LETTER

Rectifying Transformation Networks for Transformation-Invariant Representations with Power Law

Chunxiao FAN[†], **Yang LI**[†,††], ***Nonmembers***, **Lei TIAN**[†], ***and*** **Yong LI**[†a)], ***Members***

SUMMARY This letter proposes a representation learning framework of convolutional neural networks (Convnets) that aims to rectify and improve the feature representations learned by existing transformation-invariant methods. The existing methods usually encode feature representations invariant to a wide range of spatial transformations by augmenting input images or transforming intermediate layers. Unfortunately, simply transforming the intermediate feature maps may lead to unpredictable representations that are ineffective in describing the transformed features of the inputs. The reason is that the operations of convolution and geometric transformation are not exchangeable in most cases and so exchanging the two operations will yield the transformation error. The error may potentially harm the performance of the classification networks. Motivated by the fractal statistics of natural images, this letter proposes a rectifying transformation operator to minimize the error. The proposed method is differentiable and can be inserted into the convolutional architecture without making any modification to the optimization algorithm. We show that the rectified feature representations result in better classification performance on two benchmarks.

key words: *convolutional neural network, deep learning, image representation, transformation error*

1. Introduction

Current state-of-the-art algorithms in computer vision are based on deep neural networks, especially the convolutional neural networks (Convnets), such as object detection/instance segmentation [1] and pose estimation [2]–[4]. A crucial aspect of these successful work is the translation-invariant representations learned by Convnets which are matched to the statistics of natural images [5], [6]. However, natural image statistics are largely transformation-invariant to other forms as well, such as isotropic scaling and rotation. This motivates many extensive studies of the Convnets to generalize convolutional features to other geometric transformations, which can be roughly categorized as manually transforming (augmenting) images/features [7], [8], manipulating filters [9], and learning transformations [10], [11].

A lot of work adopted the three types of methods or other alternatives to handle the variations in data in computer vision tasks. Among them, data augmentation is probably one of the best-known methods. Unfortunately, only a limited number of transformations can actually result in meaningful samples and others may lead to noise instances or wrong labels [12], [13]. Bengio *et al.* [12] provided the explanation of the phenomenon from the perspective of manifold and presented that the high-level representations expand the relative volume occupied by high-probability points within feature space (Hypothesis 3), which suggests that the plausible and meaningful instances gain more opportunity to appear when sampled in feature space than in pixel space. Inspired by the work, DeVries *et al.* [14] proposed to project the feature space into a new space learned by an autoencoder then conduct feature augmentation in the learned space. Similarly, Liu *et al.* [15] presented a novel feature augmentation network for object recognition task. It is worth noting that these algorithms are specifically designed for feature space, and directly applying transformations appropriate for pixel space to feature space (as done in [8], [10], etc.) may lead to unpredictable representations, which is estimated by the 'transformation error' in this letter.

To see where the error comes from, let T denote an arbitrary affine transformation operation and W denote the convolution. It is easy to see that T and W are two linear mappings that do not generally satisfy the commutative law of multiplication. Therefore, the common fact is that transforming directly the convolutional feature maps does not give the same solution as the data augmentation gives.

To reveal this, a verification experiment is carried out about the rotation transformation on the Rotated MNIST dataset [16] via the CNN network described in [11]. Data/Feature augmentation is exerted on from the data layer to the third convolutional layer respectively.

Table 1 shows a degradation problem: with the augmentation depth increasing, performance degrades rapidly. Unexpectedly, such degradation is neither caused by optimization since all the networks reach the similar training error (see Fig. 1) nor by overfitting since they all have exactly the same architecture and number of parameters. The

Table 1 Comparisons of different augmentation on Rotated MNIST [16]. The network architecture is [conv(3x3, 3)-conv(3x3, 6)-maxpooling(2x2)-conv(3x3, 9)-conv(3x3, 12)-FC(48)-FC(10)] following [11]. The training lasts for 300 epochs using 64 batch size. The initial learning rate is 0.01 and then divided by 10 at the 150th and 225th epoch.

Augmentation on	*Data layer*	*Conv1*	*Conv2*	*Conv3*
Test error rate (%)	2.47	2.76	3.52	4.00

Manuscript received May 30, 2018.
Manuscript revised October 30, 2018.
Manuscript publicized December 4, 2018.

[†]The authors are with School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China.
[††]The author is with Department of Electronic Engineering, Tsinghua University, Beijing 100084, China.
a) E-mail: yli@bupt.edu.cn

DOI: 10.1587/transinf.2018EDL8112

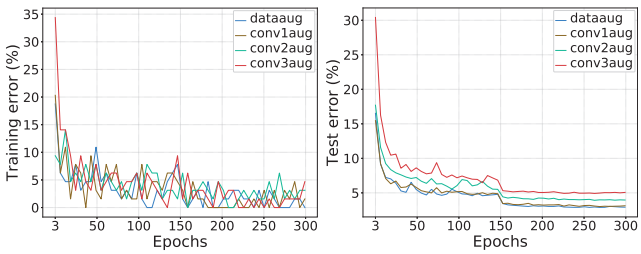

Fig. 1 Training error and test error on Rotated MNIST. All networks converge well but the network with deeper augmentation performs worse.

only difference among them is the depth of augmentation. The degradation of performance indicates that the transformation error indeed exists and is not negligible.

The potentially negative influence of the transformation error is that the solver has to optimize parts of parameters towards correcting the error, which will result in that these parameters are not used to learn the valuable semantic knowledge and lead to the consequential reduction in representation capability. Even though including extra parameters (e.g., stacking more layers) could be a compromise solution, this will result in less control over the model capacity and higher risk of overfitting.

For addressing the transformation error in a parameter-efficient way, this letter explicitly constructs a rectifying transformation network (RTN), which is motivated by the power law [17] that derives from the fractal statistics of images [18], [19]. The RTN contains a preprocessing function for normalizing the transformed features and a power-law function for rectifying the transformation error. The power-law function can be seen as the prior knowledge of the statistical relationship of the raw features and the transformed ones. By introducing the prior knowledge, the RTN is able to decrease the transformation error with considerably smaller number of parameters than the approach of stacking more layers, which in return sets free the convolutional parameters to learn valuable semantic knowledge.

2. The Proposed Method

This section proposes a method to rectify the transformation error described above. Given that the transformation error originates from the non-commutativity, we seek a new mapping to compensate for it, making the two forms equal approximately. Formally, the mapping is defined as

$$f = \arg\min_{f} \| \mathbf{X} - f(\mathbf{Y}) \|,$$
$$where,\ \mathbf{X} = W \otimes T(I),\ \mathbf{Y} = T(W \otimes I). \quad (1)$$

Here, I is the input image, W is the weights of convolutional layer, $\otimes$ denotes the convolution operation, T denotes the affine transformation, and f is the rectifying transformation mapping which can be either a linear function or a non-linear one. By this definition, the problem becomes the model selection of f and its optimization.

Model Selection. Theoretically, f can take any form, such as the Convnets or a function manually designed. In this letter, we manually define f as the linear affine function, which is motivated by the power law [17]–[19]

$$E[\phi(I_{s1})]/E[\phi(I_{s2})] = (s_1/s_2)^{-\lambda_\phi}. \quad (2)$$

Here, s is the scale factor, λ is the parameter of power function, ϕ denotes an arbitrary image statistic (features, in other words), and E denotes the expectation over an ensemble of input images. The power law suggests that the raw image features and its transformed features have potential geometric similarity. It applies to the translation-invariant low-level features. Readers are referred to [17] for details.

Considering that the Convnets have hierarchies of representations and the convolution operation covers the inherent translation-invariance of natural images, the power law can be generalized to the lower layers of Convnets. By applying the power law, Eq. (1) yields

$$f(\lambda_c, \varepsilon) = \arg\min_{(\lambda_c,\varepsilon)} \| \mathbf{X} - s^{-\lambda_c} \cdot \mathbf{Y} + \varepsilon \|, \quad (3)$$
$$where,\ \mathbf{X} = W \otimes T(I),\ \mathbf{Y} = T(W \otimes I).$$

Here, λ_c should be a channel-wise parameter according to the power law, c is the index of channels, ε is a scalar bias, and s is a parameter determined by the affine transformation. Particularly, if T is the scale transformation, s is the factor of the scale. During the forward pass of Convnets, T and the corresponding s are known. Thus, the parameters that need to be optimized are λ_c and ε.

Model Evaluation. Let e denote the transformation error. In practice, one of convenient methods to measure e could be the test error rate of classification. The method is also intuitive because the classification error is proportional to e and if e is reduced, the test error rate should be reduced as well. Consequently, to simplify the evaluation of e, we define the ratio of performance reducing

$$\eta = \max\left(\frac{E_{f(T(W\otimes X))} - E_{W\otimes T(X)}}{E_{T(W\otimes X)} - E_{W\otimes T(X)}},\ 0\right). \quad (4)$$

Here, E denotes the classification error during test. If f can rectify the error e, we obtain $E_{f(T(W\otimes X))} \approx E_{W\otimes T(X)}$, making Eq. (4) tend to be zero. If f learns nothing, we have $E_{f(T(W\otimes X))} \approx E_{T(W\otimes X)}$, driving Eq. (4) close to one. If η keeps close to zero under different transformations, we say that the model is insensitive to the transformation error.

A few particular cases are analyzed in what follows. If $E_{T(W\otimes X)} \le E_{W\otimes T(X)}$, the error does not dominate the benefits of augmenting features and therefore η is set to zero. If $E_{f(T(W\otimes X))} < E_{W\otimes T(X)}$, f may learn extra information beneficial to the model performance, for example, the channel relationship like [20]. In this case, η is zero.

Optimization. In [17], λ_c is approximately estimated by the least squares fit over a group of features of original images and their scaled ones. However, the method does not support

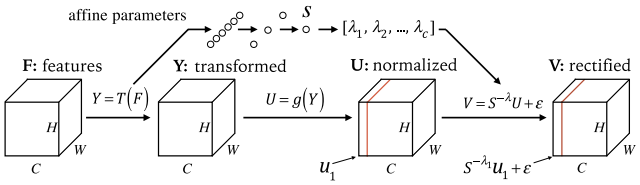

Fig. 2 The framework of the rectifying transformation networks.

the end-to-end training. For this reason, this letter chooses to optimize λ_c and ε by SGD in a data-driven manner.

Note that the transformation T could be the identity mapping in which case f should be the identity mapping as well. Nevertheless, according to He *et al.* [21], the solver (e.g., SGD) might have difficulties in approximating the identity mapping, which may potentially harm the convergence of the network. To address this, this letter proposes to exert a normalization transformation after T to avoid solving the identity mapping for f. Therefore, Eq. (3) becomes

$$f(\lambda_c, \varepsilon) = \arg\min_{(\lambda_c, \varepsilon)} \| g(\mathbf{X}) - s^{-\lambda_c} \cdot g(\mathbf{Y}) + \varepsilon \|, \quad (5)$$
$$where,\ \mathbf{X} = W \otimes T(I),\ \mathbf{Y} = T(W \otimes I).$$

Here, g is the normalization function. Eq. (5) is the proposed method of rectifying the transformation error. For each T (parameterized via s) exerted on the convolutional feature maps, we rectify the outputs by g, $s^{-\lambda_\phi}$, and ε.

Rectifying Transformation Networks. We construct the rectifying transformation networks (RTN) by embedding Eq. (5) immediately after the convolutional features, as illustrated in Fig. 2. The convolutional features **F** are firstly augmented by the transformation T. Then the result **Y** is normalized by g. The form of g should be selected according to the computer vision task and so g is not limited to a specific normalization method. In this letter, we choose to normalize the transformed features by subtracting their means and then dividing by their standard deviations. Finally, the normalized features **U** are rectified by $s^{-\lambda_c}$ and ε. Here, s is computed through a Fully-Connected network taking the parameters of T as inputs. λ_c and ε are optimized by SGD in a data-driven manner.

3. Experiments

3.1 Rectifying for Manual Augmentation

This section evaluates the RTN and compares it with two other approaches, stacking more layers and learning transformations. Experiments are still conducted on the rotated MNIST dataset [30] which containing 12k training samples and 50k test samples perturbed with the rotation transformation by an angle uniformly sampled between 0 and 2π. We continue to adopt this dataset because **1)** it is good to check if the proposed method can work on the case described in Table 1; **2)** this dataset is appropriate to examining the invariance of features and a lot of work such as [10], [11] also chose it or other variants of MNIST as the testbed.

Table 2 Comparison among stacking more layers (CNN), learning transformations (STN), and the RTN for addressing the transformation error. The rotation transformations are exerted on from the data layer to the third convolutional layer, independently and respectively. The results are reported by test error rate (and the transformation error η)(lower is better). Here, η is computed based on the baseline CNN6 results.

Network	*Capacity*	*Data layer*	*Conv1*	*Conv2*	*Conv3*
CNN6	39k	2.47	2.76	3.52	4.00
CNN7	50k	2.43	2.58 (38%)	2.97 (48%)	3.26 (52%)
CNN8	62k	2.27	2.43 (0%)	2.86 (37%)	3.01 (35%)
STN6	50k	2.30	2.36 (0%)	2.82 (33%)	3.05 (38%)
STN7	64k	2.02	2.10 (0%)	2.59 (11%)	2.76 (19%)
STN8	75k	1.92	1.97 (0%)	2.30 (0%)	2.55 (5%)
RTN6	40k	-	2.49 (7%)	2.89 (40%)	3.64 (76%)
RTN8	63k	-	**1.91 (0%)**	**2.16 (0%)**	2.96 (32%)

The network architecture is the same as that in [11] (see Table 1). The random rotation transformation (sampled uniformly from 0 to 2π) is adotpted as the augmentation method, T, for both training stage and test stage, to match the variations in the dataset. To evaluate the transformation error, the rotation transformations are exerted on from the data layer to the third convolutional layer, independently and respectively. For STN, the ST module is configured as that in the original paper and placed immediately after the augmentation operation to learn transformations. The Fully-Connected network for s in RT is configured as [FC(3)-FC(1)]. Note that the RT module allows to be inserted anywhere in the network since supporting for the end-to-end training. In this experiment, RT is placed immediately after all the convolutional features. The training strategy remains the same as that in Table 1.

Experimental Results. Table 2 shows an observable phenomenon verifying the description in Sect. 1. For the augmentation on conv1, the CNN6 achieved 2.76% test error rate, which is 0.29% higher than the baseline result (data augmentation) due to the transformation error. By stacking more layers, the CNN7 is able to reduce the transformation error to η of 38%, yielding the test error rate of 2.58%. The transformation error, however, still exists. To totally eliminate the transformation error, the network has to add more parameters and depth. The resulted CNN8 is able to reduce η to zero but the classification performance is only comparable to the baseline. These results indicate that the extra parameters were barely used to learn the semantic knowledge and discrimination capability but optimized by the solver to rectify the transformation error. This phenomenon can also be observed in other cases of augmentation. Expectedly, if there is no transformation error, the 7/8-layer CNN can enjoy the benefits of stacking more layers.

Compared to the approach of stacking more layers, learning transformations is more parameter-efficient. Table 2 shows that STN6 has the comparable parameters to CNN7 but decreases the transformation error to zero for the augmentation on conv1. Similarly, the capacity of STN7

is comparable to that of CNN8, but STN7 can reduce the transformation error and the classification error to lower level. This phenomenon suggests that apart from a method to obtain transformation-invariances, learning transformations can also be used as a technique of reducing the transformation error. We conjecture that it is because the technique is able to map the transformed features by T into a new feature space in which the realistic data points are more likely to appear.

Even though the transformation error can be reduced by stacking more layers or learning transformations, these methods are highly parameter-inefficient. CNN8 takes an extra 20k parameters to eliminate the transformation error on conv1, and STN6 introduces nearly 10k parameters to rectify the error. In contrast, the RTN is more parameter-efficient. It needs only an extra less than 1k parameters to decrease the transformation error to η of 7%.

Both CNN8 and RTN8 rectified the transformation error with comparable parameters but RTN8 further decreased the classification error to 1.91%. This result indicates that RTN could release the convolutional parameters for learning the desired semantic representation. In addition, RTN8 has similar model capacity to the STN7 but achieved lower test error rate for both conv1 and conv2 augmentation. Note that both STN and RTN include the side-branch. The difference between them is that STN reduces the transformation error by learning new affine mappings while RTN achieves the goal by manually formulating the form of the mapping using power law.

It is worth noting that RTN performs worse than CNN and STN with augmentation on conv3. It only achieved η of 76% and 32% with network depth of 6 and 8 respectively, which is a drastic reduction in performance. For this phenomenon, we argue that it is reasonable because the foundation of RTN is the power law which only works for the low-level features. For shallower networks, the conv3 features might contain high-level semantic information over low-level geometric information. Therefore, the proposed method did not work in this case.

3.2 Rectifying for Learning Transformation

This section evaluates the RTN for the case of learning transformations. The experiments are performed with the challenging real-world dataset, the Street View House Numbers (SVHN) [22], for the task of sequence prediction. This dataset consists of around 200k images as training samples and 24k images as test samples. For this task, we adopt the network architecture and training strategy from STN [10] (with 11 hidden layers). In this architecture, 'STN Single' is obtained by inserting a ST module immediately after the data layer, and 'STN Multi' is constructed by including four ST modules for the first four convolutional layers. The transformations, T, are completely determined (learned) by the ST module. The data augmentation strategy follows [23].

Table 3 The sequence error (%) on the SVHN dataset for crops of 64 × 64 / 128 × 128 pixels. *: the results are from the original papers.

Network	64 × 64 Crop Size	128 × 128 Crop Size
Maxout CNN [23]*	4.0	-
DRAM [24]*	3.9	4.5
CNN [10]*	4.0	5.6
STN Single [10]*	3.7	3.9
STN Multi [10]*	3.6	3.9
STN Multi + RTN (ours)	**3.2**	**3.6**

Experimental Results. Table 3 shows that the STN Multi performs only marginally better than the STN Single on this dataset. From the transformation error perspective, the reason is that STN Multi keeps introducing the transformation error while reducing the error. It introduces the transformation error since it acts as a spatial feature transformer like feature augmentation. On the other hand, it decreases the transformation error through the backward propagation of the sequence error (the transformation error affects the training loss). Therefore, it is difficult for the STN Multi to considerably outperform the STN Single for this task.

To evaluate the proposed method, we place the RT modules immediately after the transformed convolutional features, denoted by STN Multi + RTN in Table 3. As we can see, the sequence error decreases from 3.6% to 3.2% for the 64 × 64 crop size and from 3.9% to 3.6% for the 128 × 128 crop size. These results indicate that after rectifying the transformation error by RTN, STN Multi is able to enjoy the benefits of multiple transformers. From this we can conclude that the proposed RTN is indeed helpful for learning transformations with multiple intermediate transformers.

4. Conclusion

In this letter, we propose a rectifying transformation network (RTN) towards correcting the transformation error. Stacking more layers or learning transformations can also deal with this problem by the backward propagation of the classification error but they are very parameter-inefficient and undesired. Compared to them, our method is more parameter-efficient owing to the introduction of the prior knowledge of the power law. We test the method on two datasets and found that it not only rectifies the transformation error but further decreases the classification error.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. NSFC-61471067, 81671651), the National Great Science Specific Project (No. 2015ZX03002008), the Beijing Municipal Natural Science Foundation (No. 4172024), Beijing University of Posts and Telecommunications (No. 2013XD-04, 2015XD-02).

References

[1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," The IEEE International Conference on Computer Vision (ICCV), pp.2980–2988, 2017.
[2] H. Guo, G. Wang, X. Chen, C. Zhang, F. Qiao, and H. Yang, "Region ensemble network: Improving convolutional network for hand pose estimation," 2017 IEEE International Conference on Image Processing (ICIP), pp.4512–4516, 2017.
[3] G. Wang, X. Chen, H. Guo, and C. Zhang, "Region ensemble network: Towards good practices for deep 3d hand pose estimation," J. Vis. Commun. Image. R., vol.55, pp.404–414, 2018.
[4] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.7794–7803, 2018.
[5] J.F. Henriques and A. Vedaldi, "Warped convolutions: Efficient invariance to spatial transformations," Proc. International Conference on Machine Learning, pp.1461–1469, 2017.
[6] A. Hyvärinen, J. Hurri, and P.O. Hoyer, Natural Image Statistics - A Probabilistic Approach to Early Computational Vision, Springer, 2009.
[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Proc. International Conference Learning Representation, arXiv preprint: 1409.1556., 2015.
[8] A. Kanazawa, A. Sharma, and D. Jacobs, "Locally scale-invariant convolutional neural networks," Advances in Neural Information Processing Systems Workshops, 2014.
[9] D.E. Worrall, S.J. Garbin, D. Turmukhambetov, and G.J. Brostow, "Harmonic networks: Deep translation and rotation equivariance," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.7168–7177, 2017.
[10] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," Advances in Neural Information Processing Systems 28, pp.2017–2025, 2015.
[11] C.-H. Lin and S. Lucey, "Inverse compositional spatial transformer networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2252–2260, 2017.
[12] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai, "Better mixing via deep representations," International Conference on Machine Learning, pp.552–560, 2013.
[13] D. Laptev, N. Savinov, J.M. Buhmann, and M. Pollefeys, "TI-POOLING: Transformation-invariant pooling for feature learning in Convolutional Neural Networks," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.289–297, 2016.
[14] T. DeVries and G.W. Taylor, "Dataset augmentation in feature space," Proc. International Conference Learning Representation Workshop, arXiv preprint: 1702.05538, 2017.
[15] B. Liu, X. Wang, M. Dixit, R. Kwitt, and N. Vasconcelos, "Feature Space Transfer for Data Augmentation," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.9090–9098, 2018.
[16] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," Proc. 24th International Conference on Machine Learning - ICML '07, pp.473–480, 2007.
[17] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," IEEE Trans. Pattern Anal. Mach. Intell., vol.36, no.8, pp.1532–1545, 2014.
[18] D.L. Ruderman and W. Bialek, "Statistics of natural images: Scaling in the woods," Phys. Rev. Lett., vol.73, no.6, pp.814–817, 1994.
[19] D.L. Ruderman, "The statistics of natural images," Network: Computation in Neural Systems, vol.5, no.4, pp.517–548, 1994.
[20] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.7132–7141, 2018.
[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.770–778, 2016.
[22] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A.Y. Ng, "Reading digits in natural images with unsupervised feature learning," Advances in Neural Information Processing Systems Workshops, p.5, 2011.
[23] I.J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," Proc. International Conference Learning Representation, arXiv preprint: 1312.6082., 2014.
[24] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," Proc. International Conference Learning Representation, arXiv preprint:1412.7755, 2015.