On a Streaming Approach for Training Denoising Auto-encoders

Piotr Duda[1(✉)] and Lipo Wang[2]

[1] Czestochowa University of Technology, Czestochowa, Poland
piotr.duda@pcz.pl
[2] Nanyang Technological University, Singapore, Singapore

Abstract. Learning deep neural networks requires huge hardware resources and takes a long time. This is due to the need to process huge data sets multiple times. One type of neural networks that are particularly useful in practice are denoising autoencoders. It is, therefore, necessary to create new algorithms that reduce the training time for this type of networks. In this work, we propose a method that, in contrast to the classical approach, where each data element is repeatedly processed by the network, is focused on processing only the most difficult to analyze elements. In the learning process, subsequent data may lose their significance and others may become important. Therefore, an additional algorithm has been used to detect such changes. The method draws inspiration from boosting algorithms and drift detectors.

Keywords: Denoising autoencoders · Artificial neural networks · Drift detectors.

1 Introduction

In recent years, we can easily observe the rapid development of deep learning techniques. This mainly applies to various types of neural networks. Among the most popular techniques are convolutional neural networks [26], which are often used to images analysis; recursive neural networks, commonly used, among others, to natural language processing [40]; or restricted Boltzman machines used for density estimation or detection of changes in incoming data [21]. Less popular techniques, like spiking neural networks, are also developing significantly [28].

Much attention should be paid to autoencoders, which are a special type of neural networks. Their task is to recreate at the output the information given at the input. Depending on the application, several types of autoencoders are distinguished, such as sparse, contrastive, variational, and denoising autoencoders. In this work, we will focus on denoising encoders. This means that noisy data elements are fed to the autoencoder input, and at the output, we expect to

This work was supported by the Polish National Science Centre under grant no. 2017/27/B/ST6/02852.

L. Rutkowski et al. (Eds.): ICAISC 2020, LNAI 12416, pp. 315–324, 2020.
https://doi.org/10.1007/978-3-030-61534-5_28

receive noise-free information. This approach is particularly useful in working with missing and uncertain data.

A spectacular performance deep neural networks in solving speech and image processing problems, have made both researchers and companies pay special attention to them. However, training such models require the collection of huge amounts of data, and the learning process, epoch after epoch, makes that data are processed many times. As a result, training deep models requires both a lot of time and computers with sufficient computing power [1,18,24,27].

On the other hand, the problem of analysis of the huge amounts of data, so-called Big Data analysis (BDA), has become a separate research field [25]. Among the various approaches, one of the most promising is data stream mining (DSM). This approach requires that the data are not stored in the system but are processed as soon as arrive from the stream, and next, forgotten as soon as possible. Another important condition is to ensure that the algorithm can respond immediately, regardless of the rate at which data elements arrive. Therefore, it is not recommended to use long-term learning processes, such as epoch learning of neural networks. The last but not least feature of DSM is the ability of algorithms to detect and react to changes in the environment. This phenomenon is called concept-drift. Data stream mining can be applied in many fields, i.e. iterative learning control [35,36].

DSM algorithms can be divided by the way they process data. The on-line algorithms process single data elements immediately after arrival. This group includes, among others, the classification [20,32,33,39], regression [10,11,19,31] and density estimation algorithms [13]. This approach is also used in other, more complex systems [43]. Another approach is to process data chunks. This method is often combined with ensembles of classifiers [9,12]. There are also solutions based on storing in memory only a constant number of recently arrived data elements. This approach is called sliding windows [3].

We can also divide these algorithms by the way they react to concept drift. We distinguish a passive and active approach here. The passive approach is based on the self-adaptation of the model to changes in the environment through its continuous learning. In an active approach, the algorithm indicates the moment when the concept changed and then tries to create a new model that will be better adapted to the new environment. The change detection mechanism is called the drift detector (DD). The other approach to detecting changes is proposed in [34,37].

In the classic approach of neural network training, data from the training set are divided into batches and given into the network's input. After forward propagation, the network error is calculated and propagated backward, to change the values of the weights. Such a process is sometimes unfavorable, as the processing of data elements that do not tune the network takes the same amount of time as the processing of important data (i.e. those that have the greatest impact on the learning process). Work [8] shows how we can select subsets of training data so that the network can learn from the most important data. In contrast

to this work, which focuses only on the classification task, we concentrate now on applying this method to learning the denoising autoencoders.

The rest of the paper is divided into the following sections. In Sect. 2 recent works on autoencoders and drift detectors are presented. Section 3 describes the proposed algorithm. The simulation results are presented in Sect. 4. Finally, the conclusions are given in Sect. 5.

2 Related Works

One of the most interesting structures applicable to unsupervised learning are autoencoders [2], which learn how to reconstruct original data. In [7] the authors presented a denoising autoencoder that extracts features from data with noise. In [16] the authors proposed a convolutional denoising autoencoder to process medical images. In [42] an application of denoising autoencoders to the recommender system is presented. Solving the problem of single-channel audio signal separation is considered [17]. In [41] the authors apply autoencoders to improve electricity price forecasting.

One of the most important tools developed within SDM is the drift detector. Several approaches are proposed in the literature. The Drift Detection Method (DDM) [14] monitors the correctness of classification by the current model. Treating observations as a result of Bernoulli trials, the authors propose a statistical test to inform about warning or alarm state. In paper [15] the authors propose the Adaptive Random Forests algorithm, which combines classical random forest procedure with Hoeffding's decision trees. To react to changes in data stream, a procedure based on the ADWIN algorithm [3] and the Page-Hinkley test [30] can be applied. In [5], the authors proposed the WSTD algorithm, which applied the Wilcoxon rank-sum statistical test to improve false positive detection. In [6], the authors proposed computing multiple models explanations over time and observing the magnitudes of their changes.

It should also be noted that several authors tried to merge the fields of deep learning and data stream mining. In [4] the authors combined the evolving deep neural network with the Least Squares Support Vector Machine. Deep neural networks were also successfully applied in semi-supervised learning tasks in the context of streaming data. It was demonstrated how such structures can be used for online learning from data streams. In [29] the Deep Hybrid Boltzmann Machines and Denoising Autoencoders were proposed. In [38] the idea was to train the Deep Belief Network in an unsupervised manner based on the unlabeled data from the stream. Then, few available labeled elements were used to occasionally fine-tune the model to the current data concept. In [21] and [22] the authors proposed to apply the RBM as a concept drift detector. It was demonstrated that the properly learned RBM can be used to monitor possible changes in the underlying data distribution. This method was further analyzed from the resource-awareness perspective in [23].

3 The BBTADD Algorithm for Denoising Autoencoders

The approach presented in this paper is mainly based on the BBTA algorithm proposed in [8].

Let T be a training set consisting of N elements, where each of them is d-dimensional feature vector X_i for $i = 1, \ldots, N$, i.e.

$$T = \{X_i | i = 1, \ldots, N, X_i \in \mathbf{A}\}, \tag{1}$$

where $\mathbf{A}$ is a d-dimensional feature space. Moreover, a new factor has been added to each element describing a probability of drawing (*pod*) from the stream.

$$T^S = \{(X_i, v_i) | X_i \in T, v_i \in (0, 1)\}. \tag{2}$$

Through subsequent, independent draws of elements from the set T^S, we can create a data stream S_t as follows

$$S_t = (Y_1, \ldots, Y_t | Y_i = (X_{j_i}, c_{j_i}), 1 \leq i \leq t, 1 \leq j_i \leq N), \tag{3}$$

where t is an index of the last element coming from the stream.

The denoising autoencoder is a function mapping from set $\mathbf{A}$ to itself, $f : \mathbf{A} \rightarrow \mathbf{A}$. Without loss of generality, we can assume that the autoencoder consists of l layers. Then the function f can be expressed in the following way

$$f(X) = \phi_l \circ \phi_{l-1} \circ \cdots \circ \phi_1(X), \tag{4}$$

where X is the input vector. A single layer $\phi_j : \mathbf{Z_{j-1}} \rightarrow \mathbf{Z_j}$, where $j = 1, \ldots, l$, $\mathbf{Z_j}$ is an N_j dimensional space of $(j-1)$-th layer output values, $\mathbf{Z_0} = \mathbf{Z_l} = \mathbf{A}$, can be defined as follows

$$\phi_j(z) = [\rho_j^1(\textstyle\sum_{i=1}^{N_{j-1}} w_{i,1} z_i + b_1), \ldots, \rho_j^{N_j}(\sum_{i=1}^{N_{j-1}} w_{i,N_j} z_i + b_{N_j})] \tag{5}$$

where $z = [z_1, \ldots, z_{N_j}] \in \mathbf{Z_{j-1}}$, $w_{i,m}$ is a weight between the i-th neuron of the $(j-1)$-th layer and the m-th neuron of the j-th layer, ρ_j^m is an activation function for the m-th neuron on the j-th layer and b_m is the bias for the m-th neuron.

The classic approach to learning multidimensional neural networks involves updating weights according to the following formula

$$w_{i,m} := w_{i,m} - \eta \frac{\partial L}{\partial w_{i,m}}, \tag{6}$$

where $\eta > 0$ is the learning rate and L is a loss function.

In [8], three methods of determining v_i value were proposed. In our work we will use the NLB approach. First, the temporary values v_i are determined according to the following formula

$$v_i' = tanh(L(X_i))/M_i, \tag{7}$$

where M_i indicates the number of times the i-th data element was drawn in the past. In consequence, big values of loss function give high index of drawing this element, close to 1, and the small ones close to 0.

Next, as values v_i' are not probability mass function (since they do not have to sum up to 1), they are normalized after processing the whole mini-batch in the following way

$$v_i = \begin{cases} v_i'/Z, & \text{for } x_i \in B \\ v_i/Z, & \text{for } x_i \in T \backslash B \end{cases} \tag{8}$$

where Z is a normalization factor, given as

$$Z = \sum_{\{v_i' | X_i \in T\}} v_i'. \tag{9}$$

After processing one mini-batch of data, another one is generated and the procedure is repeated until the stopping condition is fulfilled.

Changing the weights of the network only according to elements that are not well classified can lead to network untune, which means to misclassifying elements that previously processed correctly. Another threat is the fact that constantly analyzing the same data can lead to network overfitting. Consequently, an important element of the algorithm is the drift detector, which allows indicating a moment since when elements draw according to the current *pod* values do not affect the learning process well.

As in [8], we used the CuSum algorithm as a drift detector, given by the following formula

$$Cus_0 = 0, \tag{10}$$

$$Cus_i = max(0, Cus_{i-1} + L(B_{i-1}) - L(B_i) - \alpha), \tag{11}$$

for $i = 1, 2, \ldots$, where $L(B_i)$ is a value of the loss function in the i-th mini-bath and α is a fixed parameter. The drift is detected when Cus_i exceeds the value of the threshold λ_C.

The summary of the BBTADD algorithm is presented in Algorithm 1.

4 Experimental Results

In this chapter, the application of the algorithm described in Sect. 3 is tested for autoencoders training. To this end, the MNIST data set is used. The training set consists of 60 000 elements representing hand-drawn numbers. The test set has 10 000 elements. The performance of the proposed method will be compared with the performance of the autoencoder trained by the classical approach.

To perform the simulations, a convolution neural network consists of 9 layers was used. The first 5 layers are convolution and max-pooling layers, alternately placed. The convolution layers consist of 16, 8, and 8 filters, respectively. Next, the two alternately arranged deconvolution and upsampling layers are placed.

Input: S - data stream, M - batch size, α, λ_C
1 CuSum = 0 ;
2 Collect a new batch B from the stream S;
3 **for** *every data element in* B **do**
4 Increase counter of drawn of the current element;
5 Train the network on current element;
6 Compute loss function for a current element;
7 Update v_i according to (7)
8 **for** *every data element in* T **do**
9 Update *pod*s according to (8)
10 Compute loss function on a validation set;
11 Update CuSum according to (10);
12 **if** $CuSum > \lambda_C$ **then**
13 Reinitialize *pod's* values;
14 *Return to line 1*;
15 **else**
16 *Return to line 2*;

Algorithm 1: The BBATDD algorithm.

The deconvolution layers consist of 8 and 1 filters, respectively. The sizes of the filters in all convolution and deconvolution layers was set to 3 on 3, and in pooling and upsampling to 2 on 2. The relu activation function was used. The diagram of the network is presented in Fig. 1.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d (Conv2D)	(None, 28, 28, 16)	160
max_pooling2d (MaxPooling2D)	(None, 14, 14, 16)	0
conv2d_1 (Conv2D)	(None, 14, 14, 8)	1160
max_pooling2d_1 (MaxPooling2	(None, 7, 7, 8)	0
conv2d_2 (Conv2D)	(None, 7, 7, 8)	584
up_sampling2d (UpSampling2D)	(None, 14, 14, 8)	0
conv2d_3 (Conv2D)	(None, 14, 14, 8)	584
up_sampling2d_1 (UpSampling2	(None, 28, 28, 8)	0
conv2d_transpose (Conv2DTran	(None, 28, 28, 1)	73

Fig. 1. Convolutional autoencoder

The first experiment presents a comparison of the network training in a classic way and using the BBTADD algorithm. The size of the batches used during training was set to 128 elements. The classical network has been trained by 100 epochs, which is equivalent to 46,875 batches used for learning the BBTADD algorithm. Figure 2 shows the loss function obtained for both approaches on

the test set. The loss function used during training is binary cross-entropy. It can be easily seen that the value of the loss function for the proposed algorithm decreases faster compared to the classical approach.

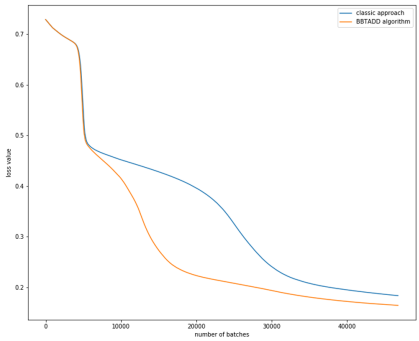

Fig. 2. The loss function computed on the test set for subsequent batches

An example of denoising images obtained by using the classic approach and the BBTADD algorithm is shown in the Figs. 3 and 4, respectively.

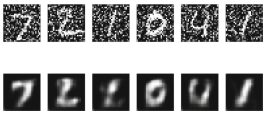

Fig. 3. The original and denoised images by the classic autoencoder

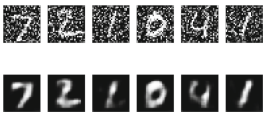

Fig. 4. The original and denoised images by the BBTADD algorithm

The next experiment was a comparison of the performance of the proposed algorithm trained on different batch sizes. For this purpose, the number of elements in a batch was set to 128, 256, 512, and 1024. The values of the loss function for these simulations are presented in Fig. 5. It shows that smaller batch sizes allow for greater accuracy. On the other hand, it is also important to compare the training time for different batch sizes. Calculations on larger batches are faster. For batches from 128 to 1024, they take 2555, 1311, 719, and 1024 s, respectively.

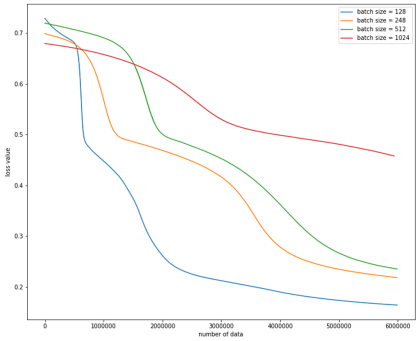

Fig. 5. The loss function values for the BBTADD algorithm trained with different sizes of batches

5 Conclusions

This paper explores the possibility of training autoencoders by selecting certain subsets of data from the training set. The carried out simulations showed the usefulness of the proposed method. The values of the loss function decreased faster than in the case of autoencoders trained classically. The effects of noise reduction on the image data were satisfactory. In the future, the proposed method will be used to train deeper models as well as it will be applied to other types of data.

References

1. Akdeniz, E., Egrioglu, E., Bas, E., Yolcu, U.: An ARMA type pi-sigma artificial neural network for nonlinear time series forecasting. J. Artif. Intell. Soft Comput. Res. **8**(2), 121–132 (2018)
2. Bengio, Y.: Learning deep architectures for AI. Found. Trends Mach. Learn. **2**(1), 1–127 (2009)
3. Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 249–260. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03915-7_22
4. Bodyanskiy, Y., Vynokurova, O., Pliss, I., Setlak, G, Mulesa, P.: Fast learning algorithm for deep evolving GMDH-SVM neural network in data stream mining tasks. In: 2016 IEEE First International Conference on Data Stream Mining Processing (DSMP), pp. 257–262, August 2016
5. deBarros, R.S.M., Hidalgo, J.I.G., de Lima Cabral, D.R.: Wilcoxon rank sum test drift detector. Neurocomputing **275**, 1954–1963 (2018)
6. Demsar, J., Bosnic, Z.: Detecting concept drift in data streams using model explanation. Expert Syst. Appl. **92**, 546–559 (2018)
7. Du, B., Xiong, W., Wu, J., Zhang, L., Zhang, L., Tao, D.: Stacked convolutional denoising auto-encoders for feature representation. IEEE Trans. Cybern. **47**(4), 1017–1027 (2016)

8. Duda, P., Jaworski, M., Cader, A., Wang, L.: On training deep neural networks using a streaming approach. J. Artif. Intell. Soft Comput. Res. **10**(1), 15–26 (2020)
9. Duda, P., Jaworski, M., Rutkowski, L.: On ensemble components selection in data streams scenario with reoccurring concept-drift. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–7. IEEE (2017)
10. Duda, P., Jaworski, M., Rutkowski, L.: Convergent time-varying regression models for data streams: tracking concept drift by the recursive Parzen-based generalized regression neural networks. Int. J. Neural Syst. **28**(02), 1750048 (2018)
11. Duda, P., Jaworski, M., Rutkowski, L.: Knowledge discovery in data streams with the orthogonal series-based generalized regression neural networks. Inf. Sci. **460**, 497–518 (2018)
12. Duda, P., Jaworski, M., Rutkowski, L.: Online GRNN-based ensembles for regression on evolving data streams. In: Huang, T., Lv, J., Sun, C., Tuzikov, A.V. (eds.) ISNN 2018. LNCS, vol. 10878, pp. 221–228. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92537-0_26
13. Duda, P., Rutkowski, L., Jaworski, M., Rutkowska, D.: On the Parzen kernel-based probability density function learning procedures over time-varying streaming data with applications to pattern classification. IEEE Trans. Cybern. **50**(4), 1683–1696 (2020)
14. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 286–295. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28645-5_29
15. Gomes, H.M., et al.: Adaptive random forests for evolving data stream classification. Mach. Learn. 1469–1495 (2017). https://doi.org/10.1007/s10994-017-5642-8
16. Gondara, L.: Medical image denoising using convolutional denoising autoencoders. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), pp. 241–246. IEEE (2016)
17. Grais, E.M., Plumbley, M.D.: Single channel audio source separation using convolutional denoising autoencoders. In: 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 1265–1269. IEEE (2017)
18. Hou, Y., Holder, L.B.: On graph mining with deep learning: introducing model r for link weight prediction. J. Artif. Intell. Soft Comput. Res. **9**(1), 21–40 (2019)
19. Jaworski, M.: Regression function and noise variance tracking methods for data streams with concept drift. Int. J. Appl. Math. Comput. Sci. **28**(3), 559–567 (2018)
20. Jaworski, M., Duda, P., Rutkowski, L.: New splitting criteria for decision trees in stationary data streams. IEEE Trans. Neural Netw. Learn. Syst. **29**(6), 2516–2529 (2017)
21. Jaworski, M., Duda, P., Rutkowski, L.: On applying the restricted Boltzmann machine to active concept drift detection. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8. IEEE (2017)
22. Jaworski, M., Duda, P., Rutkowski, L.: Concept drift detection in streams of labelled data using the restricted Boltzmann machine. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE (2018)
23. Jaworski, M., Rutkowski, L., Duda, P., Cader, A.: Resource-aware data stream mining using the restricted Boltzmann machine. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) ICAISC 2019. LNCS (LNAI), vol. 11509, pp. 384–396. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20915-5_35
24. Kamimura, R.: Supposed maximum mutual information for improving generalization and interpretation of multi-layered neural networks. J. Artif. Intell. Soft Comput. Res. **9**(2), 123–147 (2019)

25. Koren, O., Hallin, C.A., Perel, N., Bendet, D.: Decision-making enhancement in a big data environment: application of the k-means algorithm to mixed data. J. Artif. Intell. Soft Comput. Res. **9**(4), 293–302 (2019)
26. Kumarratneshk, R., Weilleweill, E., Aghdasi, F., Sriram, P.: A strong and efficient baseline for vehicle re-identification using deep triplet embedding. J. Artif. Intell. Soft Comput. Res. **10**(1), 27–45 (2020)
27. Ludwig, S.A.: Applying a neural network ensemble to intrusion detection. J. Artif. Intell. Soft Comput. Res. **9**(3), 177–188 (2019)
28. Nobukawa, S., Nishimura, H., Yamanishi, T.: Pattern classification by spiking neural networks combining self-organized and reward-related spike-timing-dependent plasticity. J. Artif. Intell. Soft Comput. Res. **9**(4), 283–291 (2019)
29. Ororbia, A.G.I., Giles, C.L., Reitter, D.: Online semi-supervised learning with deep hybrid Boltzmann machines and denoising autoencoders. CoRR, abs/1511.06964 (2015)
30. Page, E.S.: Continuous inspection schemes. Biometrika **41**(1/2), 100–115 (1954)
31. Pietruczuk, L., Rutkowski, L., Jaworski, M., Duda, P.: The Parzen kernel approach to learning in non-stationary environment. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 3319–3323. IEEE (2014)
32. Pietruczuk, L., Rutkowski, L., Jaworski, M., Duda, P.: A method for automatic adjustment of ensemble size in stream data mining. In: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 9–15. IEEE (2016)
33. Pietruczuk, L., Rutkowski, L., Jaworski, M., Duda, P.: How to adjust an ensemble size in stream data mining? Inf. Sci. **381**, 46–54 (2017)
34. Rafajłowicz, E., Rafajłowicz, W.: Testing (non-) linearity of distributed-parameter systems from a video sequence. Asian J. Control **12**(2), 146–158 (2010)
35. Rafajłowicz, E., Rafajłowicz, W.: Iterative learning in repetitive optimal control of linear dynamic processes. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2016. LNCS (LNAI), vol. 9692, pp. 705–717. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39378-0_60
36. Rafajłowicz, E., Rafajłowicz, W.: Iterative learning in optimal control of linear dynamic processes. Int. J. Control **91**(7), 1522–1540 (2018)
37. Rafajłowicz, E., Wnuk, M., Rafajłowicz, W.: Local detection of defects from image sequences. Int. J. Appl. Math. Comput. Sci. **18**(4), 581–592 (2008)
38. Read, J., Perez-Cruz, F., Bifet, A.: Deep learning in partially-labeled data streams. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC 2015, New York, NY, USA, pp. 954–959. ACM (2015)
39. Rutkowski, L., Pietruczuk, L., Duda, P., Jaworski, M.: Decision trees for mining data streams based on the McDiarmid's bound. IEEE Trans. Knowl. Data Eng. **25**(6), 1272–1279 (2013)
40. Shewalkar, A., Nyavanandi, D., Ludwig, S.A.: Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU. J. Artif. Intell. Soft Comput. Res. **9**(4), 235–245 (2019)
41. Wang, L., Zhang, Z., Chen, J.: Short-term electricity price forecasting with stacked denoising autoencoders. IEEE Trans. Power Syst. **32**(4), 2673–2681 (2016)
42. Wu, Y., DuBois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, pp. 153–162 (2016)
43. Zalasinski, M., Lapa, K., Cpalka, K., Przybyszewski, K., Yen, G.G.: On-line signature partitioning using a population based algorithm. J. Artif. Intell. Soft Comput. Res. **10**(1), 5–13 (2020)