Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

Research paper

Structure Learning for Deep Neural Networks with Competitive Synaptic Pruning

A. Ahmadi, R. Mahboobi Esfanjani *

Department of Electrical Engineering, Sahand University of Technology, Tabriz, Iran.

Article Info

Article History:
Received 26 July 2024
Reviewed 27 September 2024
Revised 16 October 2024
Accepted 31 October 2024

Keywords:
Deep neural networks
Synaptic pruning
Distillation column
PID tuning

*Corresponding Author's Email Address: mahboobi@sut.ac.ir

Abstract

Background and Objectives: A predefined structure is usually employed for deep neural networks, which results in over- or underfitting, heavy processing load, and storage overhead. Training along with pruning can decrease redundancy in deep neural networks; however, it may lead to a decrease in accuracy.

Methods: In this note, we provide a novel approach for structure optimization of deep neural networks based on competition of connections merged with brain-inspired synaptic pruning. The efficiency of each network connection is continuously assessed in the proposed scheme based on the global gradient magnitude criterion, which also considers positive scores for strong and more effective connections and negative scores for weak connections. But a connection with a weak score is not removed quickly; instead, it is eliminated when its net score reaches a predetermined threshold. Moreover, the pruning rate is obtained distinctly for each layer of the network.

Results: Applying the suggested algorithm to a neural network model of a distillation column in a noisy environment demonstrates its effectiveness and applicability.

Conclusion: The proposed method, which is inspired by connection competition and synaptic pruning in the human brain, enhances learning speed, preserves accuracy, and reduces costs due to its smaller network size. It also handles noisy data more efficiently by continuously assessing network connections.

This work is distributed under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

Introduction

The parallel-distributed nature of neural networks gives them a great capacity for learning and generalization. They have therefore been used to address a variety of issues, including modelling in automatic control. During training, information is stored in the weighted connections between the neurons, just like in the human brain. In a network, the number of hidden layers and their corresponding weights determine its structure, which is a major factor in its performance. Large and small networks both have a number of disadvantages. The contrast between curve fitting and neural network training helps to explain why networks with fewer free parameters perform better in terms of generalization, as shown by theory [1] and experience [2]. Moreover, the knowledge included in the small trained networks is easier to be understood and thus the abstraction of simple rules can be facilitated [3]. Finally, small networks require very little resources to construct in any physical computer environment. Larger networks suffer from the overfitting issue and are less able to generalize. They are also quite costly and complex. Therefore, choosing the appropriate size is crucial to having an efficient and quick network.

Deep Networks and Pruning

Deep neural networks (DNNs) have been the main reason for recent improvements in machine learning.

Numerous of these networks require a large amount of memory space and computing power [4]. These features make it impossible to implement networks in situations where resources are few, like mobile phones [5]. Therefore, along with the high potential and computing power of deep networks, it is necessary to solve the limitations of these popular networks so that they can be used optimally, at a lower cost, and at a high speed in real applications. In this regard, providing a new suitable method for this problem and reaching a network with a suitable size is the main topic of this work.

Neural network pruning—the targeted removal of parameters from an existing network—is a well-liked method for lowering these resource requirements. The objective is to build a smaller network with the same degree of precision, despite the fact that the original network was huge and precise. The pruning notion has attracted much attention in the last decade, and its popularity has increased due to the emergence of deep neural networks [6]. We need to be clear about the most important thing in order to prune effectively: which connections are the best candidates for pruning? Taking into account the pruning process in the human brain provides useful solutions [7]. During learning, frequently used synapses become stronger, while rarely used synapses become weaker and more likely to disappear [8]. On the other hand, to avoid accuracy falling, consideration must be given to the values of the connections.

Therefore, the approach presented in this work solves the usual weaknesses. In this way, it uses a more accurate criterion instead of a traditional weight factor, and secondly, it examines the behavior of connections in successive stages and It looks at strong, medium and weak connections differently. Instead of immediate removal, it measures the existence of multiple warnings regarding the connection's effectiveness.

In short, the abovementioned features are combined here to develop an efficient pruning technique. In the proposed algorithm:

First, network connections are evaluated initially based on a global gradient magnitude criterion. This evaluation also lasts for the next steps, and the scores are updated continuously. This criterion gives encouraging points for effective connections and destructive points for weak connections.

Second, inspired by the brain pruning strategy, a connection with a weak score is not removed immediately; instead, it is eliminated when its net score reaches a certain number with less than a certain threshold.

Third, dividing the connections in the network into three categories: connections with high impact, medium impact and weak impact, the first category will receive rewards and their chances of survival will increase, and the third category will be penalized and their chances of being pruned will increase.

Fourth, to enhance the network's accuracy and quality, the pruning rate is explicitly determined for each specific layer of the network.

The structure of this document is as follows: Section 2 is an overview of the related prior research. Furthermore, we provide our approach in Section 3. Comparative simulation results are shown in Section 4, and we conclude the paper in Section 5.

Related Works

This section presents some relevant works on network architecture optimization. Differently from shallow networks, which have only one hidden layer, deep networks have two or more hidden layers, which help to store and organize data efficiently; namely, they serve a more precise purpose than a superficial one. Deep networks' capacity for memorization greatly aids in managing uncertainties. Training a smaller neural network to mimic the bigger model is one method of lowering the neural network's computational complexity. Network distillation is a method that Hinton et al. [9] suggested. The primary flaw with this process is the need to predefine the smaller model's structure. Pruning, which is the process of eliminating individual neurons that provide less effect on the output of a trained network, is another method for shrinking and speeding up a model.

Therefore, compared to the above two approaches, the pruning method is usually preferred. Assuming a tight relationship between weight size and significance, the traditional and simple method is to select a threshold and prune those synapses whose weights are below it [10]. Several studies have cast doubt on this tactic [11]. Actually, by employing this method, certain advantageous synapses whose weights happen to fall below the threshold may be pruned. Some studies concentrate on creating suitable standards for assessing the significance of connections and eliminating the least important ones. Molchanov et al. considered the l2-norm of the kernel weights in addition to the feature map's mean, standard deviation, and percentage activation [12]. They also compared activations and predictions using mutual information as a criterion.

Molchanov et al. introduced first-degree Taylor expansion as a tool for evaluating synaptic significance. In order to determine synaptic significance, LeCun et al. [13] and Hassibi and Stork [14] employed a diagonal Hessian matrix and concentrated on the second-order term of a Taylor expansion. In order to eliminate the most replaceable filters that include extraneous data, He et al. proposed a geometric median-based filter-cutting approach [15]. The neuron significance score (NISP) method [16] propagates the final answers' relevance

ratings to each neuron in the network, as suggested by Yu et al. The least important neurons were then eliminated in order to prune the convolutional neural network. In addition to the feature maps they connected, Li et al. eliminated the filters with relatively low weights [17]. He et al.'s gentle pruning approach allows the model to be trained utilizing the trimmed filters after the pruning [18]. Transmits the relevance scores of the final replies to every neuron in the network, as proposed by Yu et al. The convolutional neural network was then pruned by removing the least significant neurons. Li et al. removed the filters with relatively low weights in addition to the feature maps they linked [19]. During training, every neuron in dropout is probabilistically eliminated, but during inference, they are allowed to rebound. A network's complexity does not go down while using this method. By randomly changing a portion of a neural network's weights to zero, Drop Connect is used to regularize neural networks [20]. Though it occasionally surpassed dropout, it learned more slowly than both the original network and the dropout network. One of its drawbacks is that it increases the amount of time needed for training. A dropout network often requires two to three times as long to train as a neural network with the same design that is used normally. MeProp altered a relatively small portion of the settings for each back-propagation phase [21]. These methods do not introduce any fundamental changes to the structure of the network.

An architecture for a network is optimized by evolutionary approaches. Both the topology and the weights are optimized concurrently in evolving neural networks. Numerous network structure-related factors found in the genome have been refined via evolution. An evolutionary approach evaluates a network's performance using a fitness function.

Typically, one of these functions is accurate classification [22], the other is the size of the network, which comprises factors like the quantity of connections or neurons [23]. After several rounds, an artificial neural network with evolutionary capabilities can identify the optimal network architecture. An evolutionary optimization approach was proposed by Zhao et al. [24]. A network was pruned to the ideal topology using genetic algorithms [25]. These approaches focus on network design optimization to achieve the optimal trade-off between accuracy and complexity; nonetheless, there is a significant degree of unpredictability in both approaches, and significant side trips may occur due to the lengthy development process.

In summary, all of the mentioned techniques assess connections in a single phase without keeping track of connections' behavior over time or allocating different pruning rates for different layers. As explained with reasons and references, the weighted criterion is not accurate and other single-factor criteria have weaknesses. The very important point is that we classify the connections in three categories (weak, medium and strong) and in addition to gradually reducing the weaker connections scores, we also gradually strengthen the strong connections, which is not the case in previous works in this format.

Therefore, the continuous control of connections, using our proposed criteria, which is different from the usual criteria, and on the other hand, considering the positive and negative points for network connections are the main differences between the present work and previous studies. A brain-inspired competitive synaptic pruning technique is introduced in place of the traditional omission technique.

Proposed Method

A novel brain-inspired competitive pruning technique is developed in this section. Network connections are supposed to compete for survival, and the basis of this competition is based on the weighted average score that each connection gets. Like other evolutionary-based ones, the proposed algorithm starts with an initial population (here the initial at the beginning).

Minimal-value deletion prunes all synapses whose weights are below a threshold. Magnitude-based approaches are common baselines in the literature. However, this method may prune some useful synapses whose weights are incidentally below the threshold. Gradient-based methods are less common but are more accurate, simple to implement, and have recently gained popularity.

The key idea is that after considering “Weight × Gradient” as the appraisal criterion, connections are classified into three categories: down (for example, the lower 20%), top (for example, the upper 20%), and the area between them. In each step, connections in the top class get a positive score, connections in the lowest category get a negative score, and connections in the middle set get zero points. In this way, we set a reward for good connections and a penalty for poor connections. At each stage, the net score of each connection is determined by adding the current score to the previous one. Therefore, we have updated net scores for all connections. Inspired by the human brain [26], the next important fact is that whenever the net score of a connection reaches a certain threshold—in other words, it receives a certain number of warnings—that connection is removed from the network.

It is worth noting that using the gradient in the evaluation criterion, as weight × gradient adds the rate of change to it and because of its dynamic nature increases the accuracy of the method.

Fig. 1 shows how synaptic pruning works. This issue is an important part of the presented method and it is more accurate and better than the usual methods, as inspired by the human brain, the removal of network connections happens gradually and step by step. In this way, if a connection gets lower scores several times, it is concluded that its importance for the network is low and therefore it is removed. It is clear that in synaptic pruning, there are two important parameters that we need to specify. The weight threshold shows which connections could benefit from pruning. The maximum allowed warning specifies how long the associated connection will be active before being erased. After establishing a threshold value, we keep an eye on the connection scores. In Fig. 2, the procedure of the proposed pruning technique is depicted. In the case of reaching the net score of a given connection to the threshold, it is pruned.

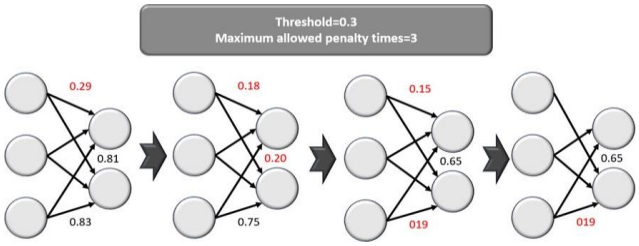

Fig. 1: Schematic of the synaptic pruning concept.

As can be seen in the flowchart, continuous evaluation of connections and obtaining a net score for each connection leads to a decision regarding removal or retention. Algorithm 1 provides a detailed presentation of the pruning pseudocode. As discussed, it is insufficient to rely just on the weighted domain, and there is a significant chance that some crucial network connections will be overlooked. Sorting the connections addressed this problem since, after the connections that may be deleted are identified, the value is not decreased all at once to complete the removal process. Actually, we warn the pruning candidates one after the other and prune them in response to these cautions. The threshold limit is chosen by trial and error. All connections at each stage are evaluated by the considered criterion and sorted based on the scores they get. The summary of the approach is that after sorting the scores of the connections, we have three categories: connections whose score are in the top 20%, as well as connections that are in the bottom 20% and connections whose scores are between these two areas. At each stage, one unit is added to the connections with them, and the connections with the lowest 20% are fined; i.e. their score are reduced by one unit. Therefore, the strong ones are strengthened and the weak ones are weakened, and they are candidate for pruning. There is a threshold value, if the connection score is less than that, a warning will be given, and after specified warnings, it will be pruned. Concisely, each link in the network is evaluated based on network error. Unless otherwise stated, we compute the errors resulting from omitting each link and, upon sorting according to the error, identify the connections with the highest number of errors. Connections are eliminated depending on the pruning rate, which is set by the designer considering factors such as layer percent.

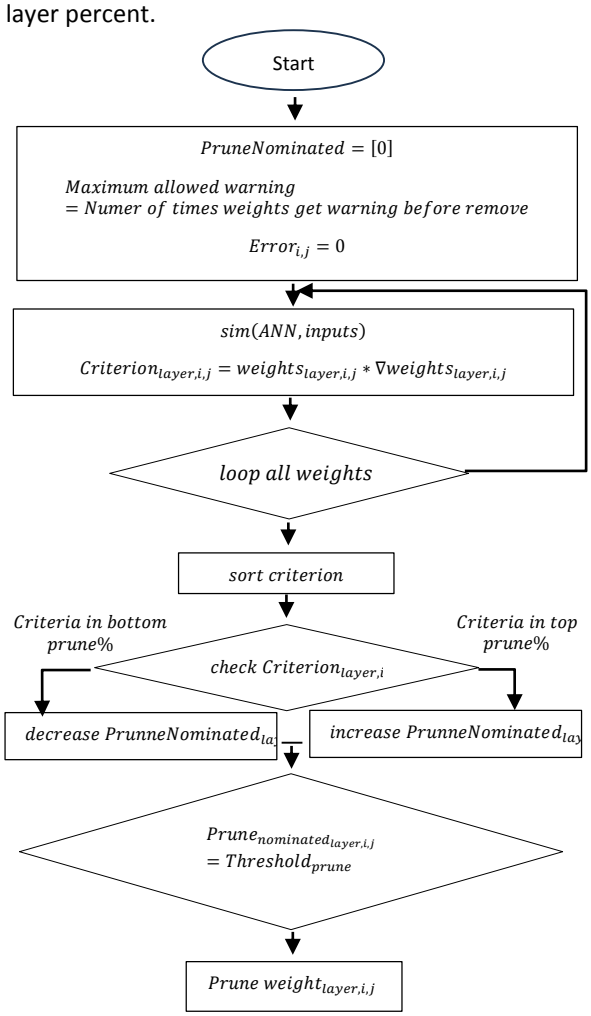

Fig. 2: Flowchart of proposed pruning.

The key idea is that, in addition to evaluation of each connection, the contribution of the layers is also taken into account in the pruning; in this way, the layer value is defined as the sum of all connection weights in the mentioned layer, and we also compute the layer percent, by dividing layer value into the total values of the network layers, as follows:

$$value_{layers} = \sum_{i=0}^{layer\ number} value_{layer_i} \quad (1)$$

$$layer\ percent_i = \frac{value_{layer_i}}{value_{layers}} \quad (2)$$

$$i = 0, \ldots, number\ of\ layers$$

Many of pruning techniques are predicated on the idea that there is a significant correlation between a weight's magnitude and significance. Recent researches have questioned this assumption and shown a notable discrepancy in the association between empirically optimum one-step decisions and weight-based pruning judgments [11].

Algorithm 1: Pruning Pseudo code

$Generate\ all\ zero\ PrunneNominated\ variable\ with$
$structure\ and\ size$
$Maximum\ allowed\ warning$
$= number\ of\ time\ we\ want\ \ a\ weight\ to\ not\ be\ cutted.$
$Initialize\ Error_{layer,i,j} = 0$
$Initialize\ Prune_{percent} = 20\%\ ,$
$shows\ percent\ of\ weights\ nominated\ for\ pruning,$
$Consider\ 20\ top\ and\ down\ for\ reward\ as\ penalty\ domain$
$Construct\ an\ evaluate\ table\ as\ below:$

$$evaluated\ table = \begin{bmatrix} error & \tau & m_{row} & m_{col} & l_{row} & l_{col} \\ \vdots & & & \vdots & & \end{bmatrix}$$

In which:

$\tau = 1$	$\tau = 2$	$\tau = 3$
$bias$	$input\ layer$	$hidden\ and\ output\ layers$
$bias\ weights\ for\ m_{row}th\ layer\ m_{col} = 1$	$weights\ of\ m_{row}th\ input\ m_{col} = 1$	$weight\ for\ layer\ m_{row}to\ m_{col}$
$row\ l_{row}of\ weight\ matrix\ l_{col} = 1$	$l_{row}\ and\ l_{col}shows\ weigh\ matrix$	$l_{row} = 1\ l_{col}shows\ weight\ matrix\ element$

$normalize\ weights\ in\ [0,1]\ range$
$loop\ weights$
$Initialize\ weight_{layer,i,j}$
$weight_{bias} = weight_{0,i,j}$
$weight_{hidden} = weight_{layer,i,j}$
$Criterion_{layer,i,j} = weights_{layer,i,j} \times \nabla weights_{layer,i,j}$
$sort\ Criterion\ \ matrix\ ascending$
$\boldsymbol{for}\ Criterion_{layer,i,j} = 1\ to\ Prune_{percent} \times count(Criterioan)$
$PrunneNominated_{layer,i,j} = PruneNominated_{layer,i,j} - 1$
$\boldsymbol{for}\ Criterion_{layer,i,j} = (1 - Prune_{percent}) \times count(Criterion) to\ count(Criterion)$
$PruneNominated_{layer,i,j} = PruneNominated_{layer,i,j} + 1$
$\boldsymbol{for}\ (layer, i, j) = PruneNominated_{layer,i,j} = maximum\ allowed\ warning$
$weight_{layer,i,j} = 0$

$$value_{layers} = \sum_{i=0}^{layer\ count} value_{laye_i}$$

$$layerpercent_i = \frac{value_{layer_i}}{value_{layers}}\ i = 0\ ..\ layer\ count$$

The use of the gradient somehow adds the rate of change to the criterion and dynamically increases the accuracy Our method, which is more accurate than other baselines, prunes the weights with the lowest absolute value of (weight * gradient), evaluated on a batch of inputs. Therefore, it is very significant and important that, firstly, our proposed criterion is much more accurate and dynamic than the simple weight criterion, and on the other hand, in addition to reducing the score of weak connections, we also give points to strong connections, and besides all this, we also use synaptic pruning in an innovative setting.

Results and Discussion

We verify the merits of the presented technique by two practical examples: identification of a refinery distillation tower and also adjusting the coefficients of a PID controller, which is the most famous and widely used controller in process industries. For the first one we use a deep feedforward neural networks as the system model, and for the second one a deep recurrent neural network as the online tuner of the controller.

Distillation Tower (Refinery)

The effectiveness of the proposed strategy is evaluated using a neural network model of the distillation tower in a refinery operation. The goal is to find out how this algorithm may improve identification accuracy and convergence speed while dealing with ideal and noisy data. Refineries are incomplete without the distillation tower, a multi-input, multi-output (MIMO) nonlinear system. One tool for separating solution components is a distillation column. Actually, the boiling point difference and volatility of the constituents of a solution are used to separate them in the distillation tower. Crude oil refining is one of the principal applications for industrial distillation towers, which are widely utilized in many process sectors. The distillation process is used in the oil business to separate various hydrocarbons according to how volatile they are. One of the most commonly utilized towers is the ethane-ethylene distillation column. The production of high-purity ethylene is necessary because of its importance. Our data comes from an identification experiment using an ethane-ethylene distillation column [27]. The data contains four series:

U_dest, Y_dest: without noise (ideal series)
U_dest_n10, Y_dest_n10:
10 percent additive white noise
U_dest_n20, Y_dest_n20:
20 percent additive white noise
U_dest_n30, Y_dest_n30:
30 percent additive white noise

There are 180 samples for neural network training. The following describes the inputs and outputs:

The inputs of the systems are: 1) the proportion between feed flow and reboiler duty; 2) the relationship between feed flow and reflux rate; 3) the proportion between the feed flow and the distillate; 4) the composition of the input ethane; and 5) the top pressure. Outputs of the system are: 1) top ethane composition; 2) bottom ethylene composition; and 3) top-bottom differential pressure. So, we employ a deep network with 90 connections, 5 inputs, and 3 outputs (Fig. 3). If we first have the proper weights training, it can be utilized the deep network's capabilities. Secondly, we can use our structural optimization approach to discover the optimal structure for the network and prevent over-fitting, which will speed up the network.

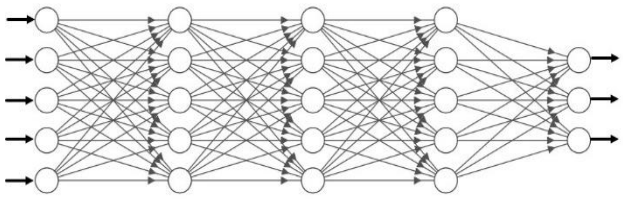

Fig. 3: Applied deep neural network.

Using the available data, we first train the network. Fig. 4 illustrates how the network's performance varies after each training epoch. Our performance function is the mean square error (MSE). Three curves with various colors are included for the test, validation, and training sets of data. The label on the horizontal axis shows the number of training cycles (epochs) of the network. The network performed its best in the 9th epoch, as evidenced by the validation data. Furthermore, Fig. 5 displays the regression curves for the test, validation, and training sets of data.

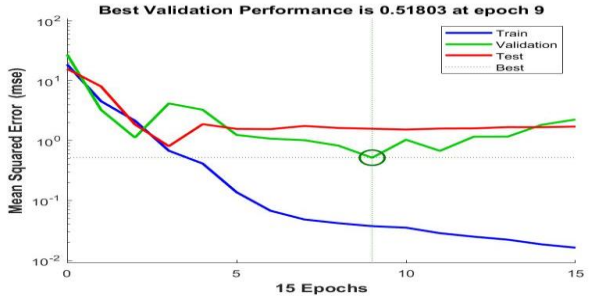

Fig. 4: Performance of the deep neural network.

More information from the training is shown in the training state visualization (Fig. 6); for instance, the "val fail" graph indicates the epoch in which the validation data evaluation was rejected. The cumulative number of failed assessments is displayed on this graph. When the network fails six consecutive assessments, training ends.

Two important measures are usually used for comparison to the simple dropout method [20]: The new size divided by the original size is defined as the compression ratio. The theoretical speedup is defined as the ratio of the initial number of multiplications and additions to the new number. Comparison statistics for two circumstances are reported in Table 1. Note that the *weight* × *gradient* is employed here.

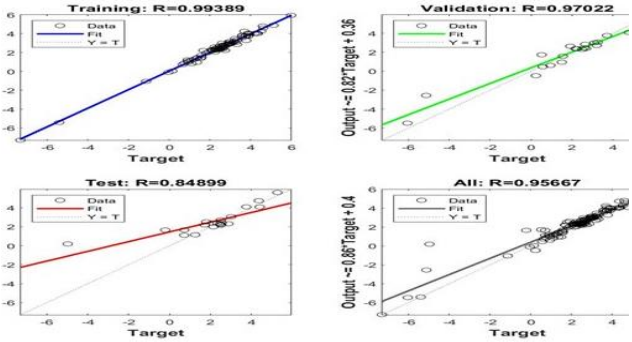

Fig. 5: Regression for the training, validation and test data.

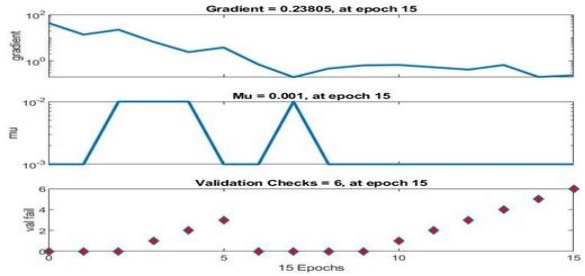

Fig. 6: Training state.

Also, in Table 2, the proposed pruning method is compared to the conventional dropout method wherein only weight is used. It verifies the superiority of the proposed scheme.

Table 1: Comparison to dropout method with weight × gradient

NN Type	Shallow (1 hidden layer)			Deep (3 hidden layers)		
	Initial	Dropout	Proposed	Initial	Dropout	Proposed
Accuracy (%)	76.6	77.1	77.9	82.6	83.8	86.3
Compression (%)	-	47	47.2	-	58	59.5
Execution Time (ms)	15	17.4	13.5	17.5	19.2	16.3

Table 2: Comparison to dropout method with only weight

NN Type	Shallow (1 hidden layer)			Deep (3 hidden layers)		
	Initial	Dropout	Proposed	Initial	Dropout	Proposed
Accuracy (%)	76.6	78.2	77.9	82.6	84.8	86.3
Compression (%)	-	48.3	47.2	-	59.4	59.5
Execution Time (ms)	15	17.3	13.5	17.5	18..2	16.3

As seen, our pruning strategy leads to a speedup in training and network performance. The suggested pruning strategy may be easily extended to other intelligent process industries. One of the most important problems in measurement and control is noisy data, which is frequently found in actual industrial settings. When working with noisy data that has 10%, 20%, and 30% noise, the outcomes of the shallow network and the deep network, which is pruned using the introduced algorithm, are compared in Fig. 7. It is clear that the suggested structure works much better, especially with noisy data.

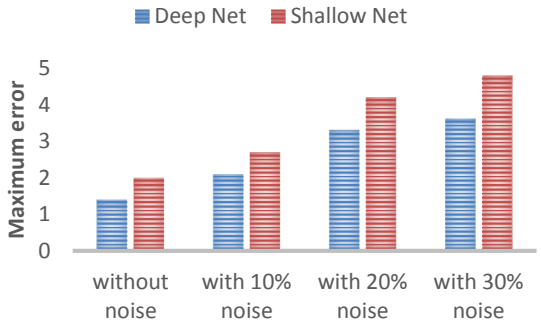

Fig. 7: Deep and shallow networks comparison in the noisy data treatment.

Briefly, the distillation tower is modelled using a deep network pruned using the suggested approach, and its effectiveness was shown in comparison to the shallow network. In order to compare the suggested model with other neural network-based models, we also compared the RMSE criteria between the model and three other structures in Table 3. The two structures that are being discussed are NARX structure-based neural networks

(which use both the Levenberg-Marquardt and the Steepest Descent algorithms) and nonlinear auto-regressive with exogenous inputs (NARX)-based ANFIS [28]-[30]. The comparison of errors clearly shows that the proposed technique is better than the other structures.

Table 3: RMSE for neural networks models, ANFIS, and the proposed CONCOMP

Outputs	Steepest Descent	Levenberg Marquardt	ANFIS	CONCOMP
Top Composition	0.639	0.2090	0.0421	0.0222
Bottom Composition	1.3127	0.4913	0.031	0.021
Pressure Difference	1.0053	0.2480	0.0189	0.0112

PID Controller

PID controllers are frequently mistuned, particularly in unreliable situations. Intelligent techniques are used recently to develop adaptive PID controllers. In order to mitigate the effects of uncertainties in the closed-loop control system, a deep dynamic neural network is used here to tune the parameters of the traditional PID controller. By using the proposed pruning method, simpler tuner is achieved and consequently the computational load is decreased.

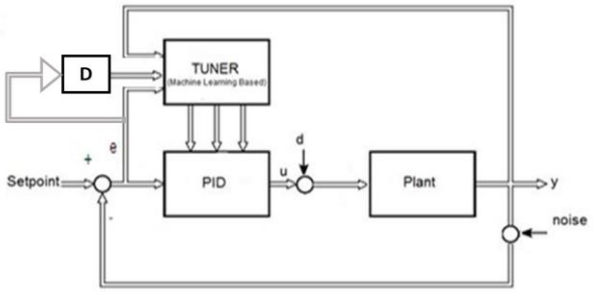

Fig. 8: Closed loop PID control with neural network tuner.

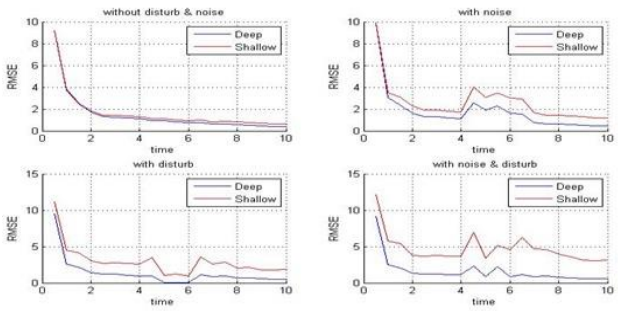

Fig. 9: RMSEs with shallow and pruned deep neural network tuner.

The transfer function of the plant in Fig. 8 is as follows:

$$H(s) = \frac{-(1.308)e^{-4.896s}}{(13.515s+1)(6.241s+1)} \quad (3)$$

The results of the Monte Carlo simulation with 100 iterations are reported in Fig. 9. At the end of the pruning process, we lost 43.5% of initial connections and reached a fast network. As can be seen, a controller with a pruned deep recurrent network tuner has superior performance compared to a shallow one. Regarding the stochastic nature of noise, deep networks better compensate for its effects.

Conclusion

This paper suggests a novel method for optimizing deep neural network topology. The weight multiplied by the gradient is employed as the criterion rather than the net weight index. Moreover, the low and high scores of connections are classified to determine the importance of the connections which compete for survival. We evaluated our method using two examples from control literature: neural network modelling of the distillation column and intelligent tuning of PID controller. We discovered that it eliminated over-fitting issues, enhanced learning speed, preserved accuracy, and reduced cost due to a smaller network size. Additionally, we demonstrated that deep neural networks in the right size and setting can handle noisy data more efficiently. This approach's main advantage over the previous ones is that a connection with a low score is not immediately terminated, and it continually assesses the effectiveness of network connections using the proposed criterion. Subsequent studies could focus on how to integrate the developed strategy with other advanced techniques like neural architecture search or automated machine learning.

Author Contributions

A. Ahmadi and R. Mahboobi Esfanjani both developed ideas, interpreted the results, wrote the manuscript, Revised and finalized it.

Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

Abbreviations

DNN Depp Neural Network
MIMO Multi-Input Multi-Output
PID Proportional–Integral–Derivative

References

[1] Z. Allen-Zhu, Y. Li, Y. Liang, "Learning and overparameterized neural networks, going beyond two layers," Adv. Neural Inf. Process. Syst., 32, 2019.

[2] C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, S. Yang, "AdaNet: Adaptive structural learning of artificial neural networks," in Proc. 34th International Conference on Machine Learning, 70: 874-883, 2017.

[3] O. A. Montesinos L., A. Montesinos L., J. C. Montesinos, Multivariate Statistical Machine Learning Methods for Genomic Prediction, Springer, 2022.

[4] S. Li, T. Hoefler, "Chimera: efficiently training large-scale neural networks with bidirectional pipelines," in Proc. International

Conference for High Performance Computing, Networking, Storage and Analysis, 2021.

[5] V. Sze, Y. H. Chen, T. J. Yang, J. Emer, "Efficient processing of deep neural networks: A tutorial and survey," arXiv preprint arXiv:1703.09039, 2017.

[6] S. A. Janowsky, "Pruning versus clipping in neural networks," Phys. Rev. A, 39(12): 6600-6603, 1989.

[7] G. Chechik, I. Meilijson, E. Ruppin, "Neuronal regulation: a biologically plausible mechanism for efficient synaptic pruning in development," Neurocomputing, 26-27: 633-639, 1999.

[8] M. V. Johnston et al., "Plasticity and injury in the developing brain," Dev. Neurosci., 31: 1-10, 2009.

[9] G. Hinton, O. Vinyals, J. Dean, "Distilling knowledge in a neural network," arXiv preprint arXiv:1503.02531, 2015.

[10] S. Han, J. Pool, J. Tran, W. J. Dally, "Learning both weights and connections for efficient neural networks," Adv. Neural Inf. Process. Syst., 2015: 1135–1143, 2015.

[11] P. Molchanov et al., "Importance estimation for neural network pruning," arXiv preprint arXiv:1906.10771, 2019.

[12] P. Molchanov et al., "Pruning convolutional neural networks for resource-efficient inference," in Proc. ICLR 2017, 2017.

[13] Y. LeCun et al., "Optimal brain damage," Adv. Neural Inf. Process. Syst., 2, 1990.

[14] B. Hassibi, D. G. Stork, "Second-order derivatives for network pruning: Optimal Brain Surgery," Adv. Neural Inf. Process. Syst., 5: 164-171, 1993.

[15] Y. He et al., "Filter pruning via geometric median for deep convolutional neural network acceleration," in Proc. CVPR 2019: 4340-4349, 2019.

[16] R. Yu et al., "NISP: pruning networks using neuron importance score propagation," in Proc. IEEE CVPR 2018: 9194-9203, 2018.

[17] H. Li et al., "Pruning filters for efficient convolutional neural networks," arXiv preprint arXiv:1608.08710, 2016.

[18] Y. He et al., "Soft filter pruning for accelerating deep convolutional neural networks," arXiv preprint arXiv:1808.06866, 2018.

[19] N. Srivastava et al., "Dropout: A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res., 15: 1929-1958, 2014.

[20] Z. Liu et al., "Dropout reduces underfitting," in Proc. 40th International Conference on Machine Learning, 202, 2023.

[21] X. Sun et al., "MeProp: Sparsified back propagation for accelerated deep learning with reduced overfitting," arXiv preprint arXiv:1706.06197, 2017.

[22] S. A. Mirjalili, Evolutionary Algorithms and Neural Networks, Springer, 2019.

[23] M. Kotyrba et al., "The influence of genetic algorithms on learning possibilities of artificial neural networks," Computers, 2022.

[24] F. Zhao et al., "Towards a brain-inspired developmental neural network by adaptive synaptic pruning," in Proc. ICONIP 2017, 2017.

[25] A. Ahmadi, B. Mashoufi, "A new optimized approach for artificial neural network training using genetic algorithms and parallel processing," Int. Rev. Comput. Softw., 7(5): 1828-6003, 2012.

[26] M. Morini et al., "Strategies and tools for studying microglial-mediated synapse elimination and refinement," Front. Immunol., 2021.

[27] R. P. Guidorzi et al., "The range error test in the structural identification of linear multivariable systems," IEEE Trans. Autom. Control, AC-27: 1044-1054, 1982.

[28] A. Saha and S. Patra, "Modeling and control of distillation column using ANFIS," J. Autom.Control, 51: 51-59, 2021.

[29] H. Zhao, J. Zhang, M. Wang, "Modeling and optimization of distillation column processes using neural networks and genetic algorithms," IEEE Access, 7: 76345-76354, 2019.

[30] A. Singh, P. Gupta, R. Verma, "Hybrid ANFIS and neural network-based approach for fault detection in industrial distillation processes," Appl. Soft Comput., 96, 106703.

Biographies

Aghil Ahmadi received B.Sc. degree in Electrical Engineering (Control Systems) from Sahand University of Technology, Tabriz, Iran, in 2004. From 2004 to 2009, he worked as a control systems and instrumentation supervisor engineer for several world-class projects in the oil and gas industries. He received the M.Sc. degree in Electronics Engineering from Urmia University, Urmia, Iran, in 2011. From 2009 to 2011, he was doing research at the High-Performance Computing Research Centre, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. He has been a Ph.D. researcher at Sahand University of Technology since 2018. His current research interests include machine learning, intelligent control systems, fuzzy systems, and professional project management.

- Email: ag_ahmadi@sut.ac.ir
- ORCID: 0009-0007-6478-1685
- Web of Science Researcher ID: NA
- Scopus Author ID: NA
- Homepage: NA

Reza Mahboobi Esfanjani received the B.SC. degree (Hons.) from the Department of Electrical Engineering, Sahand University of Technology, Tabriz, Iran, in 2002, and the M.Sc. and Ph.D. degrees from the Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2004 and 2009, respectively. In 2010, he joined the Sahand University of Technology, where he is currently a Professor in the Department of Electrical Engineering. His current research interests include analysis and control of networked dynamical systems.

- Email: mahboobi@sut.ac.ir
- ORCID: 0000-0002-0341-8141
- Web of Science Researcher ID: NA
- Scopus Author ID: NA
- Homepage: NA

How to cite this paper:
A. Ahmadi, R. Mahboobi Esfanjani, "Structure learning for deep neural networks with competitive synaptic pruning," J. Electr. Comput. Eng. Innovations, 13(1): 189-196, 2025.

DOI: 10.22061/jecei.2024.11017.758

URL: https://jecei.sru.ac.ir/article_2215.html