Training Very Deep Networks via Residual Learning with Stochastic Input Shortcut Connections

Oyebade K. Oyedotun(✉), Abd El Rahman Shabayek, Djamila Aouada, and Björn Ottersten

Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, 1855 Luxembourg City, Luxembourg
{oyebade.oyedotun,abdelrahman.shabayek,djamila.aouada, bjorn.ottersten}@uni.lu
http://wwwen.uni.lu/snt

Abstract. Many works have posited the benefit of depth in deep networks. However, one of the problems encountered in the training of very deep networks is feature reuse; that is, features are 'diluted' as they are forward propagated through the model. Hence, later network layers receive less informative signals about the input data, consequently making training less effective. In this work, we address the problem of feature reuse by taking inspiration from an earlier work which employed residual learning for alleviating the problem of feature reuse. We propose a modification of residual learning for training very deep networks to realize improved generalization performance; for this, we allow stochastic shortcut connections of identity mappings from the input to hidden layers. We perform extensive experiments using the USPS and MNIST datasets. On the USPS dataset, we achieve an error rate of 2.69% without employing any form of data augmentation (or manipulation). On the MNIST dataset, we reach a comparable state-of-the-art error rate of 0.52%. Particularly, these results are achieved without employing any explicit regularization technique.

Keywords: Deep neural networks · Residual learning · Dropout · Optimization

1 Introduction

Neural networks have been extremely useful for learning complex tasks such as gesture recognition [1] and banknote recognition [2]. More recently, as against shallow networks with one layer of feature abstraction, there has been massive interest in deep networks which compose many layers of features abstractions. There are many earlier works [3,4] which established that given a sufficiently large number of hidden units, a shallow network is a universal function approximator. Interestingly, many works addressing the benefit of depth in neural

D. Liu et al. (Eds.): ICONIP 2017, Part II, LNCS 10635, pp. 23–33, 2017.
https://doi.org/10.1007/978-3-319-70096-0_3

networks have also emerged. For example, using the concept of sum-product networks, Delalleau and Bengio [5] posited that deep networks can efficiently represent some family of functions with lesser number of hidden units as compared to shallow networks. In addition, Mhaskar et al. [6] provided proofs in their work that deep networks are capable of operating with lower Vapnik-Chervonenkis (VC) dimensions. Bianchini and Scarselli [7] employing some architectural constraints, derived upper and lower bounds for some shallow and deep architectures; they concluded that using the same resources (computation units), deep networks are capable of representing more complex functions than shallow networks. In practice, the success of deep networks have corroborated the position that deep networks have a better representational capability as compared to shallow networks; many state-of-the-art results on benchmarking datasets are currently held by deep networks [8–10].

In recent times, the aforementioned theoretical proofs, practical results and new works [11,12] now suggest that employing *even deeper* networks could be quite promising for learning even *more complex or highly varying functions*. However, it has been observed that the training of models beyond some few layers results in optimization difficulty [13,14]. In this work, for the sake of clear terms, we refer to models with 2–10 hidden layers as 'deep networks', models with more than 10 hidden layers as 'very deep networks' and use the term 'deep architecture' to refer interchangeably to a deep network or very deep network. We consider the effective training of very deep networks; that is, simultaneously overcoming optimization problems associated with model depth increase and more importantly improving generalization performance. We take inspiration from an earlier work which employed residual learning for training very deep networks [14]. However, training very deep models with millions of parameters come with the price of over-fitting. On one hand, various explicit regularization schemes such as L^1-norm, L^2-norm and max-norm can be employed for alleviating this problem. On the other hand, a more appealing approach is to explore some form of implicit regularization such as reducing the co-adaptation of model units on one another for feature learning (or activations) [19] and encouraging stochasticity during optimization [8]. In this work, we advance in this direction with some modifications on the form of residual learning that we propose for implicitly improving model regularization by *emphasizing stochasticity* during training. Our contribution is that we propose to modify residual learning for training very deep networks where we allow shortcut connections of identity mappings from the input to the hidden layers; such shortcut connections are stochastically removed during training. Particularly, the proposed training scheme is shown to improve the implicit regularization of very deep networks as compared to the conventional residual learning. We employ our proposed approach for performing extensive experiments using the USPS and MNIST datasets; results obtained are quite promising and competitive with respect to state-of-the-art results.

The rest of this paper is organized as follows. Section 2 discusses related works.

Section 3 serves as background and introduction of residual learning. Section 4 gives the description of the proposed model. Section 5 contains experiments, results and discussion on benchmark datasets. In Sect. 6, we conclude the work with our key findings.

2 Related Work

The optimization difficulty observed in training very deep networks can be attributed to the fact that input features get diluted from the input layer through the many compositional hidden layers to the output layer; this is evident in that each layer in the model performs some transformation on the input received from the preceding layer. The several transformations with model depth may make features not reusable. Here, one can conjecture that the signals (data features) which reach the output layer for error computation may be significantly less informative for effective weights update (or correction). Many works have provided interesting approaches for alleviating the problem of training deep architectures. In [15,16], carefully guided initializations were considered for specific activation functions; these initializations were found useful for improving model optimization and the rates of convergence. In another interesting work [17], batch normalization was proposed for tackling the problem of internal covariate shift which arises from non-zero mean hidden activations. Nevertheless, the problem of training (optimizing) very deep networks commonly arises when the number of hidden layers exceeds 10; see Fig. 1. For example, Srivastava et al. [13] employed *transform* gates for routing data through very deep networks; they refer to their model as a *highway network*. The concept is that the transform gates are either closed or open. When the transform gates are closed, input data are routed through the hidden layers without transformations; in fact, each hidden layer essentially copies the features from the preceding layer. However, when the transform gates are open, the hidden layers perform the conventional features transformations using layer weights, biases and activation functions. Inasmuch as the highway network was shown to allow for the optimization of very deep networks and improving classification accuracies on benchmark datasets, it comes with a price of learning additional model parameters for the transform gates. Another work, He et al. [14] has addressed the problem of feature reuse by using residual learning for alleviating the dilution (or attenuation) of features during forward propagation through very deep networks; they refer to their model as a *ResNet*. The ResNet was also shown to alleviate optimization difficulty in training very deep networks. In [33], identity shortcut connections were used for bypassing a subset of layers to facilitate training very deep networks.

3 Background: Very Deep Models and Residual Learning

3.1 Motivation

We emphasize the problem of training very deep networks using the USPS dataset. Figure 1-left shows the performance of plain deep architectures with

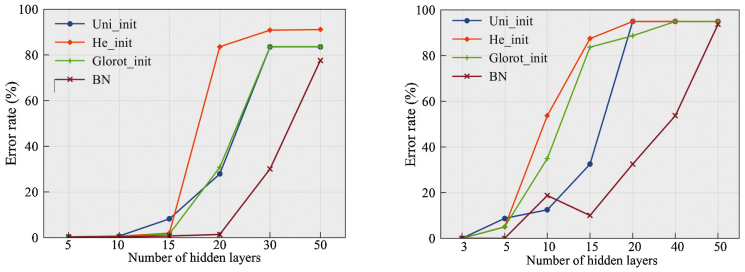

Fig. 1. Performance of deep architectures with depth. Left: Train error on USPS dataset. Right: Train error on COIL-20 dataset. It is seen that optimization becomes more difficult with depth

a different number of hidden layers. Particularly, it will be seen that the performance of the models significantly dips from over 10 hidden layers. We further emphasize this problem by going beyond the typical uniform initialization (i.e. Unit_init in Fig. 1) scheme for neural network models; we employ other initialization and training techniques which have been proposed for more effective training of deep models; these techniques include Glorot [15] initialization, He [16] initialization and batch normalization [17] which are shown as Glorot_init, He_init and BN in Fig. 1.

In addition, we investigate this problem using the COIL-20 dataset[1] which composes 1,440 samples of different objects of 20 classes. The concepts which we follow in using the COIL-20 dataset as sanity check are in two folds: (1) it is a small dataset, hence it is expected that deep architectures would easily overfit such training data (2) the dataset is of much higher dimensionality. Obviously, this training scenario can be seen as an extreme one which *indeed* favours deep models with enormous parameters for overfitting the training data. This follows directly from the concept of model complexity and curse of dimensionality with high dimensional input data as against the number of training data points. However, our experimental results do not support the overfitting intuition; instead, the difficulty of model optimization is observed when the number of hidden layers is increased beyond 10; see Fig. 1-right. It will be seen that for both USPS and COIL-20 datasets, training with batch normalization improved model optimization with depth increase. Nevertheless, model optimization remains a problem with depth increase. However, residual learning [14] has been employed in recent times for successfully training very deep networks. The idea is to scheme model training such that stacks of hidden layers learn residual mapping functions rather than the conventional transformation functions.

[1] http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html.

3.2 Residual Learning: ResNet

In this subsection, we briefly discuss residual learning as a building block for the model that we propose in this paper. In [14], residual learning was achieved by employing shortcut connections from preceding hidden layers to the higher ones. Given an input $H(x)^{l-1}$ (in block form), from layer $l-1$ feeding into a stack of specified number of hidden layers with output $H(x)^l$; in the conventional training scheme, the stack of hidden layers learns a mapping function of the form

$$H(x)^l = F^l(H(x)^{l-1}), \tag{1}$$

where the residual learning proposed in [14] uses shortcut connections such that the stack of hidden layers learns a mapping function of the form

$$H(x)^l = F^l(H(x)^{l-1}) + H(x)^{l-1}, \tag{2}$$

where $H(x)^{l-1}$ is the shortcut connection. The actual transformation function learned by the stack of hidden layers can be written as follows

$$F^l(H(x)^{l-1}) = H(x)^l - H(x)^{l-1}, \tag{3}$$

where $1 \leq l \leq L$ and $H(x)^0$ is the input data, x; L is the depth of the network. This training setup was found very effective in training very deep networks, achieving state-of-the-art results on some benchmarking datasets [14]. In a following work [18], dropping out the shortcut connections from preceding hidden layers was experimented with; however, convergence problems and unpromising results were reported.

4 Proposed Model

For improving the training of very deep models, we take inspiration from residual learning. Our proposed model incorporates some simple modifications to further improve on optimization and generalization capability as compared to the conventional ResNet. We refer to the proposed model as stochastic residual network (*S-ResNet*). The proposed training scheme is described below:

(i) There are identity shortcut connections of identity mappings from the input to hidden layers of the model; this is in addition to the shortcut connections from preceding hidden layers to the higher ones as seen in the conventional ResNets.
(ii) The identity shortcut connections from the input to the hidden layers are stochastically removed during training. Here, hidden layer units do not always have access to the untransformed input data provided via shortcut connections.
(iii) At test time, all the shortcut connections are present. The shortcut connections are not parameterized and therefore do not require rescaling at test time as in [8,33].

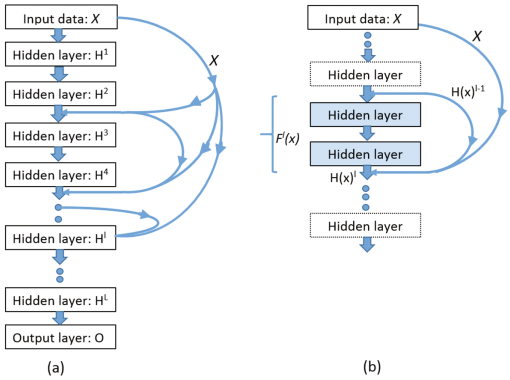

Fig. 2. (a) Proposed model with shortcut connections from the input to hidden layers (b) Closer view of the proposed residual learning with a hypothetical stack of two hidden layers

The proposed scheme for training very deep models is shown in Fig. 2(a); conventional shortcut connections from preceding hidden layers, with shortcut connections from the input to the different hidden layers are shown. For the modification that we propose in this work, the transformed output of a stack of hidden layers denoted, l, with shortcut connection from the preceding stack of hidden layers, $H(x)^{l-1}$, and shortcut connection from the input x can be written as follows

$$H(x)^l = F^l(H(x)^{l-1}) + H(x)^{l-1} + x. \tag{4}$$

where $1 \leq l \leq L \mid x = 0$ for $l = 1 \because \exists\ H(x)^0 = x$; $H(x)^l$, $F^l(H(x)^{l-1})$, $H(x)^{l-1}$ and x are of the same dimension. In this work, every stack of residual learning block composes two hidden layers. For a clearer conception of our proposed model, a single residual learning block of two hidden layers is shown in Fig. 2(b). From Fig. 2(b), assume that the underlying target function to be learned by a hypothetical residual learning block is $F^l(H(x)^{l-1})$, then using the aforementioned constraints on l, it learns a residual function of the form

$$F^l(H(x)^{l-1}) = H(x)^l - H(x)^{l-1} - x. \tag{5}$$

For dropout of shortcut connections from the input layer to the stack of hidden layers l, we can write

$$F^l(H(x)^{l-1}) = H(x)^l - H(x)^{l-1} - D * x, \tag{6}$$

where $D \in \{0, 1\}$ and $D \sim Bernoulli(p_s)$ determines that x (shortcut connection from input) is connected to the stack of hidden layers l with probability p_s; that is, $P(D = 1) = p_s$ and $P(D = 0) = 1 - p_s$ for $0 \leq p_s \leq 1$; and $*$ defines an operator that performs the shortcut connection, given the value of D. The conventional dropout probability for hidden units is denoted p_h.

5 Experiments and Discussion

For demonstrating the effectiveness of our proposed model, we train very deep networks and observe their optimization characteristics over various training settings using the USPS and MNIST datasets. The USPS dataset[2] composes handwritten digits 0–9 (10 classes) of 7,291 training and 2,007 testing samples; while the MNIST dataset[3] composes handwritten digits 0–9 of 60,000 training and 10,000 testing samples. For the USPS dataset, we use 2×2 convolutional filters, 2×2 max pooling windows and 2 fully connected layers of 300 ReLUs. For the MNIST dataset, we use 3×3 convolutional filters, 2×2 max pooling windows and 2 fully connected layers of 500 ReLUs. For both datasets, models have output layers of 10 softmax units. Our best model, 54-hidden layer S-ResNet, composes 50 convolution layers, 2 max pooling layers and 2 fully connected layers; we apply batch normalization only in the fully connected layers.

Figure 3-left shows the performance of our proposed model (S-ResNet) on the USPS dataset with different number of hidden layers at a dropout probability of $p_s = 0.8$ for the input shortcut connections to the hidden layers; for the conventional dropout of hidden units, a dropout probability of ($p_h = 0.6$) is used. It will be seen that with 54-hidden layers, our model achieves a state-of-the-art performance; that is, an error rate of 2.69%, surpassing the conventional ResNet (baseline model). In addition, Fig. 3-right shows the performance of the best proposed model (54 hidden layer S-ResNet) with different dropout probabilities for input shortcut connections to the hidden layers. Table 1 shows the error rates obtained on the test data for the USPS dataset along with the state-of-the-arts results. We observe that the models with asterisk (i.e. $*$) employed some form of data augmentation (or manipulation). For example, [26,27] extended the training dataset with 2,400 machine-printed digits; while [28] employed virtual data in addition to the original training data. However, our proposed model employs no such data augmentation tricks. The result obtained with our proposed model, 54-hiddden layer S-ResNet, surpasses many works which did not employ any form of data augmentation.

We repeat similar experiments on the MNIST dataset. Figure 4-left shows the error rates of the S-ResNets and the conventional ResNets with different number of hidden layers. It is observed that the S-ResNets are better regularized as compared to the ResNets for all the different model depths. Particularly, with 54 hidden layers, the S-ResNet achieved a result competitive with the state-of-the-art results; we reach an error rate of 0.52%. Figure 4-right shows the error rates of the 54-hidden layer S-ResNet with different dropout probabilities for the input shortcut connections to the hidden layers. In Table 2, we report the obtained error rates for our experiments, along with the best results reported in recent works. Also, for the MNIST dataset, we found that dropping out input shortcut connections to the hidden layers with a probability of 0.8 yielded the best result as given in Table 2. For both datasets, the S-ResNets employed no explicit

[2] http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html.
[3] http://yann.lecun.com/exdb/mnist/.

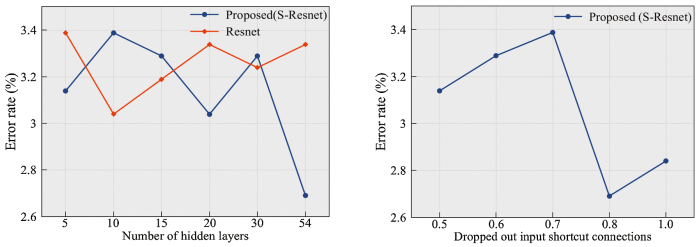

Fig. 3. Performance of deep architectures with depth on the USPS dataset. Left: Test error rate with depth. Right: Test error rate for different dropout probabilities of input shortcut connections

Table 1. Error rate (%) on the USPS dataset

Models	Test error (%)
Invariant vector supports [20]	3.00
Neural network (LetNet) [21]	4.20
Sparse Large Margin Classifiers (SLMC) [22]	4.90
Incrementally Built Dictionary Learning (IBDL-C) [23]	3.99
Neural network + boosting [21]	*2.60
Tangent distance [24]	*2.50
Human performance [24]	2.50
Kernel density + virtual data [25]	*2.40
Kernel density + virtual data + classifier combination [25]	*2.20
Nearest neighbour [25]	5.60
Baseline: Residual network (ResNet) - 54 hidden layers	3.34
Proposed model (S-ResNet) - 20 hidden layers	3.04
Proposed model (S-ResNet) - 54 hidden layers	**2.69**

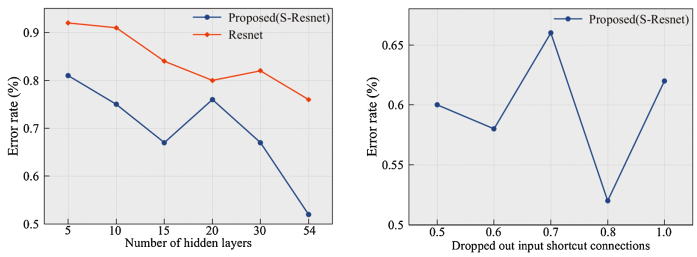

Fig. 4. Performance of deep architectures with depth on the MNIST dataset. Left: Test error rate with depth. Right: Test error rate for different dropout probabilities of input shortcut connections

Table 2. Error rate (%) on the MNIST dataset

Models	Test error (%)
Highway Net-16 [13]	0.57
Highway Net-32 [13]	0.45
Supervised Sparse Coding + linear SVM [26]	0.84
Deep Fried Convet [27]	0.71
PCANet [28]	0.62
Network in Network (NIN) [29]	0.45
Deeply Supervised Network (DSN) [30]	0.39
ConvNet + L-BFGS [31]	0.69
Neural network + adversarial examples [32]	0.78
Neural network ensemble + DropConnect [8]	0.52
Baseline: Residual network (Resnet) - 54 hidden layers	0.76
Proposed model (S-Resnet) - 15 hidden layers	0.64
Proposed model (S-Resnet) - 54 hidden layers	**0.52**

regularization technique for improving generalization capability; we relied on the implicit regularization of the models via dropout of input shortcut connections and hidden units for the S-ResNet, and dropout of hidden units only for ResNet. It is interesting to note that the proposed model do not suffer from convergence problem as reported in an earlier work which experimented with a similar training scheme [18]. In addition, the experimental results given in Tables 1 and 2 suggest that the proposed training scheme improves the implicit regularization of very deep networks; that is, lower test errors are achieved for the S-ResNets as compared to the ResNets. We conjecture that the simple modification employed for the proposed model helps to reduce the reliance of model units in one layer over others for feature learning. We observe that [8] also reported an error rate of 0.21%, however [8] employed some form of data augmentation using an ensemble of 5 neural networks; without data augmentation, they obtained a test error rate of 0.52%. Conversely, we employ no data augmentation and model ensemble.

6 Conclusion

Very deep networks suffer optimization problems even in situations that indeed favour over-fitting. Furthermore, assuming that we are able to optimize very deep networks, over-fitting is almost always inevitable due to large model capacity. We address the aforementioned problems by taking inspiration from residual learning. Our proposed model, stochastic residual network (*S-ResNet*), employs stochastic shortcut connections from the input to the hidden layers for essentially improving the implicit regularization of very deep models. Experimental results on benchmark datasets validate that the proposed approach improved implicit

regularization on very deep networks as compared to the conventional residual learning.

Acknowledgments. This work was funded by the National Research Fund (FNR), Luxembourg, under the project reference R-AGR-0424-05-D/Björn Ottersten.

References

1. Oyedotun, O.K., Khashman, A.: Deep learning in vision-based static hand gesture recognition. Neural Comput. Appl. **27**(3), 1–11 (2016)
2. Oyedotun, O.K., Khashman, A.: Banknote recognition: investigating processing and cognition framework using competitive neural network. Cogn. Neurodyn. **11**(1), 67–79 (2017)
3. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Netw. **4**(2), 251–257 (1991)
4. Funahashi, K.I.: On the approximate realization of continuous mappings by neural networks. Neural Netw. **2**(3), 183–192 (1989)
5. Delalleau, O., Bengio, Y.: Shallow vs. deep sum-product networks. In: Advances in Neural Information Processing Systems, pp. 666–674 (2011)
6. Mhaskar, H., Liao, Q., Poggio, T.: Learning functions: When is deep better than shallow. arXiv preprint (2016). arXiv:1603.00988
7. Bianchini, M., Scarselli, F.: On the complexity of neural network classifiers: a comparison between shallow and deep architectures. IEEE Trans. Neural Netw. Learn. Syst. **25**(8), 1553–1565 (2014)
8. Wan, L., Zeiler, M., Zhang, S., Cun, Y.L., Fergus, R.: Regularization of neural networks using dropconnect. In: Proceedings of the 30th International Conference on Machine Learning (ICML-2013), pp. 1058–1066 (2013)
9. Graham, B.: Fractional max-pooling. arXiv preprint (2014). arXiv:1412.6071
10. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (ELUs), arXiv preprint, arXiv:1511.07289 (2015)
11. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556 (2014)
12. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
13. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. In: Advances in Neural Information Processing Systems, pp. 2377–2385 (2015)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
15. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
16. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. AISTATS **9**, 249–256 (2010)
17. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift, arXiv preprint, arXiv:1502.03167 (2015)

18. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision, pp. 630–645 (2016)
19. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
20. Schlkopf, B., Simard, P., Smola, A., Vapnik, V.: Prior knowledge in support vector Kernels. In: Proceedings of the 10th International Conference on Neural Information Processing Systems, pp. 640–646 (1997)
21. Simard, P.Y., LeCun, Y.A., Denker, J.S., Victorri, B.: Transformation invariance in pattern recognition – tangent distance and tangent propagation. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 7700, pp. 235–269. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35289-8_17
22. Wu, M., Schlkopf, B., Bakir, G.: Building sparse large margin classifiers. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 996–1003 (2005)
23. Trottier, L., Chaib-draa, B., Giguère, P.: Incrementally built dictionary learning for sparse representation. In: Arik, S., Huang, T., Lai, W.K., Liu, Q. (eds.) ICONIP 2015. LNCS, vol. 9489, pp. 117–126. Springer, Cham (2015). doi:10.1007/978-3-319-26532-2_14
24. Simard, P., LeCun, Y., Denker, J.S.: Efficient pattern recognition using a new transformation distance. In: Advances in Neural Information Processing Systems, pp. 50–58 (1993)
25. Keysers, D., Dahmen, J., Theiner, T., Ney, H.: Experiments with an extended tangent distance. In: 15th International Conference on Pattern Recognition, Proceedings, vol. 2, pp. 38–42 (2000)
26. Yang, J., Yu, K., Huang, T.: Supervised translation-invariant sparse coding. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3517–3524 (2010)
27. Yang, Z., Moczulski, M., Denil, M., de Freitas, N., Smola, A., Song, L., Wang, Z.: Deep fried convnets. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1476–1483 (2015)
28. Chan, T.H., Jia, K., Gao, S., Lu, J., Zeng, Z., Ma, Y.: Pcanet: a simple deep learning baseline for image classification? IEEE Trans. Image Process. **24**(12), 5017–5032 (2015)
29. Lin, M., Chen, Q., Yan, S.: Network in network. In: International Conference on Learning Representations, abs/1312.4400 (2014)
30. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: Artificial Intelligence and Statistics, pp. 562–570 (2015)
31. Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Le, Q.V., Ng, A.Y.: On optimization methods for deep learning. In: Proceedings of the 28th International Conference on Machine Learning (ICML-2011), pp. 265–272 (2011)
32. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations, arXiv preprint, arXiv:1412.6572 (2015)
33. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European Conference on Computer Vision, pp. 646–661 (2016)