Simulation-based structural optimization for neural networks*

Szymon Świderski[a] and Agnieszka Jastrzębska[a,b]

[a]Warsaw University of Technology, Warsaw, Poland
[b]The John Paul II Catholic University of Lublin, Poland

Abstract. Optimizing already trained neural networks is one of the core problems in the domain of Artificial Intelligence. In this paper, we present a new approach capable of optimizing the structure of an already trained neural network. We present a new way of removing neurons from layers that is based on matrix scaling, which truly decreases the number of parameters in a model rather than zeroing weights. We present a simulation-based approach for optimizing the structure of a neural network that can select the best change to the network without causing data loss. A suite of empirical experiments demonstrated that the proposed approach can optimize already-trained neural networks, achieving up to 89% parameter reduction in the best case (OrganMNIST) with less than 0.05 decrease in performance. On average (excluding KMNIST), the method reduces parameters by 37% with minimal degradation (0.02). The resulting models use less than 2% of the parameters of ResNet-18 while maintaining comparable performance, supported by a neuron removal method based on matrix rescaling.

Keywords: neural network compression · post-training optimization · model size reduction · neuron pruning · simulation-based optimization

1 Introduction

The demand for efficient neural networks is increasing. Although there are many methods for training neural networks, there are very few tools to optimize them. Therefore, in this paper, we propose a novel method for optimizing the neural network structure while preserving its previously learned knowledge.

The method presented in this paper is general and can be applied to any existing neural network structure. Starting from a given model, the algorithm iteratively modifies the structure, gradually transforming it into a more optimized version. The term "optimization" in this context refers to reducing the number of parameters. Each structural modification is designed to preserve previously learned knowledge. However, small losses in learned information may still occur. For this reason, access to the original training dataset is required to effectively optimize the network structure.

* The research was funded by the National Science Centre, Poland, grant number 2024/53/B/ST6/00021.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_48

During optimization, the simulation may apply an unusual and complex sequence of actions to reduce the size of the model. For example, this can include removing large layers and replacing them with several smaller ones, or gradually reducing the number of neurons in selected layers. If the simulation is configured correctly, the resulting action tree can become highly complex. However, it is guided by the score function, which prioritizes maintaining accuracy while reducing the number of parameters.

To support reproducibility, we have released our implementation as an open-source package named `growingnn`. The package is available on PyPi at https://pypi.org/project/growingnn/, where all relevant links, including the GitHub repository with the source code and documentation, can be found. This tool is implemented entirely from scratch using only the NumPy library. As a result, migrating it to larger frameworks such as PyTorch or TensorFlow may be challenging, but it is still possible and is planned as future work.

2 Literature survey

Currently, one of the most well-known approaches capable of stable structural changes is the method proposed by Evci *et al.*, called GradMax, grows a network neuron-by-neuron by leveraging gradient information to decide where adding capacity will most reduce loss function [4]. GradMax demonstrated faster convergence and better final performance than static networks of equivalent size, and was validated on image classification benchmarks. Another recent paper by Yoon *et al.* proposed Dynamically Expandable Networks (DEN) for lifelong learning, which dynamically expands network capacity by performing selective retraining [14].

In addition to the aforementioned methods, which we consider most relevant to our work, there exist several other approaches. A comparative description of the approaches available in the field of dynamic neural networks can be found in two survey papers. In the first survey by Han *et al.* [7], the authors offer a clear categorization of different types of dynamic neural networks. The first family discussed is sample-wise dynamic networks. These networks can be seen as multiple subnetworks combined together. This approach was, for example, implemented in BranchyNet [13], which introduces multiple intermediate "exit" classifiers along a deep network. During inference, if an early classifier is sufficiently confident, the prediction is produced immediately, skipping subsequent layers. Bolukbasi *et al.* [1] proposed a similar idea by training an adaptive cascade of models of increasing depth.

The second interesting survey that summarizes and categorizes algorithms in the dynamic neural networks domain is titled "Dynamic Neural Network Structure: A Review of Its Theories and Applications" [5]. It proposes a completely different set of categories that better fit the idea of growing and shrinking neural networks. The first notable category is named "Dynamic architectures". These methods allow a network's topology (neurons, layers, or sub-networks) to evolve

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_48

during training which also includes pruning or structure reduction for eliminating features[1].

In addition to manual growth/pruning heuristics, another major branch of dynamic network construction methods was listed under the name "Neural Architecture Search" (NAS). A pioneering work by Zoph and Le [15] framed architecture search as a reinforcement learning problem, by training a controller RNN to generate optimal convolutional network architectures. Many subsequent NAS methods introduced alternative search strategies.

3 The method

From a high-level point of view, the proposed method aims to modify the structure of the neural network to improve the accuracy that takes place while the network is training. An essential use-case scenario for the proposed method is to take an already-trained neural network or a pre-trained neural network and use our method as a fine-tuning tool.

3.1 Simulation description

The decision-making process for choosing the best action uses a variant of Monte Carlo Tree Search (MCTS) [11], adapted for neural network structure optimization. Training is divided into generations. In each generation, the simulation starts from the current model M and expands it by considering all possible actions $a \in \mathcal{A}(M)$. Each action produces a new candidate model M_a. We then evaluate M_a using a short training step and assign it a score $S(M_a)$ based on the weighted sum. In the expansion phase, every action a is applied to the current model M, producing a new child node with model $M_a = a(M)$. From each model M_a, we simulate random actions up to depth d, briefly training and evaluating intermediate models. We use UCB1 to balance exploration and exploitation: $\mathrm{UCB1}(n) = v(n) + c \cdot \sqrt{\frac{\ln N}{n}}$ [3]. The score of M_a is a weighted combination of accuracy and parameter count:

$$S(M_a) = \frac{w_{\text{acc}} \cdot S_{\text{acc}}(M_a) \; + \; w_{\text{params}} \cdot \frac{1}{1+\beta \cdot p_a}}{w_{\text{acc}} + w_{\text{params}}}, \tag{1}$$

where $S_{\text{acc}}(M_a)$ denotes the short-run training accuracy of the model. The coefficient w_{acc} is fixed at 1.0 in all experiments, serving as a reference since accuracy is the main optimization objective. The variable p_a represents the total number of parameters in M_a, including weights, biases, and filter parameters across all layers. The meta-parameter β (currently set to 0.001) adjusts how the grade for the number of parameters is computed.

[1] Interestingly, pruning and structure reduction methods for eliminating features date back to optimal brain damage analysis (see the work of LeCun *et al.* [9]) and optimal brain surgeon concept (see the work of Hassibi *et al.* [8]).

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_48

The parameter w_{params} controls how strictly parameter count affects the simulation score and is a key focus of our experiments. Since models are compared from the same initial structure, those with similar accuracy are preferred when they have fewer parameters.

3.2 Removing neurons

The contribution presented in this paper addresses a more challenging optimization task requiring a broader set of actions. We introduce operations to **add or remove neurons**, along with a **novel mathematical formulation** for neuron removal. Unlike prior approaches [6], neurons are fully removed rather than zeroed, reducing weight matrix size while aiming to **preserve learned information** through controlled scaling.

The neuron removal action in some layer ℓ reduces the number of neurons from n to n' by scaling the weight matrix $W_\ell \in \mathbb{R}^{n \times d}$ and bias vector $b_\ell \in \mathbb{R}^n$ using a quasi-identity matrix $Q \in \mathbb{R}^{n' \times n}$:

$$W'_\ell = Q_I W_\ell, \quad b'_\ell = Q_I b_\ell \tag{2}$$

A quasi-identity matrix Q is constructed by resizing an identity matrix, preserving information through optimal linear projection rather than truncation or averaging. This approach is very beneficial because if the reduction ratio is small, for example $r = 0.1$, the matrix Q_I is almost identical to the identity matrix. From the formulas above, we can then assume that $W'_\ell \approx W_\ell$.

4 Empirical analysis

4.1 Dataset and empirical setup

Experiments were conducted on multiple datasets, including BreastMNIST, EMNIST Letters, FMNIST, MNIST, KMNIST, OrganMNIST, PneumoniaMNIST, and RetinaMNIST. For simplicity in tables and figures, we use abbreviated names (e.g., OrganMNIST as OrganM), while MNIST retains its original name. We focus on optimizing pre-trained neural networks. After selecting metaparameters, the algorithm is applied to fully trained models. We evaluate changes in accuracy and parameter count, aiming to reduce model size while maintaining performance. These effects are summarized by the **optimization score**.

The algorithm is implemented from scratch and follows a two-step procedure. First, the network is grown with early stopping to reach over 90% accuracy with minimal size. Second, the model is optimized, stopping if accuracy drops by less than 10% while parameters decrease by at least 20%, or when the maximum number of generations is reached.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_48

4.2 Classification quality of the new approach

The experiments carried out aimed to empirically validate two hypotheses concerning the proposed algorithm for structure optimization.

RH1 Reducing the number of neurons via matrix rescaling has a minimal impact on the model's predictive performance during the optimization process.

RH2 The growingNN algorithm for dynamic network structure modification can reduce the number of trainable parameters while preserving its predictive performance.

To verify the first hypothesis (RH1), we analyze how often neuron removal actions were applied during optimization and their effect on model accuracy. Table 1 summarizes the frequency of actions in all datasets.

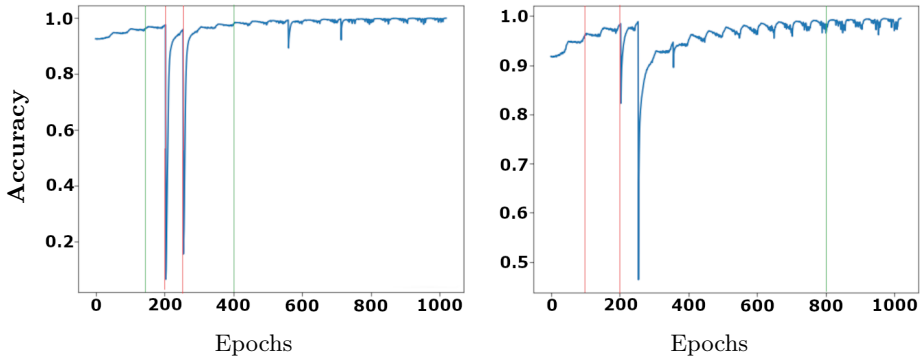
Table 1: Frequency of actions selected by the simulation across datasets.

Action	BreastM	EMDigits	EMLetters	FMNIST	KMNIST	MNIST	OrganAM	PneumoniaM	RetinaM	Sum
Add Neurons	1	1	0	0	1	0	0	0	0	3
Add Res. Conv Layer	2	0	1	1	1	1	0	0	2	8
Add Res. Layer	0	1	0	6	3	3	2	1	1	17
Add Seq. Conv Layer	2	0	0	1	0	0	0	0	1	4
Add Seq. Layer	0	0	0	3	3	0	0	1	0	7
Del Layer	1	0	1	2	3	1	1	0	1	10
Del Neurons (0.1)	0	1	0	0	2	0	0	1	2	6
Del Neurons (0.5)	0	0	0	0	4	0	0	1	1	6
Del Neurons (0.9)	0	0	0	0	2	0	0	0	0	2
Total	6	3	2	13	19	5	3	4	8	**63**

Table 1 shows that neuron removal was applied 14 times (6+6+2), more often than layer deletion. Since training stops after achieving 20% parameter reduction with less than 10% accuracy loss, the total number of actions is limited. Growth actions were generally more frequent, typically adding small layers before removing a larger one. Despite this, neuron removal was common and largely preserved accuracy (Table 2). In the original growingNN paper [10], we describe how new layers are added. To preserve performance, they must be properly initialized. Sequential layers use quasi-identity mappings, while residual layers allow more flexibility, making them more frequently used.

Figure 1 shows training histories from the optimization phase, with green lines indicating neuron removal actions. Accuracy remains stable overall, with only small, temporary drops that quickly recover, while larger drops are linked

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_48



(a) Fashion-MNIST (b) MNIST

Fig. 1: Training history during the second optimization phase with optimization weight set to 2, trainng was set to 20 generation each 50 epoch Green lines mark the execution of neuron removal actions, while red lines indicate layer removal actions. In the MNIST example, the largest accuracy drop is caused by adding a residual layer.

to other modifications. These results confirm that neuron removal via matrix rescaling preserves learned knowledge and is a stable, safe operation, supporting RH1.

The second hypothesis (RH2) was verified by analyzing changes in accuracy and parameter count in the evaluated datasets. To illustrate the optimization capacity of the proposed method, Table 2 compares the results obtained by ResNet-18 models in the same datasets [2] [12] with approximately 11 million parameters. Table 2 shows training accuracy and parameter count before and after optimization. The results demonstrate significant parameter reduction with only minor accuracy changes. Although initial models are already compact, the optimization phase further reduces their size, indicating effective action selection. Excluding the outlying result for KMNIST, the average parameter reduction is 37%, with an average performance decrease of 0.02. The method achieves up to 89% parameter reduction for OrganMNIST, with less than 0.05 decrease in performance. Even for the largest resulting models, the number of parameters remains below 2% of that of ResNet-18. Plots in Fig. 2 illustrate how structural changes impact model architecture. Early generations add small layers around a larger one, which is later removed, reducing parameters while maintaining accuracy. The process stops once parameters are reduced by over 20% without significant accuracy loss, as intended.

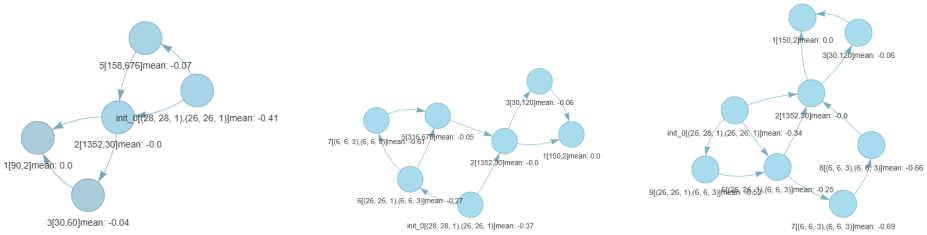
5 Conclusion

In this paper, we delivered a novel dynamic neural network training scheme that can be used to train a network from scratch and, importantly, can be used to

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_48

Table 2: Comparison of accuracy and parameter count before and after optimization across datasets.

Metric	BreastM	EM	FashionM	MNIST	KMNIST	OrganM	PneumoniaM	RetinaM
Acc. (Before)	0.91	0.91	0.92	0.98	0.90	0.91	0.91	0.83
Params (Before)	129077	144601	13629	6869	20589	20619	20349	131319
Acc. (After)	0.86	0.86	0.97	0.94	0.99	0.80	0.90	0.83
Params (After)	26853	125908	8450	7059	147059	2139	12200	118041
Param Change [%]	-79	-12	–37	2.7	614	-89	-40.0	-10
Acc. Change	-0.049	-0.048	0.049	-0.003	0.095	-0.108	-0.003	0.002
ResNet-18 Acc.	0.863	0.96	0.93	0.99	0.97	0.935	0.854	0.524
Relative Size	0.0024	0.0114	0.0008	9.0×10^{-8}	0.0134	0.0002	0.0011	0.0107



(a) Iteration 0 (b) Iteration 2 (c) Iteration 5

Fig. 2: Evolution of the neural network structure: (a) initial architecture, (b) architecture in 2nd iteration, (c) architecture in 5th (and last) iteration.

fine-tune an already-trained neural model. The method is guided by a parameter w_{params}, which influences the optimization process. Higher values lead to smaller network size but may cause larger accuracy drops. The properly chosen value of w_{params} allows a good balance between reducing network size and preserving accuracy. Each experiment was divided into a training phase and an optimization phase, simulating a practical scenario in which an already trained model is structurally optimized. The results indicate that this approach is effective in reducing model size while preserving most of the learned knowledge. A common strategy observed during the simulations was to add several small layers followed by the removal of a larger one, leading to a significant reduction in model size. This sequence of actions was selected most frequently, as also illustrated in Fig. 2. This resulted in parameter reductions of up to 89% on OrganMNIST, with only a minimal decrease in performance (below 0.05). On average (excluding KMNIST), the method achieved a 37% reduction in parameters with negligible performance degradation (0.02). Even for the largest resulting mod-

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_48

els, the number of parameters remained below 2% of those of architectures such as ResNet-18.

References

1. Bolukbasi, T., Wang, J., Dekel, O., Saligrama, V.: Adaptive neural networks for efficient inference. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 527–536. PMLR (2017)
2. Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Kazuaki, Y., Ha, D.: Deep learning for classical japanese literature (2018). https://doi.org/10.20676/00000341
3. Coquelin, P., Munos, R.: Bandit algorithms for tree search. CoRR **abs/cs/0703062** (2007), http://arxiv.org/abs/cs/0703062
4. Evci, U., Drozdzal, M., Unterthiner, T., Lamblin, P., Bacon, P.L.: Gradmax: Growing neural networks using gradient information. In: International Conference on Learning Representations (ICLR) (2022)
5. Guo, J., Chen, C.L.P., Liu, Z., Yang, X.: Dynamic neural network structure: A review for its theories and applications. IEEE Transactions on Neural Networks and Learning Systems **36**(3), 4246–4266 (2025)
6. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. In: Proceedings of the 29th International Conference on Neural Information Processing Systems. pp. 1135–1143. NIPS'15, MIT Press (2015)
7. Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y.: Dynamic neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(11), 7436–7456 (2022)
8. Hassibi, B., Stork, D., Wolff, G.: Optimal brain surgeon and general network pruning. In: IEEE International Conference on Neural Networks. vol. 1, pp. 293–299 (1993)
9. LeCun, Y., Denker, J., Solla, S.: Optimal brain damage. In: Touretzky, D. (ed.) Advances in Neural Information Processing Systems. vol. 2. Morgan-Kaufmann (1989)
10. Świderski, S., Jastrzębska, A.: Dynamic growing and shrinking of neural networks with monte carlo tree search. In: Franco, L., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) Computational Science – ICCS 2024. pp. 362–377. Springer Nature Switzerland (2024)
11. Swiechowski, M., Godlewski, K., Sawicki, B., Mandziuk, J.: Monte carlo tree search: A review of recent modifications and applications. Artificial Intelligence Review **56**(3), 2497–2562 (2023)
12. Tareen, S.A.K., Khan Tareen, F.: Optimally deep networks - adapting model depth to datasets for superior efficiency. In: 2025 5th International Conference on Digital Futures and Transformative Technologies (2025)
13. Teerapittayanon, S., McDanel, B., Kung, H.: Branchynet: Fast inference via early exiting from deep neural networks. In: 2016 23rd International Conference on Pattern Recognition (ICPR). pp. 2464–2469 (2016)
14. Yoon, J., Yang, E., Lee, J., Hwang, S.: Lifelong learning with dynamically expandable networks (2017)
15. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. CoRR **abs/1611.01578** (2016), http://arxiv.org/abs/1611.01578

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_48