Deep translation and rotation equivariance. *Preprint arXiv:1612.04642*, 2016.

Malcolm P Young and Shigeru Yamane. Sparse population coding of faces in the inferotemporal cortex. *Science*, 256(5061):1327–1331, 1992.

Matthew Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

Chiyuan Zhang, Stephen Voinea, Georgios Evangelopoulos, Lorenzo Rosasco, and Tomaso Poggio. Discriminative template learning in group-convolutional networks for invariant speech representations. In *Annual Conference of the International Speech Communication Association*, 2015.

Erklärung:

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den (Datum) .....