I.J. Modern Education and Computer Science, **2021, 3, 13-22**
Published Online June 2021 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2021.03.02

Artificial Neural Network Training Criterion Formulation Using Error Continuous Domain

Zhengbing Hu
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
Email: drzbhu@gmail.com

Mykhailo Ivashchenko
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine
Email: mivaschenko_51@lll.kpi.ua

Lesya Lyushenko
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”
Email: lyushenkol@gmail.com

Dmytro Klyushnyk
Dnipro State Agrarian and Economic University
Email: dmitriy.klyushnyk@gmail.com

Received: 08 March 2021; Accepted: 26 April 2021; Published: 08 June 2021

Abstract: One of the trends in information technologies is implementing neural networks in modern software packages [1]. The fact that neural networks cannot be directly programmed (but trained) is their distinctive feature. In this regard, the urgent task is to ensure sufficient speed and quality of neural network training procedures. The process of neural network training can differ significantly depending on the problem. There are verification methods that correspond to the task’s constraints; they are used to assess the training results. Verification methods provide an estimate of the entire cardinal set of examples but do not allow to estimate which subset of those causes a significant error. This fact leads to neural networks’ failure to perform with the given set of hyperparameters, making training a new one time-consuming.

On the other hand, existing empirical assessment methods of neural networks training use discrete sets of examples. With this approach, it is impossible to say that the network is suitable for classification on the whole cardinal set of examples.

This paper proposes a criterion for assessing the quality of classification results. The criterion is formed by describing the training states of the neural network. Each state is specified by the correspondence of the set of errors to the function range representing a cardinal set of test examples. The criterion usage allows tracking the network’s classification defects and marking them as safe or unsafe. As a result, it is possible to formally assess how the training and validation data sets must be altered to improve the network’s performance, while existing verification methods do not provide any information on which part of the dataset causes the network to underperform.

Index Terms: Neural network, neural network training, neural network verification, reachability set, verification criterion.

1. Introduction

One of the widespread tasks that are solved using neural networks is classification [2] and clustering [3]. The solution to any classification problem is a bijective relationship, “example – class.” Such a solution can be obtained by using the following types of neural networks: perceptron [4], convolutional neural networks [5], residual neural networks [6], recurrent neural networks [7]. As new architectures of neural networks emerged, the need to assess their performance formulated a specific neural network verification problem.

Solutions to the problem of verification of neural networks [8] have developed rapidly, appearing in the context of the need to build a criterion for neural networks training. Currently, a series of approaches are used to solve the verification problem [9]. However, it is not resolved in general, and the methods do not involve any analysis of the specifics of data distribution within the input and output data sets. Until recent years, the process of quality analysis had

been based on empirical approaches, which can not sufficiently guarantee that the obtained classification result proves that the neural network has been trained. More novel approaches use reachability analysis [10, 11] to solve the verification problem. Even though the methods' assessment quality surpasses one of the empirical metrics, it is not complete.

Reachability methods mark the network as safe or unsafe to use but do not provide any characteristics on which subsets of examples cause the network to falter. This paper reviews some of the latest reachability analysis methods and aims to bypass their limitation this limitation. It presents a criterion that allows identifying which clusters cause the network to falter. We believe this criterion can be used individually and as a tool to enhance the existence of neural network assessment methods.

2. Background

2.1 Perceptron

Perceptron is one of the simplest mathematical models of neural networks. It consists of several fully connected layers of neurons (synapses connect each neuron of one layer with all neurons of the previous and next layers). The learning process is carried out using direct and inverse propagation approaches.

Formally, each of the hidden layers of the perceptron is defined as follows:

$$L^{(k)} = ReLU(W_{kk-1} \times R_{k-1} + b_k)),$$

where ReLU is the layer activation function (rectified linear unit [12]) applied to each of the neurons, W_{kk-1} is a matrix of weights (represents the synapses that connect the layers numbered k and k − 1), R_{k-1} – the result of the previous layer activation, b_k – bias vector.

We used the described perceptron representation in the experiments. The architectures of the used models are described in section 5.

2.2 Reachable set of a neural network

Suppose we have the input data described as a convex polytope as:

$$L \triangleq \{x \mid Ax \leq b, x \in \boldsymbol{R}^n\}$$

and an *n*–layer perceptron $P \triangleq \{ L_1, \dots , L_n\}$.

Then the reachable set of the perceptron *P* obtained by processing the original set I can be described as follows:

$$R_{L_i} \triangleq \{r_i \mid r_i = ReLU\big(W_k x_{i-1} + b_i), x_{i-1} \in R_{L_{i-1}}\big\},$$

where the reachable set $\mathcal{R}_{L_n}$ contains the classification results of *P* over the set *I*.

The reachable set of the last layer represents the output of the neural network. The quality of the output is assessed according to the safety property, which is individually formulated for every particular task.

2.3 Safety property

Suppose we have an *n*–layer perceptron *P* and a set of linear constraints *SP*, which is superimposed on the reachable set obtained by calculations:

$$\mathrm{SP} \triangleq \{\mathrm{x} \mid \mathrm{C}_{\mathrm{R_i}} \leq \mathrm{d}\}.$$

Perceptron *P* (and the neural network model in the general case) can be considered safe if the final set does not intersect the set caused by *SP*. Otherwise, the model is called unsafe to use and cannot be used to solve the problem. The *SP* set of linear constraints is called a safe property.

3. Related Work

Consider several existing methods for verifying neural networks that have been developed.

3.1 Satisfiability modulo-theories–based approach

This method was developed by scientists at Stanford University (Stanford, USA) and involved a modified simplex method with ReLU-constraints [13].

The idea is to rebuild a set of input data (in the article – atoms) into a logical structure. The following algorithm performs logical propagation, logical partitioning, and logical conflict resolution (Boolean propagation, case-splitting, and Boolean conflict resolution). The algorithm then encodes the ReLU operations using disjunctions. As a result, a deep neural network with n ReLU nodes is transformed into $2n$ subproblems. Each subproblem is an atom disjunction.

The encoding of a ReLU operation can be implemented by creating a pair of variables (in the article – v^b and v^f) and performing a ReLU operation (v^b, v^f). v^b, a backward-facing variable, links v and nodes from the previous network layer. v^f is a forward-facing variable, links v, and nodes from the following network layer.

This method allows formulating the constraints the network needs to satisfy during the computational process. The final assessment is drawn as a result of the output analysis. SMT solvers provide the information on whether the network is safe to use or not tracking if any of the constraints have been violated. However, the method does not describe what data made the network to underperform.

3.2 Star set approach

This algorithm describes an approach that constructs both an accurate and approximated set. It was developed by scientists at Vanderbilt University and the University of Pennsylvania (USA) [14].

Star set can be formally defined as follows:

$$S = < c, V, P >,$$

where $c \in \mathbf{R}^n$ is a center, $V = \{v_1, v_1, \ldots, v_m\}$ is a basis in $\mathbf{R}^n$, $P(\alpha) \triangleq C\alpha \leq d$, where for P linear constraints, $c \in \mathbf{R}^{p \times m}$, $\alpha = [\alpha_1, \ldots, \alpha_m]^T$, $d \in \mathbf{R}^{p \times 1}$.

The algorithm described in the article states that for a layer of the neural network L, which has n neurons, the reachable set can be calculated by sequential execution of n stepReLU–operations:

$$R_L = \mathrm{ReLU}_n(\mathrm{ReLU}_n - 1(\ldots \mathrm{ReLU}_1(S)))\ldots,$$

where S is a star set of the previous layer.

Suppose the final star set has an intersection with a safety property. In that case, the neural network is considered unsafe for solving the respective classification task.

The method allows determining if the network can be used for the given task. However, when the network is marked as 'unsafe', the algorithm does not give any insight into what data caused the obtained result.

3.3 Relevance

We see that verification of neural networks is one of the new branches of artificial intelligence that has developed rapidly. The presented methods prove the effectiveness of their work. However, none of them explains why the obtained neural network model does not satisfy the established limitations. The methods allow assessing if the network is suitable to use for the particular task. However, they do not provide any information on the training data specifics that cause the flaws if the network breaks the formulated constraints.

This paper aims to find a possible resolution for this matter. It proposes a criterion that assigns a status to the assessed network. This status is determined by the network's performance on the test set. It allows identifying which examples cause the highest error. The criterion is tested and evaluated on an artificially modeled problem.

4. Task Statement

The task is to form a criterion for assessing the level of quality of neural network training.

4.1 Problem formulation

We formulate the error function as follows:

$$f(I, P) \leq e,$$

where f is the error function, I is the input set, P is the current status of the network (in terms of training process). Under the classification problem, we will understand the following – we have a finite set of objects, each of which is identified by a particular class from an available set of classes. This set of objects will be called a training data set – D_{train}. It is necessary to build an algorithm that can match a certain class to each of the examples that are not part of the training set. This set of examples will be called a test set – D_{test}. To increase the network's accuracy, we add a validation set D_{CV} to the training process, a set of examples. We will use it to assess the accuracy of classification during the training locally. The neural network that is used will be a perceptron.

The notation is the following: I – a set, $I^{(i)}$ – the i^{th} element of I, $I_j^{(i)}$ – j^{th} component of the i^{th} element of I.

The task of training a neural network includes: function range $G(\mathbf{X})$, the error boundary value ε, sets of examples D_{train}, D_{cv}, D_{test}, a set of classes C. the error boundary value ε is established for each classification task and is constant over the whole $G(\mathbf{X})$.

To form a criterion for assessing the neural network's training (by error analysis), we describe the conditions determining training quality. We decided to come up with notions that could describe the training results the same way as *underfitting* and *overfitting* do. However, the criterion's goal would be to evaluate the results taking into account the continuous domain for the given dataset. By 'continuous domain', we mean the abstract set that includes the dataset itself and all possible examples that could be obtained using interpolation on the existing ones. As an example, imagine a two-dimensional plane with points in it. If the points' coordinates x, y belong to an interval of $[a, b]$, every point would resemble an example from the dataset; the continuous domain would be represented by the rectangle, which vertices' coordinates belong to the interval of $[a, b]$. According to this, we described four possible training outcomes:

- When the network has not been trained and cannot be used for the given task (the average error on the given dataset exceeds the boundary value ε approaching infinity).
- When the network has been trained for a certain number of epochs on the given dataset, but there are still examples that cause the error above the value of ε.
- When the network has been trained for a certain number of epochs on the given dataset, and the classification error of no examples exceeds the value of ε. Besides, the network must generalize (the error on the examples that are not included in the dataset should not exceed the value of ε).
- When the net has been trained for a certain number of epochs on the given dataset, and the classification error of no examples exceeds the value of ε. However, when classifying the examples from the continuous domain that are not present in the dataset, the average error will be higher than the value of ε.

The criterion assigns a certain status to each outcome: let A be a hyperplane that contains $G(\mathbf{X})$, B – the set of classification errors. Then the statuses that compose the criterion can be described as follows:

- An *untrained neural network* is a network that has not been trained on D_{train}. Every element in A represents the example in $\mathrm{D}_{\mathrm{test}}^{(\mathrm{i})}$. The result of the classification forms a set B such that the values of each element B exceed the error boundary value ε.
- An *undertrained neural network* is a network that has been trained on D_{train}. Besides, there exists such a subset A, every element of which represents an example in D_{test}. The result of the classification forms a set B such that the values of each element in B exceed the error boundary value ε.
- A *trained neural network* is a network that has been trained on D_{train}. Every element in A represents the example $\mathrm{D}_{\mathrm{test}}^{(\mathrm{i})}$. The result of the classification forms a set B such that the values of each element in B do not exceed the error boundary value ε. The distribution of the examples in D_{test} does not have a significant influence on the classification results.
- An *overtrained neural network* is a network that has been trained on D_{train}. Every element in A represents the example $\mathrm{D}_{\mathrm{test}}^{(\mathrm{i})}$. The result of the classification forms a set B such that the values of each element in B do not exceed the error boundary value ε. The distribution of the examples in D_{test} has a significant influence on the classification results.

This formulation should not be confused with underfitting or overfitting. These notions are formed based on discrete metrics assessment while the criterion considers a continuous domain of errors.

4.2 Model description

The model used in this article can be formally described as follows:

We have a hyperplane A and a function $G(\mathbf{X})$, defined as:

$$G(\mathbf{X}) \bowtie \mathbf{Y},$$

where $\mathbf{X} \in \mathbf{R}^n$ – is a set of variables, $G(\mathbf{X})$ – a set of linear forms, $\mathbf{Y} \in \mathbf{R}^n$ – constant vectors, $\bowtie$ is a set of operators: $\{\leq, \geq, =\}$.

The model can also be described as follows:

$$G\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \lesseqgtr \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix},$$

where $G(\mathbf{X})$ and $\mathbf{Y}$ resemble the sets [D_{train}; D_{cv}; D_{test}]. The set of operators corresponds to the set of classes C. The operator ≤ is equivalent to the value "0" in the feature vector. The operator ≥ is equivalent to the value "1" in the feature vector, the operator = is equivalent to the value "2" in the feature vector.

To increase the convenience of visualization and simplify the computational process, we reduce the dimension of space to 2. Then, the resulting model will have the following wording:

$$G\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \lesseqgtr \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.$$

As the G(X) vector components represent linear forms, the task can be reformulated in the following way: locate the areas that are obtained as a result of partitioning the plane with the lines (linear forms).

$$S = \{p_i \mid p_i \triangleq \{-k \le x_i \le k\}, i = \overline{1..n}\}.$$

5. Experiment

5.1 Neural network description

This section demonstrates the classification results of the three perceptron models described in the previous paragraph.

We run the training on a machine with the following parameters:

- OS – Ubuntu 18.03 LTS.
- CPU – 5th Generation Intel® Core™ i7 Processor.
- GPU – NVIDIA® GeForce® 940MX 2 GB.
- RAM – 8 GB.

We chose perceptron (described in section 2.1) as a neural network for three experiments, as this type of network reduces the individual and total execution time compared to other architectures. The number of epochs of the training was 50, 150. The datasets used for training and testing were divided into those with examples for four constraints, including examples for eight constraints. The number of examples is 96 thousand for the training set, 12 thousand for cross-validation and testing datasets. The examples were distributed according to normal distribution. For an estimate of the error, we used the standard deviation:

$$e^{(i)}\left(p^{(i)}, c^{(i)}\right) = \frac{\sum_{j=1}^{k}(p_j^{(i)} - c_j^{(i)})^2}{k}$$

We trained ten different networks with the following structure of the hidden layers:

- Two perceptrons with two hidden layers of the following dimensions: ||L1|| = 16, ||L2|| = 8.
- Four perceptron with three hidden layers of the following dimensions: ||L1|| = 8, ||L2|| = 16, ||L3|| = 8.
- Four perceptron with six hidden layers of the following dimensions: ||L1|| = 8, ||L2|| = 16, ||L3|| = 32, ||L4|| = 32, ||L5|| = 16, ||L6|| = 8.

Each experiment involves a single perceptron trained on a dataset that contains points as examples. The experiment results are displayed in the graphs using a single color for each example. The color intensity represents the value of the classification error. The presented results include ten neural networks, trained on two different datasets. The first dataset contains the points distributed in the squares (0.25x0.25) and (0.75x0.75).

To describe the results, we introduce the following notation to identify the respective neural network: a perceptron with a structure of hidden layers P, trained on i epochs, on the dataset with j constraints, and the grid parameter k is written as *P_i_j_k*. The value of the parameter k determines the rectangle, which contains the examples.

For general information we introduce the accuracies for each perceptron, used in the presented experiments:

Table 1. The parameters of trained neural networks and the obtained accuracy values.

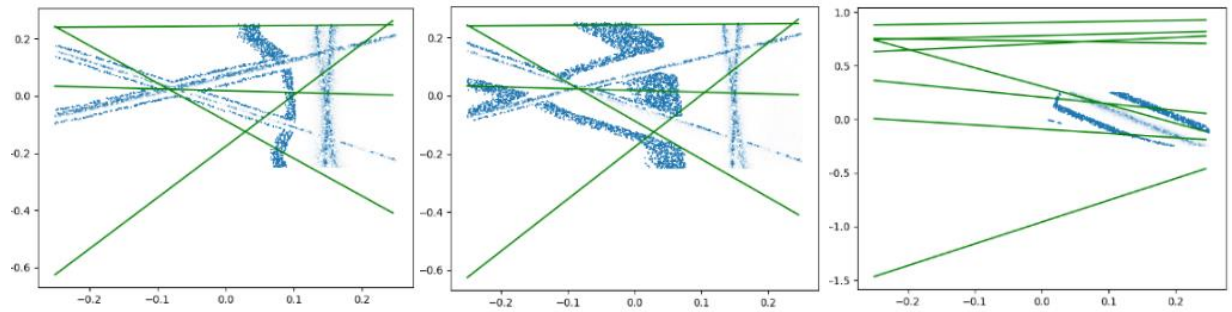
i	*j*	*k*	accuracy, %
16–8			
50	4	0.25	74
50	4	0.75	73
8–16–8			
50	4	0.25	63
50	4	0.75	53
150	8	0.25	99
150	8	0.75	78
8–16–32–32–16–8			
50	4	0.25	66
50	4	0.75	70
150	8	0.25	99
150	8	0.75	60

The classification results for each of the networks were visualized using graphs. Each graph contains:

- An area that is defined by the square k x k.
- Constraints visualization (are colored in green). A line represents each constraint as the experimental model was implemented in a two-dimensional space.
- Each point inside the square corresponds to an example from the continuous dataset. Each displayed point corresponds to an example from the discreet test set. Each point's color intensiveness represents the classification error's value for the particular example (the higher the intensity – the higher is the error value).

Meanwhile, the network obtains one of the statuses described in section 4.1. The network could be used in an independent environment only if it classified as "trained". The threshold error value of 0.2 was used in the presented experiments.

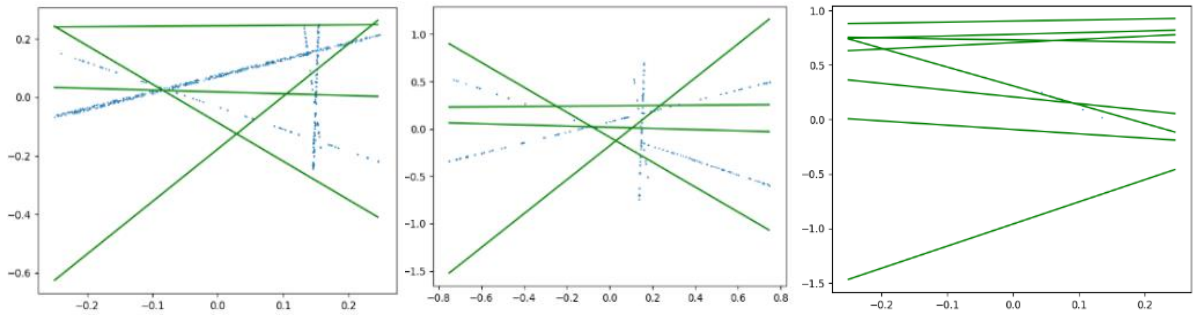
Classification results by certain networks are presented in Figure 1.



(a) (b) (c)

Fig. 1. Represents classification results for the respective *P_i_j_k* networks: (a) shows 16–8_50_4_0.25, (b) shows 8–16–8_50_4_0.25, (c) shows 8–16–32–32–16–8_150_8_0.25.

The classification results were filtered according to the set threshold, filtering out all those that do not surpass it (Fig. 2).

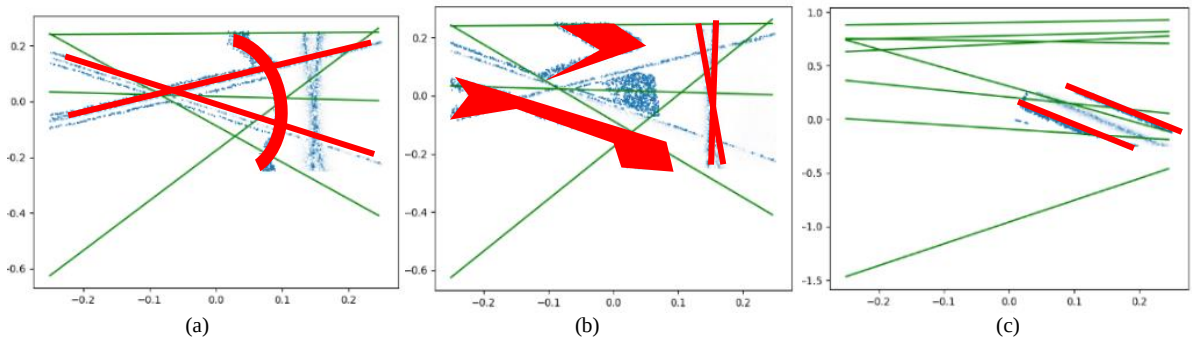


(a) (b) (c)

Fig. 2. Represents filtered classification results for the respective *P_i_j_k* networks: (a) shows 16–8_50_4_0.25, (b) shows 8–16–8_50_4_0.25, (c) shows 8–16–32–32–16–8_150_8_0.25.

According to the criterion, all the networks were marked as *undertrained*, as the results contain examples where the classification error is higher than the stated threshold (respective graphs contain the points that have the intensity above the value of ε). Thus, we can assert that the networks need to be further trained.

The advantage of the continuous domain analysis approach allows substantiating specific clusters of examples (Fig. 3). The classification error on these examples is higher than the set threshold. Each cluster defines a small subset of examples that can be separately processed and added to the initial set.



(a) (b) (c)

Fig. 3. Represents certain clusters of examples that make the networks falter (marked in red): (a) shows 3 clusters of 16–8_50_4_0.25 classification results, (b) shows 4 clusters of 8–16–8_50_4_0.25 classification results, (c) shows 2 clusters of 8–16–32–32–16–8_150_8_0.25 classification results.

Applying simple K-means clustering [22] to the obtained images allowed us to identify the certain clusters of examples that were added to the initial training set. The resulting classification accuracies are presented in Table 2.

Table 2. The parameters of trained neural networks and the obtained accuracy values after adding data to the training set.

i	*j*	*k*	accuracy, %	change, %
16–8				
50	4	0.25	85	+9
50	4	0.75	82	+9
8–16–8				
50	4	0.25	68	+5
50	4	0.75	65	+10
150	8	0.25	99	+0
150	8	0.75	81	+3
8–16–32–32–16–8				
50	4	0.25	76	+10
50	4	0.75	76	+6
150	8	0.25	99	+0
150	8	0.75	80	+20

In general, the classification accuracy for every network improved. The networks 8–16–8 and 8–16–32–32–16–8 obtained the status *trained* and could be safely used to classify the given task. Thus, the criterion allows classifying a trained or untrained network, which can be treated as a safe/unsafe answer provided by the verification methods (sections 2.3 and 3). The examination of the continuous data set allows identifying the clusters where the network's performance drops. Each cluster from the continuous perspective would help create more precise subsets of data to improve the network's accuracy (instead of generating discrete examples). A more thorough work on these subsets may also reduce adversarial attacks' effectiveness towards the given network, which is not provided by the verification methods. Besides, we recommend using more advanced clustering methods to boost the precision of clusters' location.

In order to define the centers of the clusters, different clusterization algorithms can be used (e.g. [15], [16], [17], [18], [19]). However, we believe that the location and the shape of the clusters depend on the following:

- What shape the constraints have (linear or non-linear), and where they are located with respect to the data.
- Which activation function (or functions) is used in the hidden layers of the network.

This can be noticed in Figure 3. The filtered clusters look very similar to the initial constraints. The further research path will be aimed at formalizing the aspects that identify the appearance of the cluster. We believe that this formalization could advance the following scientific fields:

- Deep learning and safe reinforcement learning using a data-driven approach [23]. Every year more and more sophisticated neural network structures emerge in order to solve classification tasks. However, it may be

necessary to pay more attention to the specifics of the data that is used for training. For this, more tools need to be developed, and the presented criterion is one of them.

- Neural network verification. Modern verification methods allow qualifying a neural network as safe or unsafe. If the network is marked as unsafe, no information on the data that caused the network to underperform is provided. It prevents from doing any further analysis on how the network could be improved. The criterion allows defining data clusters that can be improved to increase the resulting classification accuracy.

6. Conclusion

Deep neural networks have become one of the most popular tools for developing automated solutions. These models have proved to be especially useful in solving classification problems. It has been shown that with the increased accuracy of classification when using these models, the error in the activation of individual neurons within the deep layers increases [20]. Giving the neural network appropriate examples, the activation of certain neurons in the network's deep layers can significantly reduce errors in the final calculations [21]. The process of feeding such examples to a neural network is called an "adversarial attack," the purpose of which is to degrade the classification results.

People use two kinds of approaches to control the quality neural networks' training: empirical metrics (accuracy, recall, F1 score, etc.) and verification methods (the latter have developed significantly during the recent years). However, none of these approaches provides a complete description of the classification's quality performed by the neural network. Empirical metrics give only an estimate over a discrete set of examples and cannot be used to describe the entire range of values of the bounded continuum. Verification methods make it possible to estimate the whole continuum but do not determine which subset of examples causes a significant error in the classification results.

This paper proposes a criterion for assessing the classification quality, which considers the classification error as a continuum. Similarly, the test datasets represent bounded continuum sets. The criterion is formulated by describing the states of a neural network training process. Each state is specified by the correspondence of the set of errors to the range of the function:

- Untrained.
- Undertrained.
- Trained.
- Overtrained.

The described criterion allows not only to identify that the neural network is not sufficiently trained on the appropriate set of examples and assess which examples (or clusters of examples cause the error). This criterion can be used both as an independent assessment and in combination with other methods.

The article describes the model of the experiment, which represents a hyperplane and a set of constraints. The classes correspond to sequences of operators. Each operator determines a particular area into which the hyperplane is divided by the constraints (the article's experiments use two–dimensional space). We obtained the results by using three structures of perceptrons on the corresponding data sets. Visualization of the classification results shows that specific clusters of examples impact the accuracy of neural networks. Moreover, after retraining neural network models, the clusters of examples retain their topological appearance on the visualized graphs. Filtering the classification results by the margin of error demonstrates that in most cases, the clusters meet the constraints imposed on the original classification problem in some way.

A further research direction is the formalization of the criterion with respect to the clusters formed during training. It is also possible to integrate this criterion into existing verification methods. It would help obtain a clearer assessment of the neural network's safety (or unsafety) to solve the classification problem.

We believe that this formalization could advance the following scientific fields:

- Deep learning and safe reinforcement learning using a data-driven approach [23]. Every year more and more sophisticated neural network structures emerge in order to solve classification tasks. However, it may be necessary to pay more attention to the specifics of the data that is used for training. For this, more tools need to be developed, and the presented criterion is one of them.
- Neural network verification. Modern verification methods allow qualifying a neural network as safe or unsafe. If the network is marked as unsafe, no information on the data that caused the network to underperform is provided. It prevents from doing any further analysis on how the network could be improved. The criterion allows defining data clusters that can be improved to increase the resulting classification accuracy.

References

[1] Thomas Ritter, Carsten Lund Pedersen. Digitization capability and the digitalization of business models in business-to-business firms: Past, present, and future. In: Industrial Marketing Management, Vol. 86, pp. 180-190., April 2020.

[2] Kaur, Gurmeet & Bajaj, Karan. News Classification using Neural Networks. In: Communications on Applied Electronics, Vol. 5 (1), DOI:10.5120/cae2016652224, pp. 42-45, 2016.

[3] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui and J. Long. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. In: IEEE Access, vol. 6, DOI: 10.1109/ACCESS.2018.2855437, pp. 39501-39514, 2018.

[4] Toviah Moldwin, Idan Segev. Perceptron Learning and Classification in a Modeled Cortical Pyramidal Cell. In: Frontiers in Computational Neuroscience, 24 April 2020.

[5] Khan, A., Sohail, A., Zahoora, U. et al. A survey of the recent architectures of deep convolutional neural networks. In: Artificial Intelligence Review 53, 5455–5516, 2020.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. arXiv preprint arXiv:1512.03385v1. 2015.

[7] Pengfei Liu, Xipeng Qiu, Xuanjing Huang. Recurrent Neural Network for Text Classification with Multi-Task Learning. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. arXiv preprint arXiv:1512.03385v1. 2015.

[8] Andreas Venzke, Senior Member. Verification of Neural Network Behaviour: Formal Guarantees for Power System Applications. arXiv preprint arXiv: 1910.01624v4. 2020.

[9] Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark Barrett, Mykel J. Kochenderfer. Algorithms for Verifying Deep Neural Networks. arXiv preprint arXiv:1903.06758v2. 2020.

[10] S. V. Rakovic, E. C. Kerrigan, D. Q. Mayne and J. Lygeros, "Reachability analysis of discrete-time systems with disturbances," in IEEE Transactions on Automatic Control, vol. 51, no. 4, pp. 546-561, April 2006, doi: 10.1109/TAC.2006.872835.

[11] Henriksen P, Lomuscio. Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search. In: European Conference on Artificial Intelligence, 2020.

[12] Abien Fred Agarap. Deep Learning using Rectified Linear Units (ReLU). arXiv preprint arXiv:1803.08375v2. 2019.

[13] Guy Katz, Clark Barrett, David Dill, Kyle Julian, Mykel Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. arXiv preprint arXiv:1702.01135v2. 2017.

[14] Tran, Dung & Manzanas Lopez, Diego & Musau, Patrick & Yang, Xiaodong & Luan, Viet & Nguyen, Luan & Xiang, Weiming & Johnson, Taylor. (2019). Star-Based Reachability Analysis of Deep Neural Networks.

[15] Ajay Kumar, Shishir Kumar, "Density Based Initialization Method for K-Means Clustering Algorithm", *International Journal of Intelligent Systems and Applications(IJISA)*, Vol.9, No.10, pp.40-48, 2017. DOI: 10.5815/ijisa.2017.10.05.

[16] Ahmed Fahim, "Finding the Number of Clusters in Data and Better Initial Centers for K-means Algorithm", *International Journal of Intelligent Systems and Applications(IJISA)*, Vol.12, No.6, pp.1-20, 2020. DOI: 10.5815/ijisa.2020.06.01.

[17] Anand Khandare, Abrar Alvi, "Efficient Clustering Algorithm with Enhanced Cohesive Quality Clusters", *International Journal of Intelligent Systems and Applications(IJISA)*, Vol.10, No.7, pp.48-57, 2018. DOI: 10.5815/ijisa.2018.07.05.

[18] Mohammed A. H. Lubbad, Wesam M. Ashour, "Cosine-Based Clustering Algorithm Approach", *International Journal of Intelligent Systems and Applications(IJISA)*, vol.4, no.1, pp.53-63, 2012. DOI: 10.5815/ijisa.2012.01.07.

[19] Bikram K. Mishra, Amiya K. Rath, Santosh K. Nanda, Ritik R. Baidyanath, "Efficient Intelligent Framework for Selection of Initial Cluster Centers", *International Journal of Intelligent Systems and Applications(IJISA)*, Vol.11, No.8, pp.44-55, 2019. DOI: 10.5815/ijisa.2019.08.05.

[20] Zehao Douy, Stanley J. Osher, Bao Wangz. Mathematical Analysis of Adversarial Attacks. arXiv preprint arXiv:1811.06492v2. 2018.

[21] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Rocky Duan, Pieter Abbeel, Jack Clark. Attacking Machine Learning with Adversarial Examples. Available at: https://openai.com/blog/adversarial-example-research/. 2017.

[22] S. Na, L. Xumin and G. Yong, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm," 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, Jian, China, 2010, pp. 63-67, doi: 10.1109/IITSI.2010.74.

[23] Choukri Djellali, Mehdi Adda. A New Data-Driven Deep Learning Model for Pattern Categorization using Fast Independent Component Analysis and Radial Basis Function Network. Taking Social Networks resources as a case, Procedia Computer Science, ISSN 1877-0509, Volume 113, 2017, pp. 97-104, https://doi.org/10.1016/j.procs.2017.08.320.

Authors' Profiles

Zhengbing Hu. Visiting Prof., DSc Candidate in National Aviation University (Ukraine) from 2019. M.Sc. (2002), PhD. (2006) from the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". Postdoc (2008), Huazhong University of Science and Technology, China. Honorary Associate Researcher (2012), Hong Kong University, Hong Kong. Major research interests: Computer Science and Technology Applications, Artificial Intelligence, Network Security, Communications, Data Processing, Cloud Computing, Education Technology.

Deputy Director, International Center of Informatics and Computer Science, Faculty of Applied Mathematics, National Technical University of Ukraine "Kyiv Polytechnic Institute", Ukraine (2017-). http://www.icics.net/

Lyushenko Lesya is an associate professor of the Faculty of Applied Mathematics of the National Technical University of Ukraine, "Igor Sikorsky Kyiv Polytechnic Institute", Ph.D. of mathematical simulation in scientific research. The dissertation's name is "Development of the mathematical models of mainline power systems for on-line control automatization" (Ukrainian National Academy of Sciences. Institute of Simulation Problems in Power Engineering). Research interests include software engineering, mathematical modeling, machine learning, IT start-up project, etc. She is an author of research studies published in national and international journals as well as conference proceedings.

Dmytro Klyushnyk is currently working as a senior lecturer at the Mechanical department of Dnipro State Agrarian and Economic University, Dnipro, Ukraine. Also, he is working as an associated software engineer and technical consultant in ERP-systems applications. His scientific interests are concentrated in experimental data mining techniques and their applications to elastic structure dynamics and materials science.

Mykhailo Ivashchenko is a graduate student. He currently pursues Master's degrees in the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine, and at the University of Nebraska-Lincoln, Lincoln, NE, USA. The author's main research fields are dedicated to machine learning and neural networks, specifically: compositional/modular neural network learning, neural network verification, safe reinforcement learning.

How to cite this paper: Zhengbing Hu, Mykhailo Ivashchenko, Lesya Lyushenko, Dmytro Klyushnyk, " Artificial Neural Network Training Criterion Formulation Using Error Continuous Domain", International Journal of Modern Education and Computer Science(IJMECS), Vol.13, No.3, pp. 13-22, 2021.DOI: 10.5815/ijmecs.2021.03.02

