Advances in Artificial Intelligence and Machine Learning; Research 4 (2) 2324-2337 Received 7-04-2024; Accepted 21-06-2024; Published 28-06-2024

Experimental Indication to Improve the NN Learning Accuracy by Integrity Constraints From the NN Training Data

Alexander Maximilian Röser alexander_maximilian.roeser@fom-net.de
isf - Institute for Strategic Finance, FOM University of Applied Science
Essen, North Rhine–Westphalia, Germany
István Széchenyi Economics and Management Doctoral School, University of Sopron
Sopron, Hungary

Roman Alexander Englert roman.englert@fom-net.de
New Media and Information Systems, Faculty III, University of Siegen
Siegen, Germany
Computer Science, FOM University of Applied Science
Essen, North Rhine–Westphalia, Germany

Corresponding Author: Alexander Maximilian Röser

Abstract

Various approaches to improve the classification rate of neural networks (NN) exist. Nevertheless, the application of integrity constraints to mitigate this rate is novel. This paper investigates the effectiveness of integrity constraints (ICs or short: constraints) to improve the performance of neural networks (NNs). This applies in particular to data reduction in training data. The study starts with the application of ICs to the initial NN classification, focusing on the development of data set-specific constraints. These constraints are created by machine learning algorithms, such as multiple linear regression. The method consists of applying these constraints to the misclassified data sets from different tests with the aim of reducing the misclassification rate. The effectiveness of this approach is quantified by comparing the original misclassification rates with those after applying the IC, where a significant reduction was observed in three different test cases. For example, one test case can be described as significant: In Test 1, the misclassification rate decreased from 0.78% to 0.19%, which corresponds to a reduction of 75.6%. Similar improvements were observed in the subsequent tests, underlining the potential of ICs to improve classification accuracy. Thus, this study provides convincing evidence that the NNIC approach is a valuable tool for mitigating misclassification problems in neural network applications. The combination of training a NN and subsequently apply ICs from the training data (NNIC approach) is new and the experimental results indicate evidence for improving the classification rate.

Keywords: artificial intelligence, neural networks, integrity constraints

1. INTRODUCTION

Training data for AI systems lack often in integrity, since they are biased [1]. There exist only a few approaches to mitigate bias in data, e.g. [2], provide an approach that searches and adds missing knowledge to improve the quality of training data. The challenge of deficient data can also rely on the size of the data set, or with other words an inadequate number of (positive) examples. Various approaches to improve the classification rate of neural networks (NN) exist. Nevertheless, the application of integrity constraints to mitigate this rate is novel. The presented approach NNIC, to apply integrity constraints (short: IC) to neural networks (short: NN), mitigates this problem by adding so-called integrity constraints to strength positive examples, and thus, ensures the exclusion of negative examples during training.

Integrity constraints are expressions that ensure the correctness and consistency of the data [3, 4]. In this paper, a semi-automatic approach is proposed to reduce biases in the training data for NN. ICs in general cannot be logically derived from data: Training and data contain examples of general rules, the ICs are [4]. The NNIC approach consists of two phases: First, the most promising variables (aka attributes) are automatically selected from the data, then these attributes are combined into ICs. The last step may be supported by a human expert.

To evaluate the effectiveness of the ICs, the mushroom data set [5], is applied to the NNs and the extracted ICs. This data set contains various attributes of fungi, that are aimed to reason the variable edible. For this purpose, three NNs are trained with training data sets of different sizes. After training, each NN is evaluated with the corresponding test data set to determine the misclassification rate for each NN. In order to improve the classification results, the misclassified instances are identified. Subsequently, a semi-automatic check is carried out to determine which attributes lead to a correct classification of the variable edible. These attributes are combined to ICs. Hence, using ICs, the number of incorrectly classified data can be reduced. As result, the application of ICs lowers the misclassification rate, especially for training data sets with a smaller number of instances.

The paper is organized as follows: Related work for NNs and ICs is presented in Section 2. Then, in Section 3 the NNIC approach is described including basic definitions. Section 4 contains a description of the data set mushroom for the evaluation. Subsequently, two evaluations are performed and discussed: First, training NNs with the mushroom data set (Section 5), and second, applying the NNIC approach to the trained NN and evaluation of the performance increase (Section 6). The paper concludes with a summary and an outlook for future research.

2. RELATED WORK

The related work can be divided into two main components: The first section focuses on the quality of neural network models, their motivation and evaluation. In the second section, we provide an overview of constraints in the forehand data science context.

2.1 Quality of Neural Network Models

An artificial neural network can analyze data patterns to solve various types of problems [6, 7]. To use this technology, it has to be decided which challenge should be solved, for example, there are different types of problems such as classification problems or time series prediction [6, 8]. After selecting the model, the database should be divided into training and test data. In general, training data is important to analyse and learn patterns, while test data is essential to prove that the model detects the right patterns [8, 9].

To ensure model quality is one of the major challenges in implementing a NN. Model quality can be divided into data quality and real model performance. The next paragraph provides a brief overview of data quality, followed by real model performance.

For the implementation of a standardized NN, the GIGO paradigm, also known as the ”garbage in, garbage out” paradigm [10], plays an important role for data quality. This means that the quality of the input data must be good to get a good result, and if training is done on a biased or degraded data set, it will lead to worse results. Therefore, most NN implementations require a large and unbiased data set. Other approaches can be used to improve data quality, such as Principal Component Analysis (PCA). PCA is a statistical technique that eliminates redundancies within a high-dimensional vector to reduce the dimensions of a given data set [11].

Beyond data quality considerations, the performance of the model must be examined more closely. Several approaches can be used to validate the performance of a given model. For instance, precision (also known as positive predictive value) and recall (also known as sensitivity) are common performance metrics used to evaluate model performance. In addition to the specified performance metrics, a comparative analysis of different typologies of neural networks is also possible. These networks can be distinguished by their respective optimizations, activation functions, the size of the input and output dimensions, and the number of hidden layers, each of which has its own strengths and weaknesses. Therefore, before implementing an NN, a decision has to be made about the suitability of the NN topology for the given problem. If there are different possible topologies for a given problem, e.g., a classification problem can be solved by applying, both, a convolutional [12], and a recurrent NN [13], the results can be compared via a confusion matrix. To select the best model, GridSearchCV can be used to compare the different model results and select the best model according to the given problem [14]. In addition to the network topologies, the above-mentioned performance indicators can also be included [15]. After getting the results from the model with the best parameters, it is important to cross-validate the model. Cross-validation allows to check if the model has memorized the given training data or if the model recognizes a true correlation, such as a pattern. Only in the second case the specific model can be used to solve a given problem. During this process, the model is trained and tested with different training data to prove if the result will be the same as before doing the cross-validation [15, 16].

According to insights derived from performance metrics, the application of NN in different contexts inevitably leads to some degree of misclassification in predicted outcomes. To mitigate this problem and improve the accuracy of predictions, especially when working with a limited data set, the implementation of constraints can be a strategic approach.

2.2 Constraints in Logic and NN

Constraints are well-known from the fields of databases and inductive logic programming [3, 17, 18], and numerous applications [4, 19, 20]. In logic, constraints are applied as clauses and as an example, we consider the following financial implication [3]:

$$\text{credit} \rightsquigarrow \text{credit\_worthy} \tag{1}$$

which ensures that, if someone has a credit, then s/he has a credit worthiness.

Constraints are applied to broad applications: Šestak and Turkanović [4], discuss the use of constraints for graph databases and their impact on causality. Yang et al. [20], consider the management of constraints for Web applications that rely on databases. Salehi et al. [21], propose a formal and more exhaustive classification of constraints in spatiotemporal databases relying on space and time. And Linares-Vásquez et al. [22], elaborate the usage of database tables and attributes and schema constraints in database-centric applications, to name a few applications of constraints in different fields.

For NN constraints can also be applied: Usman [23], provides a constraint-based technique for repairing NN classifiers. The technique aims to fix the logic of the network at an intermediate layer or at the last layer. Their approach first uses fault localization to find potentially faulty network parameters (such as the weights) and then performs repair using constraint solving to apply small modifications to the parameters to remedy the defects. And He and Sarangapani [24], investigate and develop reinforcement learning NN-based controller, where constraints in the input of the training data tackle non-linearity in the time-based data. In contrast to their approach constraints are applied in the following NNIC approach to the output of the NN in order to improve the learning result and to reduce misclassification.

The NNIC approach in the subsequent section is based on NN and constraints. Both methods were considered separately in this section, since this kind of combining these both methods is novel as depicted in the NNIC approach.

3. THE NNIC APPROACH

The idea of the NNIC approach is to train a NN and subsequently to reduce classification errors by excluding incorrect learned patterns using ICs (FIGURE 1). The proposed approach consists of the following steps:

NN training: The NN is initially trained using a given dataset.

Identification of misclassified data: After the NN has been trained, instances that have been misclassified by the NN are identified. These misclassified instances serve as the basis for the generation of the ICs.

IC generation: The IC generation step uses pattern recognition techniques to identify common attributes of misclassified data. These patterns are then rewritten as constraints.

Application of ICs to misclassified data: The formulated ICs are then applied to the classification result in order to eliminate misclassified data.

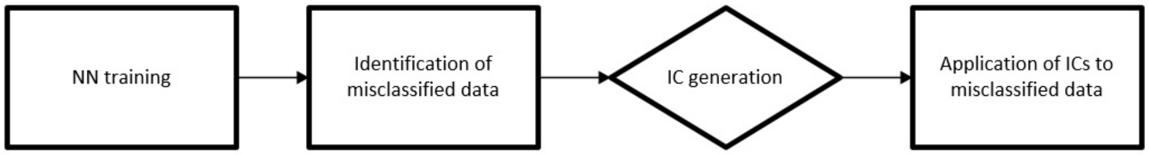
NN training
Identification of misclassified data
IC generation
Application of ICs to misclassified data

Figure 1: The NNIC approach to reduce errors in trained NNs.

ICs are based on clauses, which are defined as [17]:

Definition 1 *Clause.*

$$\mathrm{c}_1 \vee \ldots \vee \mathrm{c}_\mathrm{k}$$

where c_i are (negated) propositions.

Hence, we apply ICs from the propositional logic [17], to the NN:

Definition 2 *Integrity Constraint (short: constraint).*

$$\mathrm{p}_1 \vee \ldots \vee \mathrm{p}_\mathrm{n} \rightsquigarrow \mathrm{c}_1 \wedge \cdots \wedge \mathrm{c}_\mathrm{m}$$

where $p_i \quad (i = 0, \ldots, n)$ and $c_k \quad (k = 0, \ldots, m)$ are propositions.

An integrity constraint is a logical implication, where the premise or the conclusion can be empty and/or several conclusions may exist (a conclusion is a proposition). An empty premise means $true$ and an empty conclusion means $false$. As an example consider Eq. 1. The expressiveness of ICs exceeds that of rules, since they can represent both disjunctive and negated knowledge (see also Englert [3]). Note, the IC $a \vee b \rightsquigarrow c \wedge \neg d$ can be rewritten as $\neg a \vee \neg b \vee \neg c \vee d$, since $a \rightarrow b \Leftrightarrow \neg a \vee b$ and $\neg(a \vee b) \Leftrightarrow \neg a \wedge \neg b$.

In order to get ICs from a training sample, incorrect patterns are generalized as follows: In a first step they are marked automatically and then, in a second step the most frequent common attributes of the marked patterns build the ICs. These ICS may be revised by an expert in an optional step. Thus, NNIC is a semi-automatic approach. An expressive example is depicted and evaluated in the following sections.

4. MUSHROOM DATA SET

To evaluate the approach of using NNICs, a data set called mushroom was used. The given data set contains 8124 instances and 22 categorical attributes to describe 23 species of gilled mushrooms

Table 1: NNIC evaluation mixing rations related to the complete data set

Test	Training data	Test data	Rest data (fixed)
1	76.0%	19.0%	5.0%
2	72.5%	22.5%	5.0%
3	47.5%	47.5%	5.0%

of the families Agaricus and Lepiota as edible, poisonous or of unknown edibility and not recommended. The latter is also classified as poisonous. In conclusion, there are two labels known as poisonous and edible. As mentioned before, the other attributes are categorical [5]. To train a NN model, the classification labels of poisonous' and edible' were defined. The attributes were encoded using the one-hot method, where each value of an attribute is assigned a boolean value. For instance, the categorical attribute gill size' had two possibilities: broad (b)' and 'narrow (n)'. After one-hot encoding, there were two attributes: gill-size_broad" and gill-size_narrow," with a value of True (1) or False (0). This encoding prepared the data set for use in a NN classification, determining whether a mushroom was poisonous or edible. Once the data set was prepared, PCA was used to identify the attributes describing the highest variance. In this case, the explained variance was set to 95% which means that the adjusted data set contains only the values with a significant influence, in other words, as many attributes as were needed to reach the threshold of 95% explained variance. This step reduced the number of attributes (119 attributes after one-hot coding) to the significant ones (39 attributes after using PCA). The purpose of using the NNIC method is to minimize the data set while maintaining a high level of performance. After implementing NNIC, the amount of misclassified data with a reduced data set should be equal to or slightly worse than without the NNIC method.

5. EVALUATION OF THE MUSHROOM DATA SET WITH NNs

In order to establish a uniform testing procedure, the given data set was split into a 5% portion, called the "rest data" and a 95% portion, which was split differently between training and test data sets depending on the test. Although there are training and test data in the 95% portion, the rest of the data can be used to backtest the NNIC with an before unseen data sample. After dividing the data set, three different mixing ratios were defined as follows:

Three different tests were set up for the evaluation. Each of these tests has 5% rest data and a decreasing amount of training data accompanied by an increasing amount of test data.

Several tests were performed to evaluate the NNIC approach. The first NN was run with a large amount of training data, using 76% of the data for training. In the second test, the training data was slightly reduced to 72.5%. The third test was run with less than half of a complete data set. The rest of the data was fixed to 5% of the full data set, as mentioned above.

After implementing different tests, every test was performed due to a, for the test standardized, multi-layer perceptron (MLP). The hyperparameter optimization was also performed with the same selection of hyperparameters using GridSearchCV. It is noticeable that the hyperparameters for Test 2 & 3 remained constant compared to Test 1:

- `'activation': 'relu'`
- `'hidden_layer_sizes': (128, 64)`
- `'solver': 'adam'`

In Test 1 they are different:

- `'activation': 'tanh'`
- `'hidden_layer_sizes': (64, 32)`
- `'solver': 'adam'`

Further analysis was conducted using the topology based on the hyperparameters above. A confusion matrix was generated for each test after performing 5-fold cross-validation to demonstrate the classification's performance.

After classifying each NN without the NNIC, the results of the tests were presented as follows: For each test, a confusion matrix shows the rate of true/false positive and negative classifications. In order to compare the given results of the tests, a combined confusion matrix is implemented that shows n, the sum of misclassified data, and calculates the false classification rate for each test situation by dividing the sum of misclassified data by n. This combined confusion matrix is used to compare the results of the tests. This combined confusion matrix will be used to compare the results of Test 2. The first test, with a training data ratio of 76.0%, resulted in 824 true positives and 708 true negatives for the classification of edible and poisonous mushrooms respectively, with a total of 12 misclassified instances, leading to a false classification rate of 0.78%, see also FIGURE 2.

When the amount of training data in the second test is reduced and combined with the first test, the performance of the neural network decreases, see FIGURE 3. With 72.5% training data, true positives decreased to 319 and true negatives to 1449, while misclassified instances increased to 60, resulting in a higher false classification rate of 3.28% in TABLE 2.

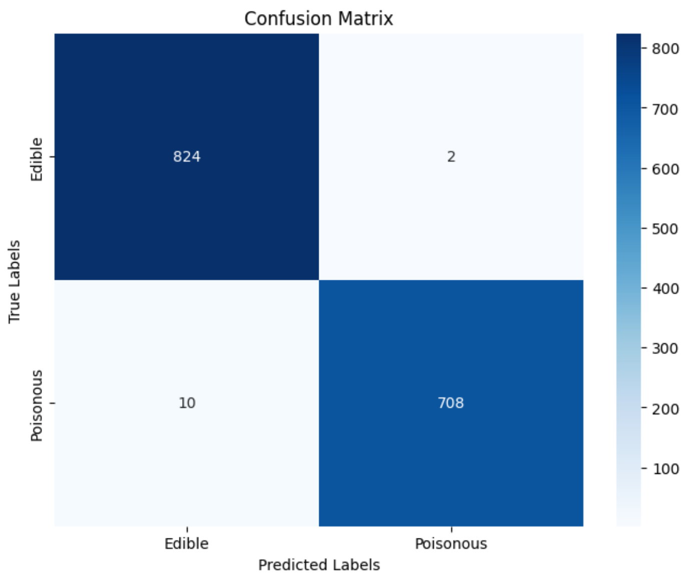

Figure 2: Confusion matrix of the first data set

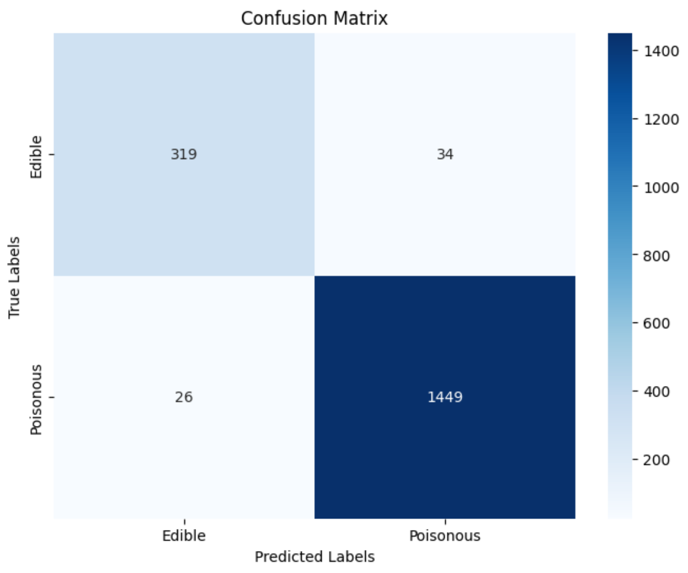

Figure 3: Confusion matrix of the second data set

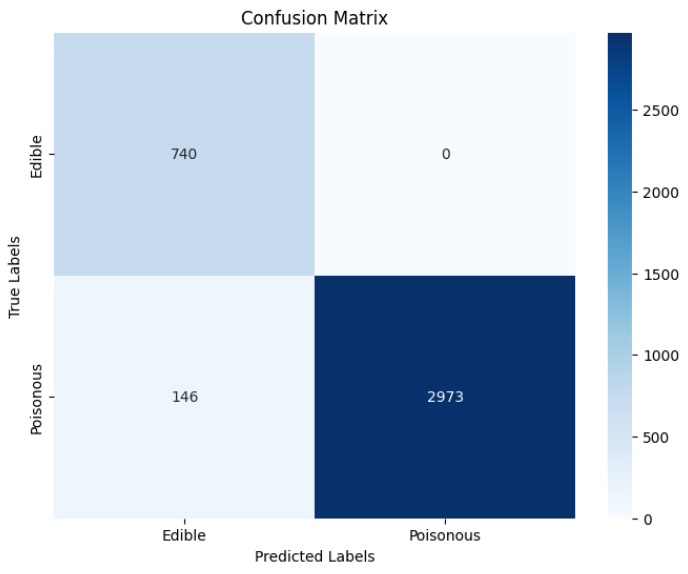

Figure 4: Confusion matrix of the third data set

Furthermore, the misclassification rate increases even more, up to 3.78% in the final test, where the training set is further reduced, see FIGURE 4, and TABLE 2. In this third test, with an equal training and test data ratio of 47.5%, the system correctly classified 740 edible and 2973 poisonous instances, but misclassified 146 instances, increasing the false classification rate to 3.78%. Reducing the size of the training data for the neural network increases the false classification rate, as shown by these results. Therefore, minimising a data set to save resources is likely to result in poor model performance, even if the misclassification rate remains below 4%.

Table 2: Results of the classification matrices of the three tests.

true label/predicted label	Test 1	Test 2	Test 3
p/p	708	1449	2973
p/e	10	26	146
e/p	2	34	0
e/e	824	319	740
n	1544	1828	3859
sum of misclassified data	12	60	146
false classification rate	0.78%	3.28%	3.78%

To address the research problem of reducing the data set while achieving comparable results, the following mushroom experiment for the NNIC approach will be implemented in the upcoming section.

6. APPLYING THE NNIC APPROACH TO THE MUSHROOM DATA SET EVALUATION AND BACKTESTING

A systematic approach was used to determine the effectiveness of ICs in improving the performance of NN in the context of data reduction. After initial classification by the NN, ICs were created by data set-specific rules or by data set-specific patterns. This process can be performed using either machine learning algorithms, such as multiple linear regression, or using the expertise of a human expert which requires additional effort and may lack in the availability of domain-specific human experts, and thus, should be avoided.

The process described below is semi-automated. To understand how this process works, it is described using the misclassified data set from Test 3. The misclassified data set (extract from the TABLE 3) contains 82 columns and 148 rows. The first row contains the column headers/variables and the last row automatically calculates the sum of the $True$ bool values. The dependent variables such as *class_edible* and *class_poisonous* are specified in the first two columns. The variables *formulation of the ICs* and *test of the ICs* are declared in the last two columns.

Table 3: Table of Misclassified Values in Test Case 3 Extract

class_edible	class_poisonous	cap-shape_bell	…	gill-size_narrow	gill-color_black	Formulation of the ICs	Testing the ICs
False	True	True	…	False	False	x	x
False	True	False	…	True	False	x	x
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
NaN	NaN	NaN	…	NaN	NaN	NaN	NaN
0	146	48	…	47	0	104	104

When creating the ICs, a detailed analysis of the predominant variables was conducted, leading to recognizable patterns. Examination of the data set showed that including a specific combination of variables — namely *odor_none*, *gill-attachment_free*, and *gill-spacing_close* — reduced the misclassification frequency from 146 to 42, as described in Formula 2. This Formula indicates that the IC clause column is marked as x if the value $True$ is assigned to these variables. It thus serves as a marker if the conditions are met for the current row. Here, x signifies that all conditions are $true$. Additionally, another column was added specifically for backtesting purposes, crucial for determining whether the set of independent variables effectively explains the dependent variable labeled *class_poisonous* (see Formula 3). The same set of variables, including the dependent variable, was tested to assess if the IC conditions also explain the dependent variable.

Formula 2: Formulation of the ICs

$$(\text{odor\_none} \land \text{gill-attachment\_free} \land \text{gill-spacing\_close}) \rightsquigarrow x \tag{2}$$

Where *odor_none*, *gill-attachment_free*, and *gill-spacing_close* are propositions.

Formula 3: Testing the ICs

$$\begin{gathered}(\text{class\_poisonous} \wedge \text{odor\_none} \wedge \\ \text{gill-attachment\_free} \wedge \text{gill-spacing\_close}) \rightsquigarrow x\end{gathered} \tag{3}$$

Where *class_poisonous*, *odor_none*, *gill-attachment_free*, and *gill-spacing_close* are propositions.

To reduce misclassification of data sets, a thorough pattern analysis of existing misclassified data sets was performed for each test. The effectiveness of the NNIC approach was assessed by calculating the net reduction in misclassified data, which was achieved by subtracting the number of correctly classified data after applying IC from the original total number of misclassified data for each test. In addition, the reduction in misclassification rate (MR) due to the application of IC was quantified for each test as described in the Formula 4.

Formula 4: Performance Test of NNICs

$$\text{Reduction of MR} = \left(\frac{\text{MR (NN)} - \text{MR after ICs}}{\text{MR (NN)}}\right) \tag{4}$$

Test 1: $\left(\frac{0.78\%-0.19\%}{0.78\%}\right) = 75.6\%$

Test 2: $\left(\frac{3.28\%-2.08\%}{3.28\%}\right) = 63.4\%$

Test 3: $\left(\frac{3.78\%-1.09\%}{3.78\%}\right) = 71.2\%$

In the Formula shown, *MR (NN)* stands for the misclassification rate (MR) before applying the NNIC approach. In contrast, *MR after ICs* stands for the MR after the application of ICs. The proportional reduction in MR due to ICs is calculated by dividing the difference between the current rate by the original MR. This is a quantifiable measure of the improvement in classification accuracy achieved by incorporating ICs into the NN framework.

In Test 1, the use of ICs led to a significant reduction in the misclassification rate from 0.78% to just 0.19%, a reduction of 75.6%. This significant reduction underlines the profound effect of ICs in reducing misclassification.

Similarly, the misclassification rate in Test 2 dropped significantly from 3.28% to 2.08%, an improvement of 63.4%. This underlines the crucial role of ICs in reducing misclassifications.

Test 3 also showed the effectiveness of the ICs with a reduction in the misclassification rate from 3.78% to 1.09%, a reduction of 71.2%. Again, the effectiveness of ICs in improving classification accuracy is remarkable.

Table 4: Results of the Classification Matrices of the Three Tests with the NNIC Approach

Feature	Test 1	Test 2	Test 3
n	1544	1828	3859
Sum of Misclassified Data (NN)	12	60	146
Misclassification Rate (NN)	0.78%	3.28%	3.78%
Correct Classification of Misclassified Data (ICs)	9	22	104
Misclassification Rate after ICs	0.19%	2.08%	1.09%
Reduction in Misclassification Rate	75.6%	63.4%	71.2%

TABLE 4 summarizes the results of the classification matrices for three different tests. It can be seen that the IC application has significantly improved the classification accuracy. This is particularly evident in the significant reduction in misclassification rates in all tests from 0.78%, 3.28% and 3.78% to 0.19%, 2.08% and 1.09% in Tests 1, 2 and 3, respectively. The corresponding reductions in misclassification rates are 75.6%, 63.4% and 71.2%, highlighting the effectiveness of ICs in improving the classification accuracy of NN.

The analysis shows that a reduction in the test data set is associated with an increase in the misclassification rate in a classical NN model. However, applying ICs to misclassified data after classification leads to a reduction of these misclassified data sets, which underlines the effectiveness of the NNIC approach in classification scenarios. The results of Test 3 are particularly significant. With an equal distribution of 47.5% for training and test data, this Test represents an NN with a significantly reduced proportion of training data. By using the IC in this scenario, the misclassification rate reaches a comparable level to Test 1, where a training data share of 95% was used.

7. OUTLOOK

The NNIC approach achieved in the mushroom experiment a similar classification result with a given data reduction compared to the application of a unreduced data set. This indicates that the NNIC approach gains low misclassification rates for small training data sets.

To further validate the NNIC approach and to prove its generalizability, more experiments are aimed. Furthermore, it could be applied to the above-mentioned ”residual data” in further investigation steps. With the same process steps, the application of the NNIC approach should lead to a reduction in the number of misclassified data in each data set. In this way, the robustness and adaptivity could also be validated on completely unseen data.

Building on the empirical evidence for the effectiveness of ICs on NN performance presented in this study, future research can expand the scope of the NNIC approach. The next research steps will include comprehensive evaluations of additional data sets to corroborate the results of the initial tests and further quantify the impact of ICs on data reduction scenarios. This may include investigating the focus on different data types to answer different classification problems.

Efforts will also focus on moving from the current semi-automated approach to a fully automated NNIC approach. This evolution will include the development of advanced machine learning algorithms capable of autonomously generating and applying NNIC with minimal human supervision. The expected progress will not only increase the efficiency of NNICs, but also represent a step towards self-improving models that adapt to different data conditions.

The overall goal is to create an NNIC approach that can be dynamically applied to different data sets, reducing the dependency on experts. In this way, the NNIC approach could revolutionize the way neural networks are trained and adapted, paving the way for more autonomous, accurate and resource-efficient machine learning solutions.

References

[1] Russell S, Norvig P. AI a Modern Approach. Learning. 2005;2:4.

[2] Englert R, Muschiol J. Training Data Improvement by Automatic Generation of Semantic Networks for Bias Mitigation. Am J Inf Sci Technol. 2022;6:1-7.

[3] Englert R. Inducing integrity constraints from knowledge bases. In: KI-95: advances in artificial intelligence: 19th Annual German Conference on Artificial Intelligence. Springer. 1995:77-88.

[4] Šestak M, Turkanović M. Extended Property-Level K-Vertex Cardinality Constraints Model for Graph Databases. J King Saud Univ Comput Inf Sci. 2023;35:126-138.

[5] `https://archive.ics.uci.edu/ml/datasets/mushroom`.

[6] Basheer IA, Hajmeer M. Artificial Neural Networks: Fundamentals, Computing, Design, and Application. J Microbiol Methods. 2000;43:3-31.

[7] Zurada JM. Introduction to Artificial Neural Systems. St Paul, MN: West Publishing Company. 1992.

[8] Patterson J, Gibson A. Deep Learning: A Practitioner's Approach. 1st ed. Sebastopol, CA: O'Reilly Media, Inc. 2017.

[9] Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-Learn: Machine Learning in Python. J Mach Learn Res. 2012;12:2825-3280.

[10] Kilkenny MF, Robinson KM. Data Quality: "Garbage In – Garbage Out". Health Inf Manag. 2018;47:103-105.

[11] Kambhatla N, Leen TK. Dimension Reduction by Local Principal Component Analysis. Neural Comput. 1997;9:1493-1516.

[12] Le Guennec A, Malinowski S, Tavenard R. Data Augmentation for Time Series Classification Using Convolutional Neural Networks. In: ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data. 2016.

[13] Kalaiselvi K, Karthikeyan C, Shenbaga Devi M, Kalpana C. Improved Classification of Brain Tumor in Mr Images Using Rnn Classification Framework. Int J Innov Technol Explor Eng. 2020;9:1098-1101.

[14] Kumari HMV, Suresh DS, Dhananjaya PE. Clinical Data Analysis and Multilabel Classification for Prediction of Dengue Fever by Tuning Hyperparameter Using Gridsearchcv. In: 14th International Conference on Computational Intelligence and Communication Networks (CICN). Saudi Arabia. IEEE PUBLICATIONS. 2022;302-307.

[15] Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, et al. API Design for Machine Learning Software: Experiences From the Scikit-Learn Project. Arxiv Preprint: https://arxiv.org/pdf/1309.0238

[16] Krogh A, Vedelsby J. Neural Network Ensembles, Cross Validation, and Active Learning. Adv Neural Inf Process Syst.1994;7:231-238,

[17] Lloyd JW. Foundations of Logic Programming. Springer Science & Business Media. 2012.

[18] Morik K, Kietz BE, Emde W, Wrobel S. Knowledge Acquisition and Machine Learning: Theory, Methods, and Applications. Knowl Based Syst. 1993.

[19] King RD, Srinivasan A, Muggleton S, Feng C, Lewis RA, et al. Drug Design Using Inductive Logic Programming. In Proceedings of the Twenty-sixth Hawaii International Conference on System Sciences. 1993;1:646-655.

[20] Yang J, Sethi U, Yan C, Cheung A, Lu S. Managing Data Constraints in Database-Backed Web Applications. In Proceedings of the ACM/IEEE. Acad Med 42nd International Conference on Software Engineering. 2020:302-303.

[21] Salehi M, Bédard Y, Mostafavi MA, Brodeur J. Formal Classification of Integrity Constraints in Spatiotemporal Database Applications. J Vis Languages Comput. 2011;22:323-339.

[22] Linares-Vásquez M, Li B, Vendome C, Poshyvanyk D. Documenting Database Usages and Schema Constraints in Database-Centric Applications. In Proceedings of the 25th International Symposium on Software Testing and Analysis. 2016:270-281.

[23] Usman M, Gopinath D, Sun Y, Noller Y, Păsăreanu CS. Nnrepair: Constraint-Based Repair of Neural Network Classifiers. In: Silva A, Rustan K, Leino M, editors. Computer Aided Verification. Cham. Springer International Publishing. 2021: 3-25.

[24] He P, Jagannathan S. Reinforcement Learning Neural-Network-Based Controller for Nonlinear Discrete-Time Systems With Input Constraints. IEEE Trans Syst Man Cybern B Cybern. 2007;37:425-436.