Towards Learning Affine-Invariant Representations via Data-Efficient CNNs

Wenju Xu[1]*, Guanghui Wang[1], Alan Sullivan[2], Ziming Zhang[3]
[1]University of Kansas, KS
[2]Mitsubishi Electric Research Laboratories (MERL), MA
[3]Worcester Polytechnic Institute, MA
{xuwenju, ghwang}@ku.edu, sullivan@merl.com, zzhang15@wpi.edu

Abstract

*In this paper we propose integrating a priori knowledge into both design and training of convolutional neural networks (CNNs) to learn object representations that are invariant to affine transformations (*i.e. *translation, scale, rotation). Accordingly we propose a novel multi-scale maxout CNN and train it end-to-end with a novel rotation-invariant regularizer. This regularizer aims to enforce the weights in each 2D spatial filter to approximate circular patterns. In this way, we manage to handle affine transformations in training using convolution, multi-scale maxout, and circular filters. Empirically we demonstrate that such knowledge can significantly improve the data-efficiency as well as generalization and robustness of learned models. For instance, on the Traffic Sign data set and trained with only 10 images per class, our method can achieve 84.15% that outperforms the state-of-the-art by 29.80% in terms of test accuracy.*

1. Introduction

Recently Sabour *et al*. [34] proposed a new network architecture, CapsNet, and a dynamic routing training algorithm to connect the capsules [17], a new type of neurons that output vectors rather than scalars in conventional neurons, in two adjacent layers and group similar features in higher layers. Later on Hinton *et al*. [16] proposed another EM-based routing-by-agreement algorithm for training CapsNet. In contrast to CNNs, the intuition behind CapsNet is to achieve "viewpoint invariance" in recognizing objects for better generalization which is inspired by inverse graphics [15]. Technically, CapsNet not only predicts classes but also encodes extra information such as geometry of objects, leading to richer representation. For instance, in [16], 4×4 pose matrices are estimated to capture the spatial relations between the detected parts and a whole. Unlike CNNs the performance of CapsNet on real and more complex data has not been verified yet, partially due to the high computation that prevents it from being applicable widely.

In fact exploring such invariant representations for object recognition has a long history in the literature of both neural science and computer vision. For instance, in [21] Isik *et al*. observed that object recognition in the human visual system is developed in stages with invariance to smaller transformations arising before invariance to larger transformations, which supports the design of feed-forward hierarchical models of invariant object recognition. In computer vision part-based representation (*e.g*. [11]) is one of the most popular invariant object representations that considers an object as a graph where each node represents an object part and each edge represents the (spatial) relation between the parts. Conceptually part-based representation is view-invariant in 3D and *affine-invariant* (*i.e*. invariant to translation, scale, and rotation) in 2D. Although the complexity of part-based models in inference on general graphs could be very high [7], for tree structures such as star graphs this complexity can be linear to the number of parts [10]. Girshick *et al*. [12] has shown that such star graph part-based models can be interpreted as CNNs.

In this paper we aim to study the following problem: *Can we design and train CNNs to learn affine-invariant representations efficiently, effectively, and robustly?*

Motivation. Besides CapsNet, we are also partially motivated by the works such as [2, 50] that utilize *a priori* knowledge as guidance to design and train neural networks efficiently and effectively. For instance, [2] proposed the notion of "neural module" to conduct certain semantic functionality using deep learning for visual question answering. Such modules can be reusable to comprise complex networks to perform certain tasks. The semantics and the network design here come from the compositional linguistic structure of questions. Thanks to these modules, the network design is much more understandable by checking whether the outputs of a module follow what we expect. [50] proposed encoding network weights as well as the architecture into a Tikhonov regularizer by lifting the ReLU

*Work was done during an internship at MERL.

activations, and accordingly developed a block coordinate descent algorithm for fast training of deep models.

In contrast to *a posteriori* knowledge such as visualization of learned filters in [47], a priori knowledge based approaches are more likely to be model-driven so that one can derive by reasoning alone, rather than being data-driven, in terms of building automatic systems such as neural networks. In this way, the networks with a priori knowledge are expected to be much easier to be understood by human, and their performance is more predictable and robust.

Contributions. Thanks to the convolution, CNNs are translation equivariant. This capability has contributed significantly to their widespread success. They, however, are not efficient or effective to capture the scaled or rotated objects, and thus enhancing CNNs with the capability of learning scale-invariant and rotation-invariant features is very challenging but appealing.

In this paper we design a novel deep multi-scale maxout [13] CNN to learn scale-invariant representations. We then propose training this network end-to-end with a novel rotation-invariant regularizer. To our best knowledge, we are the first to propose such regularization for handling rotation in deep learning. Note that we take the multi-scale maxout block and the regularizer as a priori knowledge for learning affine-invariant representations. Empirically we demonstrate the benefit of integrating such knowledge with network design and training, leading to better generalization, data-efficiency, and robustness of deep models than the state-of-the-art in learning affine-invariant representations.

2. Related Work

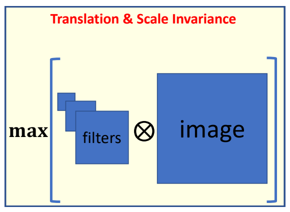
Scale-Invariant Networks. One simple way to handle the scale issue is using image pyramid in deep learning [31]. Some works [44, 24, 38] are particularly interested in extracting scale-invariant features from the networks. More broadly, multi-scale convolutional filters (or multi-kernels) are employed in networks [30, 39, 3, 28]. The inception module in GoogLeNet [37] is able to capture multi-scale information with maxout units. A similar idea has been explored in TI-Pooling [26]. ResNet [14] manages to capture multi-scale information using skip connection. Multi-scale DenseNet [19] proposes using a two-dimensional multi-scale convolutional network architecture that maintains coarse-level and fine-level features throughout the network. Note that with the increase of the number of hidden layers all the CNNs tend to extract deep features within multiple scales to a certain degree.

Rotation-Invariant Networks. Recently quite a few works focus on learning rotation-invariant features using deep networks. Cohen and Welling [6] proposed Group equivariant CNNs (GCNN) by exploiting larger groups of symmetries, including rotations and reflections, in the convolutional layers. Worrall *et al*. [42] proposed Harmonic Networks by replacing regular CNN filters with circular harmonics and returning a maximal response and orientation for every receptive field patch. Both works argue that rotating the data point is equivalent to rotating the filters. Therefore, they manage to learn rotation-invariant filters in a continuous space. In contrast, some other works such as [51, 49, 32, 18, 48, 33, 40] propose learning the filters in a discretized space by quantizing the rotation angles with predefined numbers (*e.g*. from 0 to 2π, step by $\frac{\pi}{4}$) so that the final features encode the rotation information. For instance, Rotation Equivariant Vector Field Networks (RotEqNet) [33] was proposed by applying each convolutional filter at multiple orientations and returning a vector field that represents magnitude and angle of the highest scoring orientation at every spatial location.

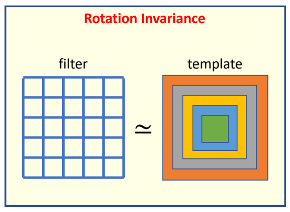
Interpretable Networks with A Priori Knowledge. Andreas *et al*. [2] proposed neural modules to mimic some basic semantic functionality using deep neural networks, based on which larger networks are constructed for specific tasks using the knowledge from natural language processing (NLP) such as grammar graphs as guidance. Belbute-Peres *et al*. [9] proposed embedding structured physics knowledge into larger systems as a differentiable physics engine that can be integrated as module in deep neural networks for end-to-end learning. Amos *et al*. [1] proposed using Model Predictive Control (MPC) as a differentiable policy class for reinforcement learning in continuous state and action spaces that leverages and combines the advantages of model-free and model-based approaches. They also showed that their MPC policies are significantly more data-efficient than a generic neural network.

Other Related Networks. Dilated convolution [45] supports exponential expansion of the receptive field (*i.e*. window) without loss of resolution or coverage and thus can help networks capture multi-scale information. Deformable Convolutional Networks (DCN) [8] proposed a more flexible convolutional operator that introduces pixel-level deformation, estimated by another network, into 2D convolution. Spatial Transformer Networks (STN) [22] learn affine-invariant representations by sequential applications of a localization network, a parameterized grid generator and a sampler. Dynamic Filter Networks (DFN) [23, 43] was proposed to learn to generate (local) filters dynamically conditioned on an input that potentially can be affine-invariant.

Data Augmentation. It is a well-known technique in deep learning for reducing the filter bias during learning by generating more (fake) data samples based on some predefined rules (or transformations) such as translation, scaling, rotation and random cropping. Trained with such augmented data, one can expect that the networks may be more robust to the transformations. For instance, TI-Pooling [26] assem-



(a) Multi-scale maxout block



(b) Regularization

Figure 1. To learn affine-invariant representations, we propose **(a)** a multi-scale maxout convolutional network block to handle translation and scale, and **(b)** a regularizer to handle rotation. We use (a) for constructing our network, and embed (b) into our learning.

bles all the transformed instances from the same data point in a pool and takes the maximal response for classification. STN [22] learns to predict a transformation matrix for each observation that can be used to augment data.

Loss Functions. From the perspective of the feature space, affine-invariant representations for an object under different transformations with translation, scale, and rotation should be mapped into a single point in the feature space ideally, or a compact cluster. To achieve this, several loss functions were proposed. For instance, the center loss [41] enforces the features from the same class to be close to the corresponding cluster center. Similar ideas have been explored in few-shot learning with neural networks [35] as well. In fact well-designed networks can generate compactly clustered features for each class with good discrimination, even if trained without such specific losses. Also such losses do not aim to learn affine-invariant features, explicitly or implicitly. Empirically we do not observe any improvement using the center loss over the cross-entropy loss, and thus we do not report the performance using the center loss.

In contrast to these previous works, we handle scale and rotation jointly in CNNs for learning affine-invariant representations. We introduce a priori knowledge into network design and training as interpretability in deep models. We demonstrate better generalization, data-efficiency, and robustness of our approach than the state-of-the-art networks.

3. Our Approach

Overview. To achieve translation and scale invariance, we propose a multi-scale maxout block as shown in Fig. 1(a), a set of filters with different predefined sizes are applied to images with convolution, and then the maxout operator is used to locate the maximum response per pixel among the filters. Mathematically this block can be formulated as

$$\max_{\omega\in\Omega} \left\{\omega \otimes \mathbf{I}_{ij}\right\}, \forall(i,j), \tag{1}$$

where $\otimes$ denotes the convolution operator, $\omega \in \Omega$ denotes a 2D spatial filter, $\mathbf{I}$ denotes an image, and $\omega \otimes \mathbf{I}_{ij}$ denotes the scalar output of the convolution at pixel (i,j).

In contrast to rotation-invariant networks such as RotEqNet, there is no rotation constraint on the design of network architectures including filters. Instead, we impose such constraint on learning with our rotation-invariant regularizer. Similar to other regularizers, ours encodes the prior knowledge of filters that we would like to learn (denoted as the template in Fig. 1(b)). Inspired by Harmonic Networks, ideally the learned filters should be symmetric along all possible directions, like circles. Due to the discretization of images, however, we propose an alternative to represent such symmetry that can be learned efficiently and effectively.

Learning Problem. In this paper we consider the following optimization problem:

$$\min_{\omega\in\Omega,\theta\in\Theta} \sum_i \ell\Big(y_i, \phi(x_i,\omega)\Big) + \lambda_1\mathcal{R}_1(\omega) + \lambda_2\mathcal{R}_2(\omega,\theta), \tag{2}$$

where $\{x_i, y_i\} \subseteq \mathcal{X} \times \mathcal{Y}$ denotes the training data with image $x_i \in \mathcal{X}, \forall i$ and its class label $y_i \in \mathcal{Y}$, $\omega \in \Omega$ denotes the parameters for the network defined by function $\phi : \mathcal{X} \times \Omega \to \mathcal{Y}$, $\theta \in \Theta$ denotes the templates in the feasible space Θ that ω should match with, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ denotes the loss function, $\mathcal{R}_1$ denotes the weight decay with ℓ_2 norm, $\mathcal{R}_2 : \Omega \times \Theta \to \mathbb{R}$ denotes the regularizer that measures the difference between ω and θ, and $\lambda_1, \lambda_2 \geq 0$ are predefined constants. Different from conventional CNNs, here we propose learning not only the network weights ω but also the matching templates θ within the feasible space Θ that encodes certain constraints on the templates such as symmetry. In the sequel we will explain how to effectively design a scale-invariant network ϕ, and how to efficiently construct a rotation-invariant regularizer $\mathcal{R}_2$.

3.1. Network Architecture

We illustrate our network in Fig. 2, where all the operations are basic and widely used in CNNs such as batch normalization (BN) [20], and "+" denotes one operation followed by the other. Due to the small image sizes (*e.g.* 32×32 pixels) in our experiments, we conduct downsampling for three times only using max-pooling. In each block the first convolutional layer is responsible for mapping the inputs into a higher dimensional space, *e.g.* $3 \to 32$, and the other two convolutional layers learn the (linear) transformation in the same space, *e.g.* $32 \to 32$. For grayscale images, the input dimension is changed from 3 to 1.

Different from existing networks such as GoogLeNet and TI-Pooling, we propose extracting features within different scales using a sequence of convolutional operations. Considering the trade-off between computational efficiency and accuracy, we only exploit three scales, *i.e.* $3 \times 3, 5 \times 5, 7 \times 7$, using fixed filter size of 3×3 in each convolutional layer, and use maxout to select a scale with the maximum response. This scale is taken as the best one to fit for the

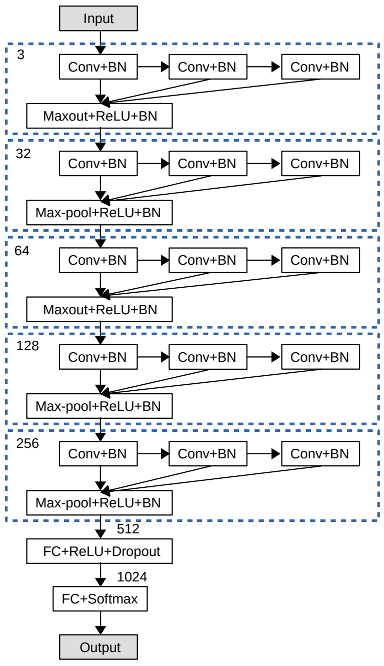


Figure 2. Illustration of the network we use in our experiments for learning affine-invariant features. Each dashed block is a multi-scale maxout block accounting for scale invariance, and the numbers here denote the default dimensions of inputs for the corresponding blocks and layers.

object. In fact we use two and three 3×3 convolutions to approximate the responses with filter sizes of 5×5 and 7×7, respectively, for efficient computation. With the increase of the network depth, information within larger scales (*i.e.* receptive field) can be extracted as well.

We also find that the network depth is more important than the network width w.r.t. the accuracy. It has been demonstrated in Wide Residual Networks (WRN) [46] that wider networks can improve the performance. In contrast to the parallel mechanism in WRN, in each block we apply convolutions sequentially. Note that the proposed mechanism can be integrated with other networks as well.

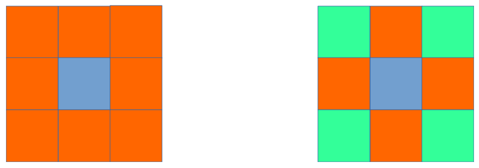
3.2. Training with Rotation-Invariant Regularizer

3.2.1 General Formulation

As illustrated in Fig. 1(b), in order to enforce the filters to satisfy certain spatial properties such as rotation invariance, the templates here need to be constructed in certain way to encode such properties. Therefore, we propose the following general formulation for rotation-invariant regularizers:

$$\mathcal{R}_2(\omega,\theta) = \mathbb{E}_{k\sim\mathcal{K}}\left[\sum_{m=-p_k}^{p_k}\sum_{n=-q_k}^{q_k} d\Big(\omega_k(m,n), \theta_k(h(m,n))\Big)\right], \tag{3}$$

where $k \in \mathcal{K}$ denotes the index of a 2D spatial filter, $\mathbb{E}_{k\sim\mathcal{K}}$ denotes the expectation over all 2D spatial filters, (m,n)



(a) (b)

Figure 3. Examples of weight patterns, defined by hash function h, that can be used to approximate circular patterns for rotation invariance. In each subfigure the same color denotes the same weight.

denotes the 2D-index of a weight in the k-th filter with size (M_k, N_k), $p_k = \left\lceil \frac{M_k}{2} \right\rceil, q_k = \left\lceil \frac{N_k}{2} \right\rceil$, $\lceil\cdot\rceil$ denotes the ceiling function, $d : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ denotes a distance function, $h : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ denotes a hash function that determines the weight pattern in the templates for matching, and correspondingly $\theta : \mathbb{R} \to \mathbb{R}$ is a learnable function.

Choices of Distance Function d. In general we do not have any explicit requirement on d. For instance, it can be ℓ_1-norm, ℓ_2-norm, or group sparsity norm such as $\ell_{2,1}$-norm. Moreover, this distance measure can be conducted in not only Euclidean but also non-Euclidean spaces such as manifold regularization [4], which will be appreciated in geometric deep learning [5].

Choices of Hash Function h. For rotation invariance, ideally it should be a circular pattern defined by $h(m,n) = (m^2+n^2)^{\frac{1}{2}}$ in a continuous space. Due to the discretization of images, however, it hardly forms circles in filters without interpolation which will significantly increase the computational complexity in convolution. Instead, we propose learning some simpler patterns that can be used to approximate circles. For instance, we illustrate two exemplar patterns for filters with size 3×3 in Fig. 3, where the patterns in (a) and (b) are defined by $h(m,n) = \left\lfloor (m^2+n^2)^{\frac{1}{2}} \right\rfloor$ and $h(m,n) = \left\lceil (m^2+n^2)^{\frac{1}{2}} \right\rceil$, respectively, and $\lfloor\cdot\rfloor$ is the floor function. Other hash functions may be also applicable here, but finding the best one is outside the scope of this paper.

3.2.2 An Empirical Showcase

In this section we will show a specific regularizer that we use in our experiments later. For the simplicity and efficiency, we decide to employ the least square loss for d and the pattern in Fig. 3(a) for h without fine-tuning the accuracy on the data sets.

Specifically we define our empirical rotation-invariant regularizer as follows:

$$\mathcal{R}_2(\omega,\theta) = \mathbb{E}_{k\sim\mathcal{K}}\left[\sum_{m,n\neq0}\left(\omega_k(m,n) - \frac{\sum_{m',n'\neq0}\omega_k(m',n')}{p_kq_k-1}\right)^2\right], \tag{4}$$

where $\theta_k(h(m,n)) = \frac{\sum_{m',n'\neq0}\omega_k(m',n')}{p_kq_k-1}$ is a scalar.

Similar to the center loss in [41], here we aims to reduce the variance among the weights in each 2D spatial filter with 3×3 pixels, on average. Meanwhile, the patterns in the templates are updated automatically with the mean of the weights. In this way we can learn filters that can better approximate 2D spatial circular patterns for rotation invariance. In backpropagation, since $\mathcal{R}_2(\omega, \theta)$ in Eq. 4 is always differentiable w.r.t. $\omega_k, \forall k$, any deep learning solver such as stochastic gradient descent (SGD) can be used to train the network with our rotation-invariant regularizer.

Discussion. Recall that Fig. 3 essentially encodes the structural patterns that we expect for learned filters to handle rotation. One may argue that we can enforce such structures into learning strictly by converting the regularizer $\mathcal{R}_2$ in Eq. 2 into constraints and solving a constrained nonconvex optimization problem. We decide not to do so because potentially the new problem will be much harder to be solved than the one in Eq. 2. Besides since the structures in Fig. 3 are already the approximation of the circular structure, we do not necessarily guarantee that all the weights with the same color are identical. More freedom as in regularization may lead to a compensation for the loss of the structural approximation in terms of accuracy.

4. Experiments

4.1. Benchmark Data with Affine Transformations

4.1.1 Experimental Setup

Data Sets. We test our approach on three benchmark data sets, affNIST [34], MNIST-rot [27], and Traffic Sign [36].

affNIST is created by applying random small affine transformations to each 28×28 grayscale image in MNIST [29] (10 classes). It is designed for testing the tolerance of an algorithm to such transformations. There are 60K training and validation samples and 10K test samples in affNIST with size 40×40 pixels. To facilitate the data processing in training, we resize all the images to 32×32 pixels.

MNIST-rot [27] is another variant of MNIST, where a random rotation between 0° and 360° is applied to each image. It has 10K/2K/50K training/validation/test samples. To facilitate the data processing in training, we again resize all the grayscale images to 32×32 pixels.

Traffic Sign contains 43 classes with unbalanced class frequencies, 34799 training RGB images, and 12630 testing RGB images with size of 32×32 pixels. It reflects the strong variations in visual appearance of signs due to distance, illumination, weather conditions, partial occlusions, and rotations, leading to a very challenging recognition problem.

Networks. We compare our approach with some state-of-the-art networks with similar model complexity to ours, *i.e.* RotEqNet [33][1], Harmonics [42][2], TI-Pooling [26][3], GCNN [6][4], STN [22][5], ResNet-32 [14][6], CapsNet [34][7], GoogLeNet [37][8], and DCN [8][9]. Specifically TI-Pooling is designed for scale invariance, RotEqNet, Harmonics, and GCNN are designed for rotation invariance. We use the public code for our comparison.

We implement our default network using Tensorflow and following the architecture in Fig. 2 with the default numbers of channels. Note that the implementation of the networks in our comparison may be different, (*i.e.* GCNN$\rightarrow$Chainer; GoogLeNet, DCN$\rightarrow$Keras; CapsNet, TI-Pooling, Harmonics, STN, ResNet-32$\rightarrow$Tensorflow; RotEqNet$\rightarrow$Pytorch) which may lead to various computational efficiency.

Training Protocols. We tune each network to report its best performance on the data sets. By default we train the networks for 42000 iterations with mini-batch size 100, weight decay $\lambda_1 = 0.0005$, and momentum 0.9. The global learning rate is set to 0.01 or 0.0001 when trained using all or a few training images per class, respectively, and it is reduced by 0.1 twice at the 20000 iteration and the 30000 iteration as well. For each network the hyper-parameter tuning starts with the default setting, and the best setting may be slightly different from the default. We follow this default setting in all the experiments and set $\lambda_2 = 150$. The numbers reported here are the average over three trials.

To do fair comparison, we follow the settings for data augmentation in the publications of most of the competitors. Specifically, by default on affNIST and Traffic Sign we do not employ data augmentation, but on MNIST-rot we do.

4.1.2 Results

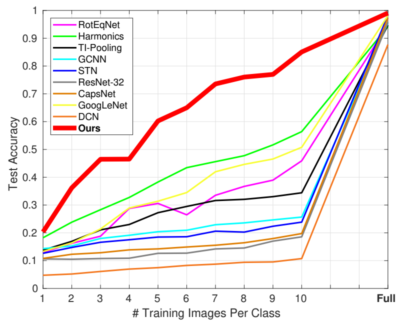
Better Generalization, Data-Efficiency, & Robustness. We summarize the test accuracy comparison in Table 1. As we see, using either all or 10 random training/validation images per class, our method consistently outperforms the competitors on the three data sets with a margin of **1.96%** or **30.37%**. Using the full set the stds of all the methods are small and similar, and thus we do not show the numbers.

To better demonstrate the data-efficiency, we illustrate test accuracy comparison using few random train-

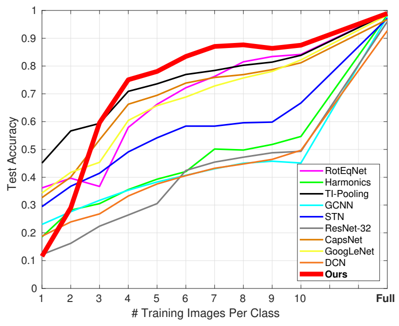
[1] https://github.com/COGMAR/RotEqNet
[2] https://github.com/deworrall92/harmonicConvolutions
[3] https://github.com/dlaptev/TI-pooling
[4] https://github.com/tscohen/gconv_experiments
[5] https://github.com/kevinzakka/spatial-transformer-network
[6] https://github.com/tensorflow/models/tree/master/research/resnet
[7] https://github.com/naturomics/CapsNet-Tensorflow
[8] https://github.com/flyyufelix/cnn_finetune/blob/master/googlenet
[9] https://github.com/felixlaumon/deform-conv

Table 1. Test accuracy (%) comparison on different datasets under two training settings: **(F)** with all the images, and **(10)** with 10 random images per class.

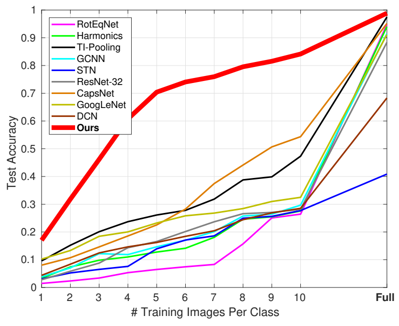
	Ours	RotEqNet	Harmonics	TI-Pooling	GCNN	STN	ResNet-32	CapsNet	GoogLeNet	DCN
aff. (F)	**99.08**	94.81	94.20	94.72	95.43	98.24	95.76	97.30	98.12	87.70
rot (F)	**98.92**	98.91	98.31	98.80	97.72	97.12	95.96	96.73	98.29	92.69
T. S. (F)	**98.87**	94.79	94.02	97.47	91.47	40.87	88.35	95.15	91.16	68.29
Ave. (F)	**98.95**	96.17	95.51	96.99	94.87	78.74	93.36	96.39	95.85	82.75
aff. (10)	**85.06±0.90**	45.91±3.85	56.41±3.66	34.40±1.54	25.67±1.99	23.85±0.12	18.56±0.27	19.74±0.39	50.77±0.20	10.74±0.52
rot (10)	**87.49±0.56**	84.18±2.17	54.67±2.65	83.86±0.88	45.12±2.48	66.72±0.72	49.31±0.30	81.17±0.17	82.20±0.35	49.69±0.33
T. S. (10)	**84.15±0.48**	26.43±0.85	27.57±0.94	47.31±1.37	29.66±0.63	27.72±1.74	28.20±2.06	54.35±0.56	32.49±0.29	28.52±0.26
Ave. (10)	**85.56**	52.17	46.21	55.19	33.48	39.43	32.02	51.75	55.15	29.65



(a) affNIST



(b) MNIST-rot



(c) Traffic Sign

Figure 4. Test accuracy comparison of different networks on the three data sets. "Full" here indicates that we use all the training images. Our approach significantly outperforms the state-of-the-art, especially with small numbers of training images.

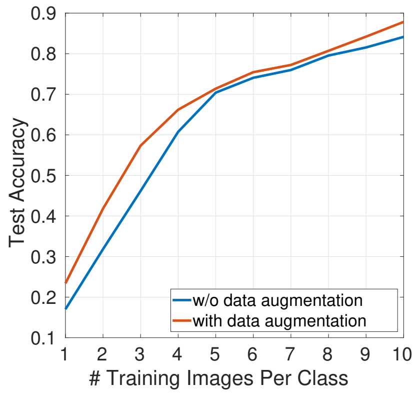


Figure 5. Data augmentation comparison on Traffic Sign.

ing/validation images per class in Fig. 4. Overall, our method works significantly better than the competitors with large margins. Note that on MNIST-rot our performance is worse than some of the competitors when using 1 or 2 images per class for training. A possible reason may come from data augmentation. Another reason is that some of the networks are designed specifically for rotation invariance and this data set just fits for this purpose. With the increase of the numbers of training samples, however, our method again beats all the competitors. It is worth mentioning that in Harmonics Networks [42], similar experiments on MNIST-rot were conducted to show data-efficiency and robustness of the approach. Using $\frac{1}{6}$ of the full training/validation data Harmonics lost about 3%. Here we compare different networks using less than $\mathbf{\frac{1}{120}}$ to show the superiority of our method over the others. Empirically we observe that our method can work very robustly with standard deviation of less than 1%, in general.

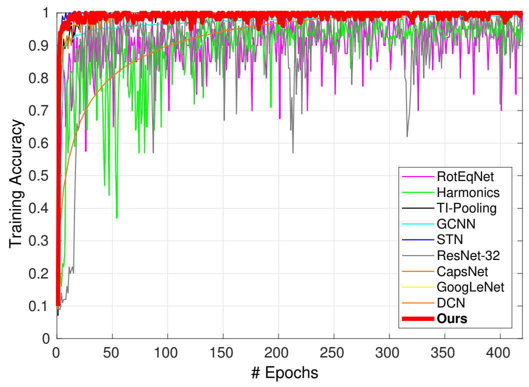
Table 2. Effect on test accuracy (%) of different multi-scale settings, where our default setting is 3×[Conv+BN].

	2×[Conv+BN]	3×[Conv+BN]	4×[Conv+BN]
affNIST (F)	99.04	99.08	98.69
MNIST-rot (F)	98.72	98.92	98.97
Traffic Sign (F)	98.42	98.87	98.42
Average	98.73	98.95	98.69

In addition, we can further improve our performance using data augmentation. In Fig. 5 we illustrate the performance comparison on Traffic Sign with or without data augmentation. As we see, using 10 random training images per class we can achieve 87.84% with 3.69% improvement.

Training & Testing Behavior. We illustrate the training and test accuracy behavior of each network on affNIST with the full training set in Fig. 6. As we see all the networks are well trained with convergence. In the testing stage our network converge faster than most of the competitors with better accuracy. Similar observations can be made in training as well. We make similar observations on the other two data sets. From this perspective, we can also demonstrate that our method has better generalization.

Effect of Multi-Scale Maxout. In Table 2 we list the test accuracy using different multi-scale settings, while fixing the parameter $\lambda_2 = 150$. As we see the changes between different settings are really marginal, which again



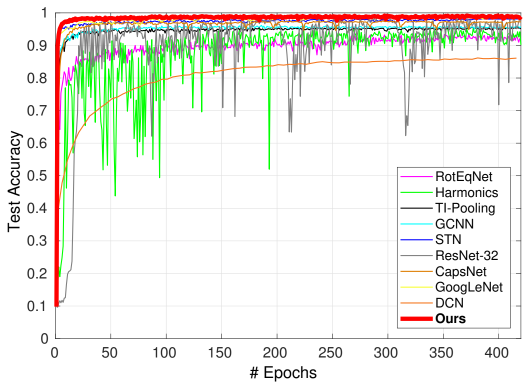


Figure 6. Illustration of training/testing behavior of different networks on affNIST.

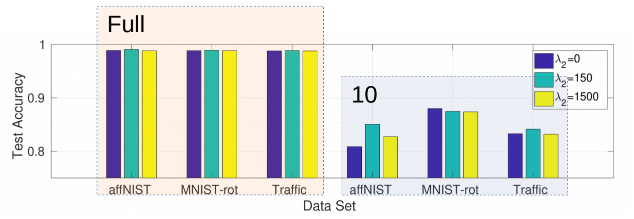


Figure 7. Illustration of the effect of λ_2 in Eq. 2 on test accuracy.

demonstrates the good generalization and robustness of our method. Considering the trade-off between accuracy and computational efficiency, we choose 3×[Conv+BN] as our default setting used in Fig. 2.

Effect of Rotation-Invariant Regularization. We illustrate such effect in Fig. 7 while using the default multi-scale maxout setting. With different values where $\lambda_2 = 0$ means no our regularizer, we can see that using the full set for training our performances are almost identical. This is probably because the number of training images is sufficiently large to capture the scaling and rotation information already. Using a few training images, *e.g*. 10 per class, the benefit of using our rotation-invariant regularizer becomes much clearer, especially on affNIST. Using $\lambda_2 = 150$ as default, there is 1.52%, on average, improvement over that without our regularizer.

We also observe that our rotation-invariant regularizer can achieve very small numbers empirically. For instance, on affNIST the value is 2.94×10^{-7}, indicating that our learned filters are very close to the spatial circular patterns.

Behavior with Different Numbers of Parameters. We reduce the number of parameters in our network by channel-wise shrinking. Specifically in ascending order of number of parameters, the corresponding network channels are set as follows: [4,4,4,4,4], [16,16,16,16,16], [32,32,32,32,32], [32,64,64,64,64], [32,64,128,128,128], [32,64,128,256,256], [32,64,128,256,512], followed by an FC of 1024 nodes and another FC for classification.

We first compare our performance using different numbers with the competitors in Fig. 8. We can see that after about 200K parameters the improvement of our approach becomes slow, while before 200K our performance drops significantly with the decrease of numbers of parameters. In the figure 200K corresponds to the setting [32,64,64,64,64], whose performance is, or on par with, the best already.

We then compare the running time per iteration in both training and testing stages in Fig. 9. We run all the code on the same machine with a Titan XP GPU. In training the running time includes the feedforward calculation and backpropagation inference (dominating training time), while in testing the running time only includes the feedforward calculation. As we see, in both training and testing our computational complexity grows exponentially, in general, with the number of parameters (note that the y-axis is in log-scale). Although some codes are written in different deep learning environments, we can still do a fair comparison with Harmonics and STN. Harmonics has fewer parameters, leading to faster backpropagation and thus shorter training time. The operations in Harmonic, however, is more complex than ours, and thus with a similar number of parameters our method is faster in testing. The operations in STN are much simpler than both Harmonics and ours, leading to faster running speed in both training and testing. Note that in order to further improve our computational efficiency, we can simply remove one Conv+BN in the multi-scale maxout block that can achieve similar accuracy (see Table 2).

4.2. Comparison on CIFAR-100 [25]

Beyond the benchmark data sets with affine transformations, we also test our method on “natural” images. For instance, we illustrate our comparison results on CIFAR-100 in Fig. 10. CIFAR-100 contains 60,000 32 × 32 color images in 100 different classes, 500/100 training/testing images per class. Following the same training protocol, we randomly sample a few images per class to further demonstrate our superiority, especially on data-efficiency.

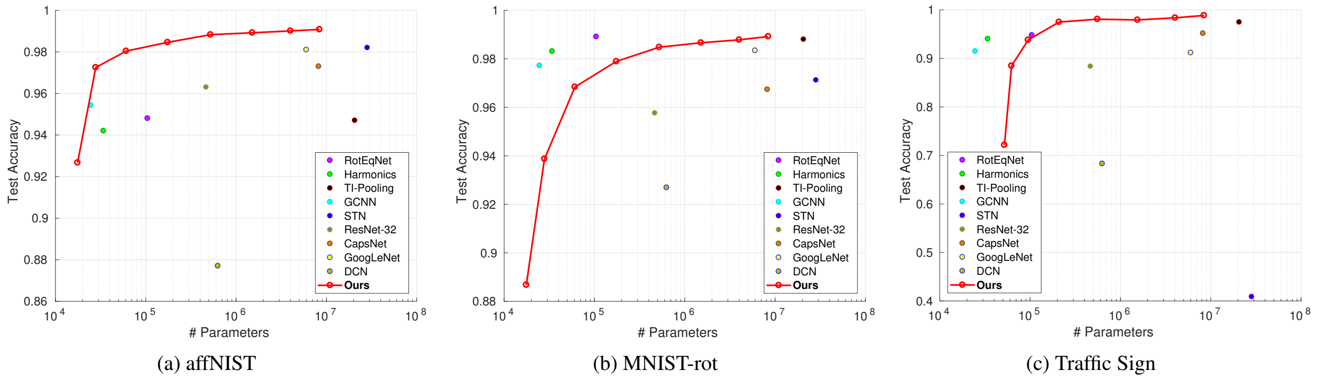


Figure 8. Test accuracy comparison with the others using different numbers of parameters. Best viewed in color.

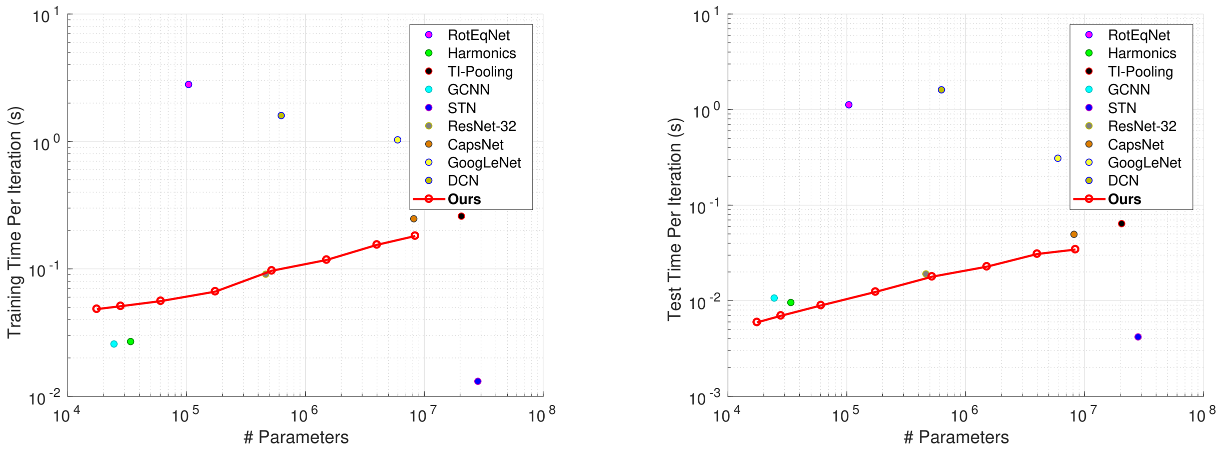


Figure 9. Training/Test time comparison with the others using different numbers of parameters. Best viewed in color.

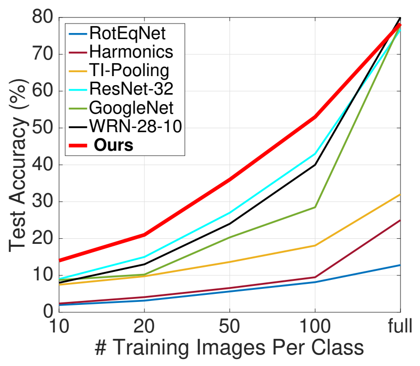


Figure 10. Test accuracy comparison of different networks on CIFAR-100. "Full" here indicates that we use all the training images. Again our approach significantly outperforms the state-of-the-art, especially with small numbers of training images.

As we see in Fig. 10, our method significantly and consistently outperforms the competitors with a few training samples. For instance, using 100 samples per class ours achieves 52.67% test accuracy with the improvement of almost 10% over ResNet-32 (the second best). Using the full training set, ours achieves 78.33% that is slightly lower than WRN-28-10 (80.75%), but higher than ResNet-32 (76.7%) and GoogleNet (78.03%), and dramatically higher than the other networks that learn the scale or rotation invariant representations such as TI-Pooling (31.77%).

5. Conclusion

In this paper we propose a novel multi-scale maxout deep CNN and a novel rotation-invariant regularizer to learn affine-invariant representations for object recognition in images. Multi-scale convolution with maxout can handle translation and scale, and enforcing 2D filters to approximate circular patterns by our regularization can manage to induce invariance to rotation. By taking these as a priori knowledge, we can easily interpret our network architecture as well as its training procedure. We test our method on three benchmark data sets as well as CIFAR-100 to demonstrate its superiority over the state-of-the-art in terms of generalization, data-efficiency, and robustness. Especially, with a few training samples our method can work significantly better, leading to the hypothesis that the introduction of a priori knowledge into deep learning can effectively reduce the amount of data to accomplish the tasks. We are planning to explore more on this topic in our future work.

References

[1] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter. Differentiable mpc for end-to-end planning and control. In *NIPS*, pages 8299–8310, 2018. 2

[2] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *CVPR*, pages 39–48, 2016. 1, 2

[3] N. Audebert, B. Le Saux, and S. Lefèvre. Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In *ACCV*, pages 180–196, 2016. 2

[4] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7(Nov):2399–2434, 2006. 4

[5] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 4

[6] T. Cohen and M. Welling. Group equivariant convolutional networks. In *ICML*, pages 2990–2999, 2016. 2, 5

[7] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *CVPR*, volume 1, pages 10–17, 2005. 1

[8] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *CVPR*, pages 764–773, 2017. 2, 5

[9] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-end differentiable physics for learning and control. In *NIPS*, pages 7178–7189, 2018. 2

[10] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. *Theory Of Computing*, 8:415–428, 2012. 1

[11] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973. 1

[12] R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In *CVPR*, pages 437–446, 2015. 1

[13] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *ICML*, pages III–1319, 2013. 2

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2, 5

[15] G. Hinton. Taking inverse graphics seriously. https://www.cs.toronto.edu/ hinton/csc2535/notes/lec6b.pdf. 1

[16] G. Hinton, N. Frosst, and S. Sabour. Matrix capsules with em routing. In *ICLR*, 2018. 1

[17] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011. 1

[18] E. Hoogeboom, J. W. Peters, T. S. Cohen, and M. Welling. Hexaconv. *arXiv preprint arXiv:1803.02108*, 2018. 2

[19] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017. 2

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. 3

[21] L. Isik, E. M. Meyers, J. Z. Leibo, and T. Poggio. The dynamics of invariant object recognition in the human visual system. *Journal of neurophysiology*, 111(1):91–102, 2013. 1

[22] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *NIPS*, pages 2017–2025, 2015. 2, 3, 5

[23] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool. Dynamic filter networks. In *NIPS*, pages 667–675, 2016. 2

[24] A. Kanazawa, A. Sharma, and D. Jacobs. Locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1412.5104*, 2014. 2

[25] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-100 (canadian institute for advanced research). 7

[26] D. Laptev, N. Savinov, J. M. Buhmann, and M. Pollefeys. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *CVPR*, pages 289–297, 2016. 2, 5

[27] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pages 473–480, 2007. 5

[28] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016. 2

[29] Y. LECUN. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*. 5

[30] Z. Liao and G. Carneiro. Competitive multi-scale convolution. *arXiv preprint arXiv:1511.05635*, 2015. 2

[31] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 2

[32] S. Luan, C. Chen, B. Zhang, J. Han, and J. Liu. Gabor convolutional networks. *TIP*, 2018. 2

[33] D. Marcos, M. Volpi, N. Komodakis, and D. Tuia. Rotation equivariant vector field networks. In *ICCV*, pages 5048–5057, 2017. 2, 5

[34] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *NIPS*, pages 3859–3869, 2017. 1, 5

[35] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087, 2017. 3

[36] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IJCNN*, pages 1453–1460, 2011. 5

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 2, 5

[38] R. Takahashi, T. Matsubara, and K. Uehara. Scale-invariant recognition by weight-shared cnns in parallel. In *ACML*, pages 295–310, 2017. 2

[39] J. Wang, Z. Wei, T. Zhang, and W. Zeng. Deeply-fused nets. *arXiv preprint arXiv:1605.07716*, 2016. 2

[40] M. Weiler, F. A. Hamprecht, and M. Storath. Learning steerable filters for rotation equivariant cnns. In *CVPR*, pages 849–858, 2018. 2

[41] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, pages 499–515, 2016. 3, 5

[42] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *CVPR*, volume 2, 2017. 2, 5, 6

[43] J. Wu, D. Li, Y. Yang, C. Bajaj, and X. Ji. Dynamic filtering with large sampling field for convnets. In *ECCV*, pages 185–200, 2018. 2

[44] Y. Xu, T. Xiao, J. Zhang, K. Yang, and Z. Zhang. Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369*, 2014. 2

[45] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 2

[46] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 4

[47] Q. Zhang, Y. Nian Wu, and S.-C. Zhu. Interpretable convolutional neural networks. In *CVPR*, pages 8827–8836, 2018. 2

[48] T. Zhang, G.-J. Qi, B. Xiao, and J. Wang. Interleaved group convolutions. In *CVPR*, 2017. 2

[49] X. Zhang, L. Liu, Y. Xie, J. Chen, L. Wu, and M. Pietikäinen. Rotation invariant local binary convolution neural networks. In *ICCV Workshops*, pages 1210–1219, 2017. 2

[50] Z. Zhang and M. Brand. Convergent block coordinate descent for training tikhonov regularized deep neural networks. In *NIPS*, pages 1721–1730. 2017. 1

[51] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao. Oriented response networks. In *CVPR*, pages 4961–4970, 2017. 2