Limited Generalization Capabilities of Autoencoders with Logistic Regression on Training Sets of Small Sizes

Alexey Potapov, Vita Batishcheva, Maxim Peterson

Limited Generalization Capabilities of Autoencoders with Logistic Regression on Training Sets of Small Sizes

Alexey Potapov[1,2], Vita Batishcheva[2], Maxim Peterson[1]

[1] St. Petersburg National Research University of Information Technologies, Mechanics and Optics, Kronverkskiy pr. 49, 197101 St. Petersburg, Russia
[2] St. Petersburg State University, Universitetskaya nab. 7-9, 199034, St.Petersburg, Russia
{pas.aicv, elokkuu, maxim.peterson}@gmail.com

Abstract. Deep learning is promising approach to extract useful nonlinear representations of data. However, it is usually applied with large training sets, which are not always available in practical tasks. In this paper, we consider stacked autoencoders with logistic regression as the classification layer and study their usefulness for the task of image categorization depending on the size of training sets. Hand-crafted image descriptors are proposed and used for training autoencoders in addition to pixel-level features. New multi-column architecture for autoencoders is also proposed. Conducted experiments showed that useful nonlinear features can be learnt by (stacked) autoencoders only using large training sets, but they can yield positive results due to redundancy reduction also on small training sets. Practically useful results (9.1% error rate for 6 classes) were achieved only using hand-crafted features on the training set containing 4800 images.

Keywords: autoencoders, logistic regression, overlearning, generalization, image categorization

1 Introduction

Classes of patterns in pattern recognition are rarely linearly separable, and nonlinear methods should be used. Classical nonlinear methods of pattern recognition can be treated as linear methods operating in some extended feature space. However, these methods use fixed sets of nonlinear transformations to map initial features into new space, and these transformations can be inappropriate to separate classes in a specific task. Thus, learning appropriate nonlinear features (or data representations in general) is the central problem in pattern recognition and particularly in its application to computer vision (e.g. [1]). One modern approach to learn such features is deep learning.

Deep learning exploits the fact that adding extra hidden layers in a multi-layer classifier helps to learn more complex features and to improve their representational power [2, 3]. The key problem here is to train such classifiers. Earlier, training of shallow feed-forward networks with the back-propagation algorithm yielded better

results than training deep networks with it, since bottom layers are difficult to train because of exponentially quick gradient vanishing, so these layers usually correspond to random (not useful or even harmful) nonlinear feature mappings. Successful training of such networks was achieved [4, 5, 6] with the help of consequent unsupervised pre-training of each hidden layer, and use of supervised training afterwards.

However, successful supervised training of multi-layer perceptrons (resulting in a very low 0.35% error rate on the MNIST benchmark) was also achieved recently [7]. The way of achieving this is remarkable. It consists in intensive training of perceptrons using relevantly (with affine and elastic image deformations) transformed patterns. Even overlearning is avoided in spite of extremely large number of neurons (free parameters), because "the continual deformations of the training set generate a virtually infinite supply of training examples" [7]. One can also see [8] that many good results for MNIST are obtained using similar deformations of training patterns and convolutional nets, which additionally exploit shift invariance. It is obvious, that if one substitutes non-image patterns for training such classifiers, their results will be rather poor. Thus, this is not an appropriate way to learn (problem-specific) invariant features, which are not known a priori.

Necessity of extremely large (with account for deformations of training patterns) training sets indicates that deep networks don't really generalize, but only approximate invariants. Moreover, classifiers with many free parameters appear to be better than with fewer parameters. It can be compared with approximation of exponent with polynomials (or some other basis functions). Very many points should be used to get precise approximation with very complex model, but only few points are enough for exact generalization (if one has means to represent and find it). At the same time, humans have capabilities to learn complex invariant features from few examples [9]. Thus, interesting question consists in learning capabilities of deep networks on small training sets, that is, whether they are able to learn useful representations from such sets.

Here we analyze this question on example of (stacked) autoencoders with logistic regression as the classification layer applied to the task of image categorization. Pixel-level and hand-crafted features are used for experiments.

2 Autoencoders

We selected stacked autoencoders as the unsupervised layers with logistic regression as the classification layer. Single autoencoder has input, hidden, and output (reconstruction) layers. The input layer accepts a vector $\mathbf{x} \in [0,1]^N$ of dimension N, which is transformed to activities of neurons of the hidden layer $\mathbf{y} = s(\mathbf{W}\mathbf{x} + \mathbf{b})$, $\mathbf{y} \in [0,1]^d$, corresponding to new features (hidden representation), where $\mathbf{W}$ is a $d \times N$ matrix of connection weights, $\mathbf{b}$ is a bias vector, and s is the activation (sigmoid) function. Activities of neurons of the last layer are calculated similarly as $\mathbf{z} = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$, $\mathbf{z} \in [0,1]^N$. Autoencoders differ from other feed-forward networks in that they are trained to minimize difference between $\mathbf{x}$ and $\mathbf{z}$ (for patterns from a

training set), that is $\mathbf{W}'$ is the matrix of the reverse mapping. $\mathbf{W}'$ is frequently taken as $\mathbf{W}^T$. One can calculate gradient of the reconstruction error relative to connection weights and bias vectors and to train the autoencoder using stochastic gradient descent.

In the case of stacked autoencoders, each autoencoder on the next level takes outputs of the hidden (not reconstruction) layer of the previous autoencoder as its input performing further nonlinear transformation of constructed latent representation of the previous level. Each next autoencoder is trained after training its preceding autoencoder. Outputs from the hidden layer of the last autoencoder are passed to the supervised feed-forward network.

Multinomial logistic regression computes probabilities of a pattern to belong to different classes based on softmax applied to linear combinations of features.

$$p(y = c \mid \mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x} + b_c)}{\sum_{c'=1}^{C} \exp(\mathbf{w}_{c'}^T \mathbf{x} + b_{c'})}, \quad (\mathbf{1})$$

where $\mathbf{x}$ is the input vector, c is the class index, C is the total number of classes, $\mathbf{W}$ is the weight matrix composed by C vectors $\mathbf{w}_c$, b_c are biases. Parameters of the classifier are learnt using stochastic gradient descent minimizing negative log-likelihood.

One common modification is denoising autoencoders [10], in which reconstruction during training is calculated using patterns with introduced noise, but reconstruction errors are calculated relative to initial patterns without this artificial noise. Thus, a denoising autoencoder learns to reconstruct a clean input from a corrupted one. This is a way to deform input patterns efficiently increasing the training set size, but in the least problem-specific fashion.

We introduce some additional modifications into denoising stacked autoencoders. One modification consists in that only the last (regression) layer is trained in the supervised fashion. This helps to check usefulness of unsupervised features.

The second modification (which is to be compared with the basic version) consists in constructing multi-column autoencoders. Multi-column deep neural networks were already used (e.g. [11]), but each column in such networks is usually trained as a separate deep network, and predictions of all columns are then averaged. In our modification, the number of columns corresponds to the number of classes. Each column is trained for its own class to produce some non-linear features. These features are then gathered and passed as the input to the logistic regression layer.

3 Image Features

We took the task of image categorization for investigating behavior of autoencoders. In this task, images can vary considerably, and special hand-crafted features are usually used, that doesn't eliminate necessity to learn non-linear features, especially when different hand-crafted features are combined.

We used three types of image features for image description: color histograms, edge direction histogram (EDH) [12] and Haralick's textural features based on gray-level co-occurrence matrices (GLCM) [13].

Color features are built as concatenation of three 1D histograms calculated for LAB components. Histogram for each component contains N_c=32 bins, so the total size of the color vector is $3N_c$=96.

In order to build an edge direction histogram we apply Deriche filtration [14] at the first step. Then we get an assessment of orientation distribution by computing the direction of gradient vector at local maximums of gradient magnitude. Initial range of edge orientation angles [0°, 360°) is quantized to N_q bins (we use 72 bins). The last bin in histogram represents the amount of image pixels that don't belong to edges. Finally, because of possible difference in image sizes, all bins are normalized by corresponding values [12]

$$\begin{aligned} &\mathbf{EDH}(i) = \mathbf{EDH}(i) / N_{edge}, \quad i \in [0, N_q - 1]; \\ &\mathbf{EDH}(N_q) = \mathbf{EDH}(N_q) / N_p, \end{aligned} \qquad (2)$$

where $\mathbf{EDH}(i)$ – value in bin i of the histogram, N_{edge} is total amount of edge points in the image, N_p is total number of pixels.

Unfortunately EDH gives no information about location of contours on the image plane, so we tried to compensate this by splitting image into K parts (K=3) and computing the histogram for each of them. Thus, the EDH descriptor size is $K(N_q+1)$=219.

The last part of our image descriptor is computed from normalized GLCM. Co-occurrence matrices represent texture properties, but they are inconvenient for matching two textures, because of big number of elements, thus we used five features of normalized GLCM $\mathbf{C_d}$, proposed in [13]:

- angular second moment: $\sum_i \sum_j \mathbf{C_d}^2(i, j)$;
- entropy: $-\sum_i \sum_j \mathbf{C_d}(i, j) \log \mathbf{C_d}(i, j)$;
- contrast: $\sum_i \sum_j (i - j)^2 \mathbf{C_d}(i, j)$;
- inverse difference moment: $\sum_i \sum_j \frac{\mathbf{C_d}(i, j)}{1 + |i - j|}$;
- correlation: $\frac{\sum_i \sum_j (i - \mu_i)(j - \mu_j) \mathbf{C_d}(i, j)}{\sigma_i \sigma_j}$,

where μ_i, μ_j, σ_i, σ_j are the means and standard deviations of rows and columns of $\mathbf{C_d}$, $\mathbf{d}$ is a displacements vector, for which co-occurrence matrix is computed (we used D options of $\mathbf{d}$ for each image).

In order to consider color information, initial RGB images were transformed to HSV representation, so GLCM was computed for hue, saturation and value

components, which in turn were preliminarily normalized to fit [0, 255] range. The size of modified GLCM descriptor (five GLCM based features for each of H, S and V component for 5 displacement vectors and 3 directions) appeared to be 5×3×5×3=225.

Thus, the image descriptor is constructed by concatenation of LAB histogram and modified EDH and GLCM parts. The total size of the image descriptor is 96+219+225=540. Reduced descriptor containing only two sets of features (96+219=315 features) was also considered for comparison.

4 Experiments

We conducted experiments with the neural networks composed of autoencoders and logistic regression layer described above. For experiments, two training sets were constructed. Variability of images in these training sets was different. The first set (DB1) contained heads of cats of 5 breeds – European Shorthair, British Shorthair, Exotic Shorthair, Burmese, Abyssinian; examples of two of them are presented in fig. 1. The second set (DB2) contained images of 6 categories – city, flowers (garden), indoor, mountains, forest, sea (beach); see fig. 2. DB1 was composed of 330 images, from which small training set was extracted with 50 (10 per class) images. DB2 was composed of 6000 images, and experiments with two training sets containing 60 (10 per class) and 4800 (800 per class) images were conducted.

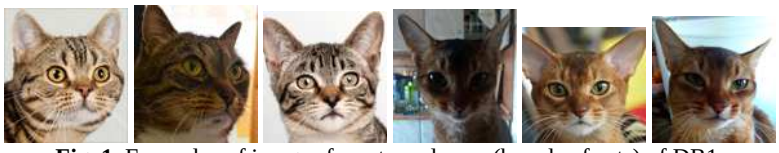

Fig. 1. Examples of images from two classes (breeds of cats) of DB1.

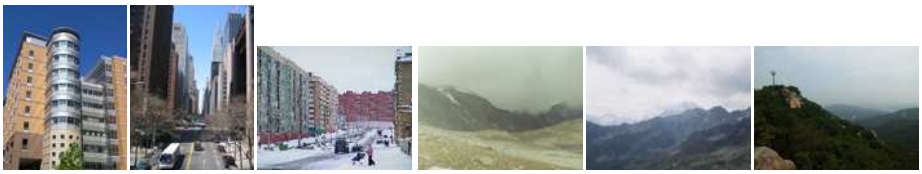

Fig. 2. Examples of images from two classes (categories of scenes) of DB2.

Logistic regression with and without autoencoders was tested separately using RGB values of pixels (on images rescaled to sizes of 64x64, 28x28, and 20x20) and hand-crafted descriptors as initial features on these two databases (three training sets). Results are presented in table 1.

It can be seen that hand-crafted features are not always very informative. Reduced image descriptors (315 features) gave worse results and enhanced descriptors (540 features) gave similar results in comparison with pixel-level features on DB1. However, for more complex image classes (DB2) these features yield better results, and results are greatly improved (in contrast to pixel-level features) with increase of

the size of training sample. Of course, pixel-level features contain more information, but it is clearly seen that classes under consideration are not linearly separable in this feature space (but better linearly separable in the space of hand-crafted features), so one might expect that autoencoders would help to improve recognition rate for pixel-level features more than for hand-crafted features. However, conducted experiments showed that this assumption is not correct. The results are presented in table 2.

Table 1. Recognition error rates of the logistic regression classifier

	DB1 (10 per class)	**DB2 (10 per class)**	**DB2 (800 per class)**
Hand-crafted features (540)	0.356	0.505	0.118
Hand-crafted features (315)	0.436	0.535	0.188
64x64x3	0.361	0.620	0.583
28x28x3	0.382	0.598	0.580
20x20x3	0.401	0.615	0.571

Table 2. Best recognition error rates of the autoencoders with logistic regression classifier

	DB1 (10 per class)	**DB2 (10 per class)**	**DB2 (800 per class)**
Hand-crafted features (540)	0.323	0.41	0.091
Hand-crafted features (315)	0.359	0.445	0.122
20x20x3	0.418	0.601	0.353

It appeared to be very difficult to train autoencoders in the case of pixel-level features and small training sets. Too small or too large number of training epochs leaded to recognition of all patters as belonging to one same class. The best achieved error rates for DB1 and DB2 (using 10 training images per class) are 0.418 and 0.601 correspondingly with the use of multi-column autoencoder with one layer containing 1000 neurons (with 20x20x3=1200 inputs). That is, autoencoders failed to learn useful features from small amount of raw data, and they only didn't ruined initial features in fortunate cases.

Training multi-column one layer autoencoders on 800 per class DB2 images resulted in 0.353 error rate (one-column autoencoders gave 0.383 error rate), which is much better than the error rate of the logistic regression on pixel-level features, but is much worse than the error rate of the logistic regression on hand-crafted features. That is, some useful nonlinear features were learnt, but they were worse than both (reduced and enhanced) hand-crafted features. It should be noted that we also tested autoencoders on the MNIST database with 50000 training images, and they considerable reduced error rate relative to the logistic regression (from 8% to 2% only

for two layers and can be reduced further) even without using convolution autoencoders or additional image transformations during training. This implies that more complex nonlinear features indeed can be learnt by autoencoders, but this result can be achieved in the case of very low variability of patterns and using large training sets.

Autoencoders trained using patterns represented by hand-crafted features appeared to be more useful (see table 2). The achieved error rate (~9%) for the scene classification task with 6 classes is rather high (e.g., in [12], 6% error rate is achieved while distinguishing only two classes – city and landscape images, and 9% error rate is achieved for forest-mountain-sunset classification), so this result is interesting by itself.

However, it is also interesting to compare results in tables 1 and 2. In the case of hand-crafted features, larger improvement is achieved in the case of small training sets. In the case of pixel-level features, results are opposite implying that nature of this improvement may be different. Possibly, dimension reduction of linear features has stronger influence in the first case (and this is achievable on small training sets), while only construction of nonlinear features helps to increase recognition rate in the second case (and this requires large training sets).

Consider also results in fig. 3-6. In all cases, error rates vary insignificantly for a wide range of hidden layer sizes. Best results are achieved on intermediate sizes, while the error rate rapidly increases for small hidden layers (in which too much information is lost), and gradually increases with the number of hidden neurons exceeding the number of features meaning that reduction of features redundancy is more useful than construction of complex nonlinear approximation on small training sets.

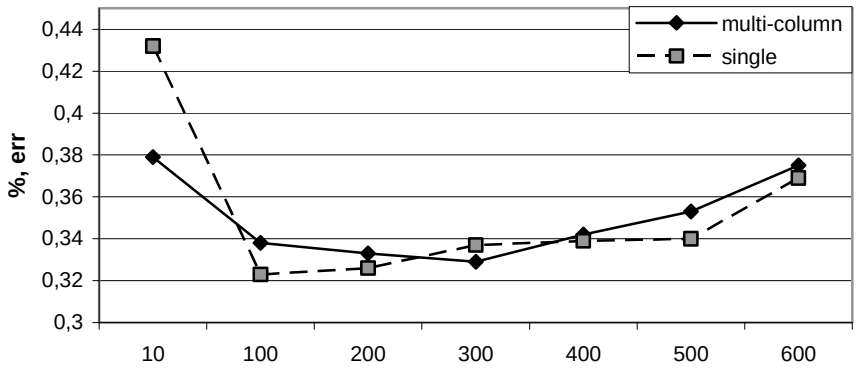

Fig. 3. Error rates of one-layer autoencoders of different hidden layer sizes with logistic regression layer on DB1 (540 features)

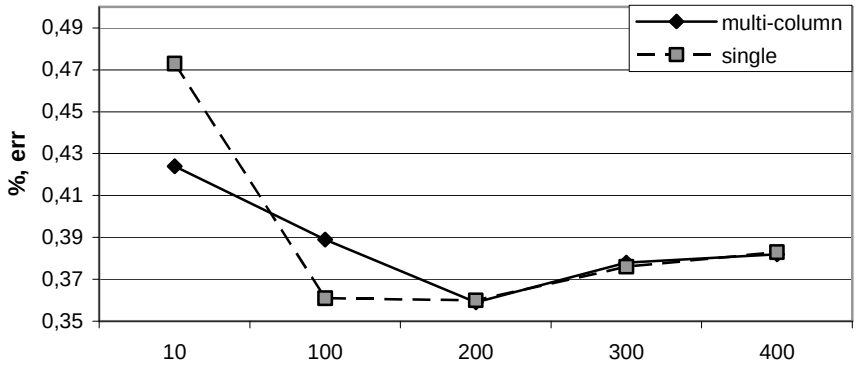

Fig. 4. Error rates of one-layer autoencoders of different hidden layer sizes with logistic regression layer on DB1 (315 features)

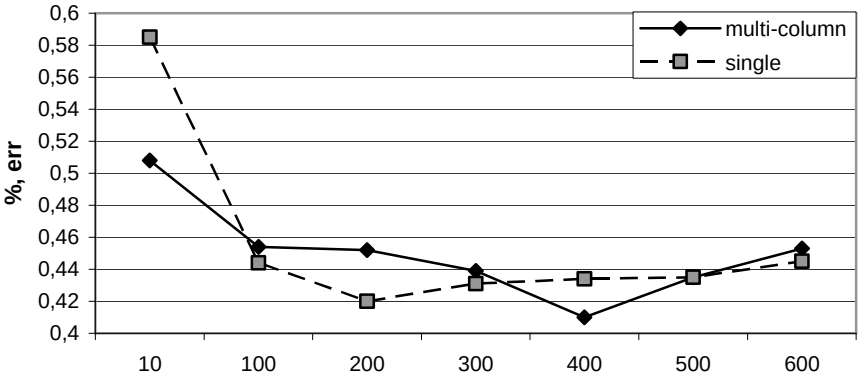

Fig. 5. Error rates of one-layer autoencoders of different hidden layer sizes with logistic regression layer on DB2 (540 features)

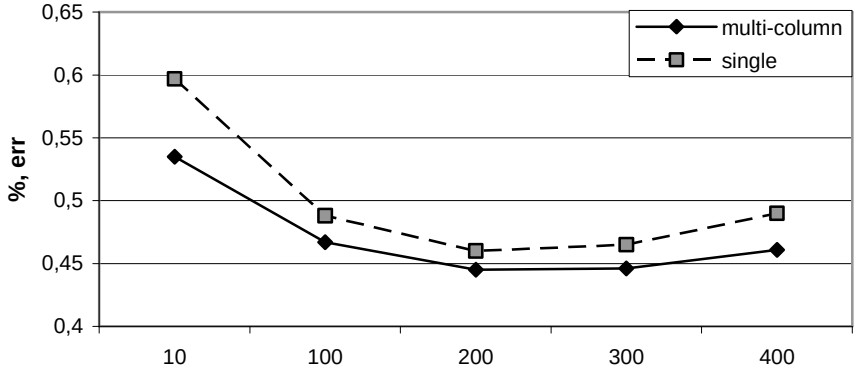

Fig. 6. Error rates of one-layer autoencoders of different hidden layer sizes with logistic regression layer on DB2 (315 features)

It can also be seen that multi-column autoencoders outperform one-column autoencoders on more difficult image classes (DB2) and have similar performance on simpler classes. Although multi-column autoencoders contain much more hidden neurons, they don't suffer from overlearning more than one-column autoencoders (with the same number of neurons per column).

Finally, stacked autoencoders were tested. However, they yielded very small decrease of the error rate. For example, multi-column autoencoders with two layers (400, 300) helped to decrease the error rate from 0.329 to 0.324, and for one-column autoencoders the error rate decreased from 0.323 to 0.319 on DB1. One might expect that multi-level autoencoders will be more useful in the case of pixel-level features and larger training sets, but adding the second layer was unsuccessful resulting in increase of the error rate from 0.353 to 0.373. Possibly, larger training sets or training image deformations are necessary for multi-level autoencoders to learn useful complex features.

5 Conclusion

Stacked autoencoders with logistic regression layer for classifying images were considered. Original modification of multi-column autoencoders was proposed. Their performance was evaluated on pixel-level features and special hand-crafted image descriptors. These descriptors were composed of color, gradient and texture features. Tests were conducted on two data sets – cat breeds and image categories (city, indoor, forest, mountains, beach, flowers).

Training on hand-crafted features appeared to be much more successful than on pixel-level features (without introducing deformations of training images). Error rates of 9.1% and 35.3% were achieved on the training set containing 4800 images for the task of recognizing 6 image categories using hand-crafted and pixel-level features correspondingly. Thus, only hand-crafted features yielded practically usable results. Training autoencoders on small (10 patterns per class) samples using pixel-level features failed. In contrast, it stably improved (in comparison with single logistic regression) recognition rate in the case of hand-crafted features.

From the gathered experimental data one can conclude that useful nonlinear features can be learnt by (stacked) autoencoders only using large training sets, since these networks don't recover true underlying regularities in data, but approximate them by complex (multi-parametric) models. Most positive effect from usage of autoencoders on small training sets is possibly connected with redundancy reduction, and not with construction of nonlinear features. Thus, some principal ways of efficiently extracting nonlinear representations of patterns from small training sets or constructing some general representations (similar to the proposed hand-crafted image descriptors which are applicable for recognizing different image categories) are required.

Acknowledgements

This work was supported by the Russian Federation President's grant Council (MD-1072.2013.9) and the Ministry of Education and Science of the Russian Federation.

References

1. He, Y., Kavukcuoglu, K., Wang, Y., Szlam, A., Qi, Y.: Unsupervised Feature Learning by Deep Sparse Coding. arXiv:1312.5783 [cs.LG] (2013)
2. Le Roux, N., Bengio, Y.: Representational Power of Restricted Boltzmann Machines and Deep Belief Networks. Neural Computation. 20(6),1631–1649 (2008)
3. Gregor, K., Mnih, A., Wierstra, D.: Deep AutoRegressive Networks. arXiv:1310.8499 [cs.LG] (2013)
4. Hinton, G. E., Osindero, S., Teh, Y.: A Fast Learning Algorithm for Deep Belief Nets. Neural Computation. 18, 1527–1554 (2006)
5. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy Layer-Wise Training of Deep Networks. Advances in Neural Information Processing Systems. 19 (NIPS 2006), 153–160 (2007)
6. Ranzato, M.A., Poultney, Ch., Chopra S., LeCun, Y.: Efficient Learning of Sparse Representations with an Energy-Based Model. Advances in Neural Information Processing Systems. 19 (NIPS 2006), (2007)
7. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition. arXiv:1003.0358 [cs.NE] (2010)
8. LeCun, Y., Cortes, C.: MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/
9. Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D.: How to Grow a Mind: Statistics, Structure, and Abstraction. Science. 331 (6022), 1279–1285 (2011)
10. Vincent, P., Larochelle, H., Bengio, Y., Manzagol P.-A.: Extracting and composing Robust Features with Denoising Autoencoders. Proc. 25th International Conference on Machine Learning. 1096–1103 (2008)
11. Cireşan, D., Meier, U., Masci, J., Schmidhuber, J.: Multi-Column Deep Neural Network for Traffic Sign Classification. Neural Networks. 32, 333–338 (2012)
12. Vailaya, A., Jain, A., Zhang, H. J.: On Image Classification: City Images vs. Landscapes. Pattern Recognition. 31 (12), 1921–1935 (1998)
13. Haralick, R.M., Shanmugam, K., Dinstein I. Textural Features for Image Classification. IEEE Transactions on Systems, Man, and Cybernetics. SMC-3 (6), 610–621 (1973)
14. Deriche, R.: Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector. International Journal of Computer Vision. 1(2), 167–187 (1987)