Improving the Sample-complexity of Deep Classification Networks with Invariant Integration

Matthias Rath[1,2] and Alexandru Paul Condurache[1,2]
[1]*Automated Driving Research, Robert Bosch GmbH, Stuttgart, Germany*
[2]*Institute for Signal Processing, University of Lübeck, Lübeck, Germany*

Keywords: Geometric Prior Knowledge, Invariance, Group Transformations, Representation Learning.

Abstract: Leveraging prior knowledge on intraclass variance due to transformations is a powerful method to improve the sample complexity of deep neural networks. This makes them applicable to practically important use-cases where training data is scarce. Rather than being learned, this knowledge can be embedded by enforcing invariance to those transformations. Invariance can be imposed using group-equivariant convolutions followed by a pooling operation.

For rotation-invariance, previous work investigated replacing the spatial pooling operation with invariant integration which explicitly constructs invariant representations. Invariant integration uses monomials which are selected using an iterative approach requiring expensive pre-training. We propose a novel monomial selection algorithm based on pruning methods to allow an application to more complex problems. Additionally, we replace monomials with different functions such as weighted sums, multi-layer perceptrons and self-attention, thereby streamlining the training of invariant-integration-based architectures.

We demonstrate the improved sample complexity on the Rotated-MNIST, SVHN and CIFAR-10 datasets where rotation-invariant-integration-based Wide-ResNet architectures using monomials and weighted sums outperform the respective baselines in the limited sample regime. We achieve state-of-the-art results using full data on Rotated-MNIST and SVHN where rotation is a main source of intraclass variation. On STL-10 we outperform a standard and a rotation-equivariant convolutional neural network using pooling.

1 INTRODUCTION

Deep neural networks (DNNs) excel in problem settings where large amounts of data are available such as computer vision, speech recognition or machine translation (LeCun et al., 2015). However, in many if not most real-world problem settings training data is scarce because it is expensive to collect, store and in case of supervised training label. Consequently, an important aspect of DNN research is to improve the sample complexity of the training process, i.e., achieving best results when the available training data is limited.

One solution to reduce the sample complexity is to incorporate meaningful *prior knowledge* to bias the learning mechanism and reduce the complexity of the possible parameter search space. One well-known example on how to embed *prior knowledge* are convolutional neural networks (CNNs) which achieve state-of-the-art performance in a variety of tasks related to computer vision. CNNs successfully employ translational weight-tying such that a translation of the input leads to a translation of the resulting feature space. This property is called *translation equivariance*.

These concepts can be expanded such that they cover other transformations of the input which lead to a predictable change of the output – or to no change at all. The former is called *equivariance* while the latter is a related concept referred to as *invariance*.

In general, DNNs for image-based object detection and classification hierarchically learn a set of features that ideally contain all relevant information to distinguish different objects while dismissing the irrelevant information contained in the input. Generally, transformations causing intraclass variance can act globally on the entire input image, e.g., global rotations or illumination changes – or locally on the objects, e.g., perspective changes, local rotations or occlusions. Prior knowledge about those transformations can usually be obtained before training a DNN and thus be incorporated to the training process or architecture. Enforcing meaningful *invariances* on

Rath, M. and Condurache, A.
Improving the Sample-complexity of Deep Classification Networks with Invariant Integration.
DOI: 10.5220/0010872000003124
In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 5: VISAPP*, pages 214-225
ISBN: 978-989-758-555-5; ISSN: 2184-4321

the learned features simplifies distinguishing relevant from irrelevant input information. One method to enforce invariance is to approximately learn it via *data augmentation*, i.e., artificially transforming the input during training. However, these learned invariances are not exact and do not cover all relevant variability.

(Cohen and Welling, 2016) first applied *group-equivariant convolutions* (G-Convs) to DNNs. G-Convs mathematically guarantee equivariance to transformations which can be modeled as a group. A DNN consisting of multiple layers is equivariant with respect to a transformation group, if each of its layers is group-equivariant or commutes with the group (Cohen and Welling, 2016). Consequently, an equivariant DNN consists of multiple group-convolutional layers as well as pooling and normalization operations that commute with the desired transformations. For a classifier, the G-Convs are usually followed by a global pooling operation over both the group dimension and the spatial domain in order to enforce invariance. These invariant features are then processed by fully connected layers to obtain the final class scores.

Invariant Integration (II) is a method to explicitly create a complete, invariant feature space with respect to a transformation group introduced by (Schulz-Mirbach, 1992; Schulz-Mirbach, 1994). Recent work showed that explicitly enforcing rotation-invariance by means of II instead of using a global pooling operation among the spatial dimensions decreases the sample complexity of rotation-equivariant CNNs used for classification tasks despite adding parameters, hence improving generalization (Rath and Condurache, 2020). However, II thus far relies on calculating monomials which are hard to optimize with usual DNN training methods. Additionally, monomial parameters have to be chosen using an iterative method based on the least square error of a linear classifier before the DNN can be trained. This method relies on an expensive pre-training step that reduces the applicability of II to real-world problems.

Consequently, in this paper we investigate how to adapt the rotation-II framework in combination with equivariant backbone layers in order to reduce the sample complexity of DNNs on various real-world datasets while simplifying the training process. Thereby, we explicitly investigate the transition between in- and equivariant features for the case of rotations and replace the spatial pooling operation by II. We start by introducing a novel monomial selection algorithm based on pruning methods. Additionally, we investigate replacing monomials altogether, using simple, well-known DNN layers such as a weighted sum (WS), a multi-layer perceptron (MLP) or self-attention (SA) instead. This contributes significantly to streamlining the entire framework. We specifically apply these approaches to 2D rotation-invariance. We achieve state-of-the-art results irrespective of limited- or full-data regime, when rotations are responsible for most of the relevant variability, such as on Rotated-MNIST and SVHN. Furthermore, we demonstrate very good performance in limited-data regimes on CIFAR-10 and STL-10, when besides rotations also other modes of intraclass variation are present.

Our **core contributions** are:

- We introduce a novel algorithm for the II monomial selection based on pruning.
- We investigate various functions to replace the monomials within the II framework including a weighted sum, a MLP and self-attention. We thereby streamline the training process of II-enhanced DNNs as the *monomial selection is no longer needed*.
- We demonstrate the performance of rotation-II on the real world datasets SVHN, CIFAR-10 and STL-10.
- We apply II to Wide-ResNet (WRN) architectures, demonstrating its general applicability.
- We establish a connection between II and regular G-Convs.
- We show that using II in combination with equivariant G-Convs reduces the sample complexity of DNNs.

2 RELATED WORK

DNNs can learn invariant representations using **group-equivariant convolutions** or **equivariant attention** in combination with pooling operations. Other methods explicitly **learn invariance**, or enforce it using **invariant integration**.

Group-equivariant convolutional neural networks (G-CNNs) are a general framework to introduce equivariance, first proposed and applied to 90° rotations and flips on 2D images by (Cohen and Welling, 2016). G-CNNs were extended to more fine-grained or continuous 2D rotations (Worrall et al., 2017; Bekkers et al., 2018; Veeling et al., 2018; Weiler et al., 2018b; Winkels and Cohen, 2019; Diaconu and Worrall, 2019b; Walters et al., 2020), processed as vector fields (Marcos et al., 2017) or further generalized to the $\mathrm{E}(2)$-group which includes rotations, translations and flips (Weiler and Cesa, 2019). Additionally, 2D scale-equivariant group convolutions have been introduced (Xu et al., 2014; Kanazawa et al., 2014; Marcos et al., 2018; Ghosh and Gupta, 2019; Zhu et al., 2019; Worrall and

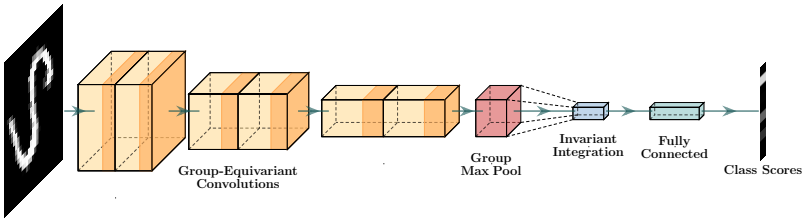

Figure 1: General invariant DNN architecture using II. The architecture includes group convolutions (orange) and group pooling (red) creating an equivariant representation, an II layer enforcing invariance (blue), and fully-connected layers (green).

Welling, 2019; Sosnovik et al., 2020). Further advances include expansions towards three-dimensional spaces (e.g., (Worrall and Brostow, 2018; Kondor et al., 2018; Esteves et al., 2018a)) or general manifolds and groups (e.g., (Cohen et al., 2019a; Cohen et al., 2019b; Bekkers, 2020; Finzi et al., 2021)) which are beyond the scope of this paper.

Recently, **equivariance** was also introduced to **attention** layers. (Diaconu and Worrall, 2019a; Romero and Hoogendoorn, 2020; Romero et al., 2020) combined equivariant attention with convolution layers to enhance their expressiveness. (Fuchs et al., 2020; Fuchs et al., 2021; Romero and Cordonnier, 2020; Hutchinson et al., 2020) introduced different equivariant transformer architectures. In order to obtain invariant representations, equivariant layers are usually combined with pooling operations.

Other methods to **learn invariant representations** include data augmentation, pooling over all transformed inputs (Laptev et al., 2016), learning to transform the input or feature spaces to their canonical representation (Jaderberg et al., 2015; Esteves et al., 2018b; Tai et al., 2019) or regularization methods (Yang et al., 2019). However, these methods approximate invariance rather than enforcing it mathematically guaranteed.

Invariant integration is a principled method to enforce invariance. It was introduced as a general algorithm in (Schulz-Mirbach, 1992; Schulz-Mirbach, 1994) and applied in combination with classical machine learning classifiers for various tasks such as rotation-invariant image classification (Schulz-Mirbach, 1995), speech recognition (Müller and Mertins, 2009; Müller and Mertins, 2010; Müller and Mertins, 2011), 3D-volume and -surface classification (Reisert and Burkhardt, 2006) or event detection invariant to anthropometric changes (Condurache and Mertins, 2012).

In (Rath and Condurache, 2020), rotation-II was applied in combination with steerable G-Convs in DNNs for image classification. The equivariant feature space learned by the G-Convs is followed by max-pooling among the group elements. Rotations of the input induce rotations in the resulting feature space, i.e., it is equivariant to rotations. While standard G-CNNs employ spatial max-pooling afterwards to achieve an invariant representation, (Rath and Condurache, 2020) and our approach replace it with II, which increases the expressibility compared to spatial max-pooling while still guaranteeing invariant features. These are finally processed with dense layers to calculate the classification scores (see Figure 1).

All previous methods including (Rath and Condurache, 2020) used II in combination with monomials which were either hand-designed or selected using expensive iterative approaches which required pre-training the entire network without II. In contrast, we propose a novel pre-selection algorithm based on pruning methods or to replace the monomials altogether. Both approaches can be applied to DNNs more natively.

3 PRELIMINARIES

In this section we concisely present the mathematical principles needed to define in- and equivariance in DNNs which rely on Group Theory. Furthermore, we introduce group-equivariant convolutions which are used to obtain equivariant features and form the backbone of our DNNs.

3.1 In- & Equivariance

A group G is a mathematical abstraction consisting of a set $\mathcal{X}$ and a *group operation* $\cdot : G \times G \to G$ that combines two elements to form a third. A group fulfills the four axioms closure, associativity, invertibility and identity. Group Theory is important for DNN research, because invertible transformations acting on feature spaces can be modeled as a group, where the left group action $G \times \mathbb{R}^n \to \mathbb{R}^n, (g,x) \mapsto L_g x$ with $g \in G$ acts on the vector space $\mathbb{R}^n$.

The concept of *in-* and *equivariance* can be mathematically defined on groups. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is defined as *equivariant*, if its output $f(x)$ transforms

predictably under group transformations

$$\forall\, g \,\exists\, g' \text{ s.t. } f(L_g x) = L_{g'} f(x), \tag{1}$$

for all $x \in \mathbb{R}^n$, and $g \in G, g' \in G'$ while G and G' may be the same or different groups. If the output does not change under transformations of the input, i.e., $\forall\, g\ \forall\, x,\ f(L_g x) = f(x)$, f is *invariant* (Cohen and Welling, 2016).

3.2 Group-equivariant Convolutions

(Cohen and Welling, 2016) first used the generalization of the convolution towards general transformation groups G in the context of CNNs. The discrete group-equivariant convolution of a signal x and a filter $\psi : G \to \mathbb{R}^n$ is defined as

$$[x \star_G \psi](g) = \sum_{h \in G} x(h)\psi(g^{-1}h). \tag{2}$$

Here, $x : G \to \mathbb{R}^n$ is used as a function. Both definitions are interchangeable. The standard convolution is a special case where $G = \mathbb{Z}^2$. The output of the group-convolution is no longer defined on the regular grid, but on group elements g and is equivariant w.r.t G. The action $L_{g'}$ in the output space depends on the group representation that is used. Two common representations used for G-CNNs are the *irreducible representation* and the *regular representation* which consists of one additional *group channel* per group element storing the responses to all transformed versions of the filters. Often, the regular representation is combined with *transformation-steerable filters* which can be transformed arbitrarily via a linear combination of basis filters, hence avoiding interpolation artifacts (Freeman and Adelson, 1991; Weiler et al., 2018b; Weiler et al., 2018a; Weiler and Cesa, 2019; Ghosh and Gupta, 2019; Sosnovik et al., 2020). In order to obtain invariant features from the equivariant ones learned by G-CNNs, a pooling operation is usually employed.

4 INVARIANT INTEGRATION

Invariant Integration introduced by (Schulz-Mirbach, 1992) is a method to create a complete feature space w.r.t. a group transformation based on the Group Average $A[f](x)$

$$A[f](x) = \int_{g \in G} f(L_g x)\, d\mu(g), \tag{3}$$

where $\int d\mu(g) = 1$ defines the Haar Measure and f is an arbitrary complex-valued function. A complete feature space implies that all patterns that are equivalent w.r.t G are mapped to the same point while all non-equivalent patterns are mapped to distinct points.

4.1 Monomials

For the choice of f, (Schulz-Mirbach, 1992) proposes to use the set of all possible monomials which form a finite basis of the signal space according to (Noether, 1916). Monomials are a multiplicative combination of different scalar input values x_i with exponents $b_i \in \mathbb{R}$

$$m(x) = \prod_{i=1}^{M} x_i^{b_i} \quad \text{with} \quad \sum_{i=1}^{M} b_i \leq |G|. \tag{4}$$

Combined with the finite group average, we obtain

$$A[m](x) = \frac{1}{|G|} \sum_{g \in G} m(L_g x) = \frac{1}{|G|} \sum_{g \in G} \prod_{i=1}^{M} (L_g x)_i^{b_i}, \tag{5}$$

(Schulz-Mirbach, 1994). When applying II with monomials to two-dimensional input data on a regular grid such as images, it is straightforward to use pixels and their neighbors for the monomial factors x_i. Consequently, monomials can be defined via the distance of the neighbor to the center pixel d_i with $d_1 = 0$. For discrete 2D rotations and translations, this results in the following formula (Schulz-Mirbach, 1995)

$$A[m](x) = \frac{1}{UV\Phi} \sum_{u,v,\phi} \prod_{i=1}^{M} x[u + \cos(\phi) d_i, v + \sin(\phi) d_i]^{b_i}, \tag{6}$$

which can be used within a DNN with learnable exponents b_i (Rath and Condurache, 2020).

4.2 Monomial Selection

While the II layer reduces the sample complexity when learning invariant representations, it introduces additional parameters which need to be carefully designed. One of them is the selection of a meaningful set of parameters d_i and b_i that define the monomials needed to obtain the invariant representation. This step is necessary since the number of possible monomials satisfying $\sum_i b_i \leq |G|$ is too extensive.

In (Rath and Condurache, 2020), an iterative approach is used based on the least square error solution of a linear classifier. While the linear classifier is easy to compute, the iterative selection is time-consuming and computationally expensive. Additionally, the base network without the II layer needs to be pre-trained which requires additional computations and prevents training the full network from scratch.

Consequently, we investigate alternative approaches for the monomial selection. Two selection approaches are introduced and explained in the following. Both enable training the network end-to-end from scratch and are computationally inexpensive compared to the iterative approach.

4.2.1 Random Selection

First, we randomly select the n_m monomials by sampling both the exponents and the distances from uniform distributions. This approach is fast only requiring a single random sampling operation and serves as a baseline to evaluate other selection methods.

4.2.2 Pruning Selection

Alternatively, monomial selection can be formulated as selecting a subset containing $n_m \ll M$ possible monomial parameters. Consequently, it is closely related to the field of pruning in DNNs whose goal is to reduce the amount of connections or neurons within DNN architectures in order to reduce the computational complexity while maintaining the best possible performance. We compare two pruning algorithms: a magnitude- and a connectivity-based approach.

Magnitude-based Approach. (Han et al., 2015) determine the importance of connections in DNNs by pre-training the network for τ epochs and sorting the weights of all layers by their magnitude $|w_{ij}|$. This approach is applied iteratively keeping the γ highest-ranked connections at each step until the final pruning-ratio γ^I is reached.

Since we aim to prune monomials instead of single connections, we sum the absolute value of weights connected to a single monomial, i.e., all weights of the first fully connected layer following the II layer:

$$s_j = \frac{1}{C_i C_o} \sum_{k=1}^{C_i} \sum_{l=1}^{C_o} |w_{klj}|, \tag{7}$$

where C_i is the number of input channels before II is applied, C_o is the number of neurons in the fully connected layer and j selects the connections belonging to the j^{th} monomial. Following (Han et al., 2015), we apply the pruning iteratively. In each step, we keep the n_i monomials with the highest calculated score s_j. We do not re-initialize our network randomly in between iterative steps but re-load the pre-trained weights from the previous step.

Connectivity-based Approach. We examine a second pruning approach based on the initial connectivity of weights inspired by (Lee et al., 2019). All monomial output connections are multiplied with an indicator mask $c \in \{0,1\}^M$ using the Hadamard product $c \odot w_l$. Here, w_l is the weight vector of the fully connected layer following the II step. Setting an individual value c_j to zero results in deleting all connections $w_{l,j}$ connected to monomial j. Consequently, the effect of deactivating a monomial can be estimated w.r.t the training loss L by calculating the connection sensitivity

$$s_j = \frac{\partial L(c, w_l; \mathcal{D})}{\partial c_j} \tag{8}$$

for each monomial using backpropagation. The n_m monomials with the highest connection sensitivity are kept. The derivative is calculated using the training dataset $\mathcal{D}$.

The connectivity-based approach can either be used directly after initializing the DNN, or after some pre-training steps. Additionally, it can either be used iteratively or in a single step.

4.2.3 Initial Selection

We investigate two different approaches for the initial selection of $M \gg n_m$ monomials. In addition to a purely random selection, we design a catalog-based initial selection in which all possible distance combinations are guaranteed to be involved in the initial set. In both cases, we sample the exponents randomly from a uniform distribution.

4.3 Replacing the Monomials

In addition to the novel monomial selection algorithm, we investigate alternatives for the monomials used to calculate the group average (Equation 3). We apply the proposed functions to the group of discrete 2D rotations and compare monomials to well-utilized DNN functions such as a weighted sum, a MLP and a self-attention-based approach.

4.3.1 Weighted Sum

One possibility for f is to use a weighted sum where the weights are a learnable kernel ψ applied at each group element g transforming the input x. We obtain

$$A[\mathrm{WS}](x) = \frac{1}{|G|} \sum_{g \in G} \sum_{y \in \mathbb{Z}^2} x(y) \psi(g^{-1} y). \tag{9}$$

For 2D-rotations, this results in translating and rotating the kernel using all group elements $g \in \mathrm{SO}(2)$. We implement two different versions of II with WS. First, we apply a global convolutional filter, i.e., the kernel size is equivalent to the size of the input feature map (*Global-WS*). Secondly, we use local filters with kernel size k which we apply at all spatial locations (u, v) and all orientations ϕ (*Local-WS*).

Relation to Group Convolutions. In the following we show the close connection between II using a WS and the group convolution introduced by (Cohen and Welling, 2016). Recall the formulation of the discrete

group convolution of an image $x : \mathbb{Z}^2 \to \mathbb{R}$ and a filter $\psi : \mathbb{Z}^{k\times k} \to \mathbb{R}$

$$[x \star \psi](g) = \sum_{y\in\mathbb{Z}^2} x(y)\psi(g^{-1}y). \quad (10)$$

The group convolution followed by global average pooling $A_G\{\cdot\}$ among all group elements is

$$A_G\{[x \star \psi](\cdot)\} = \frac{1}{|G|}\sum_{g\in G}\sum_{y\in\mathbb{Z}^2} x(y)\psi(g^{-1}y), \quad (11)$$

which is exactly the same formulation as Equation 9. Thus, using a regular lifting convolution and applying global average pooling can be formulated as a special case of II.

4.3.2 Multi-layer Perceptron

Another possibility for f is a multi-layer perceptron (MLP) which consists of multiple linear layers and non-linearities σ. In combination with the rotation-group average, we obtain

$$A[\text{MLP}](x) = \frac{1}{UV\Phi}\sum_{u,v,\phi} \sigma(\mathbf{W}_l \cdots \sigma(\mathbf{W}_1 L_{g_\phi}^{-1} x_{\mathcal{N}})), \quad (12)$$

where $\mathcal{N}$ defines the neighborhood of a pixel located at (u,v). For σ, we choose to use ReLU non-linearities.

4.3.3 Self-attention

Finally, we insert a self-attention module into the II framework. Visual self-attention $\text{SA}(\mathbf{x})$ is calculated by defining the pixels of the input image or feature space x as $N = H \cdot W$ individual tokens $\mathbf{x} \in \mathbb{R}^{HW\times C_i}$ with C_i values and learning attention scores $\mathbf{A} \in \mathbb{R}^{N\times N}$. It includes three learnable matrices: the value matrix $\mathbf{W}_V \in \mathbb{R}^{C_i\times C_h}$, the key matrix $\mathbf{W}_K \in \mathbb{R}^{C_i\times C_h}$ and the query matrix $\mathbf{W}_Q \in \mathbb{R}^{C_i\times C_h}$. It is defined as

$$\text{SA}(\mathbf{x}) = \text{softmax}(\mathbf{A})\mathbf{x}\mathbf{W}_V \text{ with } \mathbf{A} = \mathbf{x}\mathbf{W}_Q(\mathbf{x}\mathbf{W}_K)^T. \quad (13)$$

To incorporate positional information between the individual pixels, we use relative encodings $\mathbf{P}$ between query pixel x_i and key pixel x_j (Shaw et al., 2018)

$$\mathbf{A}_{\mathbf{i},\mathbf{j}} = \mathbf{x}_\mathbf{i}\mathbf{W}_Q((\mathbf{x}_\mathbf{j} + \mathbf{P}_{x_j - x_i})\mathbf{W}_K)^T. \quad (14)$$

We embed this formulation into the II framework by transforming the input using bi-linear interpolation and apply the group average over all results.

$$A[\text{SA}](x) = \frac{1}{|G|}\sum_{g\in G} \text{softmax}(L_g\mathbf{A})L_g\mathbf{x}\mathbf{W}_V, \quad (15)$$

where $L_g\mathbf{A}$ denotes calculating the attention scores using the transformed input. We also investigate multi-head self-attention (MH-SA) where H self-attention layers are calculated, concatenated and processed by a linear layer with weights $\mathbf{W}_o \in \mathbb{R}^{HC_h\times C_o}$. This formulation is related to (Romero and Cordonnier, 2020), where opposed to our approach equivariance is enforced using adapted positional encodings.

Table 1: Mean Test Error (MTE) of different monomial selection types on Rotated-MNIST using II-SF-CNN (Rath and Condurache, 2020). ✓ indicates full pre-training, ◐ iterative pre-training for a small number of epochs and x pruning at initialization.

Selection	Pre-Train	Init.	MTE [%]
SF-CNN	-	-	0.714 ± 0.022
-	x	Random	0.751 ± 0.032
LSE	✓	Random	0.687 ± 0.012
Connectivity	x	Random	0.758 ± 0.0025
Connectivity	◐	Catalog	0.708 ± 0.010
Connectivity	◐	Random	0.705 ± 0.027
Magnitude	◐	Catalog	0.704 ± 0.022
Magnitude	◐	Random	0.677 ± 0.031

5 EXPERIMENTS & DISCUSSION

We evaluate the different setups on Rotated-MNIST, SVHN, CIFAR-10 and STL-10. For each dataset, we choose a baseline architecture, assume that the feature extraction network is highly optimized and focus on the role of the II layer. We keep the number of parameters for the equivariant networks constant by adapting the number of channels per layer (see Appendix). We conduct experiments using the full training data, but more importantly limited subsets to investigate the sample complexity of the different variants. When training on limited datasets, we keep the number of total training iterations constant and adapt all hyper-parameters depending on epochs, such as learning rate decay, accordingly. All data subsets are sampled randomly with constant class ratios and are equal among all architectures. We optimized the hyper-parameters using Bayesian Optimization with Hyperband (Falkner et al., 2018) and a train-validation split of 80/20. Implementation details and hyper-parameters can be found in the Appendix.

5.1 Evaluating Monomial Selection

We evaluate the monomial selection methods on Rotated-MNIST, a dataset for hand-written digit recognition with randomly rotated inputs including 12k training and 50k testing grayscale-images (Larochelle et al., 2007). Therefore, we train a SF-CNN with five convolutional and three fully connected layers where we insert II in between (Weiler

et al., 2018b; Rath and Condurache, 2020). For all layers, we use $n_\alpha = 16$ rotations. Table 1 shows the performance of the different monomial selection algorithms. We perform five runs for each dataset size using data augmentation with random rotations and report the mean test error and the standard deviation for the full dataset.

The results in Table 1 indicate that magnitude-based pruning with random pre-selection outperforms both the LSE baseline and the connectivity-pruning approach for monomial selection. Random initial selection outperforms the catalog-based approach. Furthermore, it is evident that the monomial selection algorithm plays a key part and allows a relative performance increase of up to 10.9% compared to a purely random monomial selection. Therefore, we use randomly initialized magnitude-based pruning with pre-training for all following monomial experiments.

5.2 Evaluating Alternatives to Monomials on Digits

We further use Rotated-MNIST to evaluate the monomial replacement candidates using the training setup from above on full and limited datasets (Table 2). We observe that all variants of II outperform the baseline SF-CNN utilizing pooling and a standard seven layer CNN trained with data augmentation (as used for comparison in (Cohen and Welling, 2016)). Especially in the limited-data domain, II-enhanced networks achieve a better performance despite adding more parameters. Consequently, II successfully reduces the data-complexity and thereby improves the generalization ability. We conjecture that this is due to the II layer better preserving information that effectively contributes to successful classification compared to spatial pooling, i.e., II explicitly enforces invariance without afflicting other relevant information.

For all practical purposes, monomial-based II performs on par with the alternative functions which enable a streamlined training procedure. Thus, it seems possible to replace the monomials with other functions in order to avoid the monomial selection step while maintaining the performance. This would further reduce the training time and at the same time provide a setup in which the II layer can be optimally tuned and adapted to the other layers in the network. All proposed functions are well-known in deep learning literature which supports the practical deployment. In order to show that the benefits of II do not only stem from additional model capacity but from effectively leveraging prior knowledge, we add another steerable G-Conv and perform average pooling as a special case of II (SF-II) which performs clearly inferior.

We outperform the E(2)-CNNs (Weiler and Cesa, 2019) when they only incorporate invariance to rotations and achieve comparable results when they use a bigger invariance group including flips. The WS approach shows the most promising results among the different monomial replacement candidates.

We also conduct experiments on SVHN in order to assess the performance of II on real-world datasets that do not involve artificially induced global invariances. It contains 73k training and 10k test samples of single digits from house numbers in its core dataset (Netzer et al., 2011). We use WRN16-4 as baseline (Zagoruyko and Komodakis, 2016) and conduct experiments on the full dataset and limited subsets (Table 3). We compare the WRN to a SF-WRN and to II based on monomials, global- and local-WS with $k = 3$. For all following experiments, we use $n_\alpha = 8$ angles for the steerable convolutions as well as II and perform three runs per network and dataset size.

The II-based approach generally outperforms both the standard WRN16-4 as well as the equivariant baseline which achieves invariance using pooling. This proves that II is useful for real-world setups with non-transformed input data and can be applied to complex DNN architectures such as WRNs. The monomial and local-WS approach seem to perform best among all dataset sizes, with local-WS achieving slightly better results. We believe this is due to the fact that the architecture using this newly proposed function can be trained more efficiently. Additionally, training can be conducted in a single run without intermediate pruning steps since the monomial selection is avoided. Global-WS achieves worse results over all dataset sizes. Generally we assume that differences in performance among various methods over data size have to do with the trade-off between how good a specific architecture is able to leverage the prior knowledge on rotation invariance and how good it is able to learn and preserve other relevant invariance cues contained in real-world datasets such as color changes or illuminations.

5.3 Object Classification on Real-world Natural Images

To evaluate our approach on more complex classification settings including more variability, we use CIFAR-10 and STL-10. CIFAR-10 is an object classification dataset with 50k training and 10k test RGB-images (Krizhevsky, 2009). STL-10 is a subset of ImageNet containing 5,000 labeled training images from 10 classes (Coates et al., 2011). It is commonly used as a benchmark for semi-supervised learning and clas-

Table 2: MTE on limited subsets of Rotated-MNIST using SF-CNN as baseline (Weiler et al., 2018b).

			Number/Percentage of samples					
G-Conv	II	f	500/4.2%	1k/8.3%	2k/17%	4k/33%	6k/50%	12k/100%
x	x	-	8.635	7.205	5.586	4.684	4.324	3.664 ± 0.082
✓	x	Pooling	3.543	2.529	1.660	1.337	1.126	0.714 ± 0.022
✓	✓	Monomials	3.115	2.194	1.593	1.322	1.068	0.677 ± 0.031
✓	✓	Global-WS	3.120	2.294	1.614	1.200	1.004	0.712 ± 0.027
✓	✓	Local-WS	3.168	2.292	1.612	1.186	1.032	0.688 ± 0.032
✓	✓	MLP	3.250	2.310	1.652	1.242	1.024	0.732 ± 0.023
✓	✓	MH-SA	3.178	2.268	1.666	1.294	1.038	0.710 ± 0.022
✓	✓	SF-II	3.352	2.542	1.836	1.346	1.128	0.782 ± 0.012
E(2)-CNN, Rotation								0.705 ± 0.025
E(2)-CNN, Rotation & Flips								0.682 ± 0.022

Table 3: MTE on limited subsets of SVHN using WRN16-4 (Zagoruyko and Komodakis, 2016) as baseline.

			Number/Percentage of samples					
G-Conv	II	f	1k/1.3%	5k/6.9%	10k/14%	50k/69%	73k/100%	# Param.
x	x	-	12.72	6.37	4.96	3.29	3.00 ± 0.01	2.75M
✓	x	Pooling	11.15	5.52	4.46	3.25	2.89 ± 0.09	2.76M
✓	✓	Monomials	**10.67**	5.45	4.51	3.10	2.79 ± 0.03	2.78M
✓	✓	Global-WS	11.37	6.45	4.96	3.32	2.95 ± 0.07	2.83M
✓	✓	Local-WS	**10.70**	**5.04**	**4.31**	**3.00**	**2.69** ± 0.01	2.77M

sification with limited training data.

We use WRN28-10 and WRN16-8 as baseline architecture, respectively and test II with monomials and local-WS with $k = 3$. For CIFAR-10, we train on full data as well as on limited subsets using standard data augmentation with random crops and flips (Table 4). For STL-10 (Table 5), we use random crops, flips and cutout (Devries and Taylor, 2017).

On CIFAR-10, we notice two developments: While our networks outperform the WRN28-10 in the limited-data domain, indicating an improved sample complexity, they are unable to achieve better results in large-data regimes (Table 4). Networks employing II achieve a better performance than the pooling counterpart among all dataset sizes indicating that II better preserves the information needed for a successful classification leading to a lower sample complexity.

Local-WS performs on par or slightly worse than the monomials. We conjecture that on bigger dataset sizes, our approach with its rotation-invariant focus does not capture the complex local object-related invariant cues needed for successful classification as good as a standard WRN. We remark that for SVHN, relevant invariance cues besides rotation are rather global (e.g., color, illumination, noise), while for CIFAR these are also local and object-related (e.g., perspective changes, occlusions). Thus, our method handles global invariances well while needing additional steps to handle local invariances other than rotation.

(Weiler and Cesa, 2019) (E(2)-WRN) achieve better results than our networks in this setup. However, their approach differs from ours by loosening equivariance restrictions with depth and using a bigger invariance group including flips, thus addressing more local invariances. Nevertheless, this approach can be combined with ours in the future.

On STL-10, both II-enhanced networks outperform the equivariant baseline using pooling and the standard WRN. The local-WS approach outperforms the monomial counterpart. On this basis, we conclude that for all practical purposes, II based on local-WS delivers best results while being simpler to train than the monomial variant. Again, other methods incorporating invariance to other groups such as the general E(2)-CNN (Weiler and Cesa, 2019) or scales (SES-CNN, (Sosnovik et al., 2020)) achieve better results than our purely rotation-invariant network. This is intuitive since samples from ImageNet involve variability from an even greater source of different transformations than CIFAR-10. Consequently, the invariance cues that need to be captured by a classifier are even more complex.

6 CONCLUSION

In this contribution, we focused on leveraging prior knowledge about invariance to transformations for classification problems. Therefore, we adapted the II framework by introducing a novel monomial selection algorithm and replacing the monomials with different functions such as a weighted sum, a MLP, and self-attention. Replacing the monomials enabled a streamlined training of DNNs using II by avoiding the pre-training and selection step. This allows to optimally tune and adapt all algorithmic components at

Table 4: MTE on limited subsets of CIFAR-10 using WRN28-10 (Zagoruyko and Komodakis, 2016) as baseline.

G-Conv	II	f	Number/Percentage of samples 100/0.2%	1k/2%	10k/20%	50k/100%	#Param.
x	x	-	71.69	37.61	**9.08**	**3.89** $\pm$ 0.02	36.5M
✓	x	Pooling	76.54	37.29	12.68	4.71 $\pm$ 0.04	36.7M
✓	✓	Monomials	**69.42**	**29.83**	11.15	4.60 $\pm$ 0.12	36.8M
✓	✓	Local-WS	72.72	32.10	10.45	4.54 $\pm$ 0.15	36.9M
E(2)-WRN28-10						***2.91***	$\sim$37M

Table 5: MTE on STL-10 using WRN16-8 (Zagoruyko and Komodakis, 2016) as baseline.

G-Conv	II	f	MTE[%]	# Param.
x	x	-	12.74 $\pm$ 0.23	10.97M
✓	x	Pooling	12.51 $\pm$ 0.33	10.83M
✓	✓	Monomials	10.84 $\pm$ 0.46	10.85M
✓	✓	Local-WS	**10.09** $\pm$ 0.21	10.92M
E(2)-WRN16-8			9.80 $\pm$ 0.40	12.0M
SES-WRN16-8			***8.51***	11.0M

once promoting the application of II to complex real-world datasets and architectures, e.g., WRNs.

Our method explicitly enforces invariance which we see among the key factors to be taken into consideration by a feature-extraction engine for successful classification, especially for real-world applications, where data is often limited. Assuming that rotation invariance is required, we have shown how to design a DNN based on II to leverage this prior knowledge. In comparison to the standard approach, we replace spatial max-pooling by a dedicated layer which explicitly enforces invariance while increasing the network's expressibility. To enable the network to capture other invariance cues in particular of global nature we use a trainable weights as well.

We have demonstrated state-of-the-art sample complexity on datasets from various real-world setups. We achieve state-of-the-art results on all data regimes on image classification tasks when the targeted invariances (i.e., rotation) generate the most intraclass variance, as in the case of Rotated-MNIST and SVHN. On Rotated-MNIST, we even outperform the E(2)-CNN which also includes invariance to flips.

On CIFAR-10 and STL-10, we show top performance in limited-data regimes for image classification tasks where various other transformations besides rotation are responsible for the intraclass variance. At the same time, the performance in the full-data regime is better than the equivariant baseline, which shows that we are able to effectively make use of prior knowledge and introduce rotation invariance without afflicting other learned invariances. Specifically, monomials and local-WS achieve the best and most stable performance and consistently outperform the baseline, which uses group and spatial pooling, as well as standard convolutional architectures. Local-WS performs similarly or better than monomials while being easier to apply and optimize due to avoiding the monomial selection step. It is different to simply adding an additional group-equivariant layer and performing average pooling among rotations and spatial locations because group pooling is performed before applying the II layer. Compared to TI-Pooling (Laptev et al., 2016), our method explicitly guarantees invariance within a single forward pass. In contrast, TI-Pooling approximates invariance by pooling among the responses of a non-equivariant network needing one forward pass per group element.

Our current method is limited to problem settings where rotation invariance is desired. The expansion to other transformations is interesting future work. We also plan to investigate replacing all pooling operations with II.

ACKNOWLEDGEMENTS

The authors would like to thank their colleagues Lukas Enderich, Julia Lust and Paul Wimmer for their valuable contributions and fruitful discussions.

REFERENCES

Bekkers, E. J. (2020). B-spline cnns on lie groups. In *ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Bekkers, E. J., Lafarge, M. W., Veta, M., Eppenhof, K. A. J., Pluim, J. P. W., and Duits, R. (2018). Roto-translation covariant convolutional networks for medical image analysis. In *MICCAI 2018, Granada, Spain, September 16-20, 2018, Proceedings, Part I*, pages 440–448.

Coates, A., Ng, A. Y., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In Gordon, G. J., Dunson, D. B., and Dudík, M., editors, *AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 215–223. JMLR.org.

Cohen, T., Weiler, M., Kicanaoglu, B., and Welling, M. (2019a). Gauge equivariant convolutional networks and the icosahedral CNN. In *ICML 2019, 9-15 June 2019, Long Beach, CA, USA*, pages 1321–1330.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2990–2999.

Cohen, T. S., Geiger, M., and Weiler, M. (2019b). A general theory of equivariant cnns on homogeneous spaces. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 9142–9153.

Condurache, A. P. and Mertins, A. (2012). Sparse representations and invariant sequence-feature extraction for event detection. *VISAPP 2012*, 1.

Devries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552.

Diaconu, N. and Worrall, D. E. (2019a). Affine self convolution. *CoRR*, abs/1911.07704.

Diaconu, N. and Worrall, D. E. (2019b). Learning to convolve: A generalized weight-tying approach. In *ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 1586–1595.

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. (2018a). Learning SO(3) equivariant representations with spherical cnns. In *ECCV 2018, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, pages 54–70.

Esteves, C., Allen-Blanchette, C., Zhou, X., and Daniilidis, K. (2018b). Polar transformer networks. In *ICLR 2018*.

Falkner, S., Klein, A., and Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1436–1445.

Finzi, M., Welling, M., and Wilson, A. G. (2021). A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *CoRR*, abs/2104.09459.

Freeman, W. T. and Adelson, E. H. (1991). The design and use of steerable filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(9):891–906.

Fuchs, F., Worrall, D. E., Fischer, V., and Welling, M. (2020). Se(3)-transformers: 3d roto-translation equivariant attention networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *NeurIPS 2020, December 6-12, 2020, virtual*.

Fuchs, F. B., Wagstaff, E., Dauparas, J., and Posner, I. (2021). Iterative se(3)-transformers. *CoRR*, abs/2102.13419.

Ghosh, R. and Gupta, A. K. (2019). Scale steerable filters for locally scale-invariant convolutional neural networks. *CoRR*, abs/1906.03861.

Han, S., Pool, J., Tran, J., and Dally, W. J. (2015). Learning both weights and connections for efficient neural network. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *NeurIPS 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1135–1143.

Hutchinson, M., Lan, C. L., Zaidi, S., Dupont, E., Teh, Y. W., and Kim, H. (2020). Lietransformer: Equivariant self-attention for lie groups. *CoRR*, abs/2012.10885.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. (2015). Spatial transformer networks. In *NeurIPS 2015*, pages 2017–2025. Curran Associates, Inc.

Kanazawa, A., Sharma, A., and Jacobs, D. W. (2014). Locally scale-invariant convolutional neural networks. *CoRR*, abs/1412.5104.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kondor, R., Lin, Z., and Trivedi, S. (2018). Clebsch-gordan nets: a fully fourier space spherical convolutional neural network. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 10138–10147.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images,. Technical report.

Laptev, D., Savinov, N., Buhmann, J. M., and Pollefeys, M. (2016). TI-POOLING: transformation-invariant pooling for feature learning in convolutional neural networks. In *CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 289–297.

Larochelle, H., Erhan, D., Courville, A. C., Bergstra, J., and Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML 2007, Corvallis, Oregon, USA, June 20-24, 2007*, pages 473–480.

LeCun, Y., Bengio, Y., and Hinton, G. E. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lee, N., Ajanthan, T., and Torr, P. H. S. (2019). Snip: single-shot network pruning based on connection sensitivity. In *ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Marcos, D., Kellenberger, B., Lobry, S., and Tuia, D. (2018). Scale equivariance in cnns with vector fields. *CoRR*, abs/1807.11783.

Marcos, D., Volpi, M., Komodakis, N., and Tuia, D. (2017). Rotation equivariant vector field networks. In *ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5058–5067. IEEE Computer Society.

Müller, F. and Mertins, A. (2009). Invariant-integration method for robust feature extraction in speaker-independent speech recognition. In *INTERSPEECH 2009, Brighton, United Kingdom, September 6-10, 2009*, pages 2975–2978.

Müller, F. and Mertins, A. (2010). Invariant integration features combined with speaker-adaptation methods. In *INTERSPEECH 2010, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 2622–2625.

Müller, F. and Mertins, A. (2011). Contextual invariant-integration features for improved speaker-independent speech recognition. *Speech Communication*, 53(6):830–841.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Noether, E. (1916). Der endlichkeitssatz der invarianten endlicher gruppen. *Mathematische Annalen*, 77:89–92.

Rath, M. and Condurache, A. P. (2020). Invariant integration in deep convolutional feature space. In *ESANN 2020, Bruges, Belgium, October 2-4, 2020*, pages 103–108.

Reisert, M. and Burkhardt, H. (2006). Invariant features for 3d-data based on group integration using directional information and spherical harmonic expansion. In *ICPR 2006, 20-24 August 2006, Hong Kong, China*, pages 206–209. IEEE Computer Society.

Romero, D. W., Bekkers, E. J., Tomczak, J. M., and Hoogendoorn, M. (2020). Attentive group equivariant convolutional networks. In *ICML 2020, 13-18 July 2020, Virtual Event*, pages 8188–8199. PMLR.

Romero, D. W. and Cordonnier, J. (2020). Group equivariant stand-alone self-attention for vision. *CoRR*, abs/2010.00977.

Romero, D. W. and Hoogendoorn, M. (2020). Co-attentive equivariant neural networks: Focusing equivariance on transformations co-occurring in data. In *ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Schulz-Mirbach, H. (1992). On the existence of complete invariant feature spaces in pattern recognition. In *Pattern Recognition: Eleventh International Conference 1992*, pages 178 – 182.

Schulz-Mirbach, H. (1994). Algorithms for the construction of invariant features. In *Tagungsband Mustererkennung 1994 (16. DAGM Symposium), Reihe Informatik Xpress, Nr.5*, pages 324–332.

Schulz-Mirbach, H. (1995). Invariant features for gray scale images. In *Mustererkennung 1995, 17. DAGM-Symposium, Bielefeld, 13.-15. September 1995, Proceedings*, pages 1–14.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. In Walker, M. A., Ji, H., and Stent, A., editors, *NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 464–468. Association for Computational Linguistics.

Sosnovik, I., Szmaja, M., and Smeulders, A. W. M. (2020). Scale-equivariant steerable networks. In *ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Tai, K. S., Bailis, P., and Valiant, G. (2019). Equivariant transformer networks. In *ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6086–6095.

Veeling, B. S., Linmans, J., Winkens, J., Cohen, T., and Welling, M. (2018). Rotation equivariant CNNs for digital pathology. *CoRR*, abs/1806.03962.

Walters, R., Li, J., and Yu, R. (2020). Trajectory prediction using equivariant continuous convolution. *CoRR*, abs/2010.11344.

Weiler, M. and Cesa, G. (2019). General e(2)-equivariant steerable cnns. In *NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 14334–14345.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. (2018a). 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 10402–10413.

Weiler, M., Hamprecht, F. A., and Storath, M. (2018b). Learning steerable filters for rotation equivariant cnns. In *CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 849–858.

Winkels, M. and Cohen, T. S. (2019). Pulmonary nodule detection in CT scans with equivariant cnns. *Medical Image Anal.*, 55:15–26.

Worrall, D. E. and Brostow, G. J. (2018). Cubenet: Equivariance to 3d rotation and translation. In *ECCV 2018, Munich, Germany, September 8-14, 2018, Proceedings, Part V*, pages 585–602.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. (2017). Harmonic networks: Deep translation and rotation equivariance. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 7168–7177.

Worrall, D. E. and Welling, M. (2019). Deep scale-spaces: Equivariance over scale. In *NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 7364–7376.

Xu, Y., Xiao, T., Zhang, J., Yang, K., and Zhang, Z. (2014). Scale-invariant convolutional neural networks. *CoRR*, abs/1411.6369.

Yang, F., Wang, Z., and Heinze-Deml, C. (2019). Invariance-inducing regularization using worst-case transformations suffices to boost accuracy and spatial robustness. In *NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 14757–14768.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In Wilson, R. C., Hancock, E. R., and Smith, W. A. P., editors, *BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press.

Zhu, W., Qiu, Q., Calderbank, A. R., Sapiro, G., and Cheng, X. (2019). Scale-equivariant neural networks with decomposed convolutional filters. *CoRR*, abs/1909.11193.

APPENDIX

Implementation Details. To increase the reproducibility, we provide our exact hyper-parameter settings. We optimized the standard Wide-ResNets using stochastic gradient descent and the hyper-parameters of the corresponding paper (Zagoruyko and Komodakis, 2016). All steerable networks were optimized using Adam optimization (Kingma and Ba, 2015). We used exponential learning rate decay for Rotated-MNIST and SVHN, while we employed step-wise decay on CIFAR-10 and STL-10. All steerable filter weights were regularized using elastic net regularization with factor 10^{-7} (cf. (Weiler et al., 2018b)). For all WRNs, we additionally use ℓ_2-regularization for the learnable BatchNorm coefficients with factor 0.1. All regularization losses were then multiplied by the regularization constant.

Table 6: II-SF-CNN hyper-parameters on Rotated-MNIST.

Hyper-parameter	MH-SA	Global-WS	Local-WS	MLP	SF-Conv	Monomials	SF-CNN
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Batch Size	32	32	32	32	32	32	64
Epochs	100	100	100	100	100	100	100
n_{FC}	95	85	30	85	85	90	96
Learning Rate	5e-3	1e-4	1e-3	1e-4	5e-4	1e-4	1e-3
LR Decay	0.5	0.1	0.5	0.1	0.2	0.75	0.9
LR Decay Epoch	20	40	25	30	25	15	20
Reg. Constant	1e-3	0.1	1e-3	1e-3	0.01	0.15	1.
Dropout Rate	0.05	0.45	0.4	0.5	0.4	0.45	0.7
Attention Heads	1	-	-	-	-	-	-
Attention Dropout	0.	-	-	-	-	-	-

Table 7: Hyper-parameters on SVHN.

Hyper-parameter	SF-CNN	Global-WS	Local-WS	Monomials
Optimizer	Adam	Adam	Adam	Adam
Batch Size	128	128	128	64
Epochs	100	100	100	100
n_{FC}	32	64	36	85
Learning Rate	1e-3	5e-4	5e-4	5e-4
LR Decay	0.4	0.1	0.25	0.25
LR Decay Epoch	20	30	25	20
Reg. Constant	2e-3	0.25	0.2	0.05
Dropout Rate	0.55	0.7	0.5	0.7

Table 8: Hyper-parameters on CIFAR-10.

Hyper-parameter	SF-CNN	Local-WS	Monomials
Optimizer	Adam	Adam	Adam
Batch Size	64	64	64
Epochs	100	100	100
n_{FC}	-	90	30
Learning Rate	1e-3	5e-4	5e-4
LR Decay	0.5	0.2	0.025
LR Decay Epoch	50	50	50
Reg. Constant	0.1	5e-6	0.008
Dropout Rate	0.3	0.1	0.4

Table 9: Hyper-parameters on STL-10.

Hyper-parameter	SF-CNN	Local-WS	Monomials
Optimizer	Adam	Adam	Adam
Batch Size	96	64	32
Epochs	1000	1000	1000
n_{FC}	-	16	10
Learning Rate	5e-4	0.01	5e-4
LR Decay	0.1	0.3	0.1
LR Decay Epoch	300	300	300
Reg. Constant	1e-8	1e-9	5e-9
Dropout Rate	0.1	0.15	0.05

The hyper-parameters were optimized using Bayesian Optimization with Hyperband (BOHB, (Falkner et al., 2018)) on 80/20 validation splits, if it was not already predetermined by the dataset. They are shown in Tables 6-9. On Rotated-MNIST we used data augmentation with random rotations following (Weiler et al., 2018b). On CIFAR-10 and SVHN, we followed (Zagoruyko and Komodakis, 2016) and used random crops and flips for CIFAR-10 and no data augmentation for SVHN. On STL-10, we use random crops, flips and cutout (Devries and Taylor, 2017) .

For the monomial architectures, we applied II per channel, and pruned $M = 50$ initial monomials to $n_m = 5$ for Rot-MNIST and $n_m = 10$ monomials for SVHN, CIFAR-10 and STL-10. We used one intermediate pruning step after 10 epochs with $n_m = 25$ and train additional 5 epochs before the final pruning step. All other invariant integration layers were implemented with constant number of channels.

Number of Parameters. For our invariant architectures, we keep the number of parameters constant by reducing the number of channels accordingly. A standard convolutional filter with kernel size k, c_i input channels and c_o output channels has $k^2 c_i c_o$ parameters. A rotation-steerable filter has $2 n_F n_\alpha c_i c_o$ parameters with n_α rotations and n_F basis filters. In order to keep the number of parameters constant, we equate

$$k^2 c_i c_o = 2 n_F n_\alpha \tilde{c}_i \tilde{c}_o \quad \Leftrightarrow \quad \frac{\tilde{c}_i \tilde{c}_o}{c_i c_o} = \frac{2 n_F n_\alpha}{k^2} \tag{16}$$

We use $k = 3$, $n_\alpha = 8$, $n_F = 16$ and obtain a final ratio of $\frac{256}{9}$ by which we need to reduce $c_i c_o$. Hence, we reduce the number of channels by $\sqrt{\frac{256}{9}} = \frac{16}{3}$. For the lifting convolution, the filter is not used among all rotations, so we only need to reduce the ratio by $\sqrt{\frac{32}{9}}$.