Invariant Integration in Deep Convolutional Feature Space

Matthias Rath[1,2] and Alexandru P. Condurache[1,2]

1- University of Lübeck - Institute for Signal Processing

2- Robert Bosch GmbH - Automated Driving

Abstract. In this contribution, we show how to incorporate prior knowledge to a deep neural network architecture in a principled manner. We enforce feature space invariances using a novel layer based on invariant integration. This allows us to construct a complete feature space invariant to finite transformation groups.

We apply our proposed layer to explicitly insert invariance properties for vision-related classification tasks, demonstrate our approach for the case of rotation invariance and report state-of-the-art performance on the Rotated-MNIST dataset. Our method is especially beneficial when training with limited data.

1 Introduction

Deep neural networks (DNNs) are the state-of-the-art method to solve a wide variety of complex tasks from computer vision to speech recognition. In general, the weights of those networks are optimized exploiting a vast amount of data. However, collecting and especially labeling data, which is necessary for supervised learning, is time-consuming and labour-intensive.

Consequently, current research investigates the integration of prior knowledge to DNNs in order to improve the data efficiency of the learning process, thus improving the performance when training data is limited or scarce [1]. One promising approach is to exploit prior knowledge by enforcing in- or equivariance properties to symmetric geometrical transformations such as rotation and scaling. This can be achieved by replacing the randomly initialized weights of convolutional layers with predefined or restricted filters [2, 3, 4]. Alternatively, the *standard convolution*, which is equivariant to translations, can be substituted by a more general form called *group convolution* [5], which guarantees equivariance with respect to more sophisticated transformation groups. Weiler et al. combine both approaches by restricting the convolution weights to linear combinations of steerable filters and applying the group convolution to the activations [6]. The aforementioned methods build invariant feature spaces by pooling from equivariant feature spaces. In practice however, those feature spaces are affected by sampling effects and are thus not completely invariant. To avoid this, one could replace the simple pooling operation with an advanced operation geared towards enforcing invariance properties.

We propose to use invariant integration (II) for this purpose. II is a method for constructing a complete feature space that is invariant to a transformation group [7, 8]. It has successfully been applied to object detection from grayscale

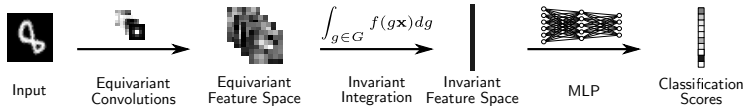

Fig. 1: We apply invariant integration in a learned equivariant feature space to form a feature space invariant to rotations and translations and use fully connected layers for classification.

images [9], automatic speech recognition [10] and event detection from image sequences [11]. However, all of these approaches use II to create a feature space used by conventional classifiers, such as a *Support Vector Machine*.

We propose a novel DNN layer based on II embedded into a convolutional neural network (CNN) architecture. The II layer (IIL) builds upon an equivariant feature layer to foster an invariant feature representation (see Figure 1). We demonstrate that the backpropagation algorithm is applicable with respect to the IIL's parameters and input. Hence, its parameters as well as preceding layers can be optimized en bloc. We test our method on the Rotated-MNIST dataset [12] and demonstrate state-of-the-art performance. The largest performance gains are achieved when only few labeled training samples are available.

2 A DNN architecture based on Invariant Integration

Group theory lays the foundation of II. A group G is a mathematical abstraction consisting of a set of elements acted upon by an operation under the axioms of closure, associativity, identity and invertibility. A function f is said to be equivariant with respect to a group G of transformations, if we can determine an exact relationship between transformations $g \in G$ of the function's input x and a corresponding transformation $g' \in G$ of the function's output:

$$f(gx) = g'f(x) \; \forall \; x \in X.$$

Invariance is a special case of equivariance, where the induced transformation in the output space g' is the identity.

An example of an equivariant function is the convolutional layer, which is equivariant to translations. In practice, we are also interested to enforce invariance to other transformation groups than translations, e.g. rotations. However, this is more difficult to achieve in an explicit manner.

II is an algorithm to construct a complete feature space with respect to a symmetric transformation first proposed by Schulz-Mirbach [7, 8]. A feature space is defined as complete, if all patterns which are equivalent with respect to a transformation group G are mapped to the same point in the feature domain, while all distinct patterns are mapped to different points. This means that a complete feature space is invariant to transformations $g \in G$ of the input signal.

In [8], Schulz-Mirbach proposes a general algorithm to construct a complete feature space relying on the group average A, which is defined by integrating an

invariant function f over transformations $g \in G$ acting on the input space $\mathbf{x}$:

$$A[f](\mathbf{x}) := \int_{g \in G} f(g\mathbf{x})\, dg.$$

For the invariant function f, the set of all possible monomials $m(\mathbf{x})$ represents a good choice for creating a complete feature space. These are defined as:

$$m(\mathbf{x}) = \prod_{i=1}^{K} x_i^{b_i} \quad \text{with} \quad \sum_i b_i \leq |G|,$$

where K is the size of the input feature and b_i are the monomial's exponents. The upper bound for the number of all possible monomials is $\binom{K+|G|}{K}$.

2.1 Rotation-Invariant Integration in Image Space

We apply II using monomials over the group of rotations and translations in image space [9]. Thus, our input is defined as $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, where H, W and C are the image's width, height and number of channels. We apply the group average independently to each channel $c = 1, \ldots, C$ and reformulate:

$$\begin{aligned} A[f](\mathbf{x}) &= \sum_u \sum_v \int_\phi m(\mathbf{x}(u, v; \phi))\, d\phi \\ &= \sum_u \sum_v \int_\phi \prod_{i=1}^{K} \mathbf{x}(u + d_{u,i} \sin(\phi), v + d_{v,i} \cos(\phi))^{b_i}\, d\phi, \end{aligned}$$

with the monomial distances $d_{u,i}$ and $d_{v,i}$, the pixel offsets u and v and the integration angle ϕ. The output is of dimension $\mathbb{R}^{C \times M}$, where M is the number of monomials. Contrarily to previous work, we choose to apply rotation-invariant integration directly in convolutional feature space.

2.2 Monomial Selection

Since including all possible monomials is computationally expensive, the question arises, how to choose a meaningful subset. While the exponents b_i can be optimized using backpropagation (see Chapter 2.4), choosing the monomial orders K and their distances is non-trivial.

We use an iterative approach for feature selection called *Feature Finding Neural Network* (FFNN) proposed by Gramss [13]. It has successfully been used to select the monomials for II in automatic speech recognition by Mueller and Mertins [10] and is based on iteratively ranking monomial combinations according to the *least square error* (LSE) of a linear classifier. While using a linear classifier facilitates good generalization ability, the LSE can be computed in closed-form, which enables an efficient calculation.

We adapt this method by utilizing the closed-form solution of a linear classifier to rank the feature combinations according to the classifiers' validation accuracies instead of the LSE. The iterative selection is stopped, if the validation accuracy has not improved within the past ten iterations.

2.3 Architecture and Training Process

The architecture for a classification network using the proposed IIL is shown in Figure 1. The IIL is used to construct an invariant feature space with respect to transformations of the convolutional feature space it acts on. To ensure that the whole DNN is invariant, the convolutional feature space must be equivariant with respect to transformations of the DNN's input. Thus, we use an equivariant backbone built out of equivariant convolutional layers and apply the IIL to build an complete invariant feature space (see Figure 1). Finally, we use fully connected layers to obtain the classification scores. In comparison, the networks proposed in [2, 6] use max- or sum-pooling to create an invariant feature space from their equivariant convolutional feature space.

During training, we first train the baseline network without the IIL. Thereafter, we apply II on the features of the last convolutional layer and select M monomials using our iterative sampling approach. Finally, we re-train the entire network including the IIL and the preceding equivariant convolutional layers.

2.4 Backpropagation for Invariant Integration Layers

To enable using II for the construction of a complete feature space in combination with DNNs it is inevitable that the proposed feature transformation is differentiable with respect to both its parameters and inputs. The gradients of a monomial with respect to a single input x_j and to the learnable exponents b_j are respectively defined as:

$$\frac{\delta m(\mathbf{x})}{\delta x_j} = b_j x_j^{b_j-1} \prod_{i=1, i\neq j}^{K} x_i^{b_i}, \quad \frac{\delta m(\mathbf{x})}{\delta b_j} = \log(x_j) x_j^{b_j} \prod_{i=1, i\neq j}^{K} x_i^{b_i}.$$

Investigating these formulas, it is evident that we need to enforce $x_i \neq 0 \ \forall i$, because $\lim_{x_i \to 0} \frac{\delta m(\mathbf{x})}{\delta x_j} = 0$ and $\lim_{x_i \to 0} \frac{\delta m(\mathbf{x})}{\delta b_j} = 0$, would lead to vanishing gradients backpropagated to earlier layers and to the exponents of the IIL, respectively.

We further restrict the input values to $\mathbb{R}^+$ because multiplying by a negative number alters the sign of the output, hence leading to great shifts in output space. Finally, we note that the identity element of a multiplication is 1, while a multiplication with values close to 0 would eradicate the result. Following the aforementioned insights, we choose to shift our input features: $\tilde{\mathbf{x}} = \max(\epsilon, \mathbf{x} - x_{min} + 1)$, with $0 < \epsilon \ll 1$. In our implementation, we shift the input per channel, while x_{min} is the minimal value that occurred within that channel when processing the training data. Lastly, we apply the max-function to ensure that there are no values smaller than ϵ during testing.

3 Experiments and Discussion

We use our approach to enforce rotation invariance for the recognition of randomly rotated hand-written digits on the Rotated MNIST dataset [12]. It contains 10,000 training, 2,000 validation and 50,000 test images. We optimize the

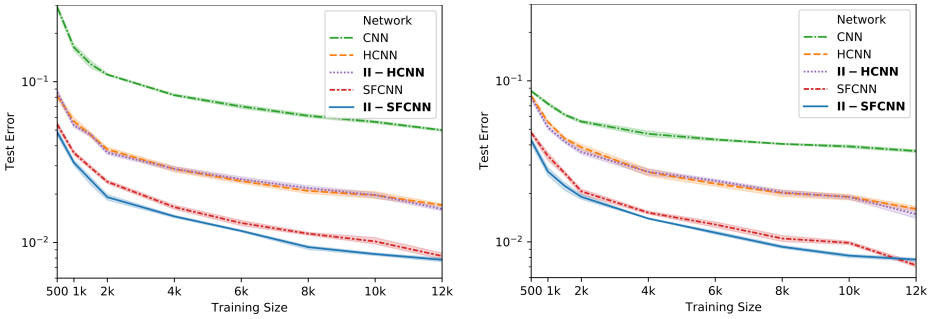

Fig. 2: Mean test error using limited training data without (*left*) and with (*right*) augmented data. The transparent corridors represent the results' standard deviation.

hyper-parameters on the validation set. Thereafter, we retrain on the training and validation sets to report the final results on the test set (cf. [2, 6]).

For the equivariant backbone, we use two different approaches that are equivariant to rotations in image space. First, we apply harmonic convolutions (HCNN) proposed by Worrall et. al [2]. We use the modulus of the final complex convolutional layer as the input to our IIL. Secondly, we use the Steerable Filter CNN (SFCNN) proposed by Weiler et. al [6]. We max-pool along the angle dimension of the final convolutional layer and apply channel-wise II. We empirically choose to use $M = 5$ monomials and to integrate over 8 angles using bilinear interpolation to obtain a good trade-off between performance and computational complexity.

Method	Augmentation	
	×	✓
CNN [2, 5]	5.130	3.772
HCNN [2]	1.730	1.606
II-HCNN	1.548	1.377
SFCNN [6]	0.880	**0.714**
II-SFCNN	**0.799**	0.776

Table 1: Mean test error [%].

We compare the different approaches by means of the mean test error over ten independent training runs. In Table 1, we show results obtained when training on the entire training and validation set both with and without data augmentation. We evaluate a standard CNN as well as the HCNN and the SFCNN without and with the IIL. The data was augmented by randomly rotating the input during training. When combining II with the HCNN, we achieve a relative improvement of 10.52 % without and 14.26 % with data augmentation. The II-SFCNN improves the baseline by 9.2 %. However, when training with augmented data, the SFCNN achieves the best performance, but only by a small margin.

Figure 2 shows the mean test error and the standard deviation when training with limited data. When using only a subset of the available training data, we achieve significant improvements over the state-of-the-art baseline algorithms. It seems that enforcing the sought invariances on top of an equivariant feature space by using II instead of the simpler pooling operation is particularly beneficial when only limited data is available. We believe this is related to the fact that a small training set typically does not include the variability needed to properly capture the sought invariances without explicitly enforcing them. Additionally, we observe that the performance boost is bigger when applying the IIL to the SFCNN because its feature representation is equivariant to sampled rotations,

while the HCNN guarantees equivariance to continuous rotations. Enforcing the invariance with the IIL seems to mitigate those sampling effects.

4 Conclusion and Future Work

We proposed a novel layer based on II which allows us to incorporate prior knowledge expressed as the need to obtain features invariant to certain geometrical transformations of the input, e.g. rotations. We apply it to enforce rotation invariance in a CNN architecture. We show competitive performance when training on the full dataset and state-of-the-art results in the low data regime. This indicates that our approach enables a better transition from equivariant to invariant features than pooling when the learned feature representation is not exactly equivariant, e.g. due to sampling effects. Our method is especially helpful, when training with fewer training samples.

In the future, II can be expanded towards other transformation groups, e.g. scale. Additionally, the IIL can be applied in more complex neural network architectures. We will investigate, if hand-designing the monomials is possible, thus allowing to incorporate additional prior knowledge and avoiding the pre-training of the baseline network necessary to enable selecting appropriate monomials.

References

[1] B. Coors, A. P. Condurache, and A. Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *ECCV 2018*, pages 525–541.

[2] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *CVPR 2017*, pages 7168–7177.

[3] E. Oyallon, E. Belilovsky, and S. Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *ICCV 2017*, pages 5619–5628.

[4] F. Cotter and N. G. Kingsbury. A learnable scatternet: Locally invariant convolutional layers. *CoRR*, abs/1903.03137, 2019.

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

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

[7] H. Schulz-Mirbach. On the existence of complete invariant feature spaces in pattern recognition. In *ICPR 1992*, pages 178 – 182.

[8] H. Schulz-Mirbach. Algorithms for the construction of invariant features. In *DAGM-Symposium 1994*, pages 324–332.

[9] H. Schulz-Mirbach. Invariant features for gray scale images. In *DAGM-Symposium 1995*, pages 1–14.

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

[11] A. P. Condurache and A. Mertins. Sparse representations and invariant sequence-feature extraction for event detection. *VISAPP 2012*, pages 679–684.

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

[13] T. Gramss. Word recognition with the feature finding neural network (ffnn). In *NNSP 1991*, pages 289–298.