ROTATION EQUIVARIANT NETWORKS VIA CONIC CONVOLUTION AND THE DFT

Anonymous authors
Paper under double-blind review

ABSTRACT

Performance of neural networks can be significantly improved by encoding known invariance for particular tasks. In particular, to aid convolutional neural networks in learning rotation invariance, we consider a simple, efficient conic convolutional scheme that encodes rotational equivariance, along with a method for integrating the magnitude response of the 2D-discrete-Fourier transform (2D-DFT) to encode global rotational invariance. We call our new method the *Conic Convolution and DFT Network* (CFNet). We evaluated the efficacy of CFNet as compared to a standard CNN and group-equivariant CNN (G-CNN) for several different image classification tasks and demonstrated improved performance, including classification accuracy, computational efficiency, and its robustness to hyperparameter selection. Taken together, we believe CFNet represents a new scheme that has the potential to improve many imaging analysis applications.

1 INTRODUCTION

Though the appeal of neural networks is their versatility for arbitrary classification tasks, there is still much benefit in designing them for particular problem settings. In particular, their effectiveness can be greatly increased by encoding invariance to uniformative augmentations of the data (LeCun et al., 1989). If such invariance is not explicitly encoded, the network must learn it from the data, perhaps with the help of data augmentation, requiring more parameters and thereby increasing its susceptibility to overfitting.

A key invariance inherent to several computer vision settings, including satellite imagery and all forms of microscopy imagery, is *rotation* (Cheng et al., 2016; Boland & Murphy, 2001). Recently, there have been a variety of proposed approaches for encoding rotation equivariance and invariance, the most promising of which have formulated convolution over groups (Cohen & Welling, 2016; Weiler et al., 2018). Notably, G-CNNs have been applied to several biological imaging tasks, producing state-of-the-art results (Weiler et al., 2018; Bekkers et al., 2018; Li et al., 2018).

Here we propose a new rotation-equivariant convolutional scheme, called *conic convolution*, which, in contrast to group convolution, encodes equivariance while still operating over only the spatial domain. Rather than convolving each filter across the entire image, as in standard convolution, rotated filters are convolved over corresponding conic regions of the input feature map that emanate from the origin, thereby transforming rotations in the input directly to rotations in the output. This scheme is intuitive, simple to implement, and computationally efficient. We also show that the method yields improved performance over group convolution on several relevant applications.

Additionally, we propose the integration of the magnitude response of the 2D-discrete-Fourier transform (2D-DFT) into a transition layer between convolutional and fully-connected layers to encode rotational invariance. Though the insight of using the DFT to encode rotational invariance has been employed for texture classification using wavelets (Do & Vetterli, 2002; Jafari-Khouzani & Soltanian-Zadeh, 2005; Ojala et al., 2002; Charalampidis & Kasparis, 2002) and for general image classification (Schmidt & Roth, 2012), as of yet, its application to CNNs has been overlooked. As in these prior works, rotations of the input are transformed to circular shifts, to which the magnitude response of the 2D-DFT is invariant, in the transformed space. Most other recently proposed rotation-invariance CNNs impose this invariance by applying a permutation-invariant operation, such as the average or maximum, over the rotation group, but since this operation is applied for each filter individually, possibly valuable pose information between filters is lost. In contrast, the 2D-DFT is able

to integrate mutual pose information between different filter responses, yielding richer features for subsequent layers.

We demonstrate the effectiveness of these two novel contributions for various applications: classifying rotated MNIST images, classifying synthetic images that model biomarker expression in microscopy images of cells, and localizing proteins in budding yeast cells (Kraus et al., 2017). We show that CFNet improves classification accuracy generally over the standard raster convolution formulation and over the equivariant method of G-CNN across these settings. We also show that the 2D-DFT clearly improves performance across these diverse data sets, and that not only for conic convolution, but also for group convolution. Source code for the implementation of CFNet will be made available on GitHub.

2 Related Work

Cohen & Welling (2016) introduced G-CNNs by formulating convolution over groups, including rotation, translation, and flips, for neural networks, which has inspired many subsequent improvements. By convolving over groups, equivariance to these groups is maintained throughout the convolutional layers, and invariance is enforced at the end of the network by pooling over groups. This work was improved upon by the design of steerable filters (Weiler et al., 2018) for convolution, similar to those proposed by Worrall et al. (2017), which allow for finer sampling of rotations of filters without inducing artifacts. Steerable filters were first proposed by Freeman & Adelson (1991) and had been explored previously for image classification (Liu et al., 2014), but as shallow features in the context of HOG descriptors.

An alternative means of encoding rotational equivariance is to transform the domain of the image to an alternative domain, such as the log-polar domain (Schmidt & Roth, 2012; Henriques & Vedaldi, 2017) in which rotation becomes some other transformation that is easier to manage, in this case, translations. The suitability of this transformation depends upon the signal of interest, since this warping will introduce distortion, as pixels near the center of the image are sampled more densely than pixels near the perimeter. In addition, its stability to translations in the original domain is of concern. Our proposed CFNet, by convolving over conic regions, also encodes global rotation equivariance about the origin, but without introducing such distortion, which greatly helps mitigate its susceptibility to translation. The recently developed spatial transform layer (Jaderberg et al., 2015) and deformable convolutional layer (Dai et al., 2017) allow the network to learn non-regular sampling patterns and can potentially help learning rotation invariance, though invariance is not explicitly enforced, which would most likely be a challenge for tasks with small training data.

A simple means for achieving rotation equivariance and invariance was proposed by Dieleman et al. (2016), in which feature maps of standard CNNs are made equivariant or invariant to rotation by combinations of cyclic slicing, stacking, rolling, and pooling. RotEqNet (Marcos et al., 2017) improved upon this idea by storing, for each feature map for a corresponding filter, only the maximal response across rotations and the value of the corresponding rotation, to preserve pose information. This approach yielded improved results and considerable storage savings over (Dieleman et al., 2016) and G-CNN. These methods are most similar to our proposed conic convolution. However, in contrast, our method applies each filter only at the appropriate rotation within each conic region, which further saves on storage.

To enforce rotation invariance, as noted, most of the previous methods apply some permutation-invariant, or pooling, operation over rotations. Cheng et al. (2016) recently proposed a strategy of encouraging a network to learn a rotation invariant transform, and follow-up work improved this learning process by incorporating a Fisher discriminant penalty (Cheng et al., 2018). However, the convolutional layers of the network do not maintain the property of rotation equivariance with the input image, which requires that the network learn this equivariance and could therefore hinder performance. Also, learning such a transform that generalizes to unseen data could prove difficult for settings with limited training data. Schmidt & Roth (2012) previously proposed the 2D-DFT for rotational invariance. However, no method has yet been proposed to integrate the 2D-DFT into a rotation-equivariant CNN.

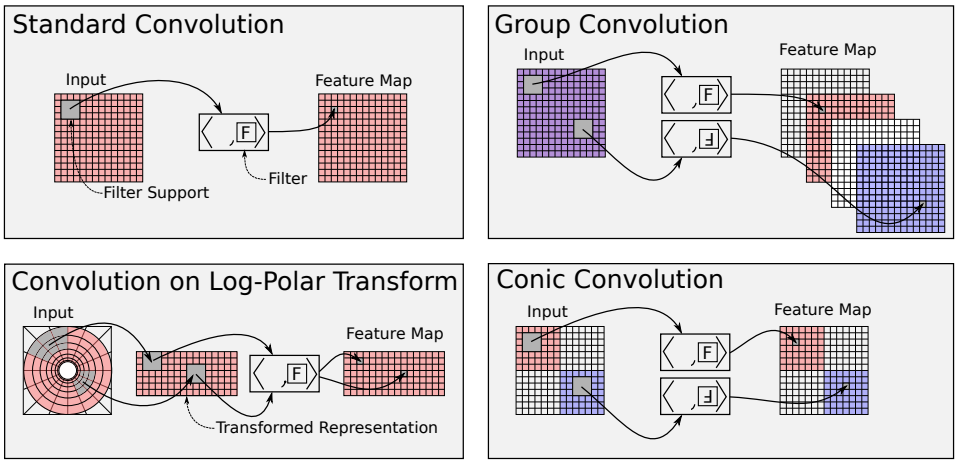


Figure 1: Comparison of convolution schemes. The domain of filter 'F' in the input and its corresponding outputs in the feature map are colored red. That of the rotation of 'F' by 180 degrees is colored blue. The local support on the domain for the convolution at a few points for each scheme are shown in gray. Conic convolution, with rotations of 90 degrees in this example, encodes rotation equivariance without introducing distortion to the support of the filter in the original domain (unlike the log-polar transform) and without requiring additional storage for feature maps (unlike group convolution). The example shown for group convolution is the first layer of a G-CNN, mapping from $\mathbb{Z}^2$ to the roto-translation group.

3 Formulation of Rotation Equivariance and Invariance

3.1 Rotation-Equivariant Quadrant Convolutional Layers

We begin our formulation with a simpler, special case of conic convolution, which we call *quadrant convolution*. Its only difference from standard convolution is that the filter being convolved is rotated by $r\pi/2$, $r \in \{0, 1, 2, 3\}$, depending upon the corresponding quadrant of the domain. We show that for quadrant convolution, rotations of $\pi/2$ in the input are straightforwardly associated with rotations in the output feature map, which is a special form of equivariance called *same-equivariance* (as coined by Dieleman et al. (2016)).

For convenience, we represent feature maps, of dimension K, $f : \mathbb{Z}^2 \to \mathbb{R}^K$, and filters, $\phi : \mathbb{Z}^2 \to \mathbb{R}^K$, of a network as functions over 2D space, as in (Cohen & Welling, 2016). Relevant to our formulation is the set, or group, G of two-dimensional rotation matrices of $\pi/2$, which can be easily parameterized by $g(r)$, and which acts on points in $\mathbb{Z}^2$ by matrix multiplication, i.e, for a given point $x = (u, v) \in \mathbb{Z}^2$,

$$g(r)x = \begin{bmatrix} \cos(r\pi/2) & \sin(r\pi/2) \\ \sin(r\pi/2) & \cos(r\pi/2) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}. \tag{1}$$

Let T_g denote the transformation of a function by a rotation in G, where $T_g f(x) = f(g^{-1}x)$ applies the inverse of g to an element of the domain of f. For an operation $\Phi : \mathcal{F} \to \mathcal{F}$, $\mathcal{F}$ being the set of K-dimensional functions f, to exhibit same-equivariance, applying rotation either before or after the operation yields the same result, i.e.

$$T_g \Phi(f) = \Phi(T_g f). \tag{2}$$

We now define quadrant convolution. The expression for convolution in a standard CNN is given by

$$\phi * f(x) = \sum_k \sum_{x' \in \mathbb{Z}^2} \phi_k(x') f_k(x - x'). \tag{3}$$

As noted in Cohen & Welling (2016), standard convolution does not exhibit rotational equivariance unless certain constraints on the filters are met.

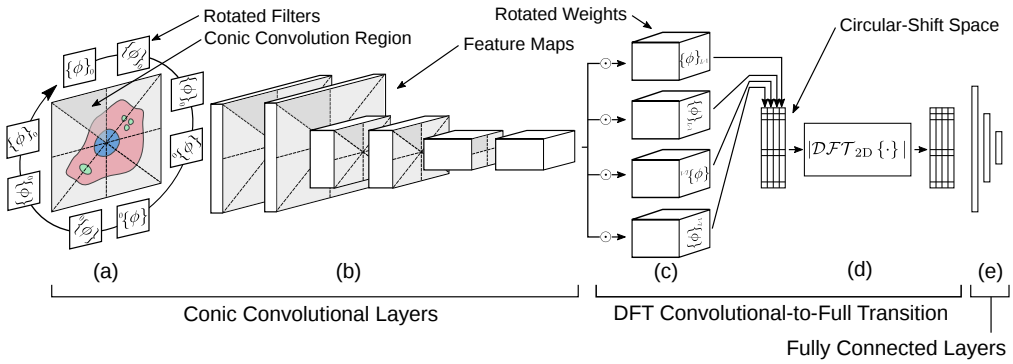


Figure 2: The overall architecture of the proposed CFNet. (a) Filtering the image by various filters at rotations in corresponding conic regions preserves rotation-equivariance. (b) Subsequent convolutional feature maps are filtered similarly. Rotation-invariance is encoded by the transition from convolutional to fully-connected layers, which consists of (c) element-wise multiplication and sum, denoted by , with rotated weight tensors, transforming rotation to circular shift, and (d) application of the magnitude response of the 2D-DFT to encode invariance to such shifts. (e) This output is reshaped and passed through the final, fully-connected layers.

Quadrant convolution can be interpreted as weighting the convolution for each rotation with a function $\omega\colon \mathsf{Z}^2 \;!\; [0,1]$ that simply "selects" the appropriate quadrant of the domain, which we define as

$$\omega_q(u,v) \;,\; \begin{cases} 1 & u > 0, v \quad 0, \\ 1/4 & (u,v) = (0,0), \\ 0 & \text{else.} \end{cases} \tag{4}$$

Since the origin does not strictly belong to a particular quadrant, it is handled simply by averaging the response of the filter at all four rotations. Boundary values are assigned arbitrarily, but consistently, by the placement of the equality for either u or v. The output of the layer is then given by

$$(f) \;,\; \sum_{g2G} [T_g\omega]\,[[T_g\phi] \quad f]\,. \tag{5}$$

Example convolutional regions with appropriate filter rotations are shown in Fig. 1.

The equivariance property is established (see Appendix) independent of the definition of ω, yet its definition will greatly influence the performance of the network. For example, if ω is simply the constant $1/4$, we have the simple example of equivariance mentioned above, equivalent to averaging the filter responses.

3.2 Generalization to Conic Convolutional Layers

The above formulation can be generalized to *conic convolution* in which the rotation angle is decreased by an arbitrary factor of $\pi/2R$, for some positive integer R, instead of being fixed to $\pi/2$. Rather than considering quadrants of the domain, we can consider conic regions emanating from the origin, defined by

$$C = \left\{(x,y)\ 2\ \mathsf{Z}^2 \colon 0 \quad \operatorname{arccot}(x/y) + \pi \mathsf{I}(y < 0) < \frac{\pi}{2R}\right\}, \tag{6}$$

where $\mathsf{I}(\,)$ is the indicator function. The weighting function is changed to have value one only over this conic region:

$$\omega_R(u,v) \;,\; \begin{cases} 1 & (u,v)\ 2\ C, \\ 1/4R & (u,v) = (0,0), \\ 0 & \text{else,} \end{cases} \tag{7}$$

of which $\omega_1 = \omega_q$ is a special case.

If we consider feature maps to be functions over the continuous domain $\mathbb{R}^2$, instead of $\mathbb{Z}^2$, and define the group G_R, with parameterization

$$g_R(r)x = \begin{bmatrix} \cos(r\pi/2R) & \sin(r\pi/2R) \\ \sin(r\pi/2R) & \cos(r\pi/2R) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}, \tag{8}$$

for $r \in \{0, 1, \ldots, 4R - 1\}$ and $x = (u, v) \in \mathbb{R}^2$, it is easy to show similarly as above that

$$_R(f) \triangleq \sum_{g \in G_R} [T_g \omega_R] [[T_g \phi] \star f] \tag{9}$$

is equivariant to G_R.

However, due to subsampling artifacts when discretizing $\mathbb{R}^2$ to $\mathbb{Z}^2$, as in an image, rotation equivariance for arbitrary values of R cannot be guaranteed and can only be approximated. In particular, the filters will have to be interpolated for rotations that are not a multiple of $\pi/2$. In our experiments, we chose nearest neighbor interpolation, which at least preserves the energy of the filter under rotations.

This defect notwithstanding, it can be shown that conic convolution maintains equivariance to rotations of $\pi/2$, and as our experiments show in the following section, the approximation of finer angles of rotation can still improve performance. Additionally, we note that R need not be the same for each layer, and it may be advantageous to use a finer discretization of rotations for early layers, when the feature maps are larger, and gradually decrease R.

3.3 Non-Linear Operations

A note must be made about subsequent nonlinear operations for a convolutional layer. It is typical in convolutional networks to perform subsampling, either by striding the convolution or by spatial pooling, to reduce the dimensionality of subsequent layers. Again, due to downsampling artifacts, rotational equivariance to rotations smaller than $\pi/2$ is not guaranteed. However, given that the indices of the plane of the feature map are in $\mathbb{Z}^2$ and are therefore centered about the origin, a downsampling of $D \in \mathbb{Z}_{>0}$ can be applied while maintaining rotational equivariance for rotations of $\pi/2$, regardless of the choice of R. After subsampling, the result is passed through a non-linear activation function $\sigma : \mathbb{R} \to \mathbb{R}$, such as ReLU, with an added offset $c_k \in \mathbb{R}$.

3.4 Efficiency of Computation and Memory Usage

In theory, the response for each rotation in conic convolution is only needed over its corresponding conic region. However, since GPUs are more efficient operating on rectangular inputs, it is faster to compute the convolution over each quadrant in which the conic region resides. In current neural network libraries, the output of conic convolution can be achieved by convolving over the corresponding quadrant, multiplying by the weighting function, summing the responses is in each quadrant together, and then concatenating the responses of quadrants. For the special case of quadrant convolution, this process incurs negligible additional computation beyond standard convolution. Additionally, conic convolution produces only one feature map per filter as in standard convolution and therefore incurs no additional storage costs, in contrast to G-CNN and cyclic slicing, which both produce one map per rotation (Cohen & Welling, 2016; Dieleman et al., 2016), and two for RotEqNet, one for the filter response and one for the orientation (Marcos et al., 2017).

3.5 Rotation-Invariant Transition using the Magnitude of the 2D-DFT

After the final convolutional layer of a CNN, some number of fully-connected layers will be applied to combine information from the various filter responses. In general, fully-connected layers will not maintain rotation equivariance or invariance properties. In a *fully-convolutional* network, convolution and downsampling are applied until the spatial dimensions are eliminated and the resulting feature map of the final convolutional layer is merely a vector, with dimension equal to the number of filters.

Rather than encoding invariance for each filter separately, as in most other recent works (Cohen & Welling, 2016; Weiler et al., 2018), we consider instead to transform the collective filter responses to a space in which rotation becomes circular shift so that the 2D-DFT can be applied to encode

invariance. The primary merit of the 2D-DFT as an invariant transform is that each output node is a function of every input node, and not just the nodes of a particular filter response, thereby capturing mutual information across responses.

Since the formulation of this transition involves the DFT, which is defined only for finite-length signals, we switch to represent feature maps as tensors, rather than functions. We denote the feature map generated by the penultimate convolutional layer by $f \in \mathbb{R}^{M \times M \times K}$, where $M \in \mathbb{Z}_{>1}$.

In a fully-convolutional network, the final convolutional layer is in reality just a fully-connected layer, in which the input f is passed through N fully-connected filters, $\phi^{(n)} \in \mathbb{R}^{M \times M \times K}$, $n \in \{0, 1, \ldots, N - 1\}$. The operation of this layer can be interpreted as the inner product of the function and filter, $\langle \phi^{(n)}, f \rangle$. If we again consider rotations of the filter from the group G_R,

$$(n, r) \mapsto \langle T_{g_R(r)} \phi^{(n)}, f \rangle, \tag{10}$$

this is equivalent to the first layer of a G-CNN, mapping from the spatial domain to G_R (though this group does not include the translation group since the convolution is only applied at the origin), and rotations of the final convolutional layer f will correspond to permutations of G_R, which are just circular shifts in of the second dimension of the matrix .

The magnitude response of the 2D-DFT can be applied to to transform these circular shifts to an invariant space,

$$\text{2D-DFT}\{f \star g\}(n, r) = \left| \sum_{n'=0}^{N-1} \sum_{r'=0}^{R-1} \quad (n', r') e^{-j2\pi \left(\frac{n' n}{N} + \frac{r' r}{4R} \right)} \right|. \tag{11}$$

This process of encoding rotation invariance corresponds to the 'Convolutional-to-Full Transition' in Fig. 2. The result is then vectorized and passed into fully-connected layers that precede the final output layer, as in a standard CNN. We note that it helped in practice to apply batch normalization after vectorizing, since the output of the magnitude of the 2D-DFT will not be normalized as such.

The 2D-DFT, as a rotation invariant transform, can also be integrated into other rotation-equivariant networks, such as G-CNN. At the final layer of a fully-convolutional G-CNN, since the spatial dimension has been eliminated through successive convolutions and spatial downsampling, rotation is encoded along contiguous stacks of feature maps $f \in \mathbb{R}^{L \times 4}$ of each filter at four rotations. In this way, rotations similarly correspond to circular shifts in the final dimension. This representation is then passed through the 2D-DFT, as in Eqn. 11.

4 Results

4.1 Application to Rotated MNIST

The rotated MNIST data set (Larochelle et al., 2007) has been used as a benchmark for several previous works on rotation invariance. As in previous works, to tune the parameters of each method, we first trained various models on a set of 10,000 images, using training augmentation of rotations of arbitrary angles as in Cohen & Welling (2016)[1], and then selected the best model based on the accuracy on a separate validation set of 5,000 images.

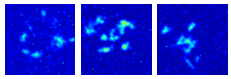
Our best CFNet architecture consisted of six convolution layers; the first were conic convolutions of $R = 8$ for the first three layers and $R = 4$ for the next four, with spatial max-pooling after the second layer. We used a filter size of three pixels, with 15 filters per layer. The final convolutional layer was the DFT transition layer as described in the previous section, which was followed by an output layer of ten nodes. This architecture was similar in terms of number of layers and filters per layer as that of the G-CNN of Cohen & Welling (2016). To evaluate the G-CNN with the DFT, the only changes we made from the reported architecture for G-CNN was to reduce the number of filters for each layer to 7, to offset the addition of the 2D-DFT, which was applied to the output of the final convolutional layer.

The results on a held-out set of 50,000 test images are shown in Table 1. Adding the DFT transition to the output of G-CNN reduces the test error by 0.28%, demonstrating the value of incorporating

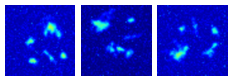
[1]Though the paper did not state the use of training augmentation, code posted by the authors at https://github.com/tscohen/gconv_experiments indicates rotations of arbitrary angles were used.

Table 1: Test error on the rotated MNIST data set.

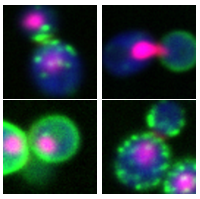
Algorithm	**Test Error (%)**
Schmidt & Roth (2012)	3.98
Cohen & Welling (2016) (CNN)	5.03
Cohen & Welling (2016) (G-CNN)	2.28
G-CNN + DFT	2.00
CFNet	1.75
Worrall et al. (2017) (H-Net)	1.69



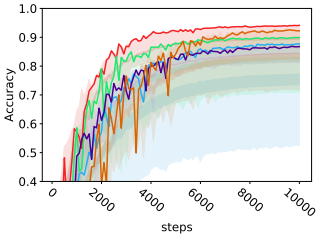
(a) Example images of three of the 50 classes.



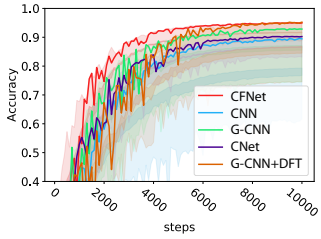
(b) Rotated examples from a single class.



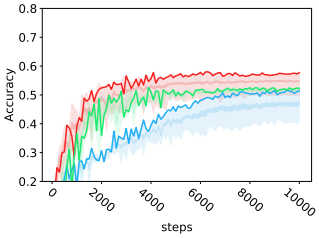
(e) Example images of cells of four of the 22 yeast phenotypes from Kraus et al. (2017).



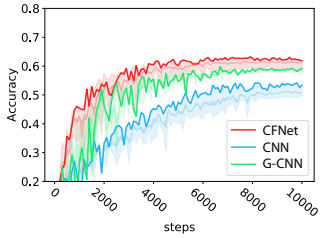
(c) $N = 50$.



(d) $N = 100$.



(f) $N = 50$



(g) $N = 100$

Figure 3: Comparison of CFNet CNet, G-CNN, G-CNN+DFT, and a standard CNN on the GMM synthetic biomarker images and on images of protein localization. (a,b) Example images, shown as heat maps for detail, showing inter- and intra-class variation. (e) Example images from yeast cell phenotype classes. Testing classification accuracy of methods on synthetic GMM images (c,d) and protein localization (f,g) with varying numbers N of training examples per class.

mutual rotational information between filters when encoding invariance. The replacement of group convolution with conic convolution in CFNet leads to an even further reduction in error of 0.25%. Even with its simple conic convolutional scheme, CFNet is able to perform comparably to H-Net[2], which constructs filters from the circular harmonic basis and operates on complex feature maps.

4.2 Application to Synthetic Biomarker Images

In order to explicitly control the manifestation of rotational invariance, we created a set of synthetic images, based upon Gaussian-mixture models (GMMs), which can also be used to emulate real-world microscopy images of biological signals (Zhao & Murphy, 2007). Example synthetic images from across and within classes are shown in Fig. 3a and Fig. 3b, respectively. We defined 50 distribution patterns and generated 50 and 100 examples per class for training and 200 examples per class for testing. Each class was defined by a mixture of ten Gaussians. The image size was 50 pixels. A batch size of 50 examples, a learning rate of 5 10 ^{3}, and a weight decay ℓ_2 penalty of 5 10 ^{4} were used during training. To help all methods, we augmented the training data by rotations and random jitter of up to three pixels, as was done during image generation.

[2]This result is without training augmentation; since H-Net can learn equivariance to arbitrary angles, augmenting with rotations might not improve performance.

Classification accuracies on the test set over training steps for various numbers of training samples, denoted by N, for several methods are shown in Figs. 3c-3d. A variety of configurations were trained for each network, and each configuration was trained three times. The darkest line shows the accuracy of the configuration that achieved the highest moving average, with a window size of 100 steps, for each method. The spread of each method, which is the area between the point-wise maximum and minimum of the error, is shaded with a light color, and three standard-deviations around the mean is shaded darker.

We observe a consistent trend of CFNet outperforming G-CNN, which in turn marginally outperforms the CNN, both in overall accuracy and in terms of the number of steps required to attain that accuracy. Additionally, the spread of CFNet is mostly above even the best performing models of G-CNN and the CNN, demonstrating that an instance of CFNet will outperform other methods even if the best set of hyperparameters has not been chosen. We also included a network consisting of conic convolutional layers, but without the DFT, noted as 'CNet', to show the relative merit of the DFT. CNet performs comparably to the standard CNN while requireing significantly less parameters to attain the same performance, though the true advantage of conic convolution is shown when integrated with the DFT to achieve global rotation invariance. In comparison, including the 2D-DFT increases the performance of G-CNN, to a comparable level with CFNet in fact, though it does not train as quickly.

4.3 Application to Protein Localization in Budding Yeast Cells

We extended our analysis to real biomarker images of budding yeast cells (Kraus et al., 2017), shown in Fig. 3e. Each image consists of four stains, where blue shows the cytoplasmic region, pink the nuclear region, red the bud neck, and green the protein of interest. The classification for each image is the cellular subcompartmental region in which the protein is expressed, such as the cell periphery, mitochondria, or eisosomes, some of which exhibit very subtle differences.

Fig. 3f-g shows the results of using CFNet, G-CNN, and a standard CNN to classify the protein localization for each image. We used the same architecture as reported in Kraus et al. (2017) for all methods, except that we removed the last convolutional layer and reduced the number of filters per layer by roughly half for CFNet and G-CNN, to offset for encoding of equivariance and invariance. The same training parameters and data augmentation were used as for the synthetic data, except that a dropout probability of 0.8 was applied at the final layer and the maximum jitter was increased to five pixels, since many examples were not well-centered. For each method, several iterations were run and the spread and the best performing model are shown. Again, CFNet outperforms G-CNN and a standard CNN, when the number of training examples per class is either 50 or 100 (see Fig. 3b-c), demonstrating that the gains of the 2D-DFT and proposed convolutional layers translate to real-world microscopy data. We note that the best reported algorithm that did not use deep learning, called ensLOC (Chong et al., 2015; Koh et al., 2015), was only able to achieve an average precision of 0.49 for a less challenging set of yeast phenotypes and with 20,000 samples, whereas all runs of CFNet achieved an average precision of between 0.60 - 0.67 with 10% of the data used for training.

5 Conclusion

We have demonstrated the effectiveness of enforcing rotation equivariance and invariance in CNNs by means of the proposed conic convolutional layer and the 2D-DFT, even for group convolution. We believe that the proposed enhancements to the standard CNN will have much utility for future applications in relevant problem settings, in particular, high-throughput molecular and cellular imaging data (Rozenblatt-Rosen et al., 2017), where training data is usually sparse, especially for rare cellular events. In future work, we believe CFNet could be even further improved by constructing steerable filters, as in (Weiler et al., 2018), to overcome sampling artifacts from rotating filters.

References

Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen AJ Eppenhof, and Josien PW Pluim. Roto-translation covariant convolutional networks for medical image analysis. *arXiv preprint arXiv:1804.03393*, 2018.

Michael V Boland and Robert F Murphy. A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of hela cells. *Bioinformatics*, 17(12):1213–1223, 2001.

Dimitrios Charalampidis and Takis Kasparis. Wavelet-based rotational invariant roughness features for texture classification and segmentation. *IEEE Transactions on Image Processing*, 11(8):825–837, 2002.

Gong Cheng, Peicheng Zhou, and Junwei Han. Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 54(12):7405–7415, 2016.

Gong Cheng, Junwei Han, Peicheng Zhou, and Dong Xu. Learning rotation-invariant and fisher discriminative convolutional neural networks for object detection. *IEEE Transactions on Image Processing*, 2018.

Yolanda T Chong, Judice LY Koh, Helena Friesen, Supipi Kaluarachchi Duffy, Michael J Cox, Alan Moses, Jason Moffat, Charles Boone, and Brenda J Andrews. Yeast proteome dynamics from single cell imaging and automated analysis. *Cell*, 161(6):1413–1424, 2015.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pp. 2990–2999, 2016.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *CoRR, abs/1703.06211*, 1(2):3, 2017.

Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pp. 1889–1898. JMLR. org, 2016.

Minh N Do and Martin Vetterli. Rotation invariant texture characterization and retrieval using steerable wavelet-domain hidden markov models. *IEEE Transactions on Multimedia*, 4(4):517–527, 2002.

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

João F Henriques and Andrea Vedaldi. Warped convolutions: Efficient invariance to spatial transformations. In *International Conference on Machine Learning*, pp. 1461–1469, 2017.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

Kourosh Jafari-Khouzani and Hamid Soltanian-Zadeh. Rotation-invariant multiresolution texture analysis using radon and wavelet transforms. *IEEE transactions on image processing*, 14(6): 783–795, 2005.

Judice L. Y. Koh, Yolanda T. Chong, Helena Friesen, Alan Moses, Charles Boone, Brenda J. Andrews, and Jason Moffat. Cyclops: A comprehensive database constructed from automated analysis of protein abundance and subcellular localization patterns in saccharomyces cerevisiae. *G3: Genes, Genomes, Genetics*, 5(6):1223–1232, 2015.

Oren Z Kraus, Ben T Grys, Jimmy Ba, Yolanda Chong, Brendan J Frey, Charles Boone, and Brenda J Andrews. Automated analysis of high-content microscopy data with deep learning. *Molecular systems biology*, 13(4):924, 2017.

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 *Proceedings of the 24th international conference on Machine learning*, pp. 473–480. ACM, 2007.

Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, pp. 143–155, 1989.

Xiaomeng Li, Lequan Yu, Chi-Wing Fu, and Pheng-Ann Heng. Deeply supervised rotation equivariant network for lesion segmentation in dermoscopy images. *arXiv preprint arXiv:1807.02804*, 2018.

Kun Liu, Henrik Skibbe, Thorsten Schmidt, Thomas Blein, Klaus Palme, Thomas Brox, and Olaf Ronneberger. Rotation-invariant hog descriptors using fourier analysis in polar and spherical coordinates. *International Journal of Computer Vision*, 106(3):342–364, 2014.

Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *ICCV*, pp. 5058–5067, 2017.

Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

Orit Rozenblatt-Rosen, Michael JT Stubbington, Aviv Regev, and Sarah A Teichmann. The human cell atlas: from vision to reality. *Nature News*, 550(7677):451, 2017.

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

Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

Ting Zhao and Robert F Murphy. Automated learning of generative models for subcellular location: building blocks for systems biology. *Cytometry Part A*, 71(12):978–990, 2007.

APPENDIX A PROOF OF ROTATION EQUIVARIANCE

We note that the transform T_g distributes over both multiplication and convolution, i.e. $T_g[f_1 f_2] = [T_g f_1][T_g f_2]$ and $T_g[f_1 \quad f_2] = [T_g f_1] \quad [T_g f_2]$. From these properties, it is easy to show that such an operation is same-equivariant for rotation.

Theorem.

$$T_g \quad (f) = \quad (T_g f)$$

Proof.

$$\begin{aligned}(T_g f) &= \sum_{h2G} [T_h \omega] [[T_h \phi] \quad [T_g f]] = \sum_{h2G} [T_h \omega][T_g \left[[T_g \ ^1 T_h \phi] \quad f\right]] \\ &= \sum_{h2G} T_g([T_g \ ^1 T_h \omega] \left[\left[T_g \ ^1 T_h \phi\right] \quad f\right]) = T_g \sum_{h2G} [T_g \ ^1 T_h \omega] \left[\left[T_g \ ^1 T_h \phi\right] \quad f\right] \\ &\overset{(a)}{=} T_g \sum_{p2G} [T_p \omega] [[T_p \phi] \quad f] = T_g \quad (f)\end{aligned}$$

where for (a) we make a change of variables, combining g and h into p, where $p = g \ ^1 h$ and $T_g \ ^1 T_h = T_g \ _1 _h = T_p$. □

APPENDIX B GENERATION OF SYNTHETIC BIOMARKER EXAMPLES

Mathematically, each class $k \ 2 \ f1, \ldots, Kg$ is described by the parameters of the GMM: $_k = f\vec{\mu}_g, \quad _g g_{g=1}^G$, where $\vec{\mu}_g \ 2 \ [\ 1, 1]^2$ and the number of Gaussians per class is a parameter G of the data set. For simplicity, we consider the image $I : \mathsf{R}^2 \ ! \ \mathsf{R}$ to be nonzero only over a region slightly larger than the $[\ 1, 1]^2$ box, so that it captures the majority of points generated by the Gaussians.

To generate a sample image from the generating distribution, first, a constant background intensity is set for the image according to $b \quad \mathsf{Exp}(0, \lambda_B)$, so $I(p) = b, \ 8p \ 2 \ \mathsf{R}^2$. Then a random angle $\theta \quad \mathsf{Uniform}[0, 2\pi]$ is drawn to determine the rotation of the image. The mean $\vec{\eta}_g \quad \mathsf{N}(\vec{\mu}_g, \quad)$ for each Gaussian of the class is drawn from an underlying Gaussian with mean $\vec{\mu}_g$, which introduces some small jitter of the relative locations of the Gaussians. A number $n_g \quad \mathsf{N}(\mu_{n;g}, \sigma_n)$ of points fpg in $[\ 1, 1]^2$, which vary for each Gaussian, are drawn from this Gaussian according to $p \quad \mathsf{N}\left(R\vec{\eta}_g, R \quad _g R \ ^1\right)$, where the realized mean and covariance have been rotated by θ by the rotation matrix:

$$R = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{12}$$

For each point p, its corresponding intensity value is drawn according to $I(p) \quad \mathsf{Uniform}[\mu_I \quad m_I, \mu_I + m_I]$, replacing the background value. Having drawn all of the points, the image is smoothed with a Gaussian kernel with variance σ_s to emulate the point-spread function of the imager and pixel noise is added: $I(p) = I(p) + \mathsf{Exp}(0, \lambda_I)$. To simulate camera jitter, the image is translated by a random offset of up to three pixels.