Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Neural Network Method for Structural Outlier Detection without Hyperparameter Tuning

Leonid N. Yasnitsky

lyasnitskiy@hse.ru

HSE University https://orcid.org/0000-0002-8212-3826

Maxim A. Goldobin

Perm National Research Polytechnic University https://orcid.org/0009-0004-9226-1482

Anton R. Neverov

LLC «DigitalBizFactory» https://orcid.org/0009-0003-1332-846X

Alexey S. Mezentsev

Joint Stock Company «OTP Bank» https://orcid.org/0000-0001-8867-8956

Yulia V. Karakulova

Academician E.A. Wagner Perm State Medical University https://orcid.org/0000-0002-7536-2060

Andrey A. Dumler

Academician E.A. Wagner Perm State Medical University, Perm, Russian Federation
https://orcid.org/0000-0002-4915-1059

Research Article

Keywords: Outlier Detection, Neural Networks, Structural Anomalies, Arnold–Kolmogorov–Hecht–Nielsen Theorem, Hyperparameter Selection

Posted Date: June 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10105267/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Neural Network Method for Structural Outlier Detection without Hyperparameter Tuning

Leonid N. Yasnitsky[1], Maxim A. Goldobin[2], Anton R. Neverov[3], Alexey S. Mezentsev[4], Yulia V. Karakulova[5], Andrey A. Dumler[5]

[1] National Research University Higher School of Economics, Perm, Russian Federation
[2] Perm National Research Polytechnic University, Perm, Russian Federation
[3] LLC «DigitalBizFactory»
[4] Joint Stock Company «OTP Bank»
[5] Academician E.A. Wagner Perm State Medical University, Perm, Russian Federation

Corresponding author:
Leonid N. Yasnitsky
E-mail: yasn1949@mail.ru
ORCID: 0000-0002-8212-3826

Abstract

The detection of structural outliers—anomalies that violate functional relationships between variables while not exhibiting extreme values in individual features—remains a challenging problem in data analysis and machine learning modeling. Existing methods are generally aimed at identifying points with anomalous feature values and, as a consequence, are ineffective for detecting structural violations. Moreover, most of them require labor-intensive hyperparameter tuning, which becomes difficult in the absence of labeled data.

This paper proposes two modifications of a neural-network approach to structural outlier detection: the NNF and NNFA methods. The approach is based on the property of certain neural networks with specific architectures to produce elevated training errors for observations that do not conform to the regularities of the domain under study. The selection of architectural parameters for such neural networks is simplified through the use of a corollary of the Arnold–Kolmogorov–Hecht–Nielsen theorem, which determines an admissible range for the number of neurons in the hidden layer. The NNF method employs a single network with the minimum number of neurons from this range, whereas the NNFA method averages the training errors of an ensemble of networks within the specified range, making it possible to apply the proposed methods without any hyperparameter tuning.

Experiments were conducted on three synthetic datasets (a parabola, a paraboloid of revolution, and a hyperbolic paraboloid) and three real-world datasets ("Concrete Compressive Strength" and "California Housing"), as well as on an additional dataset used for sensitivity analysis with respect to the noise level. Comparison with 15 standard methods from the PyOD and scikit-learn libraries shows that the proposed methods consistently achieve higher quality metrics. It is also demonstrated that the methods maintain high performance over a wide range of outlier proportions (from 1% to 10%), while the computational complexity of the NNFA method can be controlled by a parameter governing the width of the averaging interval. All source codes and datasets used in this study are publicly available.

Keywords: Outlier Detection; Neural Networks; Structural Anomalies; Arnold–Kolmogorov–Hecht–Nielsen Theorem; Hyperparameter Selection.

1 Introduction

The problem of detecting outliers in statistical data has attracted the attention of researchers for several decades. Interest in this problem is driven by two reasons. First, anomaly detection makes it possible to solve a wide range of important practical tasks, such as fraud detection, intrusion detection in computer networks, diagnosis of rare diseases, quality control in manufacturing, and others. Second, and more importantly, outliers caused by various errors often hinder or even make it impossible to build machine-learning models of acceptable quality. Sources of such outliers include equipment malfunctions, measurement errors, data collection and processing errors, dataset labeling errors, human factors, and others.

A widely cited definition by Grubbs (1969) states: *"An outlying observation is one that appears to deviate markedly from other members of the sample in which it occurs."* Other definitions describe an outlier as *"an observation that appears inconsistent with the remainder of the dataset"* (Freeman et al., 1995), *"a data point that significantly differs from other data points or does not conform to expected normal behavior"*, or *"an observation that deviates so much from other observations that it arouses suspicion that it was generated by a different mechanism"* (Hawkins, 1980), or as an observation that *"fits a particular anomalous behavior well"* (Chandola et al., 2009).

Naturally, datasets created on the basis of such mathematically imprecise definitions and intended for testing outlier detection methods lack objectivity and, consequently, reliability. In this regard, the authors of the survey paper by Campos et al. (2016) quite rightly state that *"any comparison of outlier detection methods is inappropriate unless the validity of the outliers has been established."* The same authors further note that *"the lack of suitable benchmark datasets with ground-truth annotations constitutes a substantial obstacle to the evaluation of outlier detection methods. Even when labeled datasets are available, their suitability for the outlier detection task is usually unknown. Therefore, it is difficult to determine to what extent newly proposed outlier detection methods outperform existing ones."*

The authors of Boukerche et al. (2021) reached a similar conclusion. After conducting a comprehensive survey and analysis of existing outlier detection methods, they identified the lack of reliably labeled data as one of the most important factors hindering progress in the field.

Numerous software implementations of outlier detection methods are available in Python machine-learning libraries. However, nearly all of them depend on hyperparameter tuning. Hyperparameter optimization in unsupervised outlier detection faces fundamental difficulties: the absence of reliable labels precludes direct optimization, while the sensitivity of modern detectors to hyperparameters is extremely high (Campos et al., 2016). Optimization methods developed for this purpose, such as HPOD (Zhao and Akoglu, 2024) and HyPer (Ding et al., 2024), themselves depend on additional parameters and are not always practical. Moreover, a recent large-scale study demonstrated that existing hyperparameter tuning strategies are often impractical and frequently provide no advantage over default settings (Soenen et al., 2021). Therefore, to ensure maximum objectivity and reproducibility of comparisons, we follow the standard practice in this field and evaluate all competing methods using the default parameters recommended by the authors of the PyOD (Zhao et al., 2019) and scikit-learn (Pedregosa et al., 2011) libraries. Such an approach is widely accepted in benchmarks such as ADBench and has been repeatedly used in related studies.

As shown below, applying this approach in the present study makes it possible to demonstrate, using several simple datasets corresponding to a parabola, a paraboloid of revolution, and a hyperbolic paraboloid, that classical methods implemented in Python libraries and used with their default hyperparameters are unable to detect structural outliers, i.e., outliers that violate the

functional relationships of the domain while not being extreme either in terms of neighborhood distance or statistical distribution.

The present study is aimed at overcoming these limitations.

2 Definition of Structural Outliers

As noted in the Introduction, existing definitions of outliers lack mathematical rigor. As a result, when analyzing and labeling data, it is often ambiguous whether a particular observation should be considered an outlier or not. Let us attempt, as a working hypothesis, to refine the existing formulations in order to increase their mathematical precision.

We shall define a **structural outlier** as an observation that does not conform to the regularities governing the domain under consideration.

Alternatively, in terms of geometric interpretation, a structural outlier is a point in the feature space that does not belong to the hypersurface representing the regularity governing the domain, i.e., if the distance between this point and the hypersurface exceeds a predefined threshold ε_0.

As an example, Figure 1 shows a dataset consisting of points belonging to a domain governed by the relationship described by the parabola $y = (x - 7)^2$.

The dataset contains twenty points. Nineteen of them, shown in blue, belong to the parabola, whereas point No. 10, shown in red, does not belong to it.

The coordinates of the nineteen blue points satisfy the parabola equation with an error not exceeding a threshold ε_0 (for example, $\varepsilon_0 = 0.01$), whereas the coordinates of the red point do not satisfy the equation. For this point, $\varepsilon > \varepsilon_0$.

Therefore, according to our definition, point No. 10 is an structural outlier, while the remaining nineteen points are not.

Thus, the dataset shown in Figure 1 can be regarded as reliably labeled with respect to outliers and normal observations and it can therefore be used as a benchmark dataset for testing outlier detection methods.

An important feature of our definition of a structural outlier is that it does not necessarily have extreme feature values. Moreover, it can be located quite close to other observations and not be distinguished by local density or distance to neighbors. The only criterion for its anomalousness is its violation of the patterns that govern the majority of observations in the dataset. Such observations are henceforth referred to as structural outliers.

The proposed definition of a structural outlier makes it possible to create reliably labeled benchmark datasets and thereby obtain objective evaluations of outlier detection methods.

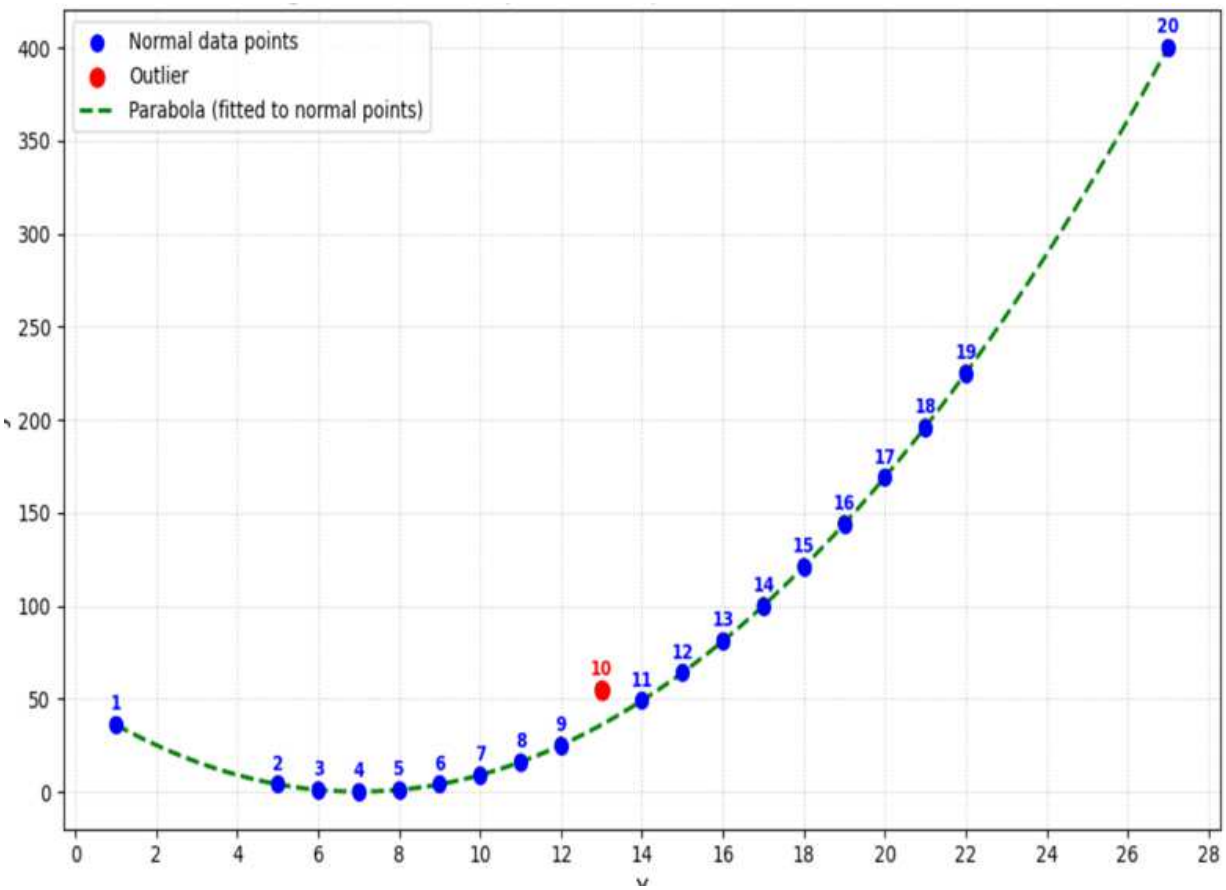


Figure 1. Dataset prepared for testing outlier detection methods. Point No. 10 does not belong to the parabola $y = (x - 7)^2$ whereas the remaining 19 points do belong to it. Therefore, point No. 10 is a reliably established structural outlier.

Table 1 presents the results obtained by applying fifteen outlier detection methods available in the PyOD and scikit-learn libraries to the proposed benchmark dataset:

• Methods using only input features: ABOD, HBOS, Isolation Forest, kNN, LOF, OCSVM, PCA, COPOD.

• Methods using the target variable: Random Forest (error), Neural Network (error), Autoencoder (reconstruction), Combined (RF+AE), One-Class SVM (with y), Isolation Forest (with y), and LOF (with y).

In our experimental comparison, we use the designation **"Neural Network (error)"** for the neural-network method based on prediction error. This is not the name of a specific method from the literature but rather a conventional identifier introduced for convenience in presenting the results. The algorithm is implemented using a multilayer perceptron (MLP) from the scikit-learn library. The trained model predicts the target variable, and observations with the largest mean squared errors (MSE) are classified as outliers. Similar approaches are widely used in anomaly detection and are known in the literature as methods based on neural-network prediction errors (Cherepanov and Yasnitsky, 2008; Chalapathy and Chawla, 2019; Pang et al., 2021).

In our experiments, all tested methods were asked to identify only one outlier in the dataset. Testing of the methods was carried out using the program:

https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_1_Testing_all_methods_on_the_Parabola_Dataset.ipynb.

As can be seen from Table 1, none of the fifteen tested methods succeeded in solving the task. The Precision, Recall, and F1 metrics were equal to zero, and none of the methods assigned rank 1 to the true outlier. The closest methods to the correct solution were Autoencoder (reconstruction) and Combined (RF+AE), which achieved ranks 2 and 3, respectively. These were followed by LOF (with y), Random Forest (error), ABOD (pyod), HBOS (pyod), and LOF (pyod), which achieved ranks 5, 7, 8, and 10, respectively.

Table 1 Quality metrics and ranks for detecting true outliers in the dataset of Figure 1. The lower the rank value, the closer the result of applying the method to the correct solution of the problem.

Method	Precision	Recall	F1	Rank
Autoencoder (reconstruction)	0	0	0	2
Combined (RF+AE)	0	0	0	3
LOF (with y)	0	0	0	5
Random Forest (error)	0	0	0	7
ABOD (pyod)	0	0	0	8
HBOS (pyod)	0	0	0	10
LOF (pyod)	0	0	0	10
IsolationForest (pyod)	0	0	0	12
kNN (pyod)	0	0	0	14
One-Class SVM (with y)	0	0	0	14
PCA (pyod)	0	0	0	16
Isolation Forest (with y)	0	0	0	16
COPOD (pyod)	0	0	0	18
OCSVM (pyod)	0	0	0	20
Neural Network (error)	0	0	0	20

The negative result of this simple computational experiment can be explained by the fact that many methods operate in the feature space without taking into account possible functional relationships among the variables.

In the dataset shown in Figure 1, nineteen points follow the quadratic relationship $y = (x - 7)^2$ whereas point No. 10 violates this relationship, having coordinates $x = 13$ and $y = 55$ which are not extreme with respect to either x or y individually.

Therefore, in the two-dimensional feature space (x, y), the outlier is not distinguishable by distance to neighboring points, local density, or statistical properties of the distribution. Consequently, classical methods based on statistical distributions and distance measures perceive as anomalies either boundary points or isolated points in the feature space. They are unable to capture violations of the structural relationship caused by the true outlier.

The goal of the present study is to propose and evaluate an outlier detection method capable of capturing such structural violations and successfully solving this type of problem without any hyperparameter tuning.

3 Proposed Methods

3.1 Neural Network Filter (NNF)

The idea of the proposed method originated from the analysis of neural network training errors. It was previously observed that some neural networks produce increased training errors on observations that do not conform to the regularities of the domain under study (Cherepanov and Yasnitsky, 2008; Yasnitsky, 2020; Beliakov et al., 2011; Ferdowsi et al., 2013).

Suppose that a neural network is trained on a dataset in which the majority of observations follow a certain functional relationship. If a structural outlier is present in the dataset, it is often more advantageous for the neural network to focus its attention on the majority of normal observations than to minimize the training error on the outlier. As a consequence, the training error corresponding to the outlier may become significantly larger than the errors associated with the remaining observations.

The Neural Network Filter (NNF) method is based on this hypothesis.

The algorithm of the method consists of the following steps:

1. Train a neural network on the dataset under analysis.
2. Compute the training errors for all observations.
3. Sort the observations according to the magnitude of their errors.
4. Consider the observations with the largest errors as candidates for structural outliers.

3.2 Dataset Governed by the Equation of a Parabola

As is well known, neural networks construct hypersurfaces during the training process (or curves in the two-dimensional case) that approximate the points of the modeled domain in the feature space. In the upper row of Figure 2, the green curves represent such approximations of the domain points shown in Figure 1. These approximations were obtained using three simple feedforward neural networks.

The networks have one input neuron, one output neuron, and one hidden layer containing N neurons. The first neural network has $N = 2$ hidden neurons, the second has $N = 8$, and the third has $N = 13$. The input and output neurons use linear activation functions, whereas the hidden neurons use hyperbolic tangent activation functions.

The neural networks were trained on the dataset shown in Figure 1 using the RPROP algorithm. The number of training epochs was 1000. In the lower row of Figure 2, the bars represent the training errors obtained at the corresponding points of the domain.

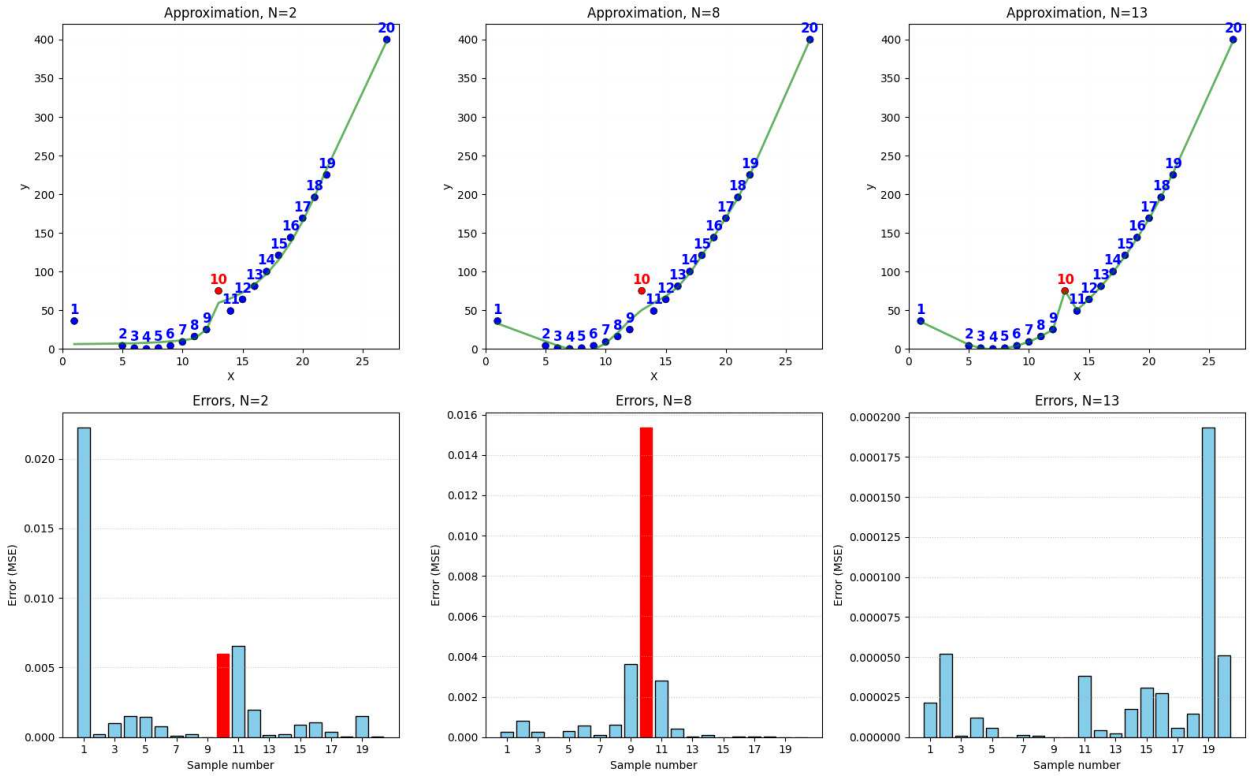


Figure 2. The upper row shows approximations of the domain points from Figure 1 obtained using neural networks with 2, 8, and 13 hidden neurons. The lower row shows the corresponding training errors of these neural networks.

As can be seen from Figure 2, only one of the three neural networks—the network containing eight hidden neurons—produced the maximum training error at point No. 10, which is the true outlier. Thus, unlike the fifteen standard outlier detection methods, the NNF method correctly identified the outlier when $N = 8$. Its Precision, Recall, and F1 metrics were equal to one.

For the other neural networks, containing 2 and 13 hidden neurons, the maximum training errors corresponded to points that were not outliers. This means that when $N = 8$ the neural network is suitable for outlier detection, whereas when $N = 2$ or $N = 13$ it is not suitable for this purpose.

The code of the program that obtained the results:
https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_2_Geometric_interpretation_of_NNF.ipynb

Further experiments are presented in Figure 3. They show that when the number of hidden neurons N varies from 0 to 80, the ability of neural networks to detect outliers appears and disappears repeatedly. In particular, Figure 3 demonstrates that:

- for $N \in [0, 2]$, neural networks are unsuitable for outlier detection;
- for $N \in [3, 11$, they are suitable;
- for $N \in [12, 14$, they become unsuitable again;
- for $N \in [15, 18$, they are again suitable;
- and so forth.

When $N > 61$ neural networks become consistently unsuitable for detecting outliers in the considered dataset.

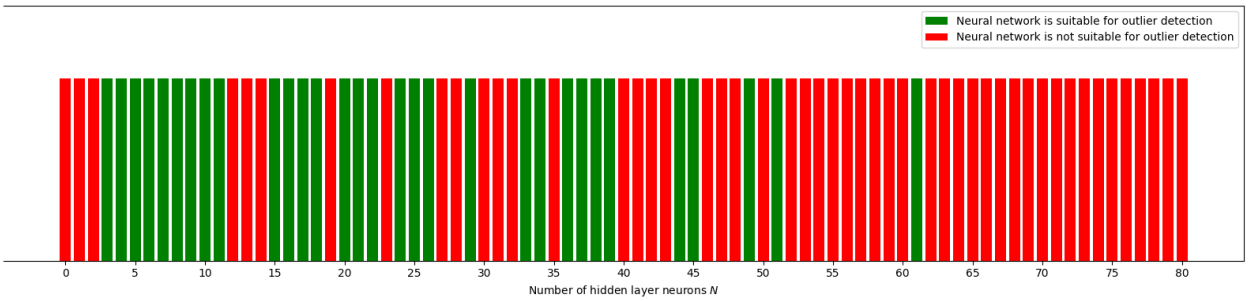


Figure 3. Ability of neural networks to detect the outlier in the dataset of Figure 1 as a function of the number of hidden neurons *N*. Green indicates values of *N* for which the neural network successfully detects the outlier, whereas red indicates values for which it fails to detect the outlier.

Thus, the question of which neural network architectures are suitable for use as outlier detectors on a particular dataset is not trivial. Therefore, the following sections are devoted to a further investigation of this issue with the goal of developing a universal algorithm for designing neural network architectures suitable for outlier detection.

3.3 Dataset Governed by the Equation of a Paraboloid of Revolution

Figure 4 presents an example of a dataset consisting of points belonging to a domain governed by the equation of a paraboloid of revolution $y = x_1^2 + x_2^2$**.** The dataset contains 45 points. Forty-four of them, shown in blue, are randomly distributed over the surface of the paraboloid, whereas point No. 24, shown in red, does not belong to the paraboloid surface. The coordinates of the 44 blue points satisfy the paraboloid equation with an error not exceeding, for example, $\varepsilon_0 = 0.01$**,** whereas the coordinates of the red point **(**$x_1 = 0,\ x_2 = 0,\ y = 0.32$**)** do not satisfy this equation. Its distance from the paraboloid surface is $\varepsilon = 0.32$**,** which exceeds the threshold value ε_0. Therefore, according to our definition of a structural outlier, point No. 24 is a structural outlier, while the remaining 44 points are not outliers.

Thus, the dataset shown in Figure 4 can be regarded as reliably labeled with respect to structural outliers and normal observations and can therefore be used for testing outlier detection methods.

As expected, none of the fifteen popular outlier detection methods passed the test. The true outlier was detected only by the NNF method when the number of hidden neurons was set to $N = 5$.

As shown in Figure 5, which presents the distribution of training MSE errors across the observations of the dataset, the largest training error corresponds to observation No. 24, which is indeed the true outlier.

This encouraging result was obtained using the corresponding program implementation: https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_4_5_Testing_on_a_paraboloid_of_revolution.ipynb

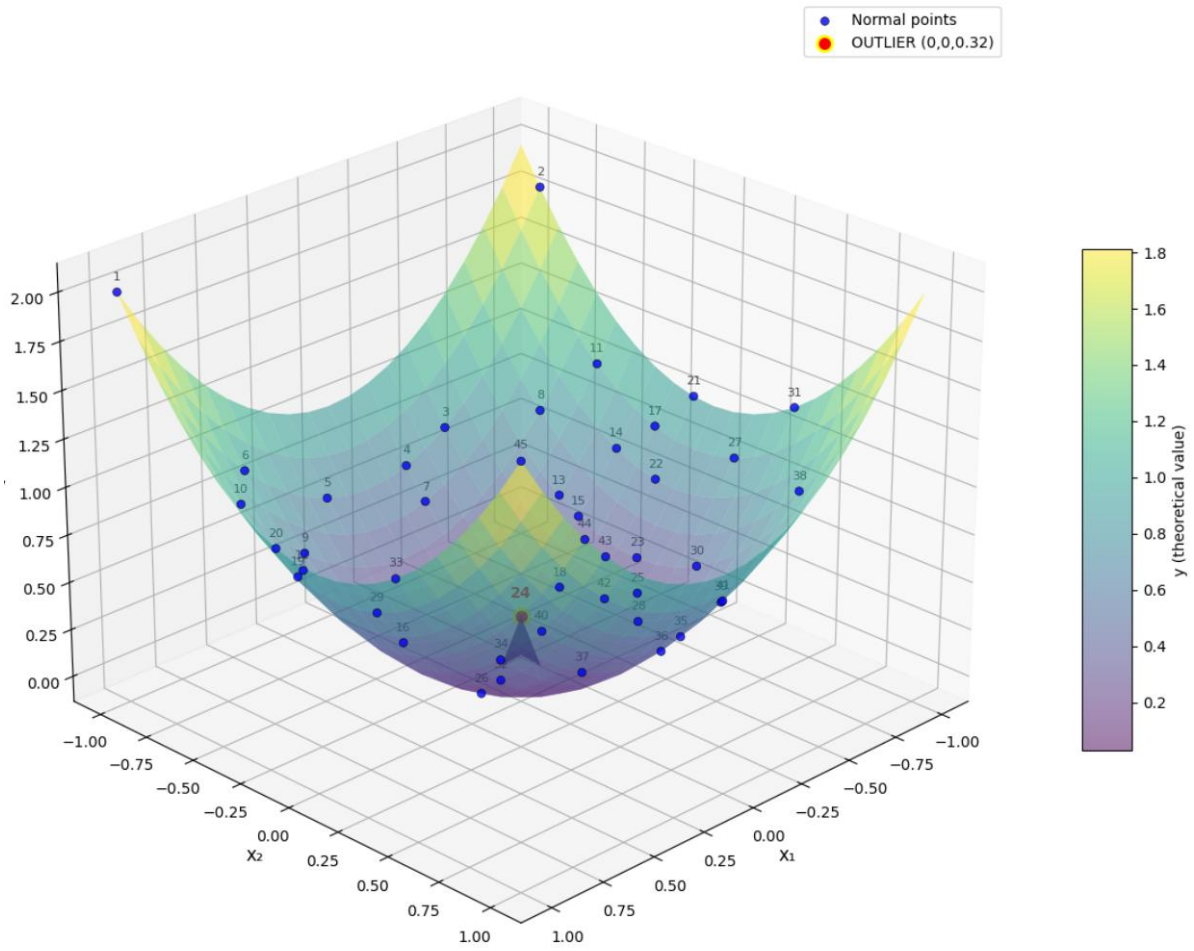


Figure 4. Dataset prepared for testing outlier detection methods. Point No. 24, having coordinates $x_1 = 0,\ x_2 = 0,\ y = 0.32$**,** is an structural outlier, whereas the remaining 44 points are not outliers because they belong to the surface of the paraboloid $y = x_1^2 + x_2^2$**.**

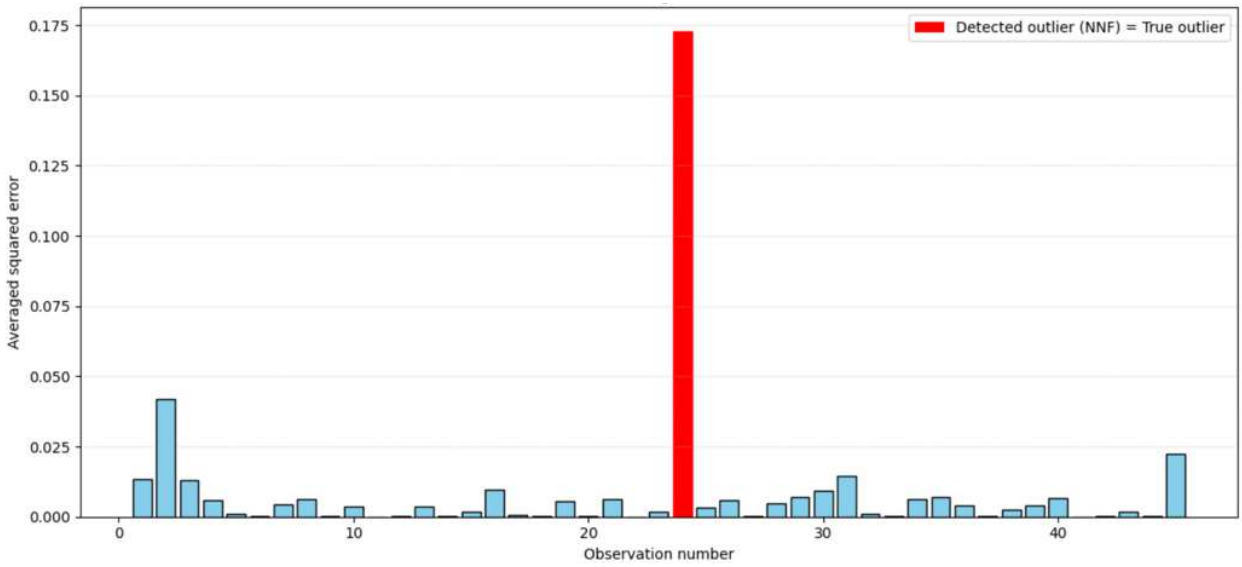


Figure 5. Distribution of neural-network training errors across the observations of the test dataset shown in Figure 4. The number of hidden neurons is $N = 5$**.** The largest error is concentrated on observation No. 24, which is the true outlier.

The results of comparing the effectiveness of the NNF method with existing outlier detection methods, obtained using the same program, are presented in Table 2.

As can be seen from Table 2, only the NNF method correctly identified the true outlier. Only this method achieved values of one for Precision, Recall, F1, and Rank.

Among the existing outlier detection methods, the closest to a successful result was the **Neural Network (error)** method, which achieved a rank of 2. All remaining methods demonstrated substantially poorer performance.

Table 2 Comparison of the performance of the new outlier detection method with existing ones using the example data set in Figure 4

Method	Precision	Recall	F1	Rank
NNF	1.0	1.0	1.0	1
Neural Network (error)	0.0	0.0	0.0	2
Autoencoder (reconstruction)	0.0	0.0	0.0	7
Combined (RF+AE)	0.0	0.0	0.0	10
ABOD (pyod)	0.0	0.0	0.0	11
Random Forest (error)	0.0	0.0	0.0	24
IsolationForest (pyod)	0.0	0.0	0.0	34
Isolation Forest (with y)	0.0	0.0	0.0	34
kNN (pyod)	0.0	0.0	0.0	37
COPOD (pyod)	0.0	0.0	0.0	37
HBOS (pyod)	0.0	0.0	0.0	42
LOF (pyod)	0.0	0.0	0.0	43
PCA (pyod)	0.0	0.0	0.0	43
LOF (with y)	0.0	0.0	0.0	44
OCSVM (pyod)	0.0	0.0	0.0	45
One-Class SVM (with y)	0.0	0.0	0.0	45

Thus, the experiment with the paraboloid of revolution confirmed the results previously obtained for the parabola.

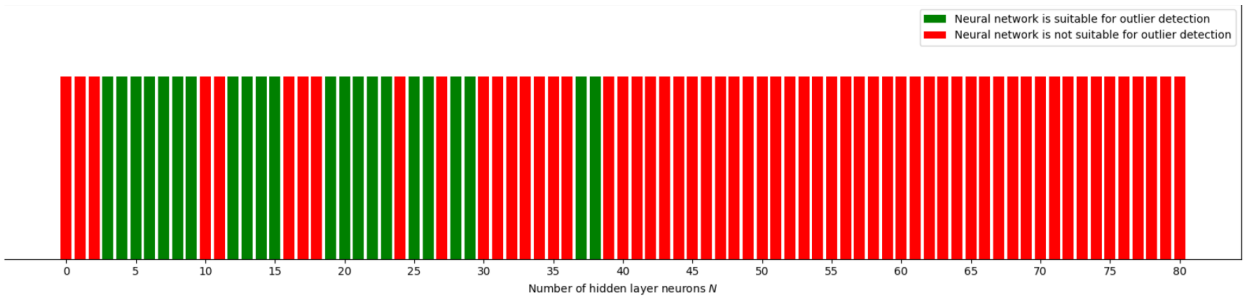


Figure 6. Ability of neural networks to detect the outlier in the dataset shown in Figure 4 (paraboloid of revolution) as a function of the number of hidden neurons N. Green indicates values of N for which the neural network successfully detects the outlier, whereas red indicates values for which it fails to detect the outlier.

However, as already noted, the success of the NNF method depends on the architecture of the neural network being used. Indeed, as shown in Figure 6, other values of the number of hidden

neurons *N*, such as 0, 1, 2, 10, 11, 16, 17, 18, and others, lead to unsatisfactory testing results for the NNF method.

This means that the problem of selecting the number of hidden neurons remains unresolved.

It is precisely this problem that becomes the central task of the subsequent investigation.

3.4 Dataset Governed by the Equation of a Hyperbolic Paraboloid

To verify the universality of the obtained result, a third test dataset was used. Figure 7 presents an example of a dataset in the form of points of a domain governed by the equation of a hyperbolic paraboloid $y = x_1^2 - x_2^2$. The dataset consists of 45 points, 44 of which are randomly scattered over the surface of the hyperbolic paraboloid, while point number 24, shown in red, does not belong to the paraboloid surface. The coordinates of the 44 points shown in blue satisfy the equation of the specified paraboloid with an error not exceeding $\varepsilon_0 = 0.01$, whereas the coordinates of the point shown in red ($x_1 = 0,\ x_2 = 0,\ y = 0.31$) do not satisfy this equation. Its distance from the paraboloid surface, $\varepsilon = 0.31$, exceeds the threshold value ε_0. Therefore, according to our definition, point number 24 is a reliably established structural outlier.

Thus, the dataset shown in Figure 7 may be considered reliably labeled and can be used for testing outlier detection methods.

The results of comparing the effectiveness of outlier detection methods, obtained using the program
https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_7_Testing_on_a_hyperbolic_paraboloid.ipynb, are presented in Table 3.

As can be seen from Table 3, only the NNF method, as before, achieved a rank equal to one and also obtained the best quality metrics Precision, Recall, and F1. Among the existing methods, the closest to obtaining a positive result this time were the methods **Random Forest (error)** and **Autoencoder (reconstruction)**, both of which achieved a rank of 3, as well as the **Combined (RF+AE)** method, which achieved a rank of 4. The **Neural Network (error)** method ranked sixth. The ranks of all remaining methods exceeded 10.

It should be noted, however, that the program
https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_8_Suitability_of_NNF_for_a_hyperbolic_paraboloid.ipynb,
which implements the NNF method, correctly identified the true outlier only when the hyperparameter *N* was selected from the set: **2, 3, 4, 5, 7, 8, 9, ...** as shown in green in Figure 8. The values of the hyperparameter *N* for which the application of the NNF method did not produce a positive result are shown in red.

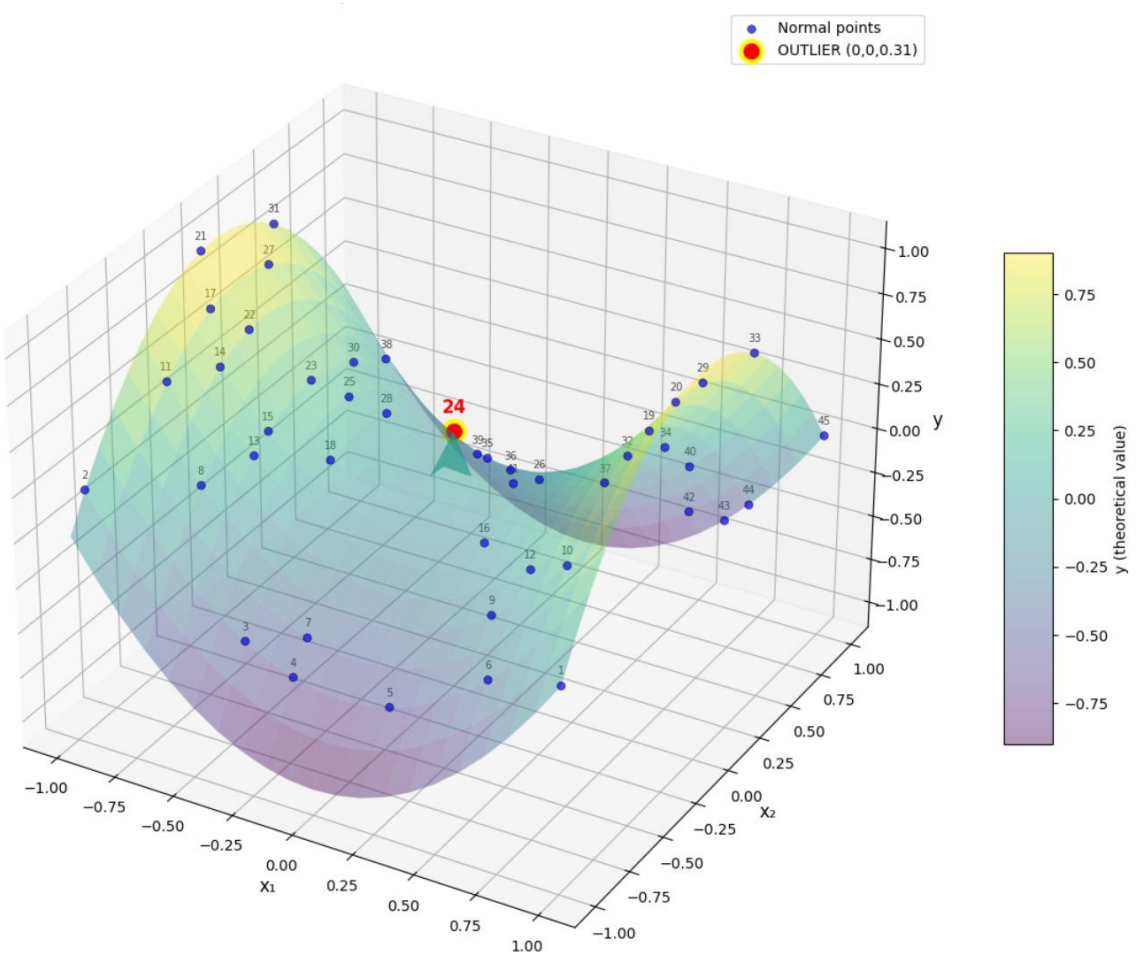


Figure 7. Dataset intended for testing outlier detection methods. Point number 24, having coordinates $x_1 = 0,\ x_2 = 0,\ y = 0.31$**,** is an outlier, whereas the remaining 44 points are not outliers because they belong to the surface of the hyperbolic paraboloid $y = x_1^2 - x_2^2$**.**

Table 3. Comparison of the effectiveness of outlier detection methods using the dataset shown in Figure 7

Method	Precision	Recall	F1	Rank
NNF	1	1	1	1
Random Forest (error)	0	0	0	3
Autoencoder (reconstruction)	0	0	0	3
Combined (RF+AE)	0	0	0	4
Neural Network (error)	0	0	0	6
ABOD (pyod)	0	0	0	12
One-Class SVM (with y)	0	0	0	24
kNN (pyod)	0	0	0	34
COPOD (pyod)	0	0	0	35
IsolationForest (pyod)	0	0	0	35
Isolation Forest (with y)	0	0	0	35
OCSVM (pyod)	0	0	0	44
LOF (pyod)	0	0	0	44
PCA (pyod)	0	0	0	44
LOF (with y)	0	0	0	44
HBOS (pyod)	0	0	0	45

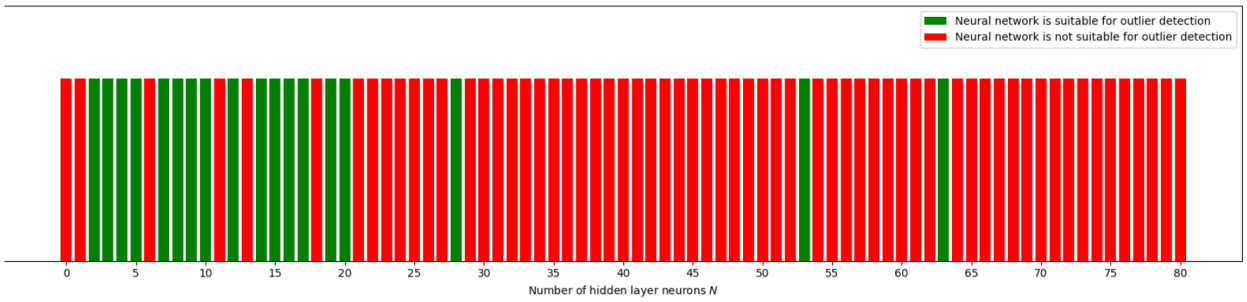


Figure 8. Ability of neural networks to detect the outlier in the dataset shown in Figure 7 (hyperbolic paraboloid) depending on the number of hidden-layer neurons N. Values of N for which the neural network is capable of detecting the dataset outlier are shown in green; values for which it is incapable of detecting the outlier are shown in red.

3.5 Generalization of the Results

Thus, unlike the well-known outlier detection methods, the proposed NNF method successfully passed all three tests. However, as shown by the histograms in Figures 3, 6, and 8, its successful application depends on the choice of the hyperparameter N. Comparing these histograms, one can observe that they exhibit a qualitatively similar pattern. It can be seen that as the number of hidden-layer neurons N increases, the probability that the neural network will be unsuitable for outlier detection gradually increases and becomes 100% when $N > 63$.

It can also be observed that some values of the hyperparameter N lead to successful testing results of the NNF method on one dataset, whereas the same values of N do not lead to positive results on another dataset. Figure 9 presents a histogram obtained by intersecting the three histograms shown in Figures 3, 6, and 8. The green color marks only those values of the hyperparameter N that ensure successful testing of the NNF method for all three datasets. As can be seen from Figure 9, there are very few such values of N. They are: **3, 4, 5, 7, 8, 9, 15, 20.**

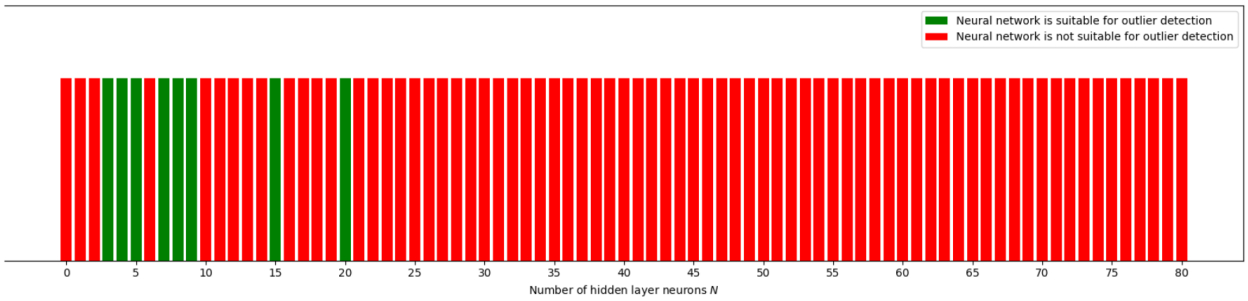


Figure 9. Histogram obtained by intersecting the histograms shown in Figures 3, 6, and 8. The green color marks the values of the hyperparameter N that ensure successful completion of all three tests

It is not difficult to guess that if we continue testing the NNF method on other datasets, the number of such universal values of the hyperparameter N, suitable for all possible cases, will become even smaller and may even disappear entirely. Thus, while noting the positive aspects of the NNF method, we are forced to acknowledge its drawback, namely the absence of a methodology for determining a universal value of the hyperparameter N that would ensure successful application of the method to arbitrary datasets.

3.5.1 Neural Network Filter Aggregation (NNFA)

The following idea makes it possible to solve this problem and formulate a universal rule for designing neural networks suitable for detecting outliers in a wide range of datasets.

The idea is to organize a loop over N (for example, from 3 to 30), accumulate the training errors for each observation by summing over N, and then compute their arithmetic mean values over N for each observation.

Naturally, one may expect that the larger the training error averaged over N for a particular observation, the more likely this observation is to be an outlier.

The results of the numerical implementation of this idea using the hyperbolic paraboloid dataset, obtained with the program https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_10_Training_errors_for_different_N.ipynb, are presented in Figures 10 and 11.

Figure 10 shows the distribution of neural-network training errors (MSE) across the observations of the dataset for networks having from $N = 3$ to $N = 30$ neurons in the hidden layer.

It can be seen that for some values of the hyperparameter N, for example $N = 6, 27, 29, 30$, the largest MSE values are concentrated not on the observation corresponding to the true outlier.

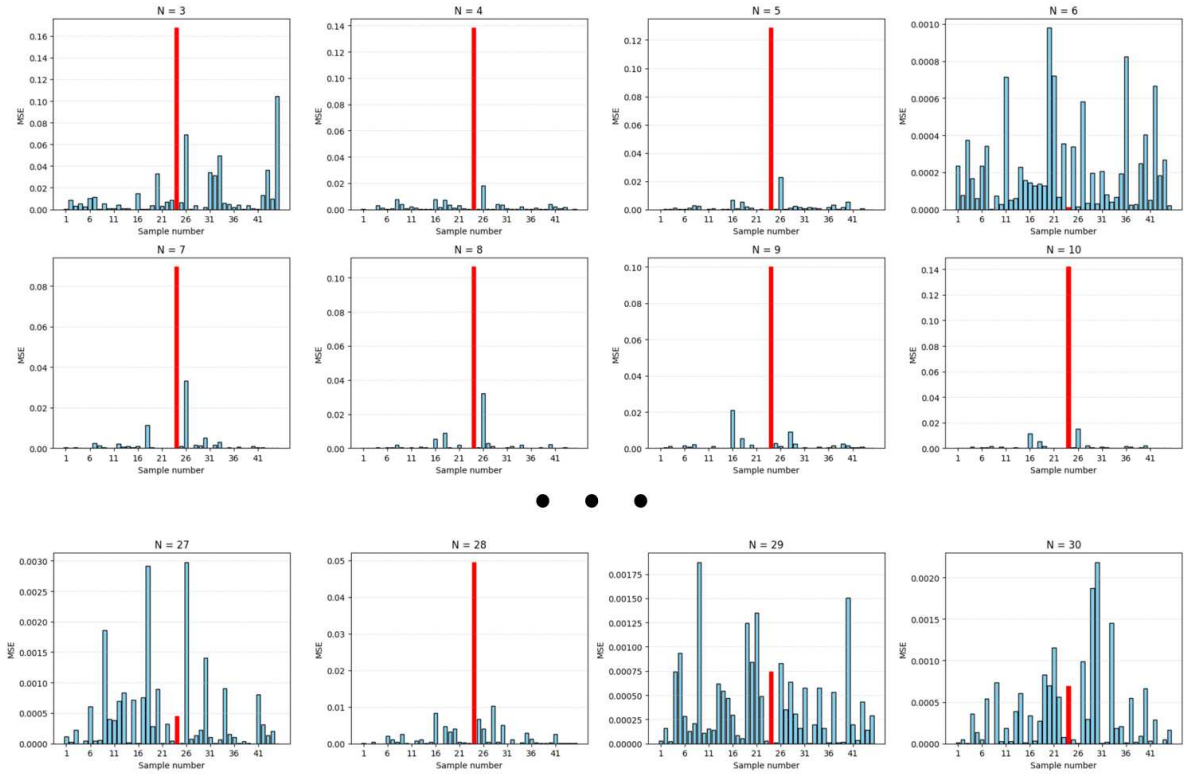


Figure 10. Distribution of neural-network training errors across the observations of the dataset for networks having from $N = 3$ to $N = 30$ neurons in the hidden layer. For $N = 6, \ldots, 27, 29, 30$, the neural network is unable to detect outliers.

Figure 11 presents the result of comparing the accumulated and averaged MSE errors over N (N=3, …, 30). As can be seen from Figure 11, the largest value of the averaged MSE errors corresponds

to observation 24, which is the true outlier. Thus, despite the fact that for individual values of the hyperparameter N the NNF method produces an incorrect recognition result for the true outlier, the procedure of accumulating and averaging these errors over a specified interval of N values makes it possible to correctly identify the true outlier.

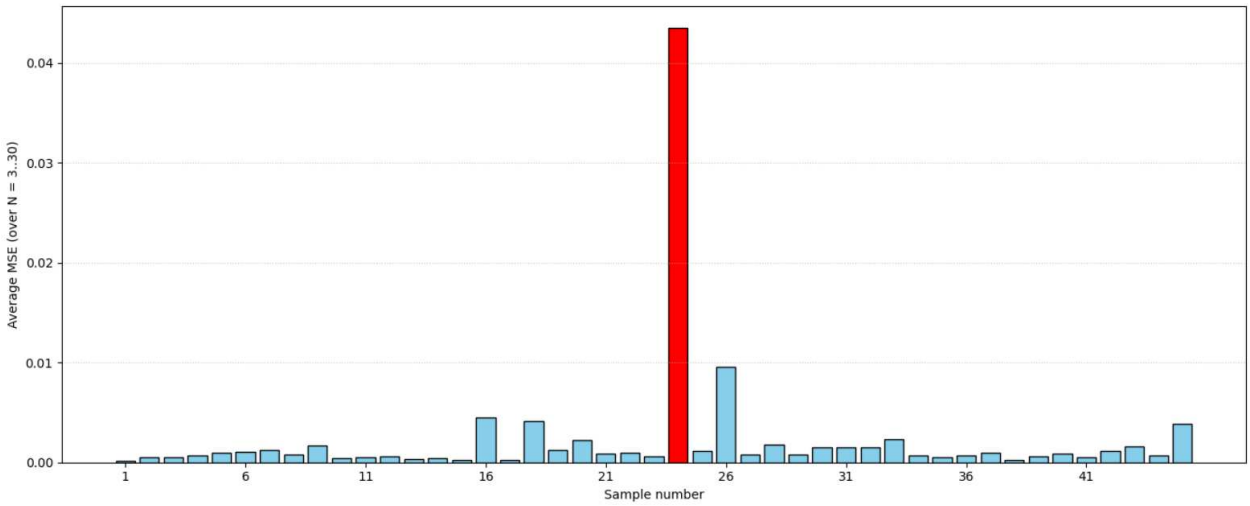


Figure 11. Distribution of MSE errors averaged over $N = 3,,30$ across the observations. The largest averaged MSE corresponds to observation 24, which is the true outlier. The ensemble of neural networks from the considered interval suppresses the negative influence of the "bad" neural networks during averaging.

The positive effect of the error accumulation and averaging procedure can be explained as follows. Figure 12 shows the same results as Figure 10, obtained using the program https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_12_Training_errors_for_different_N_on_the_same_scale_along_the_vertical_axis.ipynb but displayed using the same scale on the vertical axis. It can be clearly seen that the training errors of neural networks that are unsuitable for outlier detection (namely the networks with $N = 6, \ldots, 27, 29, 30$) are substantially smaller than the training errors of neural networks that are suitable for outlier detection (namely the networks with $N = 3, 4, 5, 7, 8.9, 10$). We shall refer to the networks unsuitable for outlier detection as **"bad" networks**. These networks simply turned out to be overfitted. Their training errors decreased almost to zero both for normal observations and for outlier observations, which made them unsuitable for outlier detection. Therefore, the contribution of the errors produced by the "bad" neural networks to the total sum of errors, and consequently to their averaged values, is always smaller than the contribution of the errors produced by the "good" neural networks.

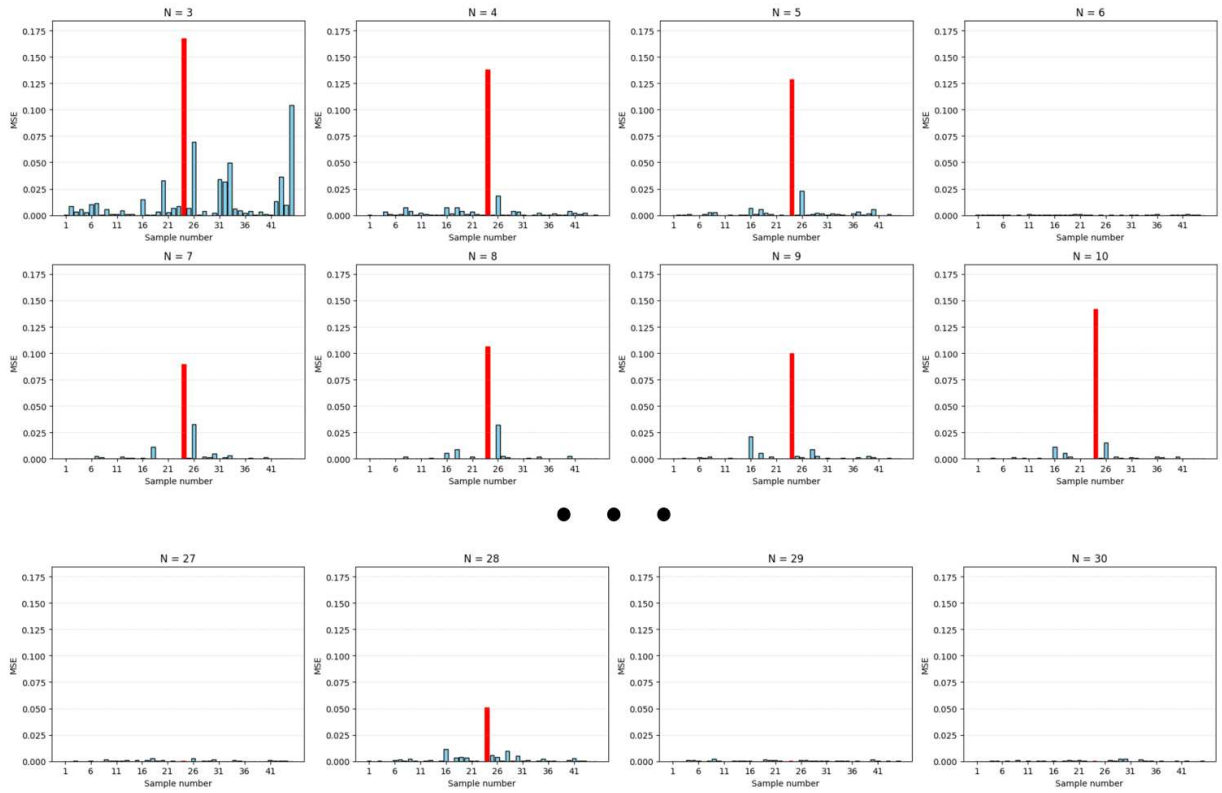


Figure 12. The results of Figure 10, plotted on a single scale along the vertical axis. The training errors of neural networks that incorrectly detect outliers ($N = 6, \dots, 27, 29, 30$) are vanishingly small compared to the errors of neural networks that correctly detect the true outlier.

Thus, the procedure of accumulation for each observation followed by averaging over *N* training errors eliminates the need to select an appropriate value for the hyperparameter *N* to ensure the success of the NNF method. From here on, we will refer to this modification of the NNF method as the NNFA (Neural Network Filter with Averaging) method. While the success of the NNF method depends on the appropriate choice of the hyperparameter N, the NNFA method does not require selecting N. Instead, it requires selecting an interval of *N* values over which to accumulate and average errors. Essentially, in the NNFA method, instead of the hyperparameter N, we obtained a new hyperparameter—an interval of *N* values over which to accumulate and average errors. In our further research, the question of choosing this new hyperparameter is resolved using a corollary to the Arnold–Kolmogorov–Hecht-Nielsen theorem.

3.5.2 Use of the Corollary of the Arnold – Kolmogorov – Hecht–Nielsen Theorem

While addressing the problem of selecting the averaging interval *N*, let us once again turn our attention to Figures 10 and 12. These figures show that, compared with the "good" neural networks, the "bad" neural networks have vanishingly small training errors. This indicates that the "bad" networks exhibit the effect of overfitting.

In this connection, it is appropriate to make use of the well-known corollary of the Arnold–Kolmogorov–Hecht–Nielsen theorem (Yasnitsky, 2016), which regulates the number of synaptic weights N_w in simple feedforward neural networks with sigmoid activation functions (including the hyperbolic tangent):

$$\frac{N_y}{1+log_2(Q)} \le N_w \le N_y\left(\frac{Q}{N_x}+1\right)\left(N_x+N_y+1\right)+N_y\ . \tag{1}$$

Here: N_x is the number of neurons in the input layer; N_y is the number of neurons in the output layer; Q is the number of observations in the dataset.

According to this corollary, when the value of N_w exceeds the upper bound of the interval specified by Formula (1), there is a high probability that the neural network will become overfitted. Conversely, when N_w falls below the lower bound of the interval, the probability of underfitting increases.

If the neural network has only one hidden layer, then the corresponding number of hidden neurons can be determined by the following obvious formula:

$$N = \frac{N_w}{N_x+ N_y}\ . \tag{2}$$

Expressing N_w from Formula (2), substituting it into Formula (1), and dividing both sides of the inequality by $(N_x + N_y)$, we obtain a formula regulating the number of hidden neurons in a neural network having a single hidden layer:

$$N_{min} \le N \le N_{max}\ , \tag{3}$$

where

$$N_{min} = \frac{N_y Q}{(1+log_2(Q))(N_x+N_y)}\ , \tag{4}$$

$$N_{max} = \frac{N_y}{N_x+N_y}\left(\left(\frac{Q}{N_x}+1\right)\left(N_x+N_y+1\right)+1\right). \tag{5}$$

Thus, Formulas (4) and (5) make it possible to determine the interval of admissible numbers of hidden neurons for a neural network intended to solve a given problem.

For example, applying Formulas (4) and (5) to the parabola dataset, for which $Q = 20,\ N_x = 1,$ $N_y = 1$, yields: $N_{min} = 1.88,\ N_{max} = 32.00$.

Applying the same formulas to the datasets corresponding to the paraboloid of revolution and the hyperbolic paraboloid, for which $Q = 45,\ N_x = 2,\ N_y = 1$, yields: $N_{min} = 2.31,\ N_{max} = 31.67$.

Comparing these values of N_{min} and N_{max} with the results shown in Figures 6, 8, and 9, it appears logical to use the interior part of the Arnold–Kolmogorov–Hecht–Nielsen interval as the desired averaging interval, namely

$$N \in (N_{min},\ N_{max}). \tag{6}$$

Thus, we arrive at the following theoretical justification of the NNFA method.

Within the Arnold–Kolmogorov–Hecht–Nielsen (AKHN) interval, a neural network possesses sufficient but not excessive capacity.

If the network is too simple, underfitting occurs. In this case, the errors become large for almost all observations and cease to carry useful information about outliers.

If the network is excessively complex, overfitting occurs. In this case, the network begins to adapt itself to outliers, causing the errors on the outliers to decrease and cease to stand out from the errors of the remaining observations.

Within the AKHN interval, a compromise between these two extremes is observed. For this reason, structural outliers form local maxima of training errors.

Based on this reasoning, the following hypothesis was proposed: when an ensemble of neural networks whose architectures belong to the AKHN interval is used, the need to tune the number of hidden neurons may be eliminated. Successful application of the NNFA method will be achieved if Formulas (4), (5), and (6) are used when accumulating and averaging training errors.

The verification of this hypothesis on real benchmark datasets is the subject of the following sections of this study.

At this point, it can already be noted that positive results were obtained on the benchmark datasets corresponding to the parabola (Figure 1), the paraboloid of revolution (Figure 4), and the hyperbolic paraboloid (Figure 7).

In our opinion, this result is quite significant, considering that, as shown above, none of the fifteen well-known outlier detection methods was able to pass these tests.

3.5.3 Parameter ξ

A possible drawback of the NNFA method is its high computational cost.

It can be reduced by decreasing the upper bound of the interval of values of parameter *N*, for example, by replacing Formula (3) with the following formula:

$$N \in (N_{min},\ N_{lim}), \tag{7}$$

Where $N_{lim} = N_{min} + \xi(N_{max} - N_{min})$ and $\xi \in [0, 1]$.

It is obvious that when $\xi = 0$, the NNFA method degenerates into the NNF method.

In the subsequent experiments, the influence of this parameter on the quality of structural outlier detection will be investigated.

4 Testing on Real-World Datasets

In the previous sections, outlier detection methods were tested on artificial datasets representing domains that strictly obey the mathematical equations of a parabola, a paraboloid of revolution, and a hyperbolic paraboloid. Outliers in these artificial datasets were created in such a way that their coordinates did not satisfy the corresponding equations of the domains, i.e., the inequality $\varepsilon > \varepsilon_0$ held for them. Therefore, according to the mathematical definition of an outlier formulated in Section 2, the correctness of labeling observations as outliers and normal observations could be considered reliable. In the real world, such reliable labeling of data into outliers and normal observations presents certain difficulties, since the regularities governing real domains are usually

unknown. In this regard, a number of authoritative authors, for example, Campos et al. (2016), quite rightly point out the "lack of reliably labeled datasets, which constitutes a substantial obstacle for the evaluation of outlier detection methods."

Our idea for creating reliable outliers is as follows. It is proposed to select from the analyzed real-world dataset an observation (or several observations) characterized, for example, by the largest and smallest values of the output variable in the dataset, and replace these values with the corresponding mean values of the same variable.

It is obvious that observations modified in this way will, with high probability, violate the regularities governing the domain under study. This means that the point representing such an observation in the feature hyperspace will not lie on the mentally imagined hypersurface that geometrically represents this regularity, which corresponds to our definition of an outlier formulated in Section 2.

This idea was used for generating artificial outliers in the subsequent experiments based on real-world datasets.

4.1 Concrete Compressive Strength Data Set

For the first series of experiments, the Concrete Compressive Strength Data Set presented by Yeh (1998) was used. The dataset contains 1030 observations, eight input features, and one output variable—the compressive strength of concrete.

4.1.1 One Outlier Case

Previously, when dealing with artificial datasets, we relied on the definition given at the beginning of Section 2 when creating outliers. In the present case, however, the hypersurface approximating the points of the domain is unknown. It can only be imagined mentally.

To begin with, let us modify this dataset as follows. We select one observation characterized by the largest compressive-strength value in the dataset and replace this value with the mean compressive-strength value of the dataset, assigning this observation the status of an "Outlier." The remaining 1029 observations are assigned the status of "Normal observation."

Obviously, we obtain a controllably labeled dataset containing an outlier that, with high probability, violates the regularity of the domain and therefore does not lie on the mentally imagined hypersurface that geometrically represents this regularity.

The results of testing various outlier detection methods on the resulting benchmark dataset, obtained using the program https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!13_Testing_on_the_Concrete_Strength_Dataset(One_Outlier).ipynb are presented in Table 4. All methods were tested under the requirement of detecting exactly one outlier.

To ensure statistical reliability, all experiments were repeated 30 times with different values of the random seed. The comparative analysis included 15 standard methods (from the PyOD and scikit-learn libraries), as well as the NNFA method with four values of parameter ξ: $\xi = 0$, 0.01, 0.5, and 1. The quality assessment was performed using the F1 metric (for the top-1 prediction) and ROC-AUC. The obtained results are presented in Table 4.

Table 4. Results of testing outlier detection methods on the Concrete Compressive Strength dataset containing one outlier

Method	F1 (mean ± std)	ROC-AUC (mean ± std)
NNFA (ξ=0)	1.000 ± 0.000	1.000 ± 0.000
NNFA (ξ=0.01)	1.000 ± 0.000	1.000 ± 0.000
NNFA (ξ=0.5)	1.000 ± 0.000	1.000 ± 0.000
NNFA (ξ=1)	1.000 ± 0.000	1.000 ± 0.000
Neural Network (error)	0.800 ± 0.400	1.000 ± 0.001
Random Forest (error)	0.000 ± 0.000	0.999 ± 0.000
Combined (RF+AE)	0.000 ± 0.000	0.986 ± 0.010
LOF (pyod)	0.000 ± 0.000	0.986 ± 0.000
COPOD (pyod)	0.000 ± 0.000	0.927 ± 0.000
Isolation Forest (with y)	0.000 ± 0.000	0.922 ± 0.036
IsolationForest (pyod)	0.000 ± 0.000	0.919 ± 0.028
OCSVM (pyod)	0.000 ± 0.000	0.911 ± 0.000
LOF (with y)	0.000 ± 0.000	0.904 ± 0.000
PCA (pyod)	0.000 ± 0.000	0.892 ± 0.000
HBOS (pyod)	0.000 ± 0.000	0.887 ± 0.000
kNN (pyod)	0.000 ± 0.000	0.871 ± 0.000
ABOD (pyod)	0.000 ± 0.000	0.845 ± 0.000
Autoencoder (reconstruction)	0.000 ± 0.000	0.806 ± 0.087
One-Class SVM (with y)	0.000 ± 0.000	0.402 ± 0.295

As follows from the presented data, all configurations of the NNFA method achieved the maximum possible quality values: F1 = 1.000 ± 0.000 and ROC-AUC = 1.000 ± 0.000. This means that in all 30 runs the true outlier was identified without error, and the anomaly scores ranked it above all other observations. The zero standard deviation confirms the absolute stability of the method regardless of random initialization.

The best among the standard methods was Neural Network (error). Its mean value was F1 = 0.800 ± 0.400 and ROC-AUC = 1.000 ± 0.001. The large standard deviation of F1 indicates a strong dependence on random initialization: in some runs the outlier was detected (F1 = 1), while in others it was not detected (F1 = 0). The ROC-AUC value close to 1.0 indicates correct anomaly ranking, but under strict top-1 selection this does not always lead to successful detection of the true outlier.

The remaining 14 standard methods (including Random Forest (error), ABOD, HBOS, kNN, LOF, PCA, COPOD, OCSVM, Isolation Forest, Autoencoder, One-Class SVM (with y), Isolation Forest (with y), LOF (with y), and Combined (RF+AE)) demonstrated F1 = 0.000 ± 0.000 in all runs. However, their ROC-AUC values were high, ranging from 0.806 to 0.999 (with the exception of One-Class SVM (with y), which achieved 0.402 ± 0.295). This means that these methods correctly rank the outlier by assigning it a high anomaly score, but not the highest one. Since the target outlier is unique, successful detection required that this observation receive the highest rank. None of these methods was able to satisfy this condition consistently.

The paired Wilcoxon signed-rank test showed that all NNFA variants statistically significantly outperform the best standard method, Neural Network (error) ($p < 0.05$). The differences among the NNFA variants themselves ($\xi = 0, 0.01, 0.5, 1$) are not statistically significant ($p = 1.0$), which confirms that there is no need to tune the hyperparameter ξ.

Thus, the conducted computational experiment demonstrated that the proposed NNFA method is the only one among the 16 considered methods that reliably and consistently detects a structural outlier.

The high standard deviation of the F1 metric obtained by Neural Network (error) makes this method unsuitable for practical applications in the absence of labeled data, where reproducible results are required. Most classical methods, despite providing good ranking quality, are unable to identify the outlier as the top-1 observation, which makes them useless in rare-anomaly detection tasks. The obtained results convincingly demonstrate the advantage of the approach based on averaging neural-network training errors over the Arnold–Kolmogorov–Hecht–Nielsen interval.

4.1.2 Multiple Outliers Case

For a more challenging evaluation of the proposed method, a dataset containing 20 structural outliers was created.

To achieve this, in the original Concrete Compressive Strength dataset (1030 observations), the target-variable values of the 10 observations with the largest values and the 10 observations with the smallest values were replaced by the arithmetic mean of the maximum and minimum values. The remaining features were left unchanged. The resulting dataset contained 1030 observations, twenty of which are structural outliers because, with high probability, they do not obey the regularities of the studied domain and therefore do not lie on the mentally imagined hypersurface that geometrically represents these regularities. The number of artificially generated outliers created in this way is approximately 2% of the total number of observations.

The experiment was repeated 30 times using different random initializations in order to evaluate the stability of the methods.

The results of testing various outlier detection methods on the resulting benchmark dataset, obtained using the program https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_14_Testing_on_the_Concrete_Strength_dataset.ipynb are presented in Table 5. All methods were tested under the requirement of detecting exactly 20 outliers.

Table 5. Results of testing outlier detection methods on the benchmark dataset “Concrete Compressive Strength” containing 2% outliers

Method	F1 (mean ± std)	ROC-AUC (mean ± std)
NNFA (ξ=0)	0.760 ± 0.051	0.984 ± 0.008
NNFA (ξ=0.01)	0.760 ± 0.037	0.986 ± 0.005
NNFA (ξ=0.5)	0.798 ± 0.009	0.992 ± 0.001
NNFA (ξ=1)	0.728 ± 0.031	0.991 ± 0.000
Neural Network (error)	0.610 ± 0.187	0.918 ± 0.047
Random Forest (error)	0.497 ± 0.050	0.853 ± 0.036
Combined (RF+AE)	0.027 ± 0.042	0.824 ± 0.042
LOF (pyod)	0.150 ± 0.000	0.628 ± 0.000
LOF (with y)	0.000 ± 0.000	0.618 ± 0.000
COPOD (pyod)	0.000 ± 0.000	0.603 ± 0.000
OCSVM (pyod)	0.000 ± 0.000	0.597 ± 0.000
PCA (pyod)	0.000 ± 0.000	0.571 ± 0.000
IsolationForest (pyod)	0.000 ± 0.000	0.564 ± 0.023
Autoencoder (reconstruction)	0.005 ± 0.015	0.553 ± 0.052
kNN (pyod)	0.000 ± 0.000	0.547 ± 0.000
Isolation Forest (with y)	0.000 ± 0.000	0.541 ± 0.029
ABOD (pyod)	0.200 ± 0.000	0.526 ± 0.000
One-Class SVM (with y)	0.012 ± 0.025	0.450 ± 0.078
HBOS (pyod)	0.000 ± 0.000	0.444 ± 0.000

The experimental results presented in Table 5 indicate the following.

1. Superiority of the NNFA method. All four NNFA configurations ($\xi = 0, 0.01, 0.5, 1$) achieved values of F1 = 0.728–0.798 and ROC-AUC = 0.984–0.992. The best result was obtained by the NNFA variant ($\xi = 0.5$): F1 = 0.798 ± 0.009, ROC-AUC = 0.992 ± 0.001. The low standard deviation ($\sigma < 0.01$) indicates high stability of the method. For $\xi = 0.01$ and $\xi = 0$, the results are also high (F1 ≈ 0.76), while for $\xi = 1$ they are somewhat lower (F1 = 0.728 ± 0.031); however, all variants statistically significantly outperform the best standard method. For all NNFA variants, the standard deviation of F1 does not exceed 0.051, confirming the high reproducibility of the method under different random initializations.
2. Instability of standard neural-network methods. The best standard method, Neural Network (error), achieved F1 = 0.610 ± 0.187. The high standard deviation indicates a strong dependence on random initialization: in some runs the method detects 12–16 outliers, while in others it detects only 4–6. The value ROC-AUC = 0.918 ± 0.047 confirms reasonably good ranking quality, but under a fixed top-20 threshold the result remains unstable.
3. Unsuitability of most classical methods for detecting structural outliers. Fifteen of the sixteen methods (including Random Forest (error), ABOD, HBOS, kNN, LOF, PCA, COPOD, OCSVM, Isolation Forest, Autoencoder, and their modifications incorporating the target variable) have F1 = 0.000 (or values close to zero) in all 30 runs. The only exception is OF (pyod) with F1 = 0.150 ± 0.000. This method detects exactly 3 outliers out of 20 (because the contamination parameter in PyOD specifies the proportion of outliers by default, but even when exactly 20 outliers are specified, the method is unable to identify the correct observations). Many methods demonstrate ROC-AUC values around 0.5–0.6 (random ranking), which confirms their complete inability to detect structural anomalies.
4. Although most standard methods have F1 = 0, some of them achieve ROC-AUC > 0.8 (for example, Random Forest (error) – 0.853, Combined (RF+AE) – 0.824). This means that they rank outliers above normal observations, but under strict top-20 selection they are unable to correctly identify the specific 20 observations that were modified. In practical problems, where the proportion of outliers is unknown, this makes such methods useless.

The paired Wilcoxon signed-rank test showed that all NNFA variants statistically significantly outperform the best standard method, Neural Network (error) ($p < 0.05$ in all comparisons). The differences between NNFA ($\xi = 0.5$) and NNFA ($\xi = 0.01$) are not statistically significant ($p > 0.05$); however, NNFA ($\xi = 0.5$) demonstrates the smallest standard deviation and the highest mean F1 score. Therefore, for practical applications, NNFA with $\xi = 0.5$ can be recommended as providing the highest and most stable recall in detecting structural outliers. When computational resources are limited, the use of $\xi = 0$ is acceptable (which corresponds to the original NNF method with a single neural network). However, in this case a somewhat higher variability of results is observed ($\sigma = 0.051$ versus 0.009), which can be explained by the absence of the suppressing effect of overfitted neural networks due to the non-application of the accumulation and averaging procedure for training errors.

Conclusion. The proposed NNFA method is the only one among the 16 methods considered that consistently and reliably detects multiple structural outliers (F1 ≈ 0.8, ROC-AUC ≈ 0.99). Standard methods, including neural-network-based ones, either produce zero results or are extremely unstable. The obtained results confirm that the accumulation and averaging of training errors,

together with the application of the corollary of the Arnold–Kolmogorov–Hecht–Nielsen theorem, make it possible to avoid hyperparameter tuning.

4.1.3 Computational Cost Analysis

As already noted, a drawback of the NNFA method is its high computational cost. It is proposed to reduce this cost by decreasing the upper boundary of the interval of parameter N values, i.e., by reducing the value of parameter ξ from 1 down to 0. The results of computational experiments measuring the execution time of the program implementing the NNFA method, obtained using the program https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_14a_Testing_on_the_Concrete_Strength_dataset_and_Time.ipynb are presented in Table 6. As can be seen from Table 6, decreasing parameter ξ substantially reduces the computational cost of the NNFA method and reaches the minimum value of 6.18 seconds when $\xi = 0$, i.e., when only one neural network with the minimum number of neurons in the hidden layer participates in training and the NNFA method degenerates into the NNF method. At the same time, as can be seen from Tables 4 and 5, the quality metrics change only slightly as ξ varies.

This property should apparently be taken into account in practical applications of the NNFA method, while understanding that when $\xi = 0$ only one neural network participates in training, and this network may turn out to be overfitted. In this case, its negative influence cannot be suppressed by averaging with other neural networks, as clearly illustrated in Figures 10–12.

Table 6. Program execution time as a function of ξ

Method	Time (seconds)
NNFA ($\xi = 0$)	6.18
NNFA ($\xi = 0.01$)	6.67
NNFA ($\xi = 0.5$)	40.28
NNFA ($\xi = 1$)	92.43

4.1.4 Sensitivity Analysis with Respect to the Degree of Data Contamination

To evaluate the robustness of the proposed method to increasing data contamination, an additional series of experiments was carried out using the Concrete Compressive Strength dataset. Structural outliers were artificially introduced into the dataset using the procedure described above in amounts ranging from 1% to 5% of the total number of observations. For each contamination level, all methods were run 30 times with different random seed values and trained for 500 epochs. he evaluation metrics were the F1 score and ROC-AUC. For each contamination level, the mean value, standard error, and 95% confidence interval (based on Student's t-distribution) were calculated. The statistical significance of differences between NNFA and the best standard method was evaluated using the paired Wilcoxon signed-rank test at the significance level $\alpha = 0.05$. The results obtained using the program https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_15_Dependence_on_the_percentage_of_outliers.ipynb are presented in Figures 13, 14, and Table 7.

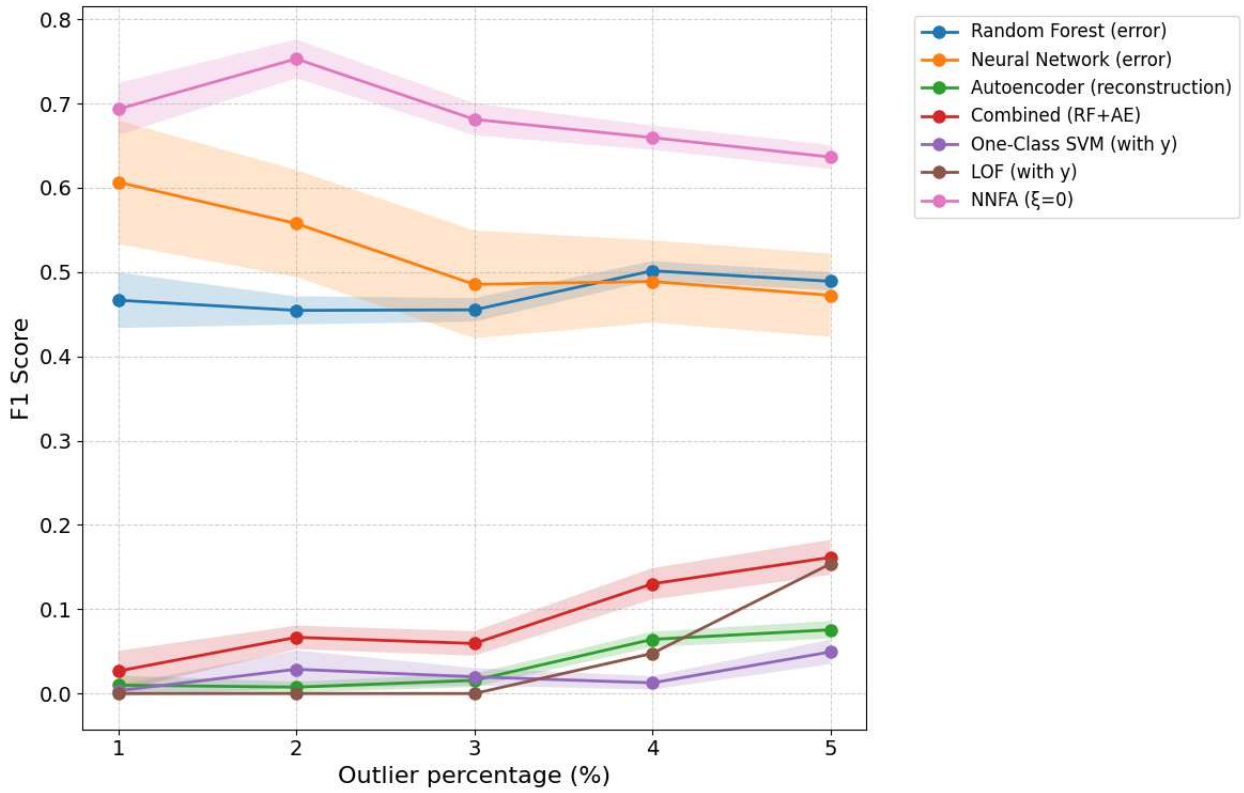


Figure 13. Dependence of the F1 score on the percentage of outliers for the seven methods that achieved the best results. Shaded regions represent 95% confidence intervals computed from 30 independent runs.

As can be seen from Figure 13, the NNFA method demonstrates the best F1-score values throughout the entire investigated range of contamination levels. The maximum F1-score value is observed at an outlier proportion of 2% and is approximately 0.75. With a further increase in the proportion of outliers, the quality of the method decreases only slightly and remains above 0.63 even at a contamination level of 5%.

It is particularly important to note that the confidence intervals of the NNFA method are substantially narrower than those of its closest competitors. This indicates a high robustness of the method with respect to random initialization of neural-network weights and confirms the reproducibility of the obtained results. In contrast, the closest competing method, Neural Network (error), demonstrates noticeably wider confidence intervals, indicating a significant dependence of its quality on random training factors.

A similar pattern is observed for the ROC-AUC metric (Figure 14). The NNFA method demonstrates ROC-AUC values in the range from 0.95 to 0.99 regardless of the proportion of outliers. At the same time, the width of the confidence intervals remains minimal.

It should be noted that as the proportion of outliers increases, the advantage of NNFA over the closest competing methods becomes more pronounced. At contamination levels of 4–5%, the gap between NNFA and the best competing method reaches approximately 0.14–0.17 in terms of ROC-AUC and about 0.14–0.16 in terms of F1-score.

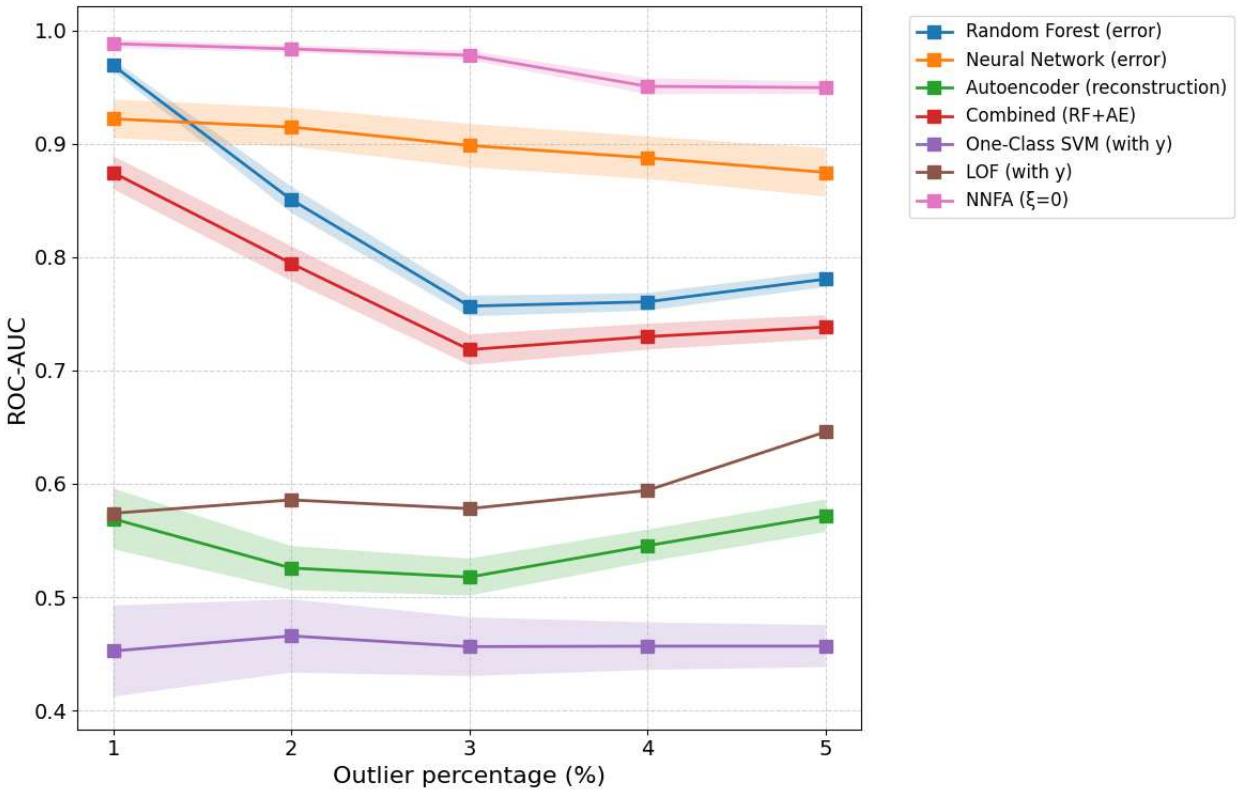


Figure 14. Dependence of ROC-AUC on the percentage of outliers for the seven methods that achieved the best results. Shaded regions represent 95% confidence intervals computed from 30 independent runs.

Table 7. Comparison of NNFA with the Best Competing Methods Using the Wilcoxon Test

Proportion of outliers	Best competitor	p-value	Conclusion
1%	Neural Network (error)	0.062236	The difference is statistically insignificant
2%	Neural Network (error)	0.000016	The difference is statistically significant
3%	Neural Network (error)	0.000005	The difference is statistically significant
4%	Random Forest (error)	0.000002	The difference is statistically significant
5%	Random Forest (error)	0.000002	The difference is statistically significant

As can be seen from Table 7, at contamination levels of 2%, 3%, 4%, and 5%, the superiority of NNFA over the best competing method is statistically significant ($p < 0.05$). The strongest differences are observed at contamination levels of 4% and 5%, where the p-values reach 2×10^{-6}. Only at the contamination level of 1% does the difference between NNFA and Neural Network (error) fail to reach statistical significance ($p = 0.062236$), although the mean values of both metrics remain higher for NNFA.

Thus, the results of the conducted study allow us to conclude that the NNFA method possesses high robustness with respect to changes in the proportion of structural outliers in the data. Moreover, an increase in the contamination level has almost no effect on its ability to detect structural violations, whereas the performance of competing methods deteriorates noticeably. The presence of narrow confidence intervals and statistically significant differences according to the

Wilcoxon test confirms that the observed superiority of NNFA is not a consequence of random factors but reflects real advantages of the proposed approach.

4.2 California Housing Dataset

To evaluate the performance of the method on a substantially larger dataset, the widely known California Housing dataset proposed by Pace and Barry (1997) was used. The dataset contains: 20,640 observations; 8 input features; 1 output variable — median housing value.

As in the previous experiments, part of the observations was artificially transformed into structural outliers. To achieve this: 200 observations with the minimum values of the output variable were replaced by the maximum housing-value observed in the dataset; 200 observations with the maximum values were replaced by the minimum housing-value observed in the dataset. As a result, 400 structural outliers were created, corresponding to approximately 2% of the entire dataset.

All methods were tested using the program https://colab.research.google.com/github/Astrohaker/OutliersDetectionResearch/blob/main/!!!_16_Testing_on_the_California_dataset.ipynb under the requirement of identifying 400 outliers. To verify the statistical significance of differences between the considered outlier-detection methods, the two-sided Wilcoxon signed-rank test was used. The analysis was performed using the results of 30 independent runs of the algorithms. The mean values of the F1 and ROC-AUC metrics are presented in Table 8, while the results of pairwise Wilcoxon comparisons are presented in Table 9.

Table 8. Results of testing outlier-detection methods on the “California Housing” benchmark dataset. Outlier proportion: 2%

Method	F1 (mean ± std)	ROC-AUC (mean ± std)
NNFA (ξ=0)	0.804 ± 0.009	0.997 ± 0.000
Neural Network (error)	0.779 ± 0.077	0.990 ± 0.008
Random Forest (error)	0.697 ± 0.014	0.980 ± 0.002
Combined (RF+AE)	0.686 ± 0.014	0.980 ± 0.002
Isolation Forest (with y)	0.140 ± 0.014	0.886 ± 0.014
HBOS (pyod)	0.042 ± 0.001	0.697 ± 0.000
IsolationForest (pyod)	0.081 ± 0.008	0.693 ± 0.011
kNN (pyod)	0.085 ± 0.000	0.681 ± 0.000
ABOD (pyod)	0.090 ± 0.000	0.661 ± 0.000
LOF (with y)	0.090 ± 0.000	0.654 ± 0.000
Autoencoder (reconstruction)	0.090 ± 0.009	0.653 ± 0.014
COPOD (pyod)	0.060 ± 0.000	0.634 ± 0.000
OCSVM (pyod)	0.085 ± 0.000	0.623 ± 0.000
LOF (pyod)	0.065 ± 0.000	0.610 ± 0.000
PCA (pyod)	0.083 ± 0.000	0.578 ± 0.000
One-Class SVM (with y)	0.012 ± 0.003	0.476 ± 0.009

Table 9. Pairwise comparisons using the Wilcoxon test

Methods	ABOD (pyod)	HBOS (pyod)	IsolationForest (pyod)	kNN (pyod)	LOF (pyod)	OCSVM (pyod)	PCA (pyod)	COPOD (pyod)	Random Forest (error)	Neural Network (error)	Autoencoder (reconstruction)	Combined (RF+AE)	One-Class SVM (with y)	Isolation Forest (with y)	LOF (with y)	NNFA (ξ=0)
ABOD (pyod)	1.00000	0.00004	0.00075	0.00001	0.00001	0.00001	0.00001	0.00001	0.00009	0.00009	0.65736	0.00009	0.00008	0.00009	1.00000	0.00009
HBOS (pyod)	0.00004	1.00000	0.00008	0.00004	0.00004	0.00004	0.00004	0.00004	0.00009	0.00009	0.00009	0.00009	0.00008	0.00009	0.00004	0.00009
IsolationForest (pyod)	0.00075	0.00008	1.00000	0.01356	0.00008	0.01356	0.17567	0.00008	0.00009	0.00009	0.00127	0.00009	0.00009	0.00009	0.00075	0.00009
kNN (pyod)	0.00001	0.00004	0.01356	1.00000	0.00001	1.00000	0.00001	0.00001	0.00009	0.00009	0.03985	0.00009	0.00008	0.00009	0.00001	0.00009
LOF (pyod)	0.00001	0.00004	0.00008	0.00001	1.00000	0.00001	0.00001	0.00001	0.00009	0.00009	0.00009	0.00009	0.00008	0.00009	0.00001	0.00009
OCSVM (pyod)	0.00001	0.00004	0.01356	1.00000	0.00001	1.00000	0.00001	0.00001	0.00009	0.00009	0.03985	0.00009	0.00008	0.00009	0.00001	0.00009
PCA (pyod)	0.00001	0.00004	0.17567	0.00001	0.00001	0.00001	1.00000	0.00001	0.00009	0.00009	0.00329	0.00009	0.00008	0.00009	0.00001	0.00009
COPOD (pyod)	0.00001	0.00004	0.00008	0.00001	0.00001	0.00001	0.00001	1.00000	0.00009	0.00009	0.00009	0.00009	0.00008	0.00009	0.00001	0.00009
Random Forest (error)	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	1.00000	0.00085	0.00009	0.00012	0.00009	0.00009	0.00009	0.00009
Neural Network (error)	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00085	1.00000	0.00009	0.00101	0.00009	0.00009	0.00009	0.38027
Autoencoder (reconstruction)	0.65736	0.00009	0.00127	0.03985	0.00009	0.03985	0.00329	0.00009	0.00009	0.00009	1.00000	0.00009	0.00009	0.00009	0.65736	0.00009
Combined (RF+AE)	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00012	0.00101	0.00009	1.00000	0.00009	0.00009	0.00009	0.00009
One-Class SVM (with y)	0.00008	0.00008	0.00009	0.00008	0.00008	0.00008	0.00008	0.00008	0.00009	0.00009	0.00009	0.00009	1.00000	0.00009	0.00008	0.00009
Isolation Forest (with y)	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	1.00000	0.00009	0.00009
LOF (with y)	1.00000	0.00004	0.00075	0.00001	0.00001	0.00001	0.00001	0.00001	0.00009	0.00009	0.65736	0.00009	0.00008	0.00009	1.00000	0.00009
NNFA (ξ=0)	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.38027	0.00009	0.00009	0.00009	0.00009	0.00009	1.00000

As can be seen from Table 8, the NNFA ($\xi = 0$) method demonstrated the best results according to both quality metrics. Its mean F1-score was 0.804 ± 0.009, while the ROC-AUC value reached 0.997 ± 0.000. The closest competitor was the Neural Network (error) method, which achieved F1 = 0.779 ± 0.077 and ROC-AUC = 0.990 ± 0.008. The remaining methods demonstrated substantially lower results. Thus, the Random Forest (error) and Combined (RF+AE) methods obtained F1 values of approximately 0.69, whereas most classical algorithms from the PyOD and scikit-learn libraries showed F1 values below 0.15.

Attention should be paid not only to the higher mean F1-score of the NNFA method, but also to its substantially smaller standard deviation. The value 0.009 indicates high robustness of the method with respect to random variations in the initial training conditions. For comparison, the standard deviation of the closest competitor, Neural Network (error), was 0.077, which is more than eight times higher. This indicates substantially greater stability of the proposed method.

To verify the statistical significance of the observed differences, the Wilcoxon signed-rank test was applied. The results of pairwise comparisons showed that the differences between NNFA and practically all competing methods are statistically significant. For most comparisons, p-values below $p < 10^{-4}$ were obtained, allowing the null hypothesis of no differences between the methods to be confidently rejected.

Of particular interest is the comparison of NNFA with its closest competitor, the Neural Network (error) method. Although the mean F1-score of NNFA was higher (0.804 versus 0.779), the Wilcoxon test yielded $p = 0.380273$, which exceeds the conventional significance level of 0.05. This means that, for the sample size considered, the difference between the two methods cannot be regarded as statistically significant. The probable reason is the relatively small difference between the mean F1 values in the presence of substantial variability in the results of the Neural Network (error) method.

Nevertheless, the practical value of the NNFA method is determined not only by its somewhat higher mean outlier-detection quality, but also by its substantially greater robustness. Unlike Neural Network (error), whose performance noticeably depends on random weight initialization and other training factors, the NNFA method demonstrates nearly unchanged results from run to run.

Thus, the conducted statistical analysis confirms that the NNFA method provides the best combination of outlier-detection quality and result stability among all considered methods. At the same time, its superiority over most classical algorithms is statistically significant, whereas relative to its closest competitor it manifests itself primarily through substantially greater stability.

Finally, Figure 15 presents, for illustrative purposes, the distribution of neural-network training errors produced by the NNFA method (for $\xi = 0$) across the observations of the test dataset.

Here, the red bars rising above the threshold dashed line represent the training errors corresponding to observations that are true outliers. The black bars are located below the threshold line. They represent the training errors obtained for normal observations.

The yellow bars also rise above the threshold line. However, they correspond to observations that were not labeled as outliers in the dataset. Since they crossed the threshold line, the NNFA algorithm "incorrectly" classified them as outliers. These "errors" of the NNFA method may be explained by the fact that these observations apparently violate the regularities of the domain to an even greater extent than the artificially created structural outliers introduced by us.

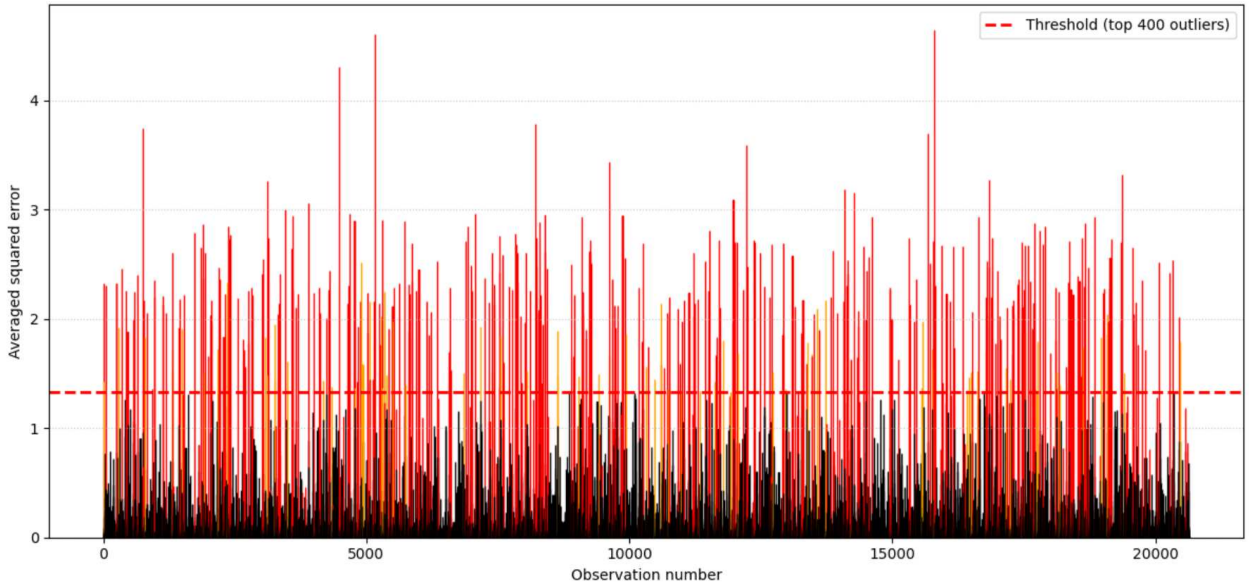


Figure 15. Distribution of neural-network training errors across the observations of the "California Housing" dataset

Nevertheless, we may state that, on the "California Housing" dataset, the NNFA method provided:

- the best outlier-detection quality;
- the best ROC-AUC values;
- the minimum variability of results;
- high robustness with respect to random network initialization.

Considering all characteristics together, NNFA proved to be the best among all considered outlier-detection methods.

5 Discussion of Results

This paper presents the results of several computational experiments that require discussion. Figure 1 shows the points of the simplest benchmark dataset governed by the equation of a parabola. The dataset contains one obvious outlier whose coordinates do not satisfy the parabola equation. Paradoxically, the results of the computational experiment presented in Table 1 show

that none of the fifteen well-known methods from Python libraries was able to detect this outlier. The result of this simple and easily reproducible experiment justifies the relevance of the objective of the present study, namely, to fill the identified gap by proposing and testing a new outlier-detection method capable of successfully solving such problems.

In this paper, two modifications of a neural-network-based outlier-detection method, NNF and NNFA, are proposed to address this problem. Both modifications are based on the idea that, for some neural networks, the largest training error is observed on observations that are outliers.

This idea itself is not new. What is new is the visual demonstration that not all neural networks possess this remarkable property (see Figure 2), as well as the identification of the conditions under which this property occurs. As follows from the results presented in Figure 2, in the simplest case of a feedforward neural network with a single hidden layer of sigmoid neurons, the presence of this property depends on the number N of hidden neurons. For some values of N, the neural network possesses this property and is therefore suitable for outlier detection, whereas for other values of N it does not possess this capability. As can be seen from Figure 3, as N increases from 0 to 80, this remarkable capability repeatedly appears and disappears.

Similar patterns can also be observed in the results of computational experiments performed on benchmark datasets corresponding to the paraboloid of revolution (Fig. 4) and the hyperbolic paraboloid (Fig. 7).

As follows from the results presented in Figures 3, 6, and 8, the NNF method is capable of correctly detecting outliers only for certain values of the hyperparameter N. Moreover, comparing the data in these figures, one can observe that these "successful" values of the hyperparameter N do not necessarily coincide for different tasks. Figure 9 presents a histogram obtained by intersecting the histograms shown in Figures 3, 6, and 8. It can be seen that, among the 81 tested values of the hyperparameter N, only eight values ensure the suitability of neural networks for outlier detection in all three considered tasks simultaneously. Thus, it becomes obvious that, with a further increase in the number of considered tasks, the number of successful values of N will continue to decrease. Therefore, it appears impossible to select a universal value of the hyperparameter N that would ensure successful application of the NNF method in all possible situations.

The idea of accumulating and averaging the training errors of an ensemble of neural networks with N values belonging to a certain interval made it possible to solve this problem and formulate a universal rule for designing neural networks suitable for detecting outliers in a wide range of datasets. The effectiveness of this idea is clearly demonstrated in Figures 10, 11, and 12. Figure 10 shows that for some values of the hyperparameter N, for example $N = 54$, the maximum training error is concentrated not on observations that are outliers. This indicates that such neural networks are unsuitable for outlier detection. In the paper, such networks are referred to as "bad" networks. However, as shown in Figure 11, despite the presence of such "bad" neural networks, the maximum value of the accumulated and averaged training errors is concentrated on observation 24, which is the true outlier.

The positive effect of averaging training errors is explained by Figure 12, which presents the results of the same experiment as in Figure 10, but displayed using a common vertical-axis scale. It can be clearly seen that the "bad" networks ($N = 54$) are overfitted, and their training errors are negligibly small in comparison with those of the "good" networks. Therefore, during averaging, they make virtually no contribution and do not affect the overall pattern of the averaged training-error distribution (Fig. 11), which is formed by the "good" neural networks and can therefore be used to detect true outliers. We called this outlier-detection approach the NNFA method, which stands for Neural Network Filter with Averaging.

Our next contribution was the idea of using the corollary of the Arnold–Kolmogorov–Hecht–Nielsen theorem, expressed by formulas (1)–(5), to determine the interval of N values over which

the averaging procedure should be performed. According to this corollary, simple feedforward neural networks with a single hidden layer and sigmoid activation functions tend to become overfitted when $N > N_{max}$ (where N_{max} is defined by Formula (5)), and tend to become underfitted when $N < N_{min}$ (where N_{min} is defined by Formula (4)). Therefore, in the NNFA method it was proposed to use the Arnold – Kolmogorov – Hecht–Nielsen interval defined by Inequality (6) as the interval for averaging training errors.

A disadvantage of the NNFA method is its high computational cost. To reduce this cost, it was proposed to decrease the upper boundary of the averaging interval (6) by replacing Formula (6) with Formula (7), in which the upper boundary of the interval is controlled by the parameter $\xi \in [0, 1]$. When $\xi = 0$, the value of N coincides with the lower boundary of Interval (6), no averaging is performed, and the NNFA method degenerates into the NNF method. When $\xi = 1$, averaging is performed over all values of N belonging to Interval (6).

A concern arose that reducing computational cost by decreasing the parameter ξ down to zero might prevent the contribution of the "good" neural networks from neutralizing the influence of the "bad" neural networks, thereby reducing the quality metrics of outlier detection. However, as can be seen from the results of subsequent computational experiments (Tables 4 and 5) performed on real-world datasets, no noticeable decrease in quality metrics occurs when the hyperparameter ξ is reduced within the interval $0 \leq \xi \leq 1$. At the same time, the quality metrics of the NNFA method exceed those of the competing methods for all investigated values of ξ.

The results of the computational experiment presented in Tables 4 and 5 and Figure 6 showed that reducing the parameter ξ makes it possible to substantially decrease computation time while causing only a negligible reduction in outlier-detection quality. This creates an opportunity for a practical trade-off between detection accuracy and computational cost.

When analyzing the results of comparing the NNFA method with other outlier detection methods, one may notice the somewhat unusual behavior of the Neural Network (error) method. While in the very first experiment with the parabola, in the ranking table (Table 1), the Neural Network (error) method occupied the last position among the competing methods, in all subsequent experiments (Tables 2, 3, 4, 5, and 9), this method, on the contrary, was among the leaders according to the F1 and ROC-AUC metrics, while at the same time exhibiting the largest standard deviations. It is interesting to note that this method is also neural-network-based, but unlike NNFA, it uses prediction errors rather than training errors. The idea of using neural-network prediction errors for outlier detection was apparently first proposed in our earlier publication (Cherepanov and Yasnitsky LN, 2008). The effectiveness of this original idea was also demonstrated there using benchmark examples. However, at that time, the idea of using neural networks for outlier detection did not gain widespread adoption because of the need to tune their hyperparameters in each specific case.

The NNFA/NNF methods proposed in the present paper are based on the idea of using neural-network training errors (rather than prediction errors). Here, users are relieved from the task of hyperparameter optimization. Using the ideas of error accumulation and averaging together with the corollary of the Arnold–Kolmogorov–Hecht–Nielsen theorem, we moved from the need to tune the hyperparameter N to the need to tune the hyperparameter ξ. However, as demonstrated by the conducted experiments, when its value is chosen within the corresponding interval, the quality of outlier detection is practically independent of the specific value of ξ. As can be seen from Tables 4 and 5, when the hyperparameter ξ changes, the F1 and ROC-AUC metrics either remain unchanged or change only insignificantly. Naturally, the absence of the need to repeatedly tune hyperparameters constitutes an important advantage of the NNFA/NNF methods compared with other outlier detection methods.

5.1 Relationship to Modern Deep Methods

Over the last decade, Deep Learning (DL) has taken a central place in anomaly-detection research (Chalapathy and Chawla 2019; Pang et al. 2021). Deep neural networks are capable of automatically extracting hierarchical feature representations, making them attractive for identifying complex nonlinear anomalous patterns in high-dimensional data (Chandola et al. 2009; Ruff et al. 2021). Currently, there exists a wide spectrum of DL approaches to anomaly detection that differ in architecture and learning paradigm. The most common categories include:

• **Autoencoders (AE)** and their variations. These networks are trained to reconstruct normal data, while anomalies are detected through large reconstruction errors (Sakurada and Yairi 2014). Owing to their simplicity, autoencoders remain among the most popular DL approaches for anomaly detection (Chalapathy and Chawla 2019).

• **Generative Adversarial Networks (GANs)**, which model the distribution of normal data and identify anomalies as observations inconsistent with this distribution (Schlegl et al. 2017; Zenati et al. 2018).

• **Recurrent Neural Networks (RNN, LSTM, GRU)** intended for anomaly detection in sequential and time-series data (Malhotra et al. 2016).

• **Convolutional Neural Networks (CNNs)**, which are effective for detecting spatial anomalies in images and spatio-temporal data (Chalapathy and Chawla 2019; Ruff et al. 2021).

• **Deep metric-learning and one-class classifiers**, such as Deep Support Vector Data Description (DSVDD) and Deep One-Class Classification (Ruff et al. 2018, 2020).

Despite their impressive performance in many applications, deep anomaly-detection methods are associated with a number of serious limitations, making their application in real-world problems labor-intensive and often insufficiently justified:

• **Complexity of hyperparameter tuning.** Deep models contain numerous hyperparameters, including the number of layers, dimensionality of hidden representations, learning rate, regularization coefficients, and others. Under unsupervised learning conditions, where true anomaly labels are unavailable, tuning these hyperparameters becomes extremely nontrivial (Campos et al. 2016; Zhao and Akoglu 2024). Autoencoders, for example, often suffer from unstable training, poor reproducibility, and high sensitivity to hyperparameter selection, which limits their reliability when working with heterogeneous datasets (An and Cho 2015; Sarvari et al. 2021; Ding et al. 2024).

• **Demanding computational requirements and large data volumes.** Deep methods require substantial computational resources (often involving GPUs) and large volumes of training data in order to achieve high performance, making them unsuitable in resource-constrained environments (Pang et al. 2021).

• **Lack of interpretability.** By their nature, deep neural networks are “black boxes.” Explaining why a particular observation has been classified as anomalous is often extremely difficult (Han et al. 2024). This is particularly critical in domains with high reliability requirements, such as medicine, cybersecurity, and industrial diagnostics (Boukerche et al. 2021).

• **Risk of overfitting.** When trained on limited datasets, deep neural networks are prone to overfitting, which leads to deterioration of their generalization ability (Sarvari et al. 2021).

In contrast to the above approaches, the NNF/NNFA method proposed in the present work is based on a simple neural network with a single hidden layer. Unlike modern deep methods, this architecture possesses several important advantages:

1. **Absence of hyperparameter tuning.** The method requires no tuning, since the number of neurons in the hidden layer is determined automatically using a closed-form formula derived from the corollary of the Arnold–Kolmogorov–Hecht–Nielsen theorem (Yasnitsky 2016).

2. **High interpretability.** The simplicity of the network makes it possible to directly analyze its operation and identify the reasons why a particular observation is recognized as an outlier.

3. **Low computational complexity.** Training a single-hidden-layer network does not require substantial computational resources and can be performed on standard processors even for large datasets.

4. **Robustness to small datasets.** The simplicity of the network reduces the risk of overfitting when working with limited samples.

Thus, although deep anomaly-detection methods demonstrate high effectiveness in a number of applications, they are associated with problems of hyperparameter tuning, low interpretability, and high computational complexity. The simple neural-network approach proposed in this paper does not suffer from these drawbacks, which makes it a practical and reliable tool for structural outlier detection.

5.2 Limitations and Future Research

1. The main advantage of the NNFA method is the absence of a need for hyperparameter tuning. This became possible due to the application of the idea of error accumulation followed by averaging, as well as due to the application of the corollary of the Arnold–Kolmogorov–Hecht–Nielsen theorem to regulate the averaging interval. However, as is known, this corollary was derived for the approximation problem rather than for the outlier-detection problem. Therefore, the idea of using the Arnold – Kolmogorov – Hecht–Nielsen corollary in our study is only a heuristic based on the authors' intuition and supported experimentally. A complete theoretical explanation of this phenomenon is a subject of future research.

2. In all our experiments, it was observed that the quality of the NNFA method is practically independent of the specific value of the hyperparameter ξ, provided that it belongs to the corresponding interval. This indicates that there is no need to tune the hyperparameter ξ when applying the NNFA method. Naturally, this fact constitutes a serious advantage of NNFA over other outlier-detection methods. However, at present this may only be regarded as a hypothesis supported by a limited number of experiments. Additional experiments are planned in the future.

3. The methods were tested on datasets containing only structural outliers without extreme deviations of individual features. In other words, the outliers created by us violated the functional relationships between variables but did not manifest themselves through extreme values of individual features. Testing the proposed methods on benchmark datasets containing more general types of outliers is a subject of future research.

4. A subject of future research is a complete answer to the question of how the methods behave in high-dimensional problems. So far, we have experimentally verified that the NNF and NNFA methods outperform 15 classical methods in the case of a problem with one input feature (Fig. 1), two input features (Figs. 4 and 7), and eight input features (the "Concrete Strength" and "California Housing" datasets). To this, it may be added that the NNF and NNFA methods performed well in projects devoted to the development of diagnostic and prognostic systems in industry (Yasnitsky LN et al. 2023), medicine (Yasnitsky et al. 2022), and economics (Yasnitsky et al. 2021), where the number of input features exceeded 40 or more.

5. In all our benchmark datasets, outliers were introduced only into the values of the output variables. In the future, it is planned to conduct more extensive experiments involving the introduction of outliers not only into y but also into X, with different degrees of intensity.

6. When analyzing the behavior of the curves shown in Figure 2, it is intuitively clear that if the dataset contained not one but many outliers located close to each other, the probability of their successful detection could decrease substantially. This means that, presumably, the NNF and NNFA methods are more suitable for detecting isolated outliers than massive groups of outliers. Therefore, it appears promising to investigate the possibility of applying the methodology of sequential detection and removal of outliers, figuratively referred to in (Liu et al. 2024) as "peeling the onion": first identifying and removing the outlier observation with the largest training error, then retraining the network, identifying and removing the next outlier with the largest training error, and so on. Investigation of the possibilities of such a methodology is planned for future work.

7. As follows from most of our experiments, the closest competitor of NNFA is the Neural Network (error) method. Its main drawback is the instability observed in our experiments, which can be explained by the catastrophic influence of neural-network architecture on their ability to detect outliers, similar to what is demonstrated in Figures 3, 6, and 8. Therefore, it appears promising in future research to eliminate this drawback of the Neural Network (error) method by further developing the approach applied in the present study. It is possible that our approach may also prove useful for improving modern outlier-detection methods based on deep neural networks, autoencoders, and transformer architectures.

8. In the future, it is planned to integrate the implementations of the NNF and NNFA methods into the PyOD library, which will make them available to a wide range of users.

6 Conclusion

The problem of outlier detection remains relevant and not fully solved, as evidenced by the large number of publications devoted to it, including recent ones. Classical methods based on estimates of statistical distributions and distances are, by virtue of their underlying principles, unable to detect structural outliers that do not stand out in terms of distance to neighbors, local density, or statistical properties of the distribution, but nevertheless violate functional relationships between features. Existing methods based on neural networks (Cherepanov and Yasnitsky 2008; Yasnitsky 2020; Beliakov et al. 2011; Malmström et al. 2022; Ferdowsi et al. 2013) are theoretically capable of capturing such functional violations, but they require hyperparameter tuning, which is difficult under real-world conditions.

Our contributions are as follows.

C1. We proposed a definition of a structural outlier and, based on this definition, created six reliably labeled datasets containing outliers and normal observations. Three of these datasets are synthetic and correspond to points belonging to a parabola, a paraboloid of revolution, and a hyperbolic paraboloid, respectively. The remaining three datasets were derived from real-world datasets related to concrete-strength evaluation and California housing-price assessment.

C2. We empirically demonstrated and theoretically substantiated that the fifteen most popular and readily available classical methods are unable to solve the problem of detecting structural outliers in the simplest synthetic datasets created by us (corresponding to a parabola, a paraboloid of revolution, and a hyperbolic paraboloid).

C3. We proposed two modifications of a neural-network-based outlier detection method, NNF and NNFA, based on the idea of accumulating and averaging training errors and on the application of the corollary of the Arnold–Kolmogorov–Hecht–Nielsen theorem, which makes it possible to avoid hyperparameter tuning. Both modifications of the neural-network method, unlike the classical methods, correctly identified outliers in all three simplest synthetic datasets and demonstrated the best results, compared with the classical methods, on all three real-world datasets.

In conclusion, we note that the NNF and NNFA methods have already been used by the authors in projects in industry (Yasnitsky LN et al. 2023), medicine (Yasnitsky et al. 2022), economics (Yasnitsky et al. 2021), and many other application domains (Yasnitsky LN 2016). Without preliminary data cleaning using NNF and NNFA, the implementation of some of these projects would have been difficult and, in certain cases, impossible.

Acknowledgements

This work was carried out within the framework of the state assignment EZID-2025-0003:

"Development of technologies for reducing mortality from cardiovascular diseases for application in the healthcare system (preventive medical examinations) based on early diagnosis, prevention, and telemedicine assessment of rehabilitation measures," No. 1024022200061-2-3.1.4; 3.2.25; 1.2.1.

Funding

Open Access funding was provided by the Academician E.A. Wagner Perm State Medical University.

Code Availability

As already mentioned, all source codes, detailed results, tested parameter values, and datasets used in this study are available for free online download at:
https://github.com/Astrohaker/OutliersDetectionResearch/tree/main

Conflict of Interest

The authors declare no conflict of interest.

References

An J, Cho S (2015) Variational autoencoder based anomaly detection using reconstruction probability. Technical report, Seoul National University

Beliakov G, Kelarev A, Yearwood J (2011) Robust artificial neural networks and outlier detection. arXiv:1110.0169v1 [math.OC]

Boukerche A, Zheng L, Alfandi O (2021) Outlier Detection: Methods, Models, and Classification. ACM Computing Surveys 53(3): Article No. 55. DOI: 10.1145/3381028

Campos GO, Zimek A, Sander J, Campello RJGB, Micenková B, Schubert E, Assent I, Houle ME (2016) On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Mining and Knowledge Discovery 30(4):891-927. DOI: 10.1007/s10618-015-0444-8

Chalapathy R, Chawla S (2019) Deep learning for anomaly detection: a survey. arXiv preprint arXiv:1901.03407

Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: A survey. ACM Computing Surveys 41(3): Article No. 15. DOI: 10.1145/1541880.1541882

Cherepanov FM, Yasnitsky LN (2008) Neural network filter for eliminating outliers in statistical information. Bulletin of Perm University. Series: Mathematics. Mechanics. Computer Science 4:151-155. URL: https://drive.google.com/file/d/1Ss4S8eqZITCl8yZ4Igx0fLXKTPLlkb_I/view

Cortez P, Cerdeira A, Almeida F, Matos T, Reis J (2009) Modeling wine preferences by data mining from physicochemical properties. Decision Support Systems 47(4):547-553. DOI: 10.1016/j.dss.2009.05.016

Ding X, Zhao Y, Akoglu L (2024) Fast Unsupervised Deep Outlier Model Selection with Hypernetworks. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Barcelona, Spain, pp 585–596. DOI: 10.1145/3637528.3672003

Ferdowsi H, Jagannathan S, Zawodniok M (2013) A neural network based outlier identification and removal scheme. In: 2013 IEEE Conference on Prognostics and Health Management (PHM), Gaithersburg, MD, USA, pp 1-6. DOI: 10.1109/ICPHM.2013.6621453

Freeman J, Barnett V, Lewis T (1995) Journal of the Operational Research Society 46(8):1034

Grubbs FE (1969) Procedures for detecting outlying observations in samples. Technometrics 11(1):1–21

Han D, Wang Z, Feng R, Jin M, Chen W, Wang Z, Yang J, Shi X, Yin X, Liu Y (2024) Rules refine the riddle: global explanation for deep learning-based anomaly detection in security applications. In: Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS'24), ACM, New York, pp 4509–4523

Hawkins D (1980) Identification of outliers. Chapman and Hall, London

Liu L, Xie C, Hu W, Li Y (2024) IEEE Access PP(99):1-1. DOI: 10.1109/ACCESS.2024.3427783

Malhotra P, Ramakrishnan A, Anand G, Vig L, Agarwal P, Shroff G (2016) LSTM-based encoder-decoder for multi-sensor anomaly detection. arXiv preprint arXiv:1607.00148

Malmström M, Skog I, Axehill D, Gustafsson F (2022) Detection of outliers in classification by using quantified uncertainty in neural networks. In: 2022 25th International Conference on Information Fusion (FUSION), Linköping, Sweden, pp 1-7. DOI: 10.23919/FUSION49751.2022.9841376

Pace RK, Barry R (1997) Sparse Spatial Autoregressions. Statistics and Probability Letters 33(30):291–297. DOI: 10.1016/S0167-7152(96)00140-X

Pang G, Shen C, Cao L, Hengel A van den (2021) Deep learning for anomaly detection: a review. ACM Computing Surveys 54(2): Article No. 38

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. In: Proceedings of the 10th International Conference on Machine Learning and Applications (ICMLA 2011), IEEE, pp 1–6

Ruff L, Vandermeulen R, Göring N, Deecke L, Siddiqui SA, Binder A, Müller E, Kloft M (2018) Deep one-class classification. In: Proceedings of the 35th International Conference on Machine Learning (ICML 2018). PMLR 80:4393–4402

Ruff L, Kauffmann JR, Vandermeulen RA, Montavon G, Samek W, Kloft M, Dietterich TG, Müller KR (2020) A unifying review of deep and shallow anomaly detection. arXiv preprint arXiv:2009.11732

Ruff L, Vandermeulen RA, Göring N, Deecke L, Siddiqui SA, Binder A, Müller E, Kloft M (2021) Deep anomaly detection with outlier exposure. arXiv preprint arXiv:2111.13627

Sakurada M, Yairi T (2014) Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proceedings of the 2nd Workshop on Machine Learning for Sensory Data Analysis (MLSDA'14), ACM, New York, pp 4–11

Sarvari H, Domeniconi C, Prenkaj B, Stilo G (2021) Unsupervised boosting-based autoencoder ensembles for outlier detection. In: Karlapalem K, Cheng H, Ramakrishnan N, Agrawal RK, Reddy PK, Srivastava J, Chakraborty T (eds) Advances in Knowledge Discovery and Data Mining (PAKDD 2021). Lecture Notes in Artificial Intelligence, vol 12712. Springer, Cham, pp 91–103

Schlegl T, Seeböck P, Waldstein SM, Schmidt-Erfurth U, Langs G (2017) Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: Niethammer M, Styner M, Aylward S, Zhu H, Oguz I, Yap PT, Shen D (eds) Information Processing in Medical Imaging (IPMI 2017). Lecture Notes in Computer Science, vol 10265. Springer, Cham, pp 146–157

Soenen J, Van Wolputte E, Perini L, Vercruyssen V, Meert W, Davis J, Blockeel H (2021) The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods. In: Proceedings of the Workshop on Outlier Detection and Description (ODD'21), Virtual Event, August 15, 2021. ACM, New York, pp 1–8

Yasnitsky LN (2016) Intelligent systems: textbook. Knowledge Laboratory, Moscow. URL: https://cloud.mail.ru/public/5H3z/4LxCpisxw, https://drive.google.com/file/d/1KMx4yQkep2_-PUl2RVMevX06GQkkQF0I/view

Yasnitsky LN (2020) Algorithm for searching and analyzing abnormal observations of statistical information based on the Arnold – Kolmogorov – Hecht-Nielsen theorem. International Journal of Advanced Trends in Computer Science and Engineering 9(2):1814-1819. DOI: 10.30534/ijatcse/2020/139922020.
URL: http://www.warse.org/IJATCSE/static/pdf/file/ijatcse139922020.pdf

Yasnitsky LN, Yasnitsky VL, Alekseev AO (2021) The Complex Neural Network Model for Mass Appraisal and Scenario Forecasting of the Urban Real Estate Market Value That Adapts Itself to Space and Time. Complexity 2021 (Article ID 5392170). DOI: 10.1155/2021/5392170

Yasnitsky LN, Dumler AA, Cherepanov FM, Yasnitsky VL, Uteva NA (2022) Capabilities of neural network technologies for extracting new medical knowledge and enhancing precise decision making for patients. Expert Review of Precision Medicine and Drug Development 7(1):70–78. DOI: 10.1080/23808993.2021.1993595

Yasnitsky LN, Gruzman VM, Martynenko SV, Mezentsev AS (2023) Prediction of Steel Articles Mechanical Properties Through Raw Stock Chemical Composition and Smelting Parameters. In: Proceedings of the 5th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA), Lipetsk State Technical University, Lipetsk, pp 813-816. DOI: 10.1109/SUMMA60232.2023.10349589

Yeh IC (1998) Modeling of Strength of High-Performance Concrete Using Artificial Neural Networks. Cement and Concrete Research 28(12):1797-1808

Zhao Y, Nasrullah Z, Li Z (2019) PyOD: a Python toolbox for scalable outlier detection. Journal of Machine Learning Research 20:1–7

Zenati H, Foo CS, Lecouat B, Manek G, Chandrasekhar VR (2018) Efficient GAN-based anomaly detection. arXiv preprint arXiv:1802.06222

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- 2GeometricinterpretationofNNF.ipynb
- 1TestingallmethodsontheParabolaDataset.ipynb
- 16TestingontheCaliforniadataset.ipynb
- 15Dependenceonthepercentageofoutliers.ipynb
- 14aTestingontheConcreteStrengthdatasetandTime.ipynb.ipynb
- 14TestingontheConcreteStrengthdataset.ipynb
- 13TestingontheConcreteStrengthDatasetOneOutlier.ipynb
- 12TrainingerrorsfordifferentNonthesamescalealongtheverticalaxis.ipynb
- 10TrainingerrorsfordifferentN.ipynb
- 8SuitabilityofNNFforahyperbolicparaboloid.ipynb
- 7Testingonahyperbolicparaboloid.ipynb
- 45Testingonaparaboloidofrevolution.ipynb