

An algorithm for training neural networks with L1 regularization

Ekaterina Gribanova, Roman Gerasimov

Department of Automated Control Systems, Faculty of Control Systems, Tomsk State University of Control Systems and Radio Electronics, Tomsk, Russia

Article Info

Article history:

Received Mar 31, 2025

Revised Jul 22, 2025

Accepted Aug 6, 2025

Keywords:

Dropout

Inverse problem

L1 regularization

Neural network

Optimization method

Pruning

ABSTRACT

This paper presents a new algorithm for building neural network models that automatically selects the most important features and parameters while improving prediction accuracy. Traditional neural networks often use all available input parameters, leading to complex models that are slow to train and prone to overfitting. The proposed algorithm addresses this challenge by automatically identifying and retaining only the most significant parameters during training, resulting in simpler, faster, and more accurate models. We demonstrate the practical benefits of the proposed algorithm through two real-world applications: stock market forecasting using the Wilshire index and business profitability prediction based on company financial data. The results show significant improvements over conventional methods: models use fewer parameters—creating simpler, more interpretable solutions—achieve better prediction accuracy, and require less training time. These advantages make the algorithm particularly valuable for business applications where model simplicity, speed, and accuracy are crucial. The method is especially beneficial for organizations with limited computational resources or that require fast model deployment. By automatically selecting the most relevant features, it reduces the need for manual feature engineering and helps practitioners build more efficient predictive models without requiring deep technical expertise in neural network optimization.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Ekaterina Gribanova

Department of Automated Control Systems, Faculty of Control Systems

Tomsk State University of Control Systems and Radio Electronics

Avenue. Lenin 40, Tomsk, Russia

Email: ekaterina.b.gribanova@tusur.ru

1. INTRODUCTION

Machine learning models, particularly neural networks, face a fundamental challenge known as overfitting—the phenomenon where a model learns the training data too closely, including its noise and specific patterns, resulting in poor performance when applied to new, unseen data. This problem becomes especially pronounced as models grow in complexity and when training data is limited relative to the number of model parameters. Regularization emerges as a critical solution to address overfitting by constraining model complexity and encouraging simpler, more generalizable solutions. The core principle of regularization is to add constraints or penalties that prevent the model from becoming overly complex, thereby improving its ability to perform well on new data. This concept represents a trade-off between fitting the training data perfectly and maintaining the model's ability to generalize to unseen examples.

Various regularization approaches have been developed to tackle this challenge [1]. These methods can be broadly categorized into several types: data-based techniques such as data augmentation that increase

training set diversity through modifications like adding Gaussian noise [2] or geometric transformations; architectural approaches including dropout [3] which randomly deactivates neurons during training; optimization strategies like early stopping based on validation performance [4]; and penalty-based methods that modify the loss function directly. Among penalty-based regularization techniques, L1 and L2 regularization have gained particular prominence due to their mathematical elegance and practical effectiveness. L2 regularization (also known as ridge regression) adds a penalty proportional to the sum of squared weights, encouraging smaller weight values and smoother solutions. L1 regularization, introduced by Tibshirani [5] as the least absolute shrinkage and selection operator (LASSO) method, applies a penalty proportional to the sum of absolute weight values. The distinguishing characteristic of L1 regularization lies in its ability to drive some weights to exactly zero, effectively performing automatic feature selection. This property makes L1 regularization particularly valuable in scenarios where identifying the most relevant features is as important as achieving good predictive performance. In contrast, L2 regularization typically shrinks weights toward zero without eliminating them entirely. Hybrid approaches like elasticNet [6] combine both penalties, while variants such as L0 regularization [7] and L1/2 regularization [8] offer different sparsity-inducing properties.

The practical applications of L1 regularization demonstrate its significance across various domains. In financial modeling [9], L1/2 regularization has been successfully applied to neural networks for predicting financial distress, enabling automatic selection of the most significant financial and non-financial variables from large feature sets. Similarly, in credit risk assessment [10], L1 regularization helped eliminate redundant information in deep neural networks, significantly improving forecasting performance. Medical applications [11], image recognition tasks [12], [13], and high-dimensional biological data analysis have all benefited from the feature selection capabilities of L1 regularization.

Despite these advantages, implementing L1 regularization in neural network training presents significant computational challenges. The absolute value function in the L1 penalty term is non-differentiable at zero, making standard gradient-based optimization methods problematic. Existing approaches include heuristic methods that exclude parameters based on their contribution ratios [14], smoothing techniques using piecewise polynomial approximations [15], and coordinate descent algorithms with cross-validation [16]. However, these methods often suffer from computational inefficiency, convergence difficulties, or require extensive hyperparameter tuning.

The motivation for this research stems from the need to overcome these computational barriers while preserving the valuable feature selection properties of L1 regularization. Current methods either compromise on the exactness of the L1 penalty or require computationally expensive procedures that limit their practical applicability, especially for large-scale neural networks. This study addresses these limitations by proposing a novel optimization approach that reformulates the L1-regularized neural network training problem as an inverse single-point problem. Our contribution lies in developing a computationally efficient algorithm that maintains the theoretical properties of L1 regularization. The practical significance of this work extends to applications requiring both high predictive accuracy and model interpretability, including medical diagnosis, financial modeling, and scientific research where understanding feature importance is crucial for decision-making.

2. MATERIALS AND METHOD

This section presents the theoretical foundation and methodological framework for the study. The first component covers neural network architecture and L1 regularization techniques for feature selection and sparsity promotion. The second component details a novel training algorithm based on constrained optimization principles with selective weight update mechanisms.

2.1. Neural network

A neural network is a computational model inspired by the way biological neural networks process information [17]. At its core, a neural network consists of interconnected processing units called neurons or nodes, organized in layers that transform input data into meaningful outputs through learned mathematical operations. The fundamental architecture comprises three main components: an input layer that receives data, one or more hidden layers that perform intermediate computations, and an output layer that produces final predictions Figure 1. Each connection between neurons has an associated weight that determines the strength and direction of information flow. The network learns by adjusting these weights during training to minimize prediction errors.

For mathematical formulation, consider input data X with dimensions $N \times M \in \mathbb{R}$, where N is the number of observations, and M is the number of features. The actual output variable Y has dimension $N \times 1$. The network processes information through successive transformations. The hidden layer values are computed as:

$$l = f_1(X \cdot w_0)$$

where w_0 are the weights of the first layer, f_l is the activation function of the first layer. Activation functions are crucial components that enable neural networks to capture complex, non-linear relationships in data.

Based on the values of neurons of the hidden layer l , the output of the neural network is calculated using the formula:

$$z = f_2(l \cdot w_1)$$

where w_1 are the weights connecting the hidden layer to the output layer, and f_2 is the activation function of the output layer. Bias terms are incorporated by adding columns of ones to both X and l matrices, providing additional flexibility in model fitting.

The adjustment of the network's weights is performed using optimization algorithms, typically based on gradient descent. The function to be optimized is the error function J , which reflects the deviation of predicted values from actual values, as defined in (1):

$$J(w) = \sum_{i=1}^N (y_i - z_i)^2 \quad (1)$$

where y is the vector of actual output values; z is the vector of predicted values; and N is the number of observations.

In classical gradient descent, after a forward pass, a backward pass is executed to compute gradients used for updating weights (f' is the derivative of the activation function). Thus, the error at the output layer is calculated as:

$$z_{error} = z - y$$

$$z_{delta} = z_{error} \odot f'_2(z)$$

where \odot is the element-wise multiplication.

The error at the hidden layer can be defined as:

$$l_{error} = z_{delta} \cdot w_1^T$$

$$l_{delta} = l_{error} \odot f'_1(l)$$

Consequently, the weight updates for the first and second layers are performed as (2):

$$\begin{aligned} w_0 &= w_0 - \eta \cdot X^T l_{delta} \\ w_1 &= w_1 - \eta \cdot l^T z_{delta} \end{aligned} \quad (2)$$

where η is the learning rate.

The L1 regularization technique can be formulated using the Lagrangian approach, which transforms the constrained optimization problem into an unconstrained form. This formulation introduces the regularization parameter λ that controls the trade-off between model fitting accuracy and sparsity, allowing practitioners to adjust the level of feature selection according to their specific requirements. The optimized function (1) with L1 regularization in Lagrangian form can be represented as (3):

$$\tilde{J}(w) = J(w) + \lambda \|w\|_1 \quad (3)$$

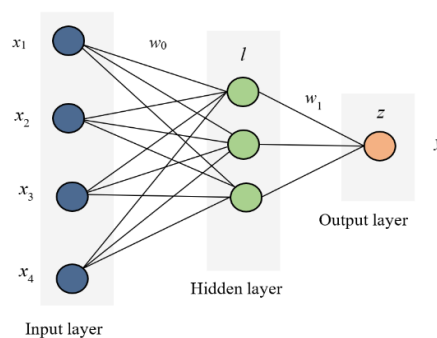


Figure 1. Neural network structure

Graphically, the regularization term takes on a diamond shape, which encourages some weight coefficients to become zero, as the minimum is often reached at its corners (Figure 2). Figure 2 illustrates the geometric interpretation of L1 regularization in a two-dimensional parameter space, where the diamond-shaped constraint region (shown in dark blue) represents the L1 penalty term $\|w\|_1 \leq t$ for some threshold t . The elliptical contours (shown in green and light blue) represent the iso-lines of the error function $J(w)$, with the innermost contour indicating the unconstrained minimum of the loss function. The optimal solution under L1 regularization occurs at the point where the smallest error function contour touches the diamond-shaped constraint region. Due to the sharp corners of the diamond constraint, this intersection frequently occurs at the axes (where one parameter equals zero), demonstrating how L1 regularization naturally produces sparse solutions by setting some coefficients to exactly zero.

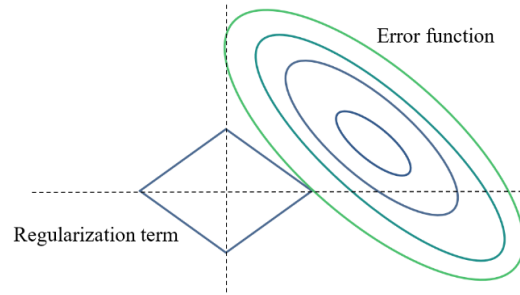


Figure 2. Graphical representation of the problem

2.2. Neural network training algorithm

The proposed neural network training algorithm is based on reformulating the minimization problem of function (3) into a constrained optimization problem. This reformulation enables the application of methods for solving inverse single-point problems [18], [19] to the neural network learning process. In the reformulated problem, the sum of the absolute values of the parameters is minimized while the training error is constrained to a specified target value, as defined in (4):

$$\begin{aligned} \|w\|_1 &\rightarrow \min \\ J(w) &= J^* \end{aligned} \quad (4)$$

where J^* is the target value of the constraint function.

The solution approach involves iteratively updating arguments according to the following rule: at each iteration, select an argument for which the absolute value of the partial derivative of the constraint function is maximized, then adjust its value using gradient descent [18], [19]. The stopping conditions for iterations include reaching the target value of the constraint function with a specified accuracy or lack of improvement in solutions, as well as reaching a predetermined number of iterations. There may be cases where the target value is not achieved during problem-solving. In such instances, values obtained through other stopping criteria are considered as solutions, being as close as possible to the target value of the constraint function with this search strategy. In this study, the target value J^* is assumed to be a small arbitrary number, under the assumption that this value will not be reached. This assumption is implemented by excluding from the stopping rules achieving the target value of the constraint function with specified accuracy.

This approach ensures differentiability of the objective function and does not require tuning of the regularization parameter. It is also worth noting that the initial values of the weight coefficients are set to zero, rather than being generated randomly as in well-known algorithms. For each weight coefficient w , an additional value u is defined to indicate its applicability in calculations. This feature can take two values: 0 or 1, reflecting whether the corresponding weight coefficient can be modified in the current iteration. Modification is excluded if adjusting the weight coefficient in the previous iteration resulted in a worsening of the optimized error function.

The proposed training algorithm incorporates a selective weight update mechanism based on gradient magnitude. This approach aims to improve convergence efficiency by preventing unnecessary updates of weights. The mechanism dynamically adjusts weight modification priorities during training, ensuring that computational resources are focused on the most beneficial parameter adjustments.

The algorithm for a single hidden layer will include the following steps:

Step 1. Error function calculation: calculate the current error function value: $J_{prev}=J(w)$ (1).

Step 2. Gradient computation: calculate gradient values for all weight coefficients (2):

$$g_0 = X^T l_{delta}$$

$$g_1 = l^T z_{delta}$$

Step 3. Selective weight update: determine the maximum value of the product between the gradient value and the applicable weight:

$$g_{max} = \max\{|g_0| \cdot u_0, |g_1| \cdot u_1\}$$

If this maximum value corresponds to the weight coefficients connecting the hidden layer to the output layer (j is the index of the maximum element), then the weight coefficient of w_1 is adjusted:

$$w_{1j}^* = w_{1j} - \eta \cdot g_{1j}$$

where η is the parameter defining the step of weight coefficient change. Otherwise, update the weight coefficient connecting the input layer to the intermediate layer:

$$w_{0j}^* = w_{0j} - \eta \cdot g_{0j}$$

Step 4. Performance evaluation and adaptation: calculate the new error function value: $J_{new}=J(w)$. If $J_{new} < J_{prev}$, u values are set to 1 for all weight coefficients, $w=w^*$, $J_{prev}=J_{new}$. Go to step 2. Otherwise, the value u for the corresponding modified weight coefficient is set to zero: $u_{0j} = 0$ (if w_{0j} was changed) or $u_{1j} = 0$ (if w_{1j} was changed). Go to step 2.

Stopping criterion: either all values of u are equal to 0, or the specified number of iterations has been completed.

3. RESULTS AND DISCUSSION

This section presents the empirical validation of the proposed neural network training algorithm through computational experiments. The evaluation is structured into two main parts to systematically demonstrate the algorithm's effectiveness across different network architectures and application domains. First, we examine the algorithm's performance on single-layer networks using financial time series data as presented in section 3.1. This analysis focuses on feature selection capabilities and prediction accuracy in forecasting stock market indices. Second, we investigate the algorithm's application to neural networks with hidden layers using enterprise financial data as discussed in section 3.2, emphasizing parameter reduction. For each experimental setting, we compare proposed algorithm against established baseline approaches, including standard adaptive moment estimation (Adam) optimization with various L1 regularization parameters and dropout techniques. The reliability of our results is ensured through rigorous experimental design, comparison with well-established optimization methods under identical conditions, and use of different activation functions and network configurations.

The proposed algorithm was implemented in Python using native NumPy operations for matrix computations and gradient calculations, while the optimization method Adam was implemented in Python using the Keras library. Additionally, we implemented L1 regularization techniques and hyperparameter optimization using GridSearchCV from the scikit-learn library [20] in the Adam method. GridSearchCV performs search over specified parameter values, evaluating each combination through cross-validation to identify optimal hyperparameters including the number of training epochs and batch size. This systematic approach ensures fair comparison between proposed algorithm and conventional regularization methods by selecting the best possible configuration for each approach.

3.1. Single-layer network

To evaluate the feature selection capabilities of proposed algorithm we conducted experiments using financial time series data from the Wilshire 2500 total market index. This dataset represents a real-world scenario where identifying the most relevant historical values for prediction is crucial for practical applications. The experimental setup involved predicting future index values based on a 40-day historical window, using daily data from January 2021 to December 2023 (754 total observations, with the last 200 observations reserved for testing). We deliberately chose this financial dataset because it exhibits the complex temporal dependencies and noise characteristics typical of real-world prediction problems, making it an appropriate testbed for evaluating feature selection methods.

The results, including mean squared error (MSE) and mean absolute error (MAE) for the test set, are presented in Table 1. Table 1 also lists the regularization parameter value λ , with the arctg used as the activation function of single-layer neural network. The proposed algorithm demonstrated superior performance compared to traditional L1 regularization approaches. While conventional L1 regularization ($\lambda=0.01$ to 1.0) failed to perform actual feature selection—retaining all 40 input features while only reducing coefficient magnitudes—the proposed algorithm successfully identified just 6 features that yielded better prediction accuracy.

The results in Table 1 reveal several important findings. First, standard Adam optimization without regularization achieved $\text{MSE}=4.39\times 10^{-5}$, while the proposed algorithm reduced this to $\text{MSE}=2.48\times 10^{-6}$, representing an 18-fold improvement. Second, the computational efficiency was enhanced, reducing training time from 11.3 seconds to 0.43 seconds. This improvement stems from the algorithm's ability to eliminate irrelevant parameters during training rather than merely penalizing them.

Table 1. Results of computational experiments based on the Wilshire 2500 index

Method	Number of selected features (p)	MSE	MAE	Time (seconds)
Adam, $\lambda=0$	40	4.39×10^{-5}	0.0053	11.3
Adam, $\lambda=0.01$	40	2.03×10^{-4}	0.0103	12.8
Adam, $\lambda=0.5$	40	0.0023	0.0105	12.8
Adam, $\lambda=1.0$	40	0.0048	0.0101	7.45
Proposed algorithm, $\eta=0.1$	6	2.48×10^{-6}	0.0013	0.43

The experimental results demonstrate the effectiveness of the proposed algorithm in both prediction accuracy and computational performance. Figure 3 demonstrates the practical prediction quality, showing close alignment between actual and predicted values on the test set. The analysis across different activation functions Figure 4 confirms the algorithm's robustness, consistently achieving lower error rates than baseline method across various function types, including specialized financial modeling functions like cloglogm and tanh [21]. The number of selected features using the proposed algorithm varied from 2 to 10 for different activation functions, allowing for higher accuracy.

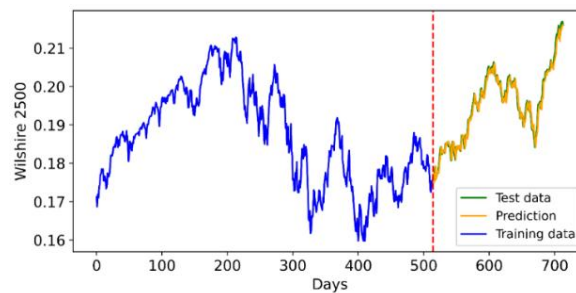


Figure 3. Actual and predicted index values

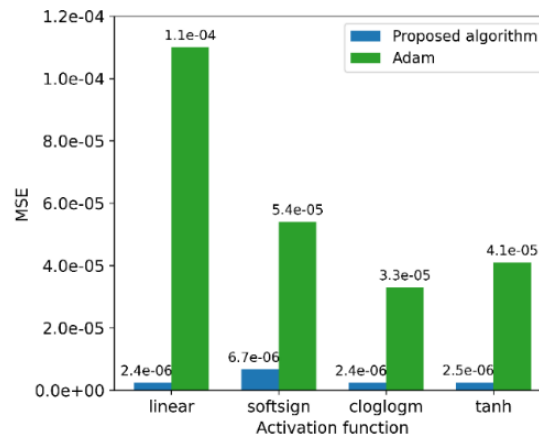


Figure 4. MSE values by activation function

3.2. Neural network with a single hidden layer

The second phase of our evaluation examined the algorithm's performance on more complex architectures with a single hidden layer [22]–[24] using enterprise financial data. We utilized a comprehensive dataset of 551 Russian enterprises with revenues exceeding 100 million rubles, spanning 2017–2020 [25]. The training dataset contains information about these 551 enterprises for 2017–2019, while the test dataset includes data for the same enterprises for 2020. The prediction task—forecasting enterprise profitability based on key financial indicators (liquidity ratio, fixed asset share, financial leverage, and asset turnover). The experimental design compared networks with 2 and 16 hidden neurons to examine how proposed algorithm performs across different complexity levels. The choice of sigmoid activation for hidden layers and linear activation for output follows established practices in financial modeling, ensuring our results are comparable to standard approaches in this domain.

The results in Table 2 reveal evidence of the algorithm's effectiveness. For the 2-neuron network, proposed algorithm reduced parameters from 13 to 8 while improving MSE from 174.79 to 156.28 and MAE from 8.54 to 8.06. More significantly, for the 16-neuron network, the algorithm achieved dramatic parameter reduction from 97 to 13 parameters while maintaining comparable accuracy (MSE=155.01 vs. 156.00), demonstrating effective model simplification without performance degradation.

Particularly noteworthy is the algorithm's superiority over dropout regularization, a widely-used technique for preventing overfitting. While dropout with 20% neuron elimination yielded MSE=195.88, proposed algorithm achieved MSE=155.01 with substantial parameter reduction, indicating more effective regularization through parameter selection rather than random elimination. The computational efficiency gains (training time reduced from 32.8 to 9.1 seconds) further demonstrate practical advantages for large-scale applications.

Figure 5 illustrates the algorithm's sensitivity to the η parameter, showing optimal performance at $\eta=10^{-4}$, where an effective balance between parameter selection and accuracy is achieved. At $\eta=10^{-2}$, parameter selection was not performed because the optimized function yielded higher values when the arguments changed significantly compared to their zero values. For $\eta=10^{-6}$, there was a slow decrease in the optimized function due to the use of a small step size. The consistency of improvements across both simple and complex architectures validates the algorithm's general applicability and suggests its potential for broader implementation in neural network optimization tasks where parameter efficiency and model interpretability are valued alongside predictive performance.

Table 2. Results of neural network modeling with a hidden layer

Method	Number of neurons in a hidden layer	Number of non-zero parameters	MSE	MAE	Time (seconds)
Adam, $\lambda=0$	2	13	174.79	8.54	33.2
Adam, $\lambda=0.01$	2	13	176.54	8.54	29.9
Adam, $\lambda=0.5$	2	13	182.19	8.54	29.2
Adam, $\lambda=1.0$	2	13	196.53	8.89	28.5
Proposed algorithm, $\eta=0.0001$	2	8	156.28	8.06	0.6
Adam, $\lambda=0$	16	97	156	8.08	25.8
Adam, $\lambda=0.01$	16	97	156.99	8.09	22.2
Adam, $\lambda=0.5$	16	97	178.7	8.34	22.9
Adam, $\lambda=1.0$	16	97	193.43	8.6	23.7
Adam, dropout (0.2)	16	97	195.88	8.7	32.8
Proposed algorithm, $\eta=0.0001$	16	13	155.01	8.08	9.1

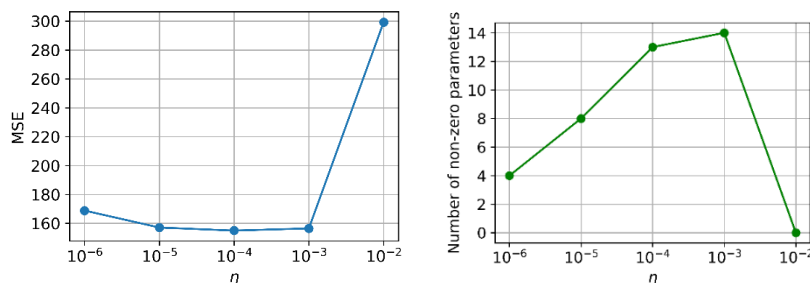


Figure 5. Changes in MSE and the number of non-zero parameters with varying η

4. CONCLUSION

An algorithm has been developed for training a neural network with L1 regularization, based on reformulating the optimization problem of the loss function as an inverse single-point problem while

An algorithm for training neural networks with L1 regularization... (Ekaterina Gribova)

minimizing the sum of modules of arguments. Experiments were conducted with two test datasets, leading to the following conclusions. The use of L1 regularization, implemented in the Keras library, did not perform parameter selection in the examined cases; however, it did reduce the absolute values of the weight coefficients. The proposed algorithm successfully performed feature selection for the first dataset. Experiments on the second dataset demonstrated that the algorithm effectively zeroed out the weight coefficients, resulting in a reduction in the number of adjustable parameters. This, in turn, simplified the network architecture and improved model accuracy. As a result, the studied examples achieved faster parameter tuning and lower MSE and MAE values. Another advantage of this method is that it eliminates the need for generating random weight values, thus removing its stochastic nature. This ensures that the method will yield identical results across multiple runs, providing stability and reproducibility of the obtained data. However, the method has some drawbacks, including high sensitivity to the choice of the parameter η . If an incorrect value is selected, all weight coefficients may become zero, as their adjustment would lead to worse values of the optimized function. For this reason, the use of the proposed algorithm is complicated for certain activation functions, such as sincos activation function. Future research will focus on developing and investigating a hybrid algorithm that synthesizes the proposed algorithm with existing neural network training methods.

FUNDING INFORMATION

This study was funded by the Russian Science Foundation, grant number 25-21-00123 “Application of methods for solving inverse ill-posed problems in machine learning”: <https://rscf.ru/project/25-21-00123/>.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Ekaterina Gribova	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Roman Gerasimov			✓	✓		✓		✓	✓	✓	✓			

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that there is no conflict of interest in relation to this paper, as well as the published research results, including the financial aspects of conducting the research, obtaining and using its results, as well as any non-financial personal relationships.

ETHICAL APPROVAL

The author confirms that did not use artificial intelligence technologies when creating the current work.

DATA AVAILABILITY

Data will be made available on reasonable request.




REFERENCES

- [1] G. Nuti, A.-I. Cross, and P. Rindler, “Evidence-based regularization for neural networks,” *Machine Learning and Knowledge Extraction*, vol. 4, no. 4, pp. 1011–1023, 2022, doi: 10.3390/make4040051.
- [2] H.-X. Dou, X.-S. Lu, C. Wang, H.-Z. Shen, Y.-W. Zhuo, and L.-J. Deng, “PatchMask: a data augmentation strategy with gaussian noise in hyperspectral images,” *Remote Sensing*, vol. 14, no. 24, 2022, doi: 10.3390/rs14246308.
- [3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [4] M. V. Ferro, Y. D. Mosquera, F. J. R. Pena, and V. M. D. Bilbao, “Early stopping by correlating online indicators in neural networks,” *Neural Networks*, vol. 159, pp. 109–124, 2023, doi: 10.1016/j.neunet.2022.11.035.
- [5] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996, doi: 10.1111/j.2517-6161.1996.tb02080.x.




- [6] Q. Li, W. Qiao, Y. Shi, W. Ba, F. Wang, and X. Hu, "Temperature modeling of wave rotor refrigeration process based on elastic net variable selection and deep belief network," *Chemometrics and Intelligent Laboratory Systems*, vol. 239, 2023, doi: 10.1016/j.chemolab.2023.104872.
- [7] Z. Wei, Q. Li, J. Wei, and W. Bian, "Neural network for a class of sparse optimization with L0-regularization," *Neural Networks*, vol. 151, pp. 211–221, 2022, doi: 10.1016/j.neunet.2022.03.033.
- [8] Z. Xu, H. Zhang, Y. Wang, X. Chang, and Y. Liang, "L 1/2 regularization," *Science China Information Sciences*, vol. 53, no. 6, pp. 1159–1169, 2010, doi: 10.1007/s11432-010-0090-0.
- [9] Y. Chen, J. Guo, J. Huang, and B. Lin, "A novel method for financial distress prediction based on sparse neural networks with L 1/2 regularization," *International Journal of Machine Learning and Cybernetics*, vol. 13, no. 7, pp. 2089–2103, 2022, doi: 10.1007/s13042-022-01566-y.
- [10] M. Yang, M. K. Lim, Y. Qu, X. Li, and D. Ni, "Deep neural networks with L1 and L2 regularization for high dimensional corporate credit risk prediction," *Expert Systems with Applications*, vol. 213, Mar. 2023, doi: 10.1016/j.eswa.2022.118873.
- [11] N. Ekwu, T. Mrziglod, and A. Schuppert, "Neural network input feature selection using structured l2-norm penalization," *Applied Intelligence*, vol. 53, no. 5, pp. 5732–5749, 2022, doi: 10.1007/s10489-022-03539-8.
- [12] B. Han *et al.*, "HSR: L 1/2-regularized sparse representation for fast face recognition using hierarchical feature selection," *Neural Computing and Applications*, vol. 27, no. 2, pp. 305–320, 2016, doi: 10.1007/s00521-015-1907-y.
- [13] E. Phaisangittisagul, "An analysis of the regularization between L2 and dropout in single hidden layer neural network," in *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, 2016, pp. 174–179, doi: 10.1109/ISMS.2016.14.
- [14] L. Görlitz, R. Loosen, and T. Mrziglod, "Topology optimization of artificial neural networks using L1-penalization," in *Proceedings 20. Workshop Computational Intelligence*, 2010, pp. 80–87.
- [15] K. S. Mohamed, "Batch gradient learning algorithm with smoothing L1 regularization for feedforward neural networks," *Computers*, vol. 12, no. 1, 2022, doi: 10.3390/computers12010004.
- [16] D. Chetverikov, Z. Liao, and V. Chernozhukov, "On cross-validated lasso in high dimensions," *The Annals of Statistics*, vol. 49, no. 3, pp. 1300–1317, 2021, doi: 10.1214/20-AOS2000.
- [17] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of Microbiological Methods*, vol. 43, no. 1, pp. 3–31, 2000, doi: 10.1016/S0167-7012(00)00201-3.
- [18] E. Gribanova, "Algorithm for solving the inverse problems of economic analysis in the presence of limitations," *EUREKA: Physics and Engineering*, vol. 1, no. 1, pp. 70–78, 2020, doi: 10.21303/2461-4262.2020.001102.
- [19] E. Gribanova, "Elaboration of an algorithm for solving hierarchical inverse problems in applied economics," *Mathematics*, vol. 10, no. 15, p. 2779, 2022, doi: 10.3390/math10152779.
- [20] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [21] G. S. S. Gomes, T. B. Luderim, and L. M. M. R. Lima, "Comparison of new activation functions in neural network for forecasting financial time series," *Neural Computing and Applications*, vol. 20, no. 3, pp. 417–439, Apr. 2011, doi: 10.1007/s00521-010-0407-3.
- [22] A. Xu, H. Chang, Y. Xu, R. Li, X. Li, and Y. Zhao, "Applying artificial neural networks (ANNs) to solve solid waste-related issues: A critical review," *Waste Management*, vol. 124, pp. 385–402, Apr. 2021, doi: 10.1016/j.wasman.2021.02.029.
- [23] F. Lolli, R. Gamberini, A. Regattieri, E. Balugani, T. Gatos, and S. Gucci, "Single-hidden layer neural networks for forecasting intermittent demand," *International Journal of Production Economics*, vol. 183, pp. 116–128, 2017, doi: 10.1016/j.ijpe.2016.10.021.
- [24] Y. Luo *et al.*, "A dissolved oxygen levels prediction method based on single-hidden layer feedforward neural network using neighborhood information metric," *Applied Soft Computing*, vol. 167, Dec. 2024, doi: 10.1016/j.asoc.2024.112328.
- [25] D. B. Vukovic, L. Spitsina, E. Gribanova, V. Spitsin, and I. Lyzin, "Predicting the performance of retail market firms: regression and machine learning methods," *Mathematics*, vol. 11, no. 8, 2023, doi: 10.3390/math11081916.

BIOGRAPHIES OF AUTHORS



Ekaterina Gribanova    holds a Doctor of Technical Sciences from Tomsk State University of Control Systems and Radio Electronics, Russian Federation, 2023. She is currently a Professor at Department of Automated Control Systems, Tomsk State University of Control Systems and Radio Electronics, Russian Federation. Her research includes metaheuristics, global optimization, machine learning, and inverse problems. She has published over 100 papers in journals and conferences. She can be contacted at email: ekaterina.b.gribanova@tusur.ru.



Roman Gerasimov    received a bachelor's degree in Applied Computer Science from Tomsk State University of Control Systems and Radioelectronics (TUSUR) with a job on "The module for processing bank documents on protested transactions in PJSC MTS Bank of Tomsk." Since 2025, he has been an Engineer at the Laboratory of Image Processing and Artificial Intelligence (TUSUR). His research interests are in modular neural networks and pattern recognition. He can be contacted at email: roman.s.gerasimov@tusur.ru.