

# Harmonic Networks: Deep Translation and Rotation Equivariance

Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov and Gabriel J. Brostow

{d.worrall, s.garbin, d.turmukhambetov, g.brostow}@cs.ucl.ac.uk

University College London\*

## Abstract

Translating or rotating an input image should not affect the results of many computer vision tasks. Convolutional neural networks (CNNs) are already translation equivariant: input image translations produce proportionate feature map translations. This is not the case for rotations. Global rotation equivariance is typically sought through data augmentation, but patch-wise equivariance is more difficult. We present Harmonic Networks or H-Nets, a CNN exhibiting equivariance to patch-wise translation and 360-rotation. We achieve this by replacing regular CNN filters with circular harmonics, returning a maximal response and orientation for every receptive field patch.

H-Nets use a rich, parameter-efficient and fixed computational complexity representation, and we show that deep feature maps within the network encode complicated rotational invariants. We demonstrate that our layers are general enough to be used in conjunction with the latest architectures and techniques, such as deep supervision and batch normalization. We also achieve state-of-the-art classification on rotated-MNIST, and competitive results on other benchmark challenges.

## 1. Introduction

We tackle the challenge of representing 360°-rotations in convolutional neural networks (CNNs) [19]. Currently, convolutional layers are constrained by design to map an image to a feature vector, and *translated* versions of the image map to proportionally-translated versions of the same feature vector [21] (ignoring edge effects)—see Figure 1. However, until now, if one *rotates* the CNN input, then the feature vectors do not necessarily rotate in a meaningful or easy to predict manner. The sought-after property, directly relating input transformations to feature vector transformations, is called *equivariance*.

A special case of equivariance is invariance, where feature vectors remain constant under all transformations of the input. This can be a desirable property globally for a model, such as a classifier, but we should be careful not to restrict all intermediate levels of processing to be transformation invariant. For example,

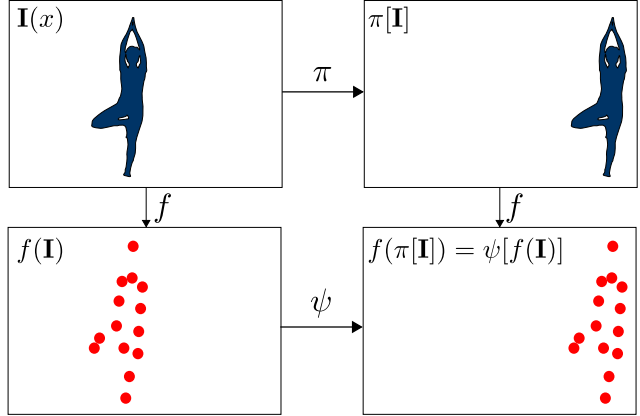


Figure 1. Patch-wise translation equivariance in CNNs arises from translational weight tying, so that a translation  $\pi$  of the input image  $\mathbf{I}$ , leads to a corresponding translation  $\psi$  of the feature maps  $f(\mathbf{I})$ , where  $\pi \neq \psi$  in general, due to pooling effects. However, for rotations, CNNs do not yet have a feature space transformation  $\psi$  ‘hard-baked’ into their structure, and it is complicated to discover what  $\psi$  may be, if it exists at all. Harmonic Networks have a hard-baked representation, which allows for easier interpretation of feature maps—see Figure 3.

consider detecting a deformable object, such as a butterfly. The pose of the wings is limited in range, and so there are only certain poses our detector should normally see. A transformation invariant detector, good at detecting wings, would detect them whether they were bigger, further apart, rotated, etc., and it would encode all these cases with the same representation. It would fail to notice nonsense situations, however, such as a butterfly with wings rotated past the usual range, because it has thrown that extra pose information away. An equivariant detector, on the other hand, does not dispose of local pose information, and so it hands on a richer and more useful representation to downstream processes. Equivariance conveys more information about an input to downstream processes, it also constrains the space of possible learned models to those that are valid under **the rules of natural image formation** [30]. This makes learning more reliable and helps with generalization. For instance, consider CNNs. The key insight is that the statistics of natural images, embodied in the correlations between pixels, are a) invariant to translation, and b) highly localized. Thus features at every layer in a CNN are computed on local receptive fields, where weights are shared

\*<http://visual.cs.ucl.ac.uk/pubs/harmonicNets/>

across translated receptive fields. This weight-tying serves both as a constraint on the translational structure of image statistics, and as an effective technique to reduce the number of learnable parameters—see Figure 1. In essence, translational equivariance has been ‘baked’ into the architecture of existing CNN models. We do the same for rotation and refer to it as *hard-baking*.

The current widely accepted practice to cope with rotation is to train with aggressive data augmentation [16]. This certainly improves generalization, but is not exact, fails to capture local equivariances, and does not ensure equivariance at every layer within a network. How to maintain the richness of local rotation information, is what we present in this paper. Another disadvantage of data augmentation is that it leads to the so-called *black-box* problem, where there is a lack of feature map interpretability. Indeed, **close inspection of first-layer weights in a CNN reveals that many of them are rotated, scaled, and translated copies of one another** [34]. Why waste computation learning all of these redundant weights?

In this paper, we present *Harmonic Networks*, or *H-Nets*. They design patch-wise  $360^\circ$ -rotational equivariance into deep image representations, by constraining the filters to the family of *circular harmonics*. The circular harmonics are *steerable filters* [7], which means that we can represent all rotated versions of a filter, using just a finite, linear combination of *steering bases*. This overcomes the issue of learning multiple filter copies in CNNs, guarantees rotational equivariance, and produces feature maps that transform predictably under input rotation.

## 2. Related Work

Multiple existing approaches seek to encode rotational equivariance into CNNs. Many of these follow a broad approach of introducing filter or feature map copies at different rotations. None has dominated as standard practice.

**Steerable filters** At the root of H-Nets lies the property of *filter steerability* [7]. Filters exhibiting steerability can be constructed at any rotation as a finite, linear combination of base filters. This removes the need to learn multiple filters at different rotations, and has the bonus of constant memory requirements. As such, H-Nets could be thought of as using an infinite bank of rotated filter copies. A work, which combines steerable filters with learning is [23]. They build shallow features from steerable filters, which are fed into a kernel SVM for object detection and rigid pose regression. H-Nets use the same filters with an added rotation offset term, so that filters in different layers can have orientation-selectivity relative to one another.

**Hard-baked transformations in CNNs** While H-Nets hard-bake patch-wise  $360^\circ$ -rotation into the feature representation, numerous related works have encoded equivariance to discrete rotations. The following works can be grouped into those, which encode global equivariance versus patch-wise equivariance, and those which rotate filters versus feature maps.

[3] introduce equivariance to  $90^\circ$ -rotations and dihedral flips in CNNs by copying the transformed filters at different

rotation–flip combinations. More recently they generalized this theory to all group-structured transformations in [4], but they only demonstrated applications on finite groups—an extension to continuous transformations would require a treatment on anti-aliasing and bandlimiting. [24] use a larger number of rotations for texture classification and [26] also use many rotated handcrafted filter copies, opting not to learn the filters. To achieve equivariance to a greater number of rotations, these methods would need an infinite amount of computation. H-Nets achieve equivariance to all rotations, but with finite computation.

[6] feed in multiple rotated copies of the CNN input and fuse the output predictions. [17] do the same for a broader class of global image transformations, and propose a novel per-pixel pooling technique for output fusion. As discussed, these techniques lead to global equivariances only and do not produce interpretable feature maps. [5] go one step further and copy each feature map at four  $90^\circ$ -rotations. They propose 4 different equivariance preserving feature map transformations. Their CNN is similar to [3] in terms of what is being computed, but rotating feature maps instead of filters. A downside of this is that all inputs and feature maps have to be square; whereas, we can use any sized input.

**Learning generalized transformations** Others have tried to learn the transformations directly from the data. While this is an appealing idea, as we have said, for certain transformations it makes more sense to hard-bake these in for interpretability and reliability. [25] construct a higher-order Boltzmann machine, which learns tuples of transformed linear filters in input–output pairs. Although powerful, they have only shown this to work on shallow architectures. [9] introduced *capsules*, units of neurons designed to mimic the action of cortical columns. Capsules are designed to be invariant to complicated transformations of the input. **Their outputs are merged at the deepest layer, and so are only invariant to global transformation.** [22] present a method to regress equivariant feature detectors using an objective, which penalizes representations, which lie far from the equivariant manifold. Again, this only encourages global equivariance; although, this work could be adapted to encourage equivariance at every layer of a deep pipeline.

## 3. Problem analysis

Many computer vision systems strive to be view independent, such as object recognition, which is invariant to affine transformations, or boundary detection, which is equivariant to non-rigid deformations. H-Nets hard-bake  $360^\circ$ -rotation equivariance into their feature representation, by constraining the convolutional filters of a CNN to be from the family of circular harmonics. Below, we outline the formal definition of equivariance (Section 3.1), how the circular harmonics exhibit rotational equivariance (Section 3.2) and some properties of the circular harmonics, which we must heed for successful integration into the CNN framework (Section 3.2).

**Continuous domain feature maps** In deep learning we use

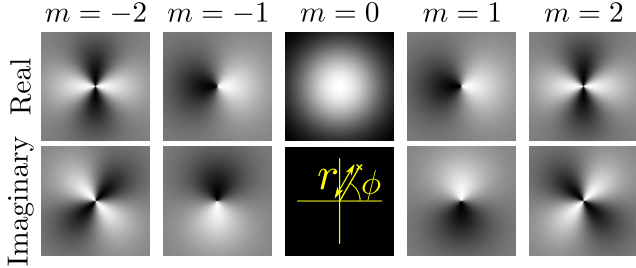


Figure 2. Real and imaginary parts of the complex Gaussian filter  $\mathbf{W}_m(r, \phi; e^{-r^2}, 0) = e^{-r^2} e^{im\phi}$ , for some rotation orders. As a simple example, we have set  $R(r) = e^{-r^2}$  and  $\beta = 0$ , but in general we learn these quantities. Cross-correlation, of a feature map of rotation order  $n$  with one of these filters of rotation order  $m$ , results in a feature map of rotation order  $m+n$ . Note the negative rotation order filters have flipped imaginary parts compared to the positive orders.

feature maps, which live in a discrete domain. We shall instead use continuous spaces, because the analysis is easier. Later on in Section 4.2 we shall demonstrate how to convert back to the discrete domain for practical implementation, but for now we work entirely in continuous Euclidean space.

### 3.1. Equivariance

Equivariance is a useful property to have because transformations  $\pi$  of the input produce predictable transformations  $\psi$  of the features, which are interpretable and can make learning easier. Formally, we say that feature mapping  $f: \mathcal{X} \rightarrow \mathcal{Y}$  is *equivariant* to a group of transformations if we can associate every transformation  $\pi \in \Pi$  of the input  $\mathbf{x} \in \mathcal{X}$  with a transformation  $\psi \in \Psi$  of the features; that is,

$$\psi[f(\mathbf{x})] = f(\pi[\mathbf{x}]). \quad (1)$$

This means that the order, in which we apply the feature mapping and the transformation is unimportant—they *commute*. An example is depicted in Figure 1, which shows that in CNNs the order of application of integer pixel-translations and the feature map are interchangeable. An important point of note is that  $\pi \neq \psi$  in general, so if we seek for  $\Pi$  to be rotations in the image domain, we do not require to find the set of  $f$ , such that  $\Psi$  “looks like” a rotation in feature space, rather we are searching for the set of  $f$ , such that there exists an *equivalent* class of transformations  $\Psi$  in feature space. A special case of equivariance is *invariance*, when  $\Psi = \{\mathbb{I}\}$ , the identity.

### 3.2. The Complex Circular Harmonics

With data augmentation CNNs may learn some rotation equivariance, but this is difficult to quantify [21]. H-Nets take the simpler approach of hard-baking this structure in. If  $f$  is the feature mapping of a standard convolutional layer, then 360°-rotational equivariance can be hard-baked in by restricting the filters to be of the form from the circular harmonic family (proof in Supplementary Material)

$$\mathbf{W}_m(r, \phi; R, \beta) = R(r) e^{i(m\phi + \beta)}. \quad (2)$$

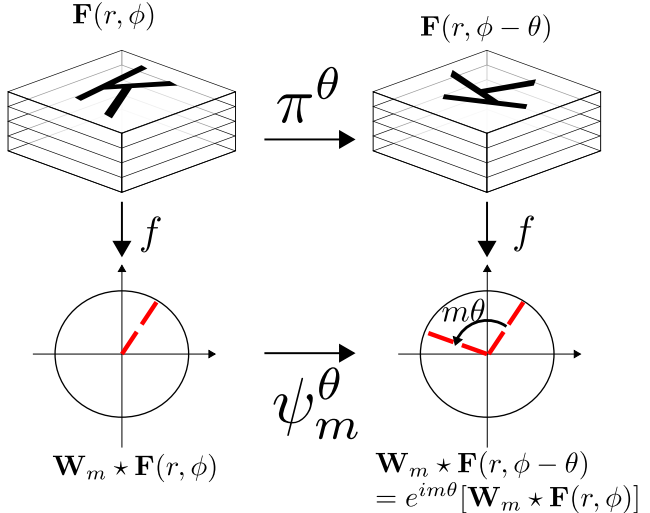


Figure 3. DOWN: Cross-correlation of the input patch with  $\mathbf{W}_m$  yields a scalar complex-valued response. ACROSS-THEN-DOWN: Cross-correlation with the  $\theta$ -rotated image yields another complex-valued response. BOTTOM: We transform from the unrotated response to the rotated response, through multiplication by  $e^{im\theta}$ .

Here  $r, \phi$  are the spatial coordinates of image/feature maps, expressed in polar form,  $m \in \mathbb{Z}$  is known as the *rotation order*,  $R: \mathbb{R}_+ \rightarrow \mathbb{R}$  is a function, called the *radial profile*, which controls the overall shape of the filter, and  $\beta \in [0, 2\pi)$  is a *phase offset* term, which gives the filter orientation-selectivity. During training, we learn the radial profile and phase offset terms. Examples of the real component of  $\mathbf{W}_m$  for a ‘Gaussian envelope’ and different rotation orders are shown in Figure 2. Since we are dealing with complex-valued filters, all filter responses are complex-valued, and we assume from now on that the reader understands that all feature maps are complex-valued, unless otherwise specified. Note that there are other works (e.g., [32]), which use complex filters, but our treatment differs in that the complex phase of the response is explicitly tied to rotation angle.

#### Rotational Equivariance of the Circular Harmonics

Some deep learning libraries implement cross-correlation  $\star$  rather than convolution  $*$ , and since the understanding is slightly easier to follow, we consider correlation. Strictly, cross-correlation with complex functions requires that one of the arguments is conjugated, but we do not do this in our model/implementation, so

$$[\mathbf{W} \star \mathbf{F}](p', q') = \int \mathbf{W}(p-p', q-q') \mathbf{F}(p, q) dp dq \quad (3)$$

$$[\mathbf{W} \star \mathbf{F}](p', q') = \int \mathbf{W}(p'-p, q'-q) \mathbf{F}(p, q) dp dq. \quad (4)$$

Consider correlating a circular harmonic of order  $m$  with a rotated image patch. We assume that the image patch is only able to rotate locally about the origin of the filter. This means that the cross-correlation response is a scalar function of input image patch rotation  $\theta$ . Using the notation from Equation 1, and recalling that we are working in polar coordinates  $(r, \phi)$ ,

counter-clockwise rotation of an image  $\mathbf{F}(r, \phi)$  about the origin by an angle  $\theta$  is  $\mathbf{F}(r, \pi^\theta[\phi]) = \mathbf{F}(r, \phi - \theta)$ . As a shorthand we denote  $\mathbf{F}^\theta := \mathbf{F}(r, \pi^\theta[\phi])$ . It is a well-known result [23, 7] (proof in Supplementary Material) that

$$[\mathbf{W}_m \star \mathbf{F}^\theta] = e^{im\theta} [\mathbf{W}_m \star \mathbf{F}^0], \quad (5)$$

where we have written  $\mathbf{W}_m$  in place of  $\mathbf{W}_m(r, \phi; R, \beta)$  for brevity. We see that the response to a  $\theta$ -rotated image  $\mathbf{F}^\theta$  with a circular harmonic of order  $m$  is equivalent to the cross-correlation of the unrotated image  $\mathbf{F}^0$  with the harmonic, followed by multiplication by  $e^{im\theta}$ . While the rotation is done in input space, multiplication by  $e^{im\theta}$  is performed in feature space, and so, using the notation from Equation 1,  $\psi_m^\theta[\bullet] = e^{im\theta} \cdot \bullet$ . This process is shown in Figure 3. Note that we have included a subscript  $m$  on the feature space transformation. This is important, because the kind of feature space transformation we apply is dependent on the rotation order of the harmonic. Because the phase of the response rotates with the input at frequency  $m$ , we say that the response is an  $m$ -equivariant feature map. By thinking of an input image as a complex-valued feature map with zero imaginary part, we could think of it as 0-equivariant.

The rotation order of a filter defines its response properties to input rotation. In particular, rotation order  $m=0$  defines invariance and  $m=1$  defines linear equivariance. For  $m=0$  this is because, denoting  $f_m := [\mathbf{W}_m \star \mathbf{F}^0]$ , then  $\psi_0^\theta[f_m] = e^{i \cdot 0 \theta} \cdot f_m = f_m$ , which is independent of  $\theta$ . For  $m=1$ ,  $\psi_1^\theta[f_m] = e^{i \cdot 1 \theta} f_m$ —as the input rotates,  $e^{i\theta} f_m$  is a complex-valued number of constant magnitude  $f_m$ , spinning round with a phase equal to  $\theta$ . Naturally, we are not constrained to using rotation orders 0 or 1 only, and we make use of higher and negative orders in our work.

**Arithmetic and the Equivariance Condition** Further important properties of the circular harmonics, which are proven in the Supplementary Material, are: 1) **Chained cross-correlation of rotation orders  $m_1$  and  $m_2$  lead to a new response with rotation order  $m_1 + m_2$ .** 2) Point-wise nonlinearities  $h: \mathbb{C} \rightarrow \mathbb{C}$ , acting solely on the magnitudes maintain rotational equivariance, so we can interleave cross-correlations with typical CNN nonlinearities adapted to the complex domain. 3) The summation of two responses of the same order  $m$  remains of order  $m$ . Thus to construct a CNN where the output is  $M$ -equivariant to the input rotation, we require that the sum of rotation orders along any path equals  $M$ , so

$$\sum_{i=1}^N m_i = M. \quad (6)$$

This is the fundamental condition underpinning the equivariance properties of H-Net, so we call it the *equivariance condition*.

We note here that for our purposes, our filter  $\mathbf{W}_{-m} = \overline{\mathbf{W}_m}$  (the complex conjugate), which saves on parameters, but this does not necessarily imply conjugacy of the responses unless  $\mathbf{F}$  is real, which is only true at the input.

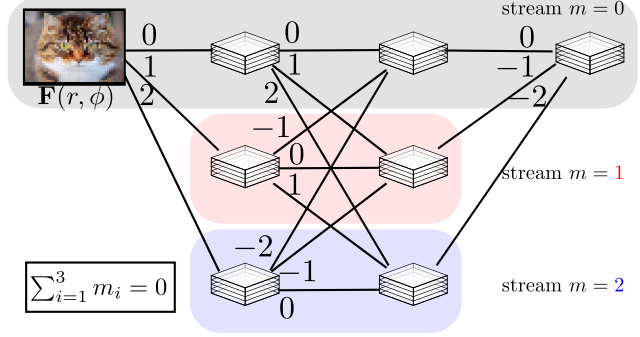


Figure 4. An example of a 2 hidden layer H-Net with  $m=0$  output, input–output left-to-right. Each horizontal stream represents a series of feature maps (circles) of constant rotation order. The edges represent cross-correlations and are numbered with the rotation order of the corresponding filter. The sum of rotation orders along any path of consecutive edges through the network must equal  $M=0$ , to maintain disentanglement of rotation orders.

## 4. Method

We have considered the  $360^\circ$ -rotational equivariance of feature maps arising from cross-correlation with the circular harmonics, and we determined that the rotation orders of chained cross-correlations sum. Next, we use these results to construct a deep architecture, which can leverage the equivariance properties of circular harmonics.

### 4.1. Harmonic Networks

The rotation order of feature maps and filters sum upon cross-correlation, so to achieve a given output rotation order, we must obey the equivariance condition. In fact, at every feature map, the equivariance condition must be met, otherwise, it should be possible to arrive at the same feature map along two different paths, with different summed rotation orders. The problem is that combining complex features, with phases, which rotate at different frequencies, leads to *entanglement* of the responses. The resultant feature map is no longer equivariant to a single rotation order, making it difficult to work with. We resolve this by enforcing the equivariance condition at every feature map.

Our solution is to create separate streams of constant rotation order responses running through the network—see Figure 4. These streams contain multiple layers of feature maps, separated by rotation order zero cross-correlations and nonlinearities. Moving between streams, we use cross-correlations of rotation order equal to the difference between those two streams. It is very easy to check that the equivariance condition holds in these networks.

When multiple responses converge at a feature map, we have multiple choices of how to combine them. We could stack them, we could pool across them, or we could sum them [5]. To save on memory, we chose to sum responses of the same rotation order

$$\mathbf{Y}_p = \sum_{m, n: m+n=p} \mathbf{W}_m \star \mathbf{F}_n. \quad (7)$$



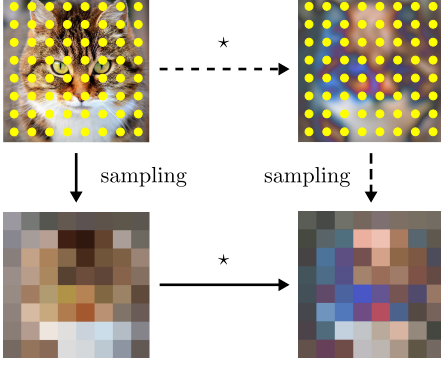


Figure 5. H-Nets operate in a continuous spatial domain, but we can implement them on pixel-domain data because sampling and cross-correlation commute. The schematic shows an example of a layer of an H-Net (magnitudes only). The solid arrows follow the path of the implementation, while the dashed arrows follow the possible alternative, which is easier to analyze, but computationally infeasible. The introduction of the sampling defines *centers of equivariance* at pixel centers (yellow dots), about which a feature map is rotationally equivariant.

$\mathbf{Y}_p$  is then fed into the next layer. Usually in our experiments, we use streams of orders 0 and 1, which we found to work well and is justified by the fact that CNN filters tend to contain very little high frequency information [12].

Above, we see that the structure of the Harmonic Network is very simple. We replaced regular CNN filters with radially reweighted and phase shifted circular harmonics. This causes each filter response to be equivariant to input rotations with order  $m$ . To prevent responses of different rotation order from entangling upon summation, we separated filter responses into streams of equal rotation order.

**Complex nonlinearities** Between cross-correlations, we use complex nonlinearities, which act on the magnitudes of the complex feature maps only, to preserve rotational equivariance. An example is a complex version of the ReLU

$$\mathbb{C}\text{-ReLU}_b(X e^{i\phi}) = \text{ReLU}(X + b) e^{i\phi}. \quad (8)$$

We can provide similar analogues for other nonlinearities and for Batch Normalization [11], which we use in our experiments.

We have thus far presented the Harmonic Network. Each layer is a collection of feature maps of different rotation orders, which transform predictably under rotation of the input to the network and the  $360^\circ$ -rotation equivariance is achieved with finite computation. Next we show how to implement this in practice.

## 4.2. Implementation: Discrete cross-correlations

Until now, we have operated on a domain with continuous spatial dimensions  $\Omega = \mathbb{R} \times \mathbb{R} \times \{1, k_\ell\}$ . However, the H-Net needs to operate on real-world images, which are sampled on a 2D-grid, thus we need to anti-alias the input to each discretized layer. We do this with a simple Gaussian blur. We can then use a regular CNN architecture without any problems. This works on the fact that the order of bandlimited sampling and

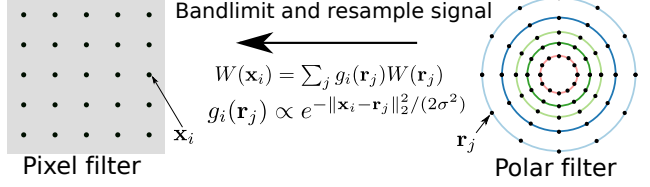


Figure 6. Images are sampled on a rectangular grid but our filters are defined in the polar domain, so we bandlimit and resample the data before cross-correlation via Gaussian resampling.

cross-correlation is interchangeable [7]; so either we correlate in continuous space, then downsample, or downsample then correlate in the discrete space. Since point-wise nonlinearities and sampling also commute, the entire H-Net, seen as a deep feature-mapping, commutes with sampling. This could allow us to implement the H-Net on non-regular grids; although, we did not explore this.

Viewing cross-correlation on discrete domains sheds some insight into how the equivariance properties behave. In Figure 5, we see that the sampling strategy introduces multiple origins, one for each feature map patch. We call these, *centers of equivariance*, because a feature map will exhibit local rotation equivariance about each of these points. If we move to using more exotic sampling strategies, such as strided cross-correlation or average pooling, then the centers of equivariance are ablated or shifted. If we were to use max-pooling, then the center of equivariance would be a complicated nonlinear function of the input image and harmonic weights. For this reason we have not used max-pooling in our experiments.

**Complex cross-correlations** On a practical note, it is worth mentioning, that complex cross-correlation can be implemented efficiently using 4 real cross-correlations

$$\underbrace{\mathbf{W}_m^{\text{Re}} \star \mathbf{F}^{\text{Re}} - \mathbf{W}_m^{\text{Im}} \star \mathbf{F}^{\text{Im}}}_{\text{real response}} + i \underbrace{\mathbf{W}_m^{\text{Re}} \star \mathbf{F}^{\text{Im}} + \mathbf{W}_m^{\text{Im}} \star \mathbf{F}^{\text{Re}}}_{\text{imaginary response}}. \quad (9)$$

So circular harmonics can be implemented in current deep learning frameworks, with minor engineering. We implement a grid-resampled version of the filters  $W(\mathbf{x}_i) = \sum_j g_i(\mathbf{r}_j) W(\mathbf{r}_j)$ , with  $g_i(\mathbf{x}_j) \propto e^{-\|\mathbf{r}_i - \mathbf{x}_j\|_2^2 / (2\sigma^2)}$  (see Figure 6). The polar representation  $(r_j, \phi_j)$  can be mapped from the components  $\mathbf{r}_j$  by  $\mathbf{r}_j = [r_j \cos \phi_j, r_j \sin \phi_j]^\top$ . If we stack all the polar filter samples into a matrix we can write each point as the outer product of a radial tensor  $\mathbf{R}_j$  and trigonometric angular tensor  $[\cos m \Phi_{r_j}, i \sin m \Phi_{r_j}]^\top$ . The phase offset  $\beta$  can be separated out by noting that

$$W_m(\mathbf{r}_j) = \sum_{i=1}^I R(r_j) \begin{bmatrix} \mathbf{I} \cos \beta & -\mathbf{I} \sin \beta \\ \mathbf{I} \sin \beta & \mathbf{I} \cos \beta \end{bmatrix} \begin{bmatrix} \cos m \Phi_{r_j} \\ i \sin m \Phi_{r_j} \end{bmatrix} \quad (10)$$

where the complex exponential and trigonometric terms are element-wise, and  $\mathbf{I}$  is the identity matrix. This is just a reweighting of the ring elements. In full generality, we could also use a per-radius phase  $\beta_{r_i}$ , which would allow for spiral-like left- and right-handed features, but we did not investigate this.

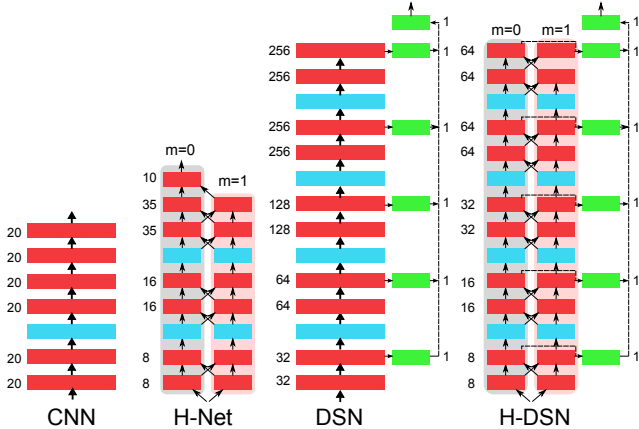


Figure 7. Networks used in our experiments. LEFT: MNIST networks, as per [3]. RIGHT: deeply-supervised networks (DSN) [20] for boundary segmentation, as per [33]. Red boxes denote feature maps. Blue boxes are pooling (max for CNNs and average for H-Nets). Green boxes are **side feature maps** as per [33]; these are connected to the DSN with dashed lines for ease of viewing. All main cross-correlations are  $3 \times 3$ , unless otherwise stated in the experiments section.

### 4.3. Computational cost

We have increased the computational cost of cross-correlation, in return for continuous rotational equivariance. Here we analyze the computational cost in terms of number of multiplications. In the standard cross-correlation, for an input of size  $h \cdot w \cdot i_Z$ , (height, width, input channels) and filters of size  $k \cdot k \cdot o_Z$  (height, width, output channels), the number of multiplications to form a feature map of the same size as the input is  $M(\mathbb{Z}) = hwk^2 i_Z o_Z$ . In the H-Net, we have  $f$  rotation orders on the input and  $r$  rotation orders on the output, so perform  $fr$  complex cross-correlations. Each complex cross-correlation can be formed from 4 real cross-correlations, so the number of multiplications is  $4M(\mathbb{H})fr$ , where  $i_{\mathbb{H}}$  and  $o_{\mathbb{H}}$  are the number of input and output channels, respectively. Thus for similar computational cost we equate the two to yield  $M(\mathbb{Z}) = 4M(\mathbb{H})fr$ . Rearranging; setting  $i_{\mathbb{H}} = o_{\mathbb{H}}$ ,  $i_Z = o_Z$  and  $f = r$ ; and taking the square root of both sides, we arrive at a simple rule of thumb for network design,  $i_Z = 2fi_{\mathbb{H}}$ . For example, if we want to build an H-Net with similar computational cost to a regular CNN with 64 channels per layer, then if we use 2 rotation orders  $m \in \{0, 1\}$ , then the number of H-Net channels is  $64/(2 \cdot 2) = 16$ .

## 5. Experiments

We validate our rotation equivariant formulation below, performing some introspective investigations, and measuring against relevant baselines for classification on **the rotated-MNIST dataset** [18] and boundary detection on the Berkeley Segmentation Dataset [1]. We selected our baselines as representative examples of the current state-of-the-art and to demonstrate that H-Nets can be used on different architectures for different tasks. The networks we used are in Figure 13.

Method	Test error (%)	# params
SVM [18]	11.11	-
Transformation RBM [31]	4.2	-
Conv-RBM [27]	3.98	-
CNN [3]	5.03	22k
CNN [3] + data aug*	3.50	22k
P4CNN rotation pooling [3]	3.21	25k
P4CNN [3]	2.28	25k
H-Net (Ours)	<b>1.69</b>	33k

Table 1. Results. Our model sets a new state-of-the-art on the rotated MNIST dataset, reducing the test error by 26%. \* Our reimplemention

## 5.1. Benchmarks

Here we compare H-Nets for classification and boundary detection. **Classification is a typical rotation invariant task**, and should suit H-Nets very well. In contrast, **boundary detection is a rotation equivariant task**. The key to the success of the H-Net is that it can achieve global equivariance, without sacrificing local equivariance of features.

**MNIST** Of course, this is a small dataset, with simple visual structures, but it is a good indication of how introducing the right equivariances into CNNs can aid inference. We investigate classification on the rotated MNIST dataset (new version) [18] as a baseline. This has 10000 training images, 2000 validation images, and 50000 test images. **The 360°-rotations and small training set size make this a difficult task for classical CNNs**. We compare against a collection of previous state-of-the-art papers and [3], who build a deep CNN with filter copies at 90°-rotations. We try to mimic their network architecture for H-Nets as best as we can, using 2 rotation order streams with  $m \in \{0, 1\}$  through to the deepest layer, and complex-valued versions of ReLU nonlinearities and Batch Normalization (see Method). We also replace max-pooling with mean-pooling layers, as shown in Figure 13. We perform stochastic gradient descent on a cross-entropy loss using Adam [13] and **an adaptive learning rate**, which we divide by 10 if there has been **no improvement in validation accuracy in the last 10 epochs**. We train multiple models with randomly chosen hyperparameters, and report the test error of the model that performed best on the validation set, training on a combined training and validation set Table 1 lists our results. This model actually has 33k parameters, which is about 50% larger than the standard CNN and [3], which have 22k. This is because it uses  $5 \times 5$  convolutions instead of  $3 \times 3$ . Interestingly, it does not overfit on such a small dataset and it still outperforms the standard CNN trained with rotation augmentations, which we do not use. We set the new state-of-the-art, with a 26% improvement on the previous best model.

**Deep Boundary Detection** Boundary detection is equivariant to non-rigid transformations; although edge presence is locally invariant to orientation. The current state-of-the-art

Method	ODS	OIS	# params
HED, [33]*	0.640	0.650	2346k
HED, low # params [33]*	0.697	0.709	115k
Kivinen et al. [14]	0.702	0.715	-
H-Net (Ours)	<b>0.726</b>	<b>0.742</b>	116k
CSCNN <sup>†</sup> , [10]	0.741	0.759	
DeepEdge <sup>†</sup> , [2]	0.753	0.772	
$N^4$ -Fields <sup>†</sup> , [8]	0.753	0.769	
DeepContour <sup>†</sup> , [28]	0.756	0.773	
HED <sup>†</sup> , [33]	0.782	0.804	2346k
DCNN + sPb <sup>†</sup> , [15]	<b>0.813</b>	<b>0.831</b>	

Table 2. Our model beats the non-pretrained neural networks baselines on BSD500 [1]. \*Our implementation. †ImageNet pretrained

depends on fine-tuning ImageNet-pretrained networks to regress boundary probabilities on a per-patch basis. To demonstrate that hard-baked rotation equivariance serves as a strong generalization tool, we compared against a previous state-of-the-art architecture [33], *without pretraining*. We tried to mimic [33] as closely as possible, as shown in Figure 13. The main difference is that we divide the number of all feature maps by 2, for faster, more stable training. They use a VGG network [29] extended with deeply supervised network (DSN) [20] *side-connections*. These are  $1 \times 1$ -convolutions, which perform weighted averages across all relevant feature maps, resized to match the input. A binary cross-entropy loss is applied to each side connection, to stabilize learning. A final ‘fusion’ layer is created by taking a weighted linear combination of the side-connections, this is the final output. We adapt side-connections to H-Nets, by using the complex magnitude of feature maps before taking a weighted average. This means that the resultant boundary predictions are locally invariant to rotation. We added a small sparsity regularizer to our cost function, because we found it improved results slightly. We call the Harmonic variant of the DSN, an *H-DSN*. We also compare against [33] with the number of parameters matched to H-DSN (the first layer has 7 features, instead of 16, and so on).

We also compared with [14], who use a mean-and-covariance-RBM. Their technique has five main contributions: 1) zero-mean, unit variance normalization of inputs, 2) sparsity regularization of hidden units, 3) averaged ground truth edge annotations, 4) average outputs to 16 input rotations, 5) non-maximum suppression of results by the Canny method. We perform the first 2 methods, but leave the last 3 alone. In particular, they do not pretrain on ImageNet, and attempt some kind of rotation averaging for global equivariance, so are a good baseline to measure against. We tested on the **Berkeley Segmentation Dataset** (BSD500) [1]. As shown in Table 2 for non-pretrained models, H-Nets deliver superior performance over current state-of-the-art architectures, including [14], who also encode rotation equivariance. Most noticeable of all is that we only use 5% of the parameters of [33], showing how by restricting

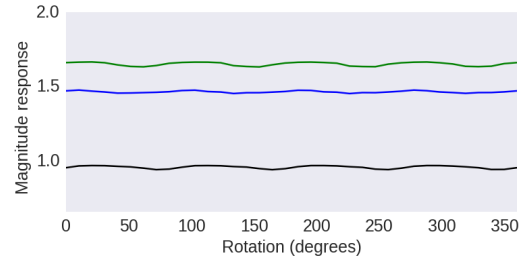


Figure 8. Stability of the response magnitude to input rotation angle. Black  $m=0$ , blue  $m=1$ , green  $m=2$ .

the search space of learnable models through hard-baking local rotation equivariance, we need not learn so many parameters.

## 5.2. Model Insight

Here we investigate some of the properties of the H-Net implementation, making sure that the motivations behind H-Net design are achieved by the implementation.

**Rotational stability** As a sanity check, we measured the invariance of the magnitude response to rotation for  $m \in \{0, 1, 2\}$ . We show the result of rotating a random input to an H-Net layer in Figure 8. The response is very flat, with periodic small fluctuations due to the inexactness of anti-aliasing.

**Filter Visualization** The real parts of the filters, from the first layer of the boundary-detection-trained H-Net, are shown in Figure 9. They are aligned at zero phase ( $\beta = 0$ ) for ease of viewing. Since the network is trained on zero-mean, unit variance, normalized color images, the weights do not have the natural colors we would see in real-world images. Nonetheless, there is useful information we can glean from inspecting these. Most 1<sup>st</sup> layer filters detect color boundaries, there are no blank filters as one usually sees in CNNs, and there are few reoriented copies. We also see from the phase histograms that all phases are utilized by filters throughout the network, indicating full use

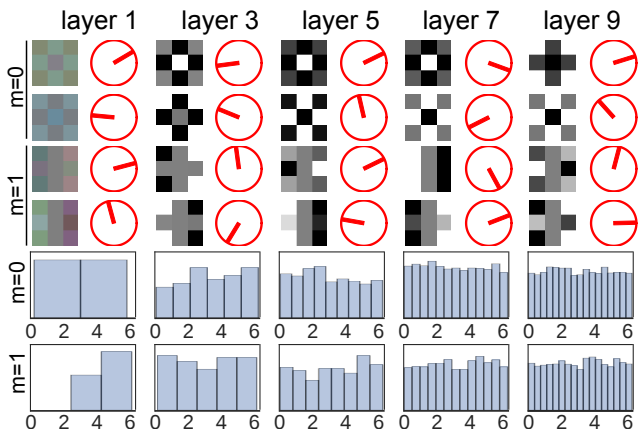


Figure 9. Randomly selected filters and phase histograms from the BSDS500 trained H-DSN. Filter are aligned at  $\beta = 0$ ; and the oriented circles represent phase. We see **few filter copies and no blank filters, as usually seen in CNNs**. We also see a balanced distribution over phases, indicating that boundaries, and their deep feature representations, are uniformly distributed in orientation.

of the phase information. This is interesting, because it means that the model’s parameters are being used fully, with low redundancy, which we surmise comes from easier optimization on the equivariant manifold.

**Data ablation** Here we investigate H-nets data-efficiency. CNNs are massively data-hungry. Krizhevsky’s landmark paper [16] used 60 million parameters, trained on 1.2 million  $256 \times 256$  RGB images quantized to 256 bits and split between 1000 classes, for a total of 10 bits of information per weight. Even this vast amount of data was insufficient for training, and data augmentation was needed to improve results. We ran an experiment on the rotated MNIST dataset to show that with hard-baked rotation equivariance, we require less data than competing methods, which is indeed the case (see Figure 10). Interestingly, and predictably, **regular CNNs trained with data augmentation still perform worse than H-Nets, because they only learn global invariance to rotation, rather than local equivariances at each layer.**

**Feature maps** We visualize feature maps in the lower layers of an MNIST trained H-Net (see Figure 11). For given input, we see the feature maps encode very complicated structures. Left to right, we see the H-Net learns to detect edges, corners, object presence, negative space, and outlines of objects. We perform this for the BSD500 trained H-DSN (see Figure 12). It shows equivariance is preserved through to the deepest feature maps. It also highlights the compact representation of feature presence and pose, which regular CNNs cannot do.

## 6. Conclusions

We presented a convolutional neural network that is locally equivariant to patch-wise translation and, for the first time, to continuous  $360^\circ$ -rotation. We achieved this by restricting the filters to circular harmonics, essentially hard-baking rotation into the architecture. This can be implanted onto other architectures too. The use of circular harmonics pays dividends in that we receive full rotational equivariance using few parameters. This leads to good generalization, even with less (or less augmented) training data. The only disadvantage we’ve seen so far is the higher per-filter computational cost, but our guidance for network design balances that cost against the more expressive

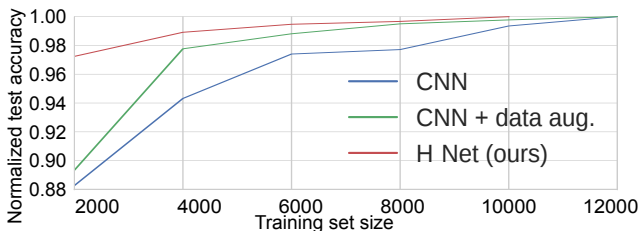


Figure 10. Data ablation study. On the rotated MNIST dataset, we experiment with test accuracy for varying sizes of the training set. We normalize the maximum test accuracy of each method to 1, for direct comparison of the falloff with training size. Clearly H-Nets are more data-efficient than regular CNNs, which need more data to discover rotational equivariance unaided.

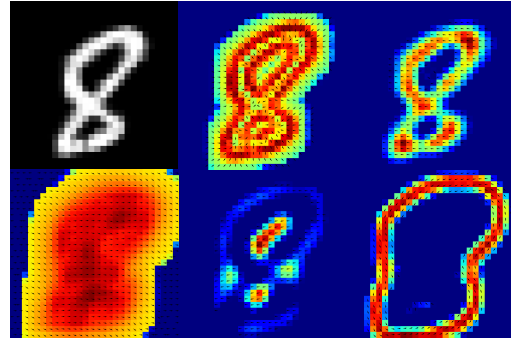


Figure 11. Feature maps from the MNIST network. The arrows display phase, and the colors display magnitude information (jet color scheme). There are diverse features encoding edges, corners, whole objects, negative spaces, and outlines.

representation. The better interpretability of the feature maps is a bonus, because we know how they transform under input image rotations. We applied our network to the problem of classifying rotated-MNIST, setting a new state-of-the-art. We also applied our network to boundary detection, again achieving state-of-the-art results, for non-pretrained networks. We have shown that  $360^\circ$ -rotational equivariance is both possible and useful. Our TensorFlow™ implementation is available on the project website.

**Future work** Extension of this work could involve hard-baking yet more transformations into the equivariance properties of the Harmonic Network, possibly extending to 3D. This will allow yet more expressibility in network representations, extending the benefits we have seen afforded by rotation equivariance to a larger class of models and applications.

**Acknowledgements** Support is from Fight for Sight UK, a Microsoft Research PhD Scholarship, EPSRC Nature Smart Cities EP/K503745/1 and ENGAGE EP/K015664/1.



Figure 12. View best in color. Orientated feature maps for the H-DSN. The color wheel shows orientation coding. Note that between layers boundary orientations are colored differently because each feature has a different  $\beta$ . This visualization demonstrates the consistency of orientation within a feature map and across multiple layers. The images are taken from layers 2, 4, 6, 8, and 10 in a clockwise order from largest to smallest.



## References

- [1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011. [6](#), [7](#)
- [2] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4380–4389, 2015. [7](#)
- [3] T. S. Cohen and M. Welling. Group equivariant convolutional networks. *arXiv preprint arXiv:1602.07576*, 2016. [2](#), [6](#)
- [4] T. S. Cohen and M. Welling. Steerable cnns. *CoRR*, abs/1612.08498, 2016. [2](#)
- [5] S. Dieleman, J. De Fauw, and K. Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. *arXiv preprint arXiv:1602.02660*, 2016. [2](#), [4](#)
- [6] B. Fasel and D. Gatica-Perez. Rotation-invariant neoperceptron. In *18th International Conference on Pattern Recognition (ICPR 2006), 20-24 August 2006, Hong Kong, China*, pages 336–339, 2006. [2](#)
- [7] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991. [2](#), [4](#), [5](#)
- [8] Y. Ganin and V. S. Lempitsky.  $N^4$ -fields: Neural network nearest neighbor fields for image transforms. *CoRR*, abs/1406.6558, 2014. [7](#)
- [9] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I*, pages 44–51, 2011. [2](#)
- [10] J. Hwang and T. Liu. Pixel-wise deep learning for contour detection. *CoRR*, abs/1504.01989, 2015. [7](#)
- [11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456, 2015. [5](#)
- [12] J. Jacobsen, J. C. van Gemert, Z. Lou, and A. W. M. Smeulders. Structured receptive fields in cnns. *CoRR*, abs/1605.02971, 2016. [5](#)
- [13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. [6](#)
- [14] J. J. Kivinen, C. K. I. Williams, and N. Heess. Visual boundary prediction: A deep neural prediction network and quality dissection. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, pages 512–521, 2014. [7](#)
- [15] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. *arXiv preprint arXiv:1511.07386*, 2015. [7](#)
- [16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012. [2](#), [8](#)
- [17] D. Laptev, N. Savinov, J. M. Buhmann, and M. Pollefeys. TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. *CoRR*, abs/1604.06318, 2016. [2](#)
- [18] H. Larochelle, D. Erhan, A. C. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 473–480, 2007. [6](#)
- [19] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 396–404, 1989. [1](#)
- [20] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, 2015. [6](#), [7](#)
- [21] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 991–999, 2015. [1](#), [3](#)
- [22] K. Lenc and A. Vedaldi. Learning covariant feature detectors. *CoRR*, abs/1605.01224, 2016. [2](#)
- [23] K. Liu, Q. Wang, W. Driever, and O. Ronneberger. 2d/3d rotation-invariant detection using equivariant filters and kernel weighted mapping. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 917–924, 2012. [2](#), [4](#)
- [24] D. Marcos, M. Volpi, and D. Tuia. Learning rotation invariant convolutional filters for texture classification. *arXiv preprint arXiv:1604.06720*, 2016. [2](#)
- [25] R. Memisevic and G. E. Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation*, 22(6):1473–1492, 2010. [2](#)
- [26] E. Oyallon and S. Mallat. Deep roto-translation scattering for object classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 2865–2873, 2015. [2](#)
- [27] U. Schmidt and S. Roth. Learning rotation-aware features: From invariant priors to equivariant descriptors. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 2050–2057, 2012. [6](#)
- [28] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3982–3991, 2015. [7](#)
- [29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. [7](#)
- [30] S. Soatto. Actionable information in vision. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, pages 2138–2145, 2009. [1](#)
- [31] K. Sohn and H. Lee. Learning invariant representations with local transformations. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012. [6](#)

- [32] M. Tygert, J. Bruna, S. Chintala, Y. LeCun, S. Piantino, and A. Szlam. **A mathematical motivation for complex-valued convolutional networks.** *Neural Computation*, 28(5):815–825, 2016. [3](#)
- [33] S. Xie and Z. Tu. Holistically-nested edge detection. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1395–1403, 2015. [6, 7](#)
- [34] M. D. Zeiler and R. Fergus. **Visualizing and understanding convolutional networks.** In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, pages 818–833, 2014. [2](#)

# Supplementary Material

## Abstract

We include some proofs and derivations of the rotational equivariance properties of the circular harmonics, along with a demonstration of how we calculate the number of parameters for various network architectures.

### A. Equivariance properties

In Section 3.2 we mentioned that cross-correlation with the circular harmonics is a  $360^\circ$ -rotation equivariant feature transform. Here we provide the proof, and some of the properties mentioned in **Arithmetic and Equivariance Condition**.

#### A.1. Equivariance of the Circular Harmonics

We are interested in proving that there exists a filter  $\mathbf{W}_m$ , such that cross-correlation of  $\mathbf{F}$  with  $\mathbf{W}_m$  yields a rotationally equivariant feature map. The proof requires us to introduce two different kinds of transformation: rotation  $\mathcal{R}$  and translation  $\mathcal{T}$ . To simplify the math, we use vector notation, so the spatial domain of the filter/image is  $\mathbb{R}^2$ . We write the filter as  $\mathbf{W}_m(\mathbf{x})$  and image as  $\mathbf{F}(\mathbf{x})$  for  $\mathbf{x} \in \mathbb{R}^2$ . We define the transformation operators  $\mathcal{R}_\theta$  and  $\mathcal{T}_\mathbf{t}$ , such that  $\mathcal{R}_\theta \mathbf{F} = \mathbf{F}(\mathbf{R}_{-\theta} \mathbf{x})$  and  $\mathcal{T}_\mathbf{t} \mathbf{F} = \mathbf{F}(\mathbf{x} - \mathbf{t})$ , where  $\mathbf{R}_\theta$  is a 2D rotation matrix for a  $\theta$  counter-clockwise rotation. We introduce rotational cross-correlation  $\star$ . This is defined as

$$[\mathbf{W}_m \star \mathbf{F}] = \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_\phi \hat{\mathbf{x}}) \mathbf{F}(r \mathbf{R}_\phi \hat{\mathbf{x}}) dr d\phi, \quad (11)$$

where we have used the decomposition  $\mathbf{x} = r \hat{\mathbf{x}}$ , with  $r = \|\mathbf{x}\|_2 \geq 0$  and  $\hat{\mathbf{x}} = \mathbf{x}/r$ . The rotational cross-correlation is performed about the origin of the image. If we rotate the image, then we have

$$[\mathbf{W}_m \star \mathcal{R}_\theta \mathbf{F}] = \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_\phi \hat{\mathbf{x}}) \mathbf{F}(r \mathbf{R}_\phi \mathbf{R}_{-\theta} \hat{\mathbf{x}}) dr d\phi \quad (12)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_\phi \hat{\mathbf{x}}) \mathbf{F}(r \mathbf{R}_{\phi-\theta} \hat{\mathbf{x}}) dr d\phi \quad (13)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_{\phi'+\theta} \hat{\mathbf{x}}) \mathbf{F}(r \mathbf{R}_{\phi'} \hat{\mathbf{x}}) dr d\phi'. \quad (14)$$

If we define  $\mathbf{W}_m(\mathbf{x}) = \mathbf{W}_m(r \hat{\mathbf{x}}) = R(r) e^{i(m\phi + \beta)}$ , where  $\phi = \angle \hat{\mathbf{x}}$ , then

$$[\mathbf{W}_m \star \mathcal{R}_\theta \mathbf{F}] = \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_{\phi'+\theta} \hat{\mathbf{x}}) \mathbf{F}(r \mathbf{R}_{\phi'} \hat{\mathbf{x}}) dr d\phi' \quad (15)$$

$$= \int_{\Phi} \int_R R(r) e^{i(m(\phi'+\theta) + \beta)} \mathbf{F}(r \mathbf{R}_{\phi'} \hat{\mathbf{x}}) dr d\phi' \quad (16)$$

$$= e^{im\theta} \int_{\Phi} \int_R R(r) e^{i(m\phi' + \beta)} \mathbf{F}(r \mathbf{R}_{\phi'} \hat{\mathbf{x}}) dr d\phi' \quad (17)$$

$$= e^{im\theta} [\mathbf{W}_m \star \mathbf{F}]. \quad (18)$$

And so rotational cross-correlation is rotationally equivariant about the origin of rotation. In the next part, we build up to a result needed for proving the chained cross-correlation result.

**Cross-correlation about  $\mathbf{t}$**  To perform the rotational cross-correlation about another point  $\mathbf{t}$ , we first have to translate the image such that  $\mathbf{t}$  is the new origin, so  $F_\mathbf{t}(\mathbf{x}) = \mathbf{F}(\mathbf{x} - \mathbf{t})$ , then perform the rotational cross-correlation, so

$$[\mathbf{W}_m \star \mathcal{T}_\mathbf{t} \mathbf{F}] = [\mathbf{W}_m \star \mathbf{F}_\mathbf{t}] \quad (19)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_\phi \hat{\mathbf{x}}) \mathbf{F}_\mathbf{t}(r \mathbf{R}_\phi \hat{\mathbf{x}}) dr_x d\phi \quad (20)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_\phi \hat{\mathbf{x}}) \mathbf{F}(r \mathbf{R}_\phi \hat{\mathbf{x}} - \mathbf{t}) dr_x d\phi. \quad (21)$$

**Cross-correlation about  $\mathbf{t}$  with rotated  $\mathbf{F}$  about  $\mathbf{t}$**  In general, for every  $\mathbf{t}$  this expression returns a different value. The response of a  $\theta$ -rotated image about  $\mathbf{t}$  is then

$$[\mathbf{W}_m \star \mathcal{R}_\theta \mathcal{T}_t \mathbf{F}] = [\mathbf{W}_m \star \mathcal{R}_\theta \mathbf{F}_t] \quad (22)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_\phi \hat{\mathbf{x}}) \mathbf{F}_t(r \mathbf{R}_{-\theta} \mathbf{R}_\phi \hat{\mathbf{x}}) dr_x d\phi \quad (23)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_\phi \hat{\mathbf{x}}) \mathbf{F}_t(r \mathbf{R}_{\phi-\theta} \hat{\mathbf{x}}) dr_x d\phi \quad (24)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_{\phi'+\theta} \hat{\mathbf{x}}) \mathbf{F}_t(r \mathbf{R}_{\phi'} \hat{\mathbf{x}}) dr_x d\phi' \quad (25)$$

$$= e^{im\theta} \int_{\Phi} \int_{R_z} R(r_z) e^{i(m\phi'+\beta)} \mathbf{F}_t(r \mathbf{R}_{\phi'} \hat{\mathbf{x}}) dr d\phi' \quad (26)$$

$$= e^{im\theta} [\mathbf{W}_m \star \mathcal{T}_t \mathbf{F}]. \quad (27)$$

**Cross-correlation about  $\mathbf{t}$  with rotated  $\mathbf{F}$  about origin** Say we wish to perform the rotational cross-correlation about a point  $\mathbf{t}$ , when the image has been rotated about the origin. Denoting  $\mathbf{F}^\theta = \mathcal{R}_\theta \mathbf{F}$ , then the response is

$$[\mathbf{W}_m \star \mathcal{T}_t \mathcal{R}_\theta \mathbf{F}] = [\mathbf{W}_m \star \mathcal{T}_t \mathbf{F}^\theta] \quad (28)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_\phi \hat{\mathbf{x}}) \mathbf{F}^\theta(r_x \mathbf{R}_\phi \hat{\mathbf{x}} - \mathbf{t}) dr_x d\phi \quad (29)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_\phi \hat{\mathbf{x}}) \mathbf{F}(r_x \mathbf{R}_{-\theta} \mathbf{R}_\phi \hat{\mathbf{x}} - \mathbf{R}_{-\theta} \mathbf{t}) dr_x d\phi \quad (30)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_\phi \hat{\mathbf{x}}) \mathbf{F}(r_x \mathbf{R}_{\phi-\theta} \hat{\mathbf{x}} - \mathbf{R}_{-\theta} \mathbf{t}) dr_x d\phi \quad (31)$$

$$= \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_{\phi'+\theta} \hat{\mathbf{x}}) \mathbf{F}(r_x \mathbf{R}_{\phi'} \hat{\mathbf{x}} - \mathbf{R}_{-\theta} \mathbf{t}) dr_x d\phi' \quad (32)$$

$$= e^{im\theta} \int_{\Phi} \int_R \mathbf{W}_m(r \mathbf{R}_{\phi'} \hat{\mathbf{x}}) \mathbf{F}(r_x \mathbf{R}_{\phi'} \hat{\mathbf{x}} - \mathbf{R}_{-\theta} \mathbf{t}) dr_x d\phi' \quad (33)$$

$$= e^{im\theta} [\mathbf{W}_m \star \mathcal{T}_{\mathbf{R}_{-\theta} \mathbf{t}} \mathbf{F}]. \quad (34)$$

Thus we see that cross-correlation of the rotated signal  $\mathbf{F}^\theta$  with the circular harmonic filter  $\mathbf{W}_m = R(r) e^{i(m\phi+\beta)}$  is equal to the response at zero rotation  $[\mathbf{W}_m \star \mathbf{F}]$ , multiplied by a complex phase shift  $e^{im\theta}$ . In the notation of the paper, we denote this multiplication by  $e^{im\theta}$  as  $\psi_m^\theta[\bullet] = e^{im\theta} \bullet$ . Thus cross-correlation with  $\mathbf{W}_m$  yields a rotationally equivariant feature mapping.

## A.2. Properties

### A.2.1 Chained cross-correlation

We claimed in **Arithmetic and Equivariance Condition**, that the rotation order of a feature map resulting from chained cross-correlations is equal to the sum of the the rotation orders of the filters in the chain. We prove this for a chain of two filters, and the rest follows by induction. Consider taking a  $\theta$ -rotated image  $\mathbf{F}$  about the origin, then cross-correlating it with a filter  $\mathbf{W}_m$  as every point in the image plane  $\mathbf{t} \in \mathbb{R}^2$ , followed by cross-correlation with  $\mathbf{W}_n$  as a point  $\mathbf{s} \in \mathbb{R}^2$ . We already know that the response to the rotation is  $[\mathbf{W}_m \star \mathcal{T}_t \mathcal{R}_\theta \mathbf{F}] = e^{im\theta} [\mathbf{W}_m \star \mathcal{T}_{\mathbf{R}_{-\theta} \mathbf{t}} \mathbf{F}]$ , for all rotations  $\theta$  of the input and all points  $\mathbf{t}$  in the response plane, so we can write the chained convolution as

$$[\mathbf{W}_n \star \mathcal{T}_s [\mathbf{W}_m \star \mathcal{T}_t \mathcal{R}_\theta \mathbf{F}]] = [\mathbf{W}_n \star \mathcal{T}_s e^{im\theta} [\mathbf{W}_m \star \mathcal{T}_{\mathbf{R}_{-\theta} \mathbf{t}} \mathbf{F}]] \quad (35)$$

$$= e^{im\theta} [\mathbf{W}_n \star \mathcal{T}_s [\mathbf{W}_m \star \mathcal{T}_{\mathbf{R}_{-\theta} \mathbf{t}} \mathbf{F}]] \quad (36)$$



We have used the property that the cross-correlation is linear and that we may pull the scalar factor  $e^{im\theta}$  outside. If we write  $\mathbf{G}(\mathbf{t}) = [\mathbf{W}_m \star \mathcal{T}_t \mathbf{F}]$  then  $[\mathbf{W}_m \star \mathcal{T}_{\mathbf{R}_{-\theta} \mathbf{t}} \mathbf{F}] = \mathbf{G}(\mathbf{R}_{-\theta} \mathbf{t}) = [\mathcal{R}_\theta \mathbf{G}](\mathbf{t})$ , so

$$[\mathbf{W}_n \star \mathcal{T}_s [\mathbf{W}_m \star \mathcal{T}_t \mathcal{R}_\theta \mathbf{F}]] = e^{im\theta} [\mathbf{W}_n \star \mathcal{T}_s [\mathbf{W}_m \star \mathcal{T}_{\mathbf{R}_{-\theta} \mathbf{t}} \mathbf{F}]] \quad (37)$$

$$= e^{im\theta} [\mathbf{W}_n \star \mathcal{T}_s \mathcal{R}_\theta \mathbf{G}] \quad (38)$$

$$= e^{im\theta} e^{in\theta} [\mathbf{W}_n \star \mathcal{T}_{\mathbf{R}_{-\theta} \mathbf{s}} \mathbf{G}]. \quad (39)$$

Thus we see that the chained cross-correlation results in a summation of the rotation orders of the individual filters  $\mathbf{W}_m$  and  $\mathbf{W}_n$ . Setting  $\mathbf{s} = \mathbf{0}$ , such that we evaluate the cross-correlation at the center of rotation, we regain an equation similar to 18.

### A.2.2 Magnitude nonlinearities

Point-wise nonlinearities acting on the magnitude of a feature map maintain rotational equivariance. Consider that we have a point on a feature map of rotation order  $m$ , which we can write as  $F e^{im\theta}$ , where  $F \geq 0$  is the magnitude of the feature map and  $e^{im\theta}$  is the phase component. The output of the nonlinearity  $g: \mathbb{R}_+ \rightarrow \mathbb{R}$  is

$$g(F e^{im\theta}) = g(F) e^{im\theta}, \quad (40)$$

since  $g$  only acts on magnitudes. Since for fixed  $F$  the output is a function of  $m$  and  $\theta$  only, the point-wise magnitude-acting nonlinearity preserves rotational equivariance.

### A.2.3 Summation of feature maps

The summation of feature maps of the same rotation order is a new feature map of the same rotation order. Consider two feature maps  $\mathbf{F}_1$  and  $\mathbf{F}_2$  of rotation order  $m$ . Summation is a pointwise operation, so we only consider two corresponding points in the feature maps, which we denote  $F_1 e^{i(m\theta + \beta_1)}$  and  $F_2 e^{i(m\theta + \beta_2)}$ , where  $\beta_1$  and  $\beta_2$  are phase offsets. The sum is

$$F_1 e^{i(m\theta + \beta_1)} + F_2 e^{i(m\theta + \beta_2)} = e^{im\theta} (F_1 e^{i\beta_1} + F_2 e^{i\beta_2}), \quad (41)$$

which for fixed  $F_1, F_2, \beta_1, \beta_2$  is a function of  $m$  and  $\theta$  only and so also rotationally equivariant with order  $m$ .

## B. Number of parameters

Here we list a break down of how we computed the number of parameters for the various network architectures in the experiments section. The networks architectures used are in Figure 13. Red boxes are cross-correlations, blue boxes are pooling (average for H-Nets, max for regular CNNs), green boxes are  $1 \times 1$ -cross-correlations.

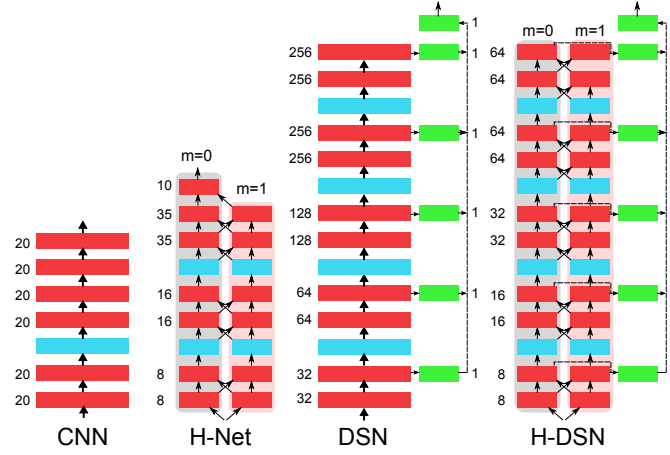


Figure 13. Networks used

### B.1. Standard CNN

For a standard CNN layer with  $i$  input channels and  $o$  output channels, and  $k \times k$  sized weights, the number of learnable parameters is  $io k^2$ . Since there is one bias per output layer, this increases to  $io k^2 + o$ . If using batch normalization, then there is an extra per-channel scaling factor, which increases the number of learnable parameters to  $io k^2 + 2o$ . The standard CNN for the rotated MNIST experiments has 6 layers of  $3 \times 3$  cross-correlations, and 1 layer of  $4 \times 4$ -cross-correlations, with 20 feature maps per layer and 3 batch normalization layers so the number of learnable parameters is 21570. The calculations are shown in Table 3.

Layer	Weights	Batch Norm/Bias	#Params
1	$3 \cdot 3 \cdot 1 \cdot 20$	20	200
2	$3 \cdot 3 \cdot 20 \cdot 20$	$2 \cdot 20$	3640
3	$3 \cdot 3 \cdot 20 \cdot 20$	20	3620
4	$3 \cdot 3 \cdot 20 \cdot 20$	$2 \cdot 20$	3640
5	$3 \cdot 3 \cdot 20 \cdot 20$	20	3620
6	$3 \cdot 3 \cdot 20 \cdot 20$	$2 \cdot 20$	3640
7	$4 \cdot 4 \cdot 20 \cdot 10$	10	3210
Total			21570

Table 3. Number of parameters for a regular CNN.

### B.2. Harmonic networks

The learnable parameters of a Harmonic Network are the radial profile and the the per-filter phase offset. For a  $k \times k$  filter, the number of radial profile elements is equal to the number of rings of equal distance from the center of the filter. For example, consider the Figure 14, which is an excerpt from the main paper. This is a  $5 \times 5$  filter, with 6 rings of equal distance from the center of the filter (the smallest ring is just a single point). So this filter has 6 radial profile terms and 1 phase offset to learn. This contrasts with a regular filter, which would have 25 learnable parameters. Note, that for filters with rotation order  $m \neq 0$ , the center pixel of the filter is in fact always zero, and so for a  $5 \times 5$  rotation order  $m \neq 0$  filter, the number of radial profile terms is  $6 - 1 = 5$ . So for the H-Net in the main paper with  $5 \times 5$  filters and batch normalization in layers 2, 4, & 6, the number of learnable parameters is 33347. The calculations are in Table 4. Note that the final layer contains just one set of biases and no phase offsets. A similar set of calculations can be performed for the deeply supervised networks.

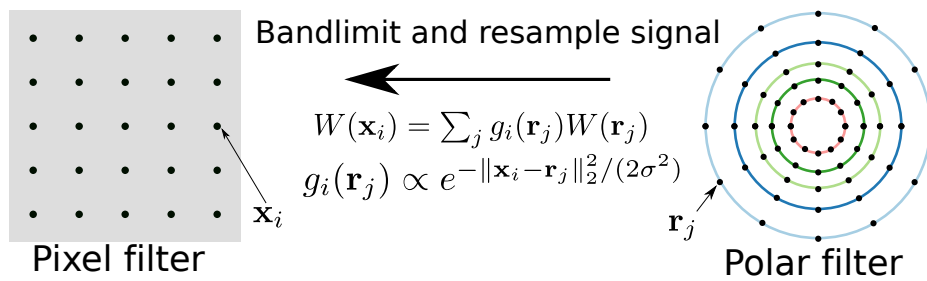


Figure 14. Each radius has a single learnable weight. Then there is a bias for the whole filter.

Layer	$m=0$	$m=1$	Batch Norm/Bias	#Params
1	$6 \cdot 1 \cdot 8 + 1 \cdot 8$	$5 \cdot 1 \cdot 8 + 1 \cdot 8$	2 · 8	120
2	$6 \cdot 8 \cdot 8 + 8 \cdot 8$	$5 \cdot 8 \cdot 8 + 8 \cdot 8$	2 · 16	864
3	$6 \cdot 8 \cdot 16 + 8 \cdot 16$	$5 \cdot 8 \cdot 16 + 8 \cdot 16$	2 · 16	1696
4	$6 \cdot 16 \cdot 16 + 16 \cdot 16$	$5 \cdot 16 \cdot 16 + 16 \cdot 16$	2 · 32	3392
5	$6 \cdot 16 \cdot 35 + 16 \cdot 35$	$5 \cdot 16 \cdot 35 + 16 \cdot 35$	2 · 35	7350
6	$6 \cdot 35 \cdot 35 + 35 \cdot 35$	$5 \cdot 35 \cdot 35 + 35 \cdot 35$	2 · 70	16065
7	$6 \cdot 35 \cdot 10$	$5 \cdot 35 \cdot 10$	10	3860
Total				33347

Table 4. Number of parameters for H-Net.