Artificial Neural Network Pruning to Extract Knowledge

1rd Evgeny M. Mirkes
School of Mathematics and Actuarial Science
University of Leicester
Leicester, UK
Lobachevsky State University
Nizhni Novgorod, Russia
em322@leicester.ac.uk

***Abstract*—Artificial Neural Networks (NN) are widely used for solving complex problems from medical diagnostics to face recognition. Despite notable successes, the main disadvantages of NN are also well known: the risk of overfitting, lack of explainability (inability to extract algorithms from trained NN), and high consumption of computing resources. Determining the appropriate specific NN structure for each problem can help overcome these difficulties: Too poor NN cannot be successfully trained, but too rich NN gives unexplainable results and may have a high chance of overfitting. Reducing precision of NN parameters simplifies the implementation of these NN, saves computing resources, and makes the NN skills more transparent. This paper lists the basic NN simplification problems and controlled pruning procedures to solve these problems. All the described pruning procedures can be implemented in one framework. The developed procedures, in particular, find the optimal structure of NN for each task, measure the influence of each input signal and NN parameter, and provide a detailed verbal description of the algorithms and skills of NN. The described methods are illustrated by a simple example: the generation of explicit algorithms for predicting the results of the US presidential election.**

***Index Terms*—artificial neural networks, network pruning, gradient descent, knowledge generation**

I. Introduction

The universality of artificial neural networks (NN) as a tool for approximating continuous functions [1], [2] is a mathematical basis for widespread use of NN in many applications. However, this universality create another problem: what is required NN structure? How many neurons are needed to solve a particular problem? The trend of publications in the field of Neural Networks can be illustrated by the graph in Fig. 1. Reducing the number of scientific papers related to NN can be explained by the fact that the methods and algorithms of NN creation, training, and testing are mainly developed. Fig. 1 also illustrates the trend of the NN pruning methodology publications. We can see continuous monotonous increase in the number of such papers. It is very interesting to consider fraction of NN pruning papers among total NN related papers (see the bottom graph in Fig. 1). It can be seen that the first

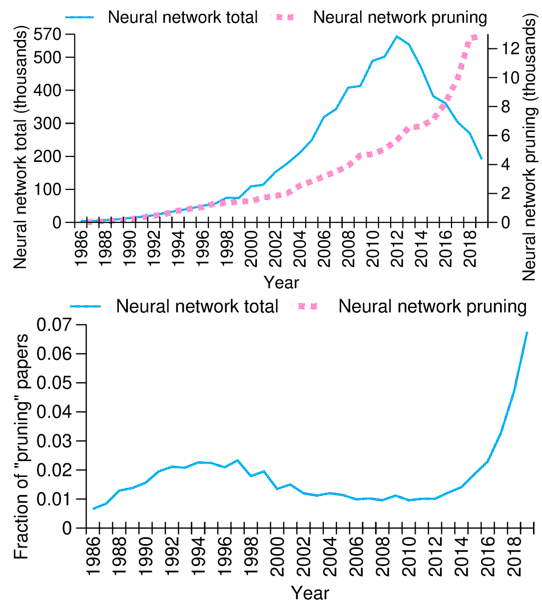

Fig. 1. Number of published papers with keywords ***"neural" AND "network" AND "pruning"*** (dotted line in the top graph), with keywords ***"neural" AND "network"*** (solid line in the top graph) and ratio, of these two amounts in the bottom graph

NN pruning works (e.g. [3]) was published approximately at the same time when back propagation NN was developed [4].

Why NN pruning attracts so much attention during so long time? There are several main drawbacks of NN [5]. First of all, NN have the same drawback as most machine learning methods: the tendency to overfitting if NN is too rich. A detailed descriptions of this problem is provided in [6], [7]. However, notion "too rich" is not well defined. There is an iterative procedure that allows the user to identify one of the suitable NN structures: first, the redundant NN is trained and then the complexity of the NN is reduced by the pruning procedure. The universality of NN with arbitrary nonlinearity for function approximation, proved in [1], [2] and illustrated in [8], makes it possible to successfully train NN if it is rich enough. Universal training procedures are described in [4] (partial case) and [9] (general approach). Some

The project is supported by the Ministry of Education and Science of the Russian Federation (Project No 14.Y26.31.0022).

special methods for acceleration and simplification the training process developed by A.N. Gorban and his team are described in [9], [10], [11]. The universal model of a neurocomputer with description of various implementations of its components is presented in [12]. Methods of back propagation based NN pruning are described in many papers: [6], [9], [13], [14], [3], [15], [16], [17], [18]. The most general and universal approach to NN pruning, including methods of zero, first and second order, a theoretical comparison of these methods and recommendations for their use are presented by A.N. Gorban in [15]. Several examples of successful applications of this two step techniques in medicine are presented in [15], [19], [20]. Paper [21] presents a detailed description of proposed approach. The most general description of the first order methods for assessing the sensitivity of the loss function for various elements of NN structure (inputs, synapses, neurons, layers, etc.) is presented in [9], [22], [12]. There are many modification of this approach, for example, [23], [24]. NN growth [25], [26], [27] is an alternative approach. Some recent studies [28], [29] have shown that special regularization through the loss function can significantly reduce the influence of overparametrisation.

The second NN drawback is unreadability: trained NN solves the problem, but we cannot understand this solution. This problem can also be solved by the training-pruning process (see, for example, [15], [21], [30]) with the subsequent interpretation of NN structure. Detailed description of this three step approach is presented in [13] and in this paper. A result of this approach is one or more algorithms described in natural language. The presence of several different algorithms may be considered as drawback of method but, from the other side, several algorithms can be used to determine the points of difference and plan further data collection (it is reasonable to collect data points in regions where several algorithms produce different answers).

The third problem is implementation of fitted NN. It is almost impossible to use extended or even double precision variables for mobile phone or Raspberry Pi NN implementation: it is too slow and require a lot of memory. Moreover, it is not necessary because of redundancy of NN. Usage of special type of pruning to simplify NN specification and reduce required precision is described in [13], [31]. This approach also has an alternative implementation in the form of growing NN [32]. Another problem that is closely related to this is the problem of the robustness of NN with respect to input noise. Reduction of implementation precision increases robustness.

The fourth problem is closely related to the third and can be formulated as the problem of backward feature selection [33]: select the minimal set of input features that provides a solution to the problem with the required accuracy. Back propagation NN provides the ability to calculate derivative of loss function with respect any elements of NN and with respect to any output, intermediate, and input signals. This allows the user to identify the influence of each input feature and remove one or more of the least important.

All described before problems were formulated for so named "shallow" NN and related to the first NN pruning peak in 1990-2000 (see Fig. 1). The new generation NN techniques, Deep Learning (DL) NN or Convolutional Neural Network (CNN) as the main representative of DL began to be widely used in the early 21st century (see review [34]). The number of papers related to DL NN pruning monotonously increased during this period (see, for example, [35], [36], [37], [38], [39]). The robustness of DL NN solutions is one of the main problems in the DL NN applications. The study of DL NN stability is presented in [40], [41], [42], [43].

There is also one more interesting way to use trained DL CNN: as was suggested first in [44], it is possible to use trained CNN without final classification layers (usually one or two fully connected layers) as a feature generator, and then apply any classification methods in this space of generated features. This approach was also used for person re-identification problem [45], [46]. [35] describes the use of trained CNN as a generator of features with different number of layers in the generating part. It is shown that reducing the depth of the generating part increases the number of errors but allows to solve problem with high enough accuracy.

As a result, we can conclude that in NN pruning there are several problems that are important in the past and now:

1) Feature selection: removal of neurons in the input layer. This problem is mostly relevant for shallow NN.
2) Identification the appropriate NN architecture to solve the problem and prevent overfitting. For this, various NN elements are removed (usually neurons in shallow NN or filters in CNN).
3) Reduction of precision of synaptic weights to provide cheap and fast implementation of trained NN.
4) Replacement of the activation function of neurons with a simple function. For example, a threshold or piecewise linear function instead of a sigmoid one. This problem also reduces the cost of implementing NN.
5) Uniform network simplification with a decrease in the maximum number of synapses associated with each neuron. This pruning problem was introduced especially for the problem of knowledge extraction: it is easier to interpret a statement (given by one neuron) if the neuron has a small number (two or three) of input signals.
6) Usage of CNN to generate features and pruning by removing the last layers of deep NN.

The rest of this paper is organised as follows: the section II describes methods for estimating the sensitivity indicators, the section III describes the pruning procedures and strategies for various pruning problems, the section IV describes pruning procedure which are specific for knowledge extraction and presents a example of knowledge extraction for the task of prediction the result of the USA presidential election, and the section V presents discussion of proposed methods.

II. Sensitivity indicators evaluation

To prune NN we need to know the importance of each NN element. There are several different measures of this importance and different names for them: 'sensitivity indicators' [9],

[6], [12], 'effectiveness of hidden units' [23], [24], 'second order sensitivity analysis' [9], [18], [47], [48]. A.N. Gorban in [9] described many different approaches but recommended to use the simplest: the first order sensitivity indicators. The main difference of proposed in [9], [15] sensitivity indicators is averaging over several training epoches, described below.

Let us consider shallow NN. The same approach can be used for deep NN, but the description will be slightly more complex. NN consists of neurons and layers. Standard representation of an artificial neuron is shown in Fig. 2. A regular neuron contains linear function $\sigma^r = w_0^r + \sum_{i=1}^n x_i w_i^r$ (x_i is i input of neuron, w_i^r is i synaptic weight of neuron r, and w_0^r is bias of neuron r) and the nonlinear activation function $y^r = f(\sigma^r)$. To build neural networks from neurons, one of the most detailed descriptions is [12]. This book also contains a detailed description of NN pruning. Further we consider the pruning on base of the first order sensitivity indicators firstly described in detail in [9].

Let us consider one d dimensional input vector u^j and the corresponding output z^j. This input vector propagates through NN, and for each neuron, the output values y are calculated and, finally, the NN output value $\hat{z}^j$) is calculated. Then we calculate the loss function $L^j = L(z^j, \hat{z}^j)$ (it can be usual mean square deviation function or more sophisticated SVM like function [11], [12]). Further back propagation allows us to calculate the derivatives of the loss function with respect to each input signal of NN and synaptic weight, bias, and neuron output:

$$\frac{\partial L^j}{\partial u_i^j}, \frac{\partial L^j}{\partial w_i^r}, \frac{\partial L^j}{\partial y^r}.$$

Detailed algorithms of all this derivatives calculations can be found, for example, in [9], [15], [12].

Let us have training set with N pairs (u^j, z^j). This means that the goal of the training process is to minimise the total loss function, which is the sum of the individual loss functions:

$$L = \sum_{j=1}^N L^j.$$

Let us now consider the problem of feature selection. For each individual element (u^j, z^j) from the training set, we can apply the Taylor series for the loss function with respect to input signals

$$L^j(u^j - v^j) = L^j(u^j) + \sum_{i=1}^d \frac{\partial L^j}{\partial u_i^j}(u_i^j - v_i^j) + o(u^j - v^j),$$

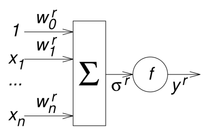

Fig. 2. The regular representation of the artificial neuron number r: 1 is a constant unit signal, x_i are input signals of the r neuron, w_0^r is the bias, w_i^r are the synaptic weights, Σ is summator, σ^r is summator output, f is an activation function

where v^j is the modified input vector. Since we are interested in feature selection we can consider vector v^j which is the same as u^j exclude one element: $v_k^j = 0,\ v_i^j = u_i^j\ \forall i \neq k$. In this case we can evaluate the cost of removing the input feature k for linear approximation as

$$\chi_k^j = \left|L^j(u^j - v^j) - L^j(u^j)\right| = \left|\frac{\partial L^j}{\partial u_k^j} u_k^j\right|.$$

The indicator of sensitivity is χ_k^j. It is clear that this indicator depends on the element of training set. There are several possible methods of evaluation the sensitivity to the input signal on the entire training set. For example, we can define maximal rating or average rating:

$$\chi_k^{max} = \max_{j=1,\dots,N} \chi_k^j, \quad \chi_k^{avg} = \frac{1}{N}\sum_{j=1}^N \chi_k^j. \qquad (1)$$

χ_k^{max} evaluates the maximum influence of the input signal (this signal has never been more important than χ_k^{max}). A widely used alternative is mean influence or simple sum: χ_k^{avg} evaluates the average influence of the input signal.

For the problem of reducing the precision of synaptic weights (include bias) the value of the loss function when w_i^r changes by v_i^r is

$$L^j(w - v) = L^j(w) + \sum_{r=1}^m \sum_{i=0}^{n^r} \frac{\partial L^j}{\partial w_i^r}(w_i^r - v_i^r) + o(w_i^r - v_i^r),$$

where m is the number of neurons, n^r is the number of input signals of neuron r, v_i^r is the new value of weight i of neuron r. Usually $v_i^r = w_i^r$ or is the closest to the value w_i^r from the set S of valid weights values. For example, it can be useful to train NN with extended precision and later transfer all synaptic weights to single precision or to smaller set of values. The set of valid values can be different for different problems. For example, to extract explicit knowledge from NN it is useful to use weights only with values from the set $S = \{-1, 0, 1\}$. For a low-cost implementation it is possible to convert weights to an integer and apply cheaper software (see, e.g. [49]). For the problem of removing synaptic weights the set of valid weights is $S = \{0\}$. Modified synaptic weights must be removed from the set of elements used for further modification. This means that modified synaptic weights are marked as non trainable and never change values. The effect of changing of one weight w_i^r only ($v_l^k = w_l^k\ \forall k \neq r$ or $i \neq l$) is

$$\chi_k^{js} = \left|L^j(w - v) - L^j(w)\right| = \left|\frac{\partial L^j}{\partial w_k^s}(w_k^s - v_k^s)\right|.$$

For the problems of removing or reducing of precision of synaptic weights the sensitivity indicators χ_k^{js} depend on the element of training set. In this case we can use modification of formulae (1) to evaluate the impact for the entire training set:

$$\chi_{sk}^{max} = \max_{j=1,\dots,N} \chi_k^{js} = |w_k^s - v_k^s| \max_{j=1,\dots,N}\left|\frac{\partial L^j}{\partial w_k^s}\right|, \qquad (2)$$

$$\chi_{sk}^{avg} = \frac{1}{N}\sum_{j=1}^{N}\chi_k^{js} = \frac{|w_k^s - v_k^s|}{N}\sum_{j=1}^{N}\left|\frac{\partial L^j}{\partial w_k^s}\right|. \qquad (3)$$

The problem of neuron removing can be solved by removing synaptic weights. This approach is not the best because in the bad case the number of synaptic weights can be drastically reduced without removing any neurons. In the worst case, each neuron may have only one synaptic weight but all neurons will affect NN output. To avoid such situation, we can consider the sensitivity indicators to the entire neuron instead of synaptic weights. To evaluate these indicators we can consider the following representation of loss function:

$$L^j(y^j - v^j) = L^j(y^j) + \sum_{r=1}^{m}\frac{\partial L^j}{\partial y^{jr}}(y^{jr} - v^{jr}) + o(y^j - v^j),$$

where y^{jr} is the output of the neuron r for the element j of the training set and v^{jr} is the new value of this output. Since we are interested in removing neurons the value of the new signals v^{jr} can be the same as in y^{jr} or zero. To evaluate the sensitivity to the neuron k only, let us consider $v^{jk} = 0,\ v^{jr} = y^{jr}\ \forall r \neq k$. In this case the sensitivity indicator is

$$\chi_k^j = \left|L^j(y^j - v^j) - L^j(y^j)\right| = \left|\frac{\partial L^j}{\partial y^{jk}}y^{jk}\right|.$$

To evaluate the sensitivity indicator for the entire training set, one of the formulae (1) can be used. The similar approach for neuron removal problem is proposed in [50].

Now we have sensitivity indicators for the problems of input feature selection, removing or reducing precision of synaptic weights, and neuron removing. All of these indicators are based on the gradient of the loss function. However, gradient of any function has one property [51]: if we precisely find minimum in the direction of the anti-gradient, then the gradient calculated at the new point will be orthogonal to the previous gradient. This means that the gradients of several successive epoches of learning vary greatly. On the other hand, the sensitivity indicators defined above depend on the gradient. It must be emphasized that other sensitivity indicators, such as 'hidden unit efficiency' or the Hessian matrix in the 'second order sensitivity analysis', are also strongly influenced by the current set of network parameters (synaptic weights and activation function parameters). This means that the pruning result will be very sensitive to the NN weights at the start of the pruning.

A.N. Gorban [15] suggested averaging the sensitivity indicators over several epoches of training. In this case the sensitivity indicators become more objective and less dependent on NN state at the pruning start moment. The results of pruning based on average sensitivity indicators are presented in [15], [19], [20].

III. Pruning procedures and strategies

The general pruning procedure is shown in Fig. 3. Step of calculating the sensitivity indicators involves several epoches of NN training with the accumulation of the sum of the sensitivity indicators.

```
Training NN
loop
    Save current NN.
    Calculate sensitivity indicators.
    Select element for modification.
    Modification of NN.
    Retraining modified NN
    if total loss function is greater than threshold then
        Break loop
    end if
end loop
Restore the last saved NN
```

Fig. 3. Algorithm of NN pruning

Procedure for selecting an element for modification is simple for most of pruning problems: the first candidate is trainable element with minimal value of sensitivity indicator (some of the elements can be marked as non trainable during the pruning process). The only exception is the problem of uniform network simplification. In this case the first candidate for modification is the synaptic weight w_i^r with the smallest sensitivity indicator χ_i^r such that the r neuron has maximal number of synaptic weights n^r. This definition of the first candidate for removal allows us to uniformly simplify the structure of NN.

Modification of NN has different meaning for different pruning problems. For problems involving deleting an element, modification means simple deleting of selected element and possibly related elements. Removing related NN elements involves, for example, removing from a NN a subnetwork whose outputs are not connected (directly or indirectly) to output neurons. For the problem of the reduction of a synaptic weight precision, modification of selected element w_k^s involves replacing it by v_k^s and marking this synaptic weight as non trainable. This means that in any subsequent training periods, this synaptic weight will be constant.

Retraining of the modified NN should begin with the modified network. It is easy to understand that in many cases a modified network can be successfully trained, but training NN with exactly the same structure but with randomly generated initial weights can be unsuccessful.

The described procedure for pruning NN is very simple, but for a rich enough NN this can take a lot of time. There is a very simple modification of the proposed procedure for its acceleration (see Fig. 4). Let us denote the number of elements for simultaneous modification/deletion as M. In the first step M can be half of the total number of elements (input features, neurons, synaptic weights) in NN. After calculating the sensitivity indicators select M elements for modification. If the modified NN cannot be successfully trained, then we return to the last saved NN and divide M by 2: $M = M/2$. Then we repeat the pruning without recalculating the sensitivity indicators. If modification of one element ($M = 1$) is unsuccessful, then the last saved NN is the minimal NN, and the procedure stops.

NN after completion of the pruning procedure is minimal. This means that there are no elements of the considered type

```
Training NN
M = half of total number of elements to remove.
loop
    Save current NN.
    Calculate sensitivity indicators.
    loop
        Select M elements for modification.
        Modification of NN.
        Retraining modified NN
        if total loss function is greater than threshold then
            Break internal loop
        end if
        if M > 1 then
            M = M/2
        else
            Break external loop
        end if
    end loop
end loop
Restore the last saved NN
```

Fig. 4. Accelerated algorithm of NN pruning

which could be modified without destroying NN skills. The notion of a minimal network depends on type of pruning problem to be solved: for feature selection problem the minimal network used a minimal set of input features, for neuron removing problem the minimal NN contains the minimal number of neurons, for synaptic weights precision reduction problem the minimal network contains minimal (usually zero) number of non modified elements. Really it is possible to combine different procedures. For example, we can initially minimise the set of used input features, then minimise the number of neurons, minimise the number of synaptic weights, and, finally, reduce precision of synaptic weights. The final network of this procedure will be minimal from all points of view.

All described pruning procedures requires retraining modified NN. It should be emphasized that in the proposed framework (see Fig. 3 and Fig. 4), retraining does not require time comparable with initial training [50].

IV. Knowledge extraction

To extract knowledge from trained NN a specially developed pruning procedure [13] can be used. The main step of this procedure is to uniformly reduce the number of inputs of each neuron. It is easy to interpret output of neuron with one input. For two inputs interpretation is usually possible but for three inputs it is not so clear. On the other hand, if all synaptic weights can be converted to plus or minus one, then interpreting a neuron with three inputs will become simple. This means that to extract explicit knowledge from NN it is necessary to apply uniform reducing of number of inputs for each neuron, than removing of synaptic weights can be applied, and finally all synaptic weights has to be converted to one of the values from the set $S = \{-1, 0, 1\}$. Usually such network allows us to substitute the step function

$$h(x) = \begin{cases} -1, & \text{if } x < 0 \\ 1, & \text{otherwise.} \end{cases} \quad (4)$$

instead of the continuous activation functions. NN, which was successfully pruned in this way and has no more than three inputs for each neuron we called logically transparent. If at least one of the requirements is not satisfied, then NN is not logically transparent and cannot be represented in "if-then" style algorithm. Such NN can be used to generate "fuzzy if-then" rules [30]. Obviously, explicit if-then rules are preferable, but sometimes it is not possible to find such a description. In such situations, an increase in the number of layers and the number of units (neurons) in hidden layers can sometimes help.

To illustrate the extraction of knowledge from NN, we consider the database for the problem of forecasting the results of the US presidential election [52]. Each database entry contains answers to 12 questions:

1) Has the incumbent party been in office more than a single term?
2) Did the incumbent party gain more than 50% of the vote cast in the previous election?
3) Was there major third party activity during the election year?
4) Was there a serious contest for the nomination of the incumbent party candidate?
5) Was the incumbent party candidate the sitting president?
6) Was the election year a time of recession or depression?
7) Was there a growth in the gross national product of more than 2.1% in the year of the election?
8) Did the incumbent president initiate major changes in national policy?
9) Was there major social unrest in the nation during the incumbent administration?
10) Was the incumbent administration tainted by major scandal?
11) Is the incumbent party candidate charismatic or a national hero?
12) Is the challenging party candidate charismatic or a national hero?

This is binary classification problem with two classes: P is the victory of power (incumbent) party and O is the victory of opposition (challenging) party. The database contains 31 records (elections from 1860 to 1980). The following NN structure was chosen: two hidden layers with 10 neurons in each and two output neurons P and O. Network answer was 'victory of power party' if the output of neuron P was greater than the output of neuron O, and answer was 'victory of opposition party' in the opposite situation. The input was coded as 1 for the answer 'yes' and -1 for the answer 'no'. The first neural network solution for this problem was presented in [22], [53]. The authors showed that this problem can be solved using 5 inputs and two neurons. Really, a single neuron is enough to solve this problem (see Fig. 5(e)). Paper [54] presented NN for prediction of the result of the UK General Election.

For experiment we generated, trained and pruned several networks with different initial weights and different pruning

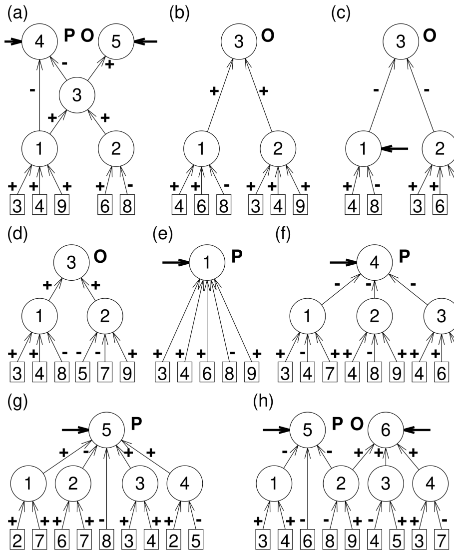

Fig. 5. Examples of minimal NNs for the presidential election problem: (a), (b), (c), (d), (f), and (h) are logically transparent and (e) and (g) are not logically transparent; a rectangle means input neuron; a bold arrow means a unit bias; a thin arrow means the connection between neurons; the signs "+" and "−" mean synaptic weight of 1 or -1, respectively; "O" ("P") means the positive signal of neuron corresponds to victory of opposition (power) party

procedures. Minimal NNs are presented in Fig. 5. For networks (a), (b), (c) and (e) the first step in the pruning was input feature selection. As a result these 4 networks have only five inputs. The second pruning procedure for networks (a), (b), and (c) and the first for networks (d), (f), and (h) was uniform structure simplification (the goal was to have no more than 3 inputs for each neuron). We can see that each neuron of all these NNs has no more than 3 inputs. The next pruning procedure was the removal of neurons. After described pruning procedures, all NNs was pruned by removing of synaptic weights. The final step was to modify the synaptic weights to the values from the set $S = \{-1, 0, 1\}$. In all presented networks, the sigmoid activation function was replaced by step function (4). Networks (e) and (g) are minimal but not logically transparent because the output neurons of both networks have five inputs. On the other hand, NN in Fig. 5(e) can be described as a rule "the power party candidate will win if at least two answers for questions 3, 4, 6, and 9 are positive or at least one of these answers is positive and answer for question 8 is negative". This rule can be considered appropriate. Logically transparent networks allow us to form a clearer verbal description of the algorithm.

Let us verbalise the network (b) in Fig. 5. Using medical terminology, the inputs can be called "symptoms", the outputs of neurons 1 and 2 are "syndromes" and output of neuron 3 is "diagnosis". The first syndrome appears if at least two of the following symptoms are observed: "There was a serious contest for the nomination of the incumbent party candidate", "The election year was a time of recession or depression", and "The incumbent president did not initiate any major changes in national policy" (note that the negation of the last statement was used). All three statements characterise quality of the current president governance: governance is inadequate if two of the three symptoms listed are observed. The second syndrome appears if at least two of the following symptoms are observed: "There was major third party activity during the election year", "There was a serious contest for the nomination of the incumbent party candidate", and "There was major social unrest in the nation during the incumbent administration". The second syndrome can be called "political instability syndrome": the situation is politically unstable if at least two of the three symptoms listed above are observed. Neuron 3 produces a positive output if at least one of the syndromes appears.

Now we can finally formulate the algorithm A1 extracted from the NN in Fig. 5(b):

1) *Inadequate governance syndrome* appears if at least two of the following symptoms are observed: "There was a serious contest for the nomination of the incumbent party candidate", "The election year was a time of recession or depression", and "The incumbent president did not initiate any major changes in national policy".
2) *Political instability syndrome* appears if at least two of the following symptoms are observed: "There was major third party activity during the election year", "There was a serious contest for the nomination of the incumbent party candidate", and "There was major social unrest in the nation during the incumbent administration".
3) The opposite (challenging) party will win if either governance is inadequate or situation is politically unstable.

Analogously the NN in Fig. 5(d) can be interpreted as the following algorithm A2:

1) *Syndrome of political instability or stagnation* appears if at least two of the following symptoms are observed: "There was major third party activity during the election year", "There was a serious contest for the nomination of the incumbent party candidate", and "The incumbent president did not initiate any major changes in national policy".
2) *Syndrome of instability* appears if at least two of the following symptoms are observed: "The incumbent party candidate is not the sitting president", "The a growth in the gross national product was less than 2.1% in the year of the election", and "There was major social unrest in the nation during the incumbent administration".
3) The opposite (challenging) party will win if one of instability syndromes appears.

Two different algorithms were generated. Since both algorithms use 7 attributes together, and all attributes are binary, the number of different combinations of these attributes is 128: all attributes -1; the first is 1, and the remaining six are -1, etc. In 98 cases (more than 75%), the answers of both algorithms are the same. There are 19 cases where the A1 algorithm predicts the victory of the party in power, and the A2 algorithm predicts the victory of the opposite party. Finally, there are 11 cases where Algorithm A1 predicts the victory of the opposite party, and Algorithm A2 predicts the victory of the party in power. For further data collection, it is advisable to choose one of 30 cases when two algorithms give different forecasts.

V. Discussion

In this paper we presented a general description of the NN pruning problems and possible solutions based on the first order estimation of sensitivity indicators. The first order evaluation of sensitivity indicators were proposed 30 years ago [9], explained in detail in [12], [15], and illustrated by various examples [21], [22]. The sensitivity analysis was developed not only for synaptic weights, but also for any signals or group of signals in NN. This approach allows us to evaluate sensitivity to each neuron directly instead of an indirect estimate of the zero order [3], [7], [16], [17], [26], [24] or a second order estimate [23].

The listed pruning procedures can be implemented in a unified framework: all the procedures correspond to the same algorithm in Fig. 4 and differ in the assessment of sensitivity indicator and the selection of candidates for modification. There are two different ways to evaluate the sensitivity indicator: the sensitivity to synaptic weight or the sensitivity to signals (input, output, or intermediate). There are also two described approaches for selecting candidates for modification: globally or in several local sets (for the uniform simplification of all neurons or other structural blocks).

Described approach can be easily applied to CNN. The main advantage of CNN is uniformity of all calculations. This means that procedure for removing of one synaptic weight becomes useless: it can complicate calculations instead of simplifying them. Essentially more reasonable is operation of removing of filters [36] or, even, channels [38]. Proposed by A.N. Gorban [9], [15] algorithms of sensitivity indicators estimation calculate them for each filter and each channel. This allows us to apply effectively the algorithm (Fig.4) to CNN. Procedures of precision reduction can be applied directly to DL NN.

Finally, the special sequences of pruning procedures allow us to form special class of shallow networks: logically transparent NN and then produce a verbal description of an explicit algorithm for the problem solution. This approach was demonstrated on the problem of forecasting the result of the USA Presidential election. Proposed method can give several explicit algorithms. Is this an advantage or disadvantage? We find this property useful and reasonable. To further improve the theory and discriminate models we can identify areas where different algorithms will produce different results, and collect new data from these areas . New data can be used to reduce the number of models or falsify all of them and create new explicit algorithms.

Generalisation of knowledge extraction methods for DL NN assumes splitting these networks into two parts: a part for feature generation and a part for final decision (classification, prediction, etc.). The natural way to prune the feature generation part is to remove unnecessary filters and channels. The second part can be considered as a shallow NN and transformed into logically transparent one.

The proposed pruning procedures, based on sensitivity indicators averaged over several training epochs, work ”on the fly” during training and avoid the expensive retraining that is common for methods with separate training and pruning procedures.

References

[1] V. Y. Kreinovich, “Arbitrary nonlinearity is sufficient to represent all functions by neural networks: A theorem,” *Neural Networks*, vol. 4, no. 3, pp. 381–383, jan 1991. [Online]. Available: https://doi.org/10.1016%2F0893-6080%2891%2990074-f

[2] A. Gorban, “Approximation of continuous functions of several variables by an arbitrary nonlinear continuous function of one variable, linear functions, and their superpositions,” *Applied Mathematics Letters*, vol. 11, no. 3, pp. 45–49, 1998. [Online]. Available: https://doi.org/10.1016%2Fs0893-9659%2898%2900032-9

[3] J. Sietsma and R. J. Dow, “Neural net pruning - why and how,” in *IEEE International Conference on Neural Networks*, vol. 1. IEEE, 1988, pp. 325–333. [Online]. Available: https://doi.org/10.1109%2Ficnn.1988.23864

[4] D. Rumelhart, G. Hinton, and R. Williams, “Learning internal representations by error propagation,” in *Readings in Cognitive Science*. Elsevier, 1988, pp. 399–421. [Online]. Available: https://doi.org/10.1016%2Fb978-1-4832-1446-7.50035-2

[5] D. A. Elizondo and M. A. Góngora, “Current trends on knowledge extraction and neural networks,” in *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005*, W. Duch, J. Kacprzyk, E. Oja, and S. Zadrożny, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 485–490.

[6] A. N. Gorban *et al.*, *Neuroinformatics*. Novosibirsk: Nauka (RAN), 1998.

[7] J. Sietsma and R. J. Dow, “Creating artificial neural networks that generalize,” *Neural Networks*, vol. 4, no. 1, pp. 67–79, jan 1991. [Online]. Available: https://doi.org/10.1016/0893-6080(91)90033-2

[8] C.-H. Choi and J. Choi, “Construction of neural networks to approximate arbitrary continuous functions of one variable,” *Electronics Letters*, vol. 28, no. 2, p. 151, 1992. [Online]. Available: https://doi.org/10.1049%2Fel%3A19920094

[9] A. N. Gorban, *Training neural networks*. USSR-USA JV Paragraph, 1990.

[10] S. E. Gilev, A. N. Gorban, and E. M. Mirkes, “Several methods for accelerating the training process of neural networks in pattern recognition,” *Advances in Modelling and Analysis-A*, vol. 12, pp. 29–29, 1992.

[11] A. N. Gorban and E. M. Mirkes, “Estimations and answer interpreters for dual-functioning neural networks,” *Izvestiya Vuzov. Priborostroyenie*, vol. 39, no. 1, pp. 5–14, 1996.

[12] E. M. Mirkes, *Neurocomputer, Project of Standard*. Novosibirsk: Nauka (RAN), 1998.

[13] A. Gorban, E. Mirkes, and V. Tsaregorodtsev, “Generation of explicit knowledge from empirical data through pruning of trainable neural networks,” in *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, vol. 6. IEEE, 1999, pp. 4393–4398. [Online]. Available: https://doi.org/10.1109%2Fijcnn.1999.830876

[14] E. D. Karnin, “A simple procedure for pruning back-propagation trained neural networks,” *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 239–242, 1990. [Online]. Available: https://doi.org/10.1109/72.80236

[15] A. N. Gorban and D. A. Rossiev, *Neural networks on personal computer*. Novosibirsk: Nauka (RAN), 1996.

[16] M. C. Mozer and P. Smolensky, “Skeletonization: A technique for trimming the fat from a network via relevance assessment,” in *Advances in neural information processing systems (NIPS 1988)*, vol. 1, 1989, pp. 107–115.

[17] M. Hagiwara, “Removal of hidden units and weights for back propagation networks,” in *International Joint Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, vol. 1. IEEE, 1993, pp. 351–354. [Online]. Available: https://doi.org/10.1109%2Fijcnn.1993.713929

[18] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[19] A. Gorban *et al.*, "Medical, psychological and physiological applications of MultiNeuron neural simulator," in *The Second International Symposium on Neuroinformatics and Neurocomputers*. IEEE, 1995. [Online]. Available: https://doi.org/10.1109%2Fisninc.1995.480831

[20] S. Gilev *et al.*, "Multineuron" neural simulator and its medical applications," in *Proceedings of the International Conference on Neural Information Processing (Oct. 17-20, Seoul, Korea)*, vol. 2, 1994, pp. 1261–1266.

[21] P. Gordienko, "Construction of efficient neural networks: Algorithms and tests," in *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, vol. 1. IEEE, 1993, pp. 313–316. [Online]. Available: https://doi.org/10.1109%2Fijcnn.1993.713920

[22] A. Gorban and C. Waxman, "Neural networks for political forecast," in *Proceedings of the WCNNWCNN'95 (World Congress on Neural Networks'95, Washington DC, July 1995)*, J. DeWitte, Ed., vol. 1. Lawrence Erlbaum Associates, 1995, pp. 179–184.

[23] N. Majima, A. Watanabe, A. Yoshimura, and T. Nagano, "A new criterion "effectiveness factor" for pruning hidden units and its effectiveness," *The Brain & Neural Networks*, vol. 1, no. 2, pp. 64–68, 1994. [Online]. Available: https://doi.org/10.3902%2Fjnns.1.64

[24] A. Yoshimura and T. Nagano, "A new measure for the estimation of the effectiveness of hidden units," in *Proceedings Of the Annual Conference Of JNNS*, 1992, pp. 82–83.

[25] M. Ishikawa, "Structural learning with forgetting," *Neural Networks*, vol. 9, no. 3, pp. 509–521, apr 1996. [Online]. Available: https://doi.org/10.1016%2F0893-6080%2896%2983696-3

[26] K. Murase, Y. Matsunaga, and Y. Nakade, "A backpropagation algorithm which automatically determines the number of association units," in *International Joint Conference on Neural Networks*, vol. 1. IEEE, 1991, pp. 783–788. [Online]. Available: https://doi.org/10.1109%2Fijcnn.1991.170496

[27] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in neural information processing systems*, 1990, pp. 524–532.

[28] T. Poggio, K. Kawaguchi, Q. Liao, B. Miranda, L. Rosasco, X. Boix, J. Hidary, and H. Mhaskar, "Theory of deep learning iii: the non-overfitting puzzle," *CBMM Memo*, p. 073, 2017.

[29] H. Mhaskar and T. Poggio, "An analysis of training and generalization errors in shallow and deep networks," *Neural Networks*, vol. 121, pp. 229–241, jan 2020. [Online]. Available: https://doi.org/10.1016%2Fj.neunet.2019.08.028

[30] H. Ishibuchi and M. Nii, "Generating fuzzy if-then rules from trained neural networks: linguistic analysis of neural networks," in *Proceedings of International Conference on Neural Networks (ICNN'96)*, vol. 2. IEEE, 1996, pp. 1133–1138. [Online]. Available: https://doi.org/10.1109%2Ficnn.1996.549057

[31] H. Keegstra, W. Jansen, J. Nijhuis, L. Spaanenburg, H. Stevens, and J. Udding, "Exploiting network redundancy for low-cost neural network realizations," in *Proceedings of International Conference on Neural Networks (ICNN'96)*, vol. 2. IEEE, 1996, pp. 951–955. [Online]. Available: https://doi.org/10.1109%2Ficnn.1996.549025

[32] M. Hoehfeld and S. Fahlman, "Learning with limited numerical precision using the cascade-correlation algorithm," *IEEE Transactions on Neural Networks*, vol. 3, no. 4, pp. 602–611, jul 1992. [Online]. Available: https://doi.org/10.1109%2F72.143374

[33] K. Kira, L. A. Rendell *et al.*, "The feature selection problem: Traditional methods and a new algorithm," in *AAAI-92*, vol. 2, 1992, pp. 129–134.

[34] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015. [Online]. Available: https://doi.org/10.1016/j.neunet.2014.09.003

[35] A. N. Gorban, E. M. Mirkes, and I. Y. Tyukin, "How deep should be the depth of convolutional neural networks: a backyard dog case study," *Cognitive Computation*, 2019. [Online]. Available: https://doi.org/10.1007%2Fs12559-019-09667-7

[36] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[37] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems (NIPS 2015)*, vol. 28, 2015, pp. 1135–1143.

[38] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, oct 2017, pp. 1398–1406. [Online]. Available: https://doi.org/10.1109%2Ficcv.2017.155

[39] F. Tung and G. Mori, "Deep neural network compression by in-parallel pruning-quantization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018. [Online]. Available: https://doi.org/10.1109%2Ftpami.2018.2886192

[40] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, oct 2019. [Online]. Available: https://doi.org/10.1109%2Ftevc.2019.2890858

[41] N. Mani, "On adversarial attacks on deep learning models," Ph.D. dissertation, San Jose State University, 2019. [Online]. Available: https://doi.org/10.31979%2Fetd.49ee-sknc

[42] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017. [Online]. Available: https://arxiv.org/pdf/1706.06083.pdf

[43] L. Deng, "Deep learning: Methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3-4, pp. 197–387, 2014. [Online]. Available: https://doi.org/10.1561%2F2000000039

[44] F. J. Huang and Y. LeCun, "Large-scale learning with SVM and convolutional for generic object categorization," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1. IEEE, 2006. [Online]. Available: https://doi.org/10.1109%2Fcvpr.2006.164

[45] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2014. [Online]. Available: https://doi.org/10.1109%2Fcvpr.2014.220

[46] Y. Sun, X. Wang, and X. Tang, "Deeply learned face representations are sparse, selective, and robust," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015. [Online]. Available: https://doi.org/10.1109%2Fcvpr.2015.7298907

[47] B. Hassibi, D. Stork, and G. Wolff, "Optimal brain surgeon and general network pruning," in *IEEE International Conference on Neural Networks*, vol. 1. IEEE, 1993, pp. 293–299. [Online]. Available: https://doi.org/10.1109%2Ficnn.1993.298572

[48] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in neural information processing systems*, 1993, pp. 164–171.

[49] C. Bertin, N. Brisebarre, B. D. de Dinechin, C.-P. Jeannerod, C. Monat, J.-M. Muller, S.-K. Raina, and A. Tisserand, "A floating-point library for integer processors," in *Advanced Signal Processing Algorithms, Architectures, and Implementations XIV*, F. T. Luk, Ed. SPIE, oct 2004. [Online]. Available: https://doi.org/10.1117%2F12.557168

[50] C. Siegel, J. Daily, and A. Vishnu, "Adaptive neuron apoptosis for accelerating deep learning on large scale systems," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, dec 2016, pp. 753–762. [Online]. Available: https://doi.org/10.1109%2Fbigdata.2016.7840668

[51] L. Kantorovich and G. Akilov, "The method of steepest descent," in *Functional Analysis*. Elsevier, 1982, pp. 443–473. [Online]. Available: https://doi.org/10.1016%2Fb978-0-08-023036-8.50021-7

[52] A. J. Lichtman and V. I. Keilis-Borok, "Pattern recognition applied to presidential elections in the united states, 1860-1980: Role of integral social, economic, and political traits," *Proceedings of the National Academy of Sciences*, vol. 78, no. 11, pp. 7230–7234, nov 1981. [Online]. Available: https://doi.org/10.1073%2Fpnas.78.11.7230

[53] C. Waxman, "The history of US presidential elections from siberian NC point of view," in *[Proceedings] 1992 RNNS/IEEE Symposium on Neuroinformatics and Neurocomputers, Rostov-on-Don, Russia*, vol. 2. IEEE, 1992, pp. 1000–1010. [Online]. Available: https://doi.org/10.1109%2Frnns.1992.268530

[54] R. Borisyuk, G. Borisyuk, C. Rallings, and M. Thrasher, "Forecasting the 2005 general election: A neural network approach," *The British Journal of Politics and International Relations*, vol. 7, no. 2, pp. 199–209, may 2005. [Online]. Available: https://doi.org/10.1111%2Fj.1467-856x.2005.00182.x