Adapting the Size of Artificial Neural Networks Using Dynamic Auto-Sizing

Vojtech Cahlik
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czechia
cahlivoj@fit.cvut.cz

Pavel Kordik
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czechia
pavel.kordik@fit.cvut.cz

Miroslav Cepek
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czechia
miroslav.cepek@fit.cvut.cz

***Abstract*—We introduce dynamic auto-sizing, a novel approach to training artificial neural networks which allows the models to automatically adapt their size to the problem domain. The size of the models can be further controlled during the learning process by modifying the applied strength of regularization. The ability of dynamic auto-sizing models to expand or shrink their hidden layers is achieved by periodically growing and pruning entire units such as neurons or filters. For this purpose, we introduce weighted L1 regularization, a novel regularization method for inducing structured sparsity. Besides analyzing the behavior of dynamic auto-sizing, we evaluate predictive performance of models trained using the method and show that such models can provide a predictive advantage over traditional approaches.**

***Index Terms*—regularization, structured sparsity, neural architecture search, AutoML, auto-sizing**

I. Introduction

The field of searching for optimal model architecture and size for a given problem is quite old. One of the older approaches is *GMDH* [2] and its evolutions such as [3], and in recent years, the search for optimal structure of deep learning models attracted new focus as every learning iteration can be very costly. A simple but computationally intensive approach is to tune or search for the hyperparameters controlling the sizes of the hidden layers, namely the numbers of individual units such as neurons or filters. More sophisticated methods known as neural architecture search have been proposed, notably the family of methods based on the approach known as **auto-sizing** [4, 5, 6, 7]. Auto-sizing techniques automatically set the sizes of hidden layers without introducing significant computational overhead, as they operate during training itself by pruning parameters using a standard gradient-based optimizer. However, as these methods prune units only at the end of training, they are still computationally ineffective; moreover, they can not grow the models beyond their original sizes. More complex auto-sizing techniques such as *MorphNet* have been introduced that allow models to grow new units in hidden layers [8, 9], however the sizes of the models are constant during training with respect to resource constraints such as the number of floating-point operations per second (FLOPS), instead of adapting to the difficulty of the task.

This work was supported by the Student Summer Research Program 2021 of FIT CTU in Prague. The paper is based on the master's thesis of the main author [1] available at http://bit.ly/3UxoMau.

We build upon original auto-sizing by introducing **dynamic auto-sizing (DAS)**, a novel method for training feed-forward artificial neural network models that can dynamically change their size and structure according to the difficulty of the problem by automatically shrinking or expanding their hidden layers during training. This is achieved using the novel *weighted l_1 regularization* technique which induces structured sparsity in a model by penalizing every additional unit more, until some units are regularized so much that they effectively do not contribute to the model's outputs. During training, new units are periodically grown and unnecessary neurons are periodically pruned by being completely removed from the hidden layers, until the model stabilizes at a final architecture. Notably, the size of resulting models increases with growing complexity of the problem domain, and can be further controlled by adjusting the strength of regularization. In this paper, we experimentally analyze DAS on selected tasks and perform a comparison against selected baseline models. The promising results hint at the potential of use of the method in everyday model training, pruning, and growth, or in situations where the size of the model must react to concept drift in on-line or reinforcement learning tasks.

II. Related Work

A. Parameter Pruning

A classical technique on neural network parameter pruning is *Optimal Brain Damage* [10], which uses the second derivative of the objective function to estimate the change caused by deleting each parameter, and then prunes the least important parameters. A more recent approach is to use a sparsity-inducing regularizer, such as the l_1 norm [11]. *Connection pruning* was utilized for example in [12] and [13] for reducing the sizes of AlexNet and VGG-16 convolutional models, although with no major computational speedups, as with dense representations of artificial neural networks, parameters are grouped into units such as neurons or filters. Nevertheless, such groups of parameters can be pruned using a structured sparsity regularizer such as *group lasso* [14, 15].

B. Auto-Sizing

1) Original Paper: Auto-sizing [4] is a technique which automatically determines the numbers of neurons in hidden

Cahlik, V., Kordik, P., Cepek, M. (2022). Adapting the Size of Artificial Neural Networks Using Dynamic Auto-Sizing. https://doi.org/10.1109/CSIT56902.2022.10000471 | 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT) | 979-8-3503-3431-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/CSIT56902.2022.10000471

layers of a fully-connected artificial neural network, and can additionally be used to prune previously trained models. The method works as follows: first the weights of incoming connections of every neuron in the model are grouped, the model is regularized using a structured sparsity regularizer, and subsequently trained. If the regularization term is sufficiently large, then at the end of training some of the neurons have all of the weights of the incoming connections set to a value close to zero. These neurons can thus be pruned from the model altogether. In the original paper, two structured sparsity regularizers were used, namely the $l_{2,1}$ and $l_{\infty,1}$ norm. It was shown that auto-sizing can lower the perplexity of neural language models while decreasing the number of parameters.

2) Related Techniques: Approaches very similar to auto-sizing were investigated in [5, 6] with deeper convolutional models, and it was shown that the methods can considerably reduce the number of parameters of a model while retaining its predictive accuracy. Auto-sizing was also examined further by its original authors with Transformer models in [7]. Another paper [8] presented *MorphNet*, a neural network architecture design method which aims to find an optimal model under the specified resource constraint (such as model size or inference speed). The method works by iteratively shrinking and growing the model while training. Shrinking is achieved by utilization of a sparsity-inducing regularizer, more specifically the l_1 norm on the γ_L variables of batch normalization [16]. Growing is performed by uniformly expanding all hidden layer sizes as much as the resource constraint allows, thus restructuring the model while approximately preserving its original size. A similar technique is proposed in [9], capable of activation and deactivation of entire hidden layers by using indicator variables indicating the presence of each component. These variables are continuous, but effectively approach binary values as training progresses.

C. Neural Architecture Search

Neural architecture search is a subfield of AutoML that focuses on automating the process of designing artificial neural network architectures. One noteworthy technique is the *differentiable neural architecture search* (DARTS) framework presented in [17], which allows to find an optimal architecture of deep learning models by working with high-level cells composed of arbitrary operations, and selecting over possible combinations of these cells using softmax. Another notable method is *EfficientNet* [18], which optimizes the architecture of convolutional neural networks using a small baseline model by estimating its optimal ratio between increasing the number of layers, number of filters per layer, and the resolution of inputs.

Also worth mentioning are approaches utilizing growth of new hidden units, for example custom learning algorithms like *Cascade-Correlation* [19, 20, 21]. Growth is also employed in *Topology and Weight Evolving Artificial Neural Network* (TWEANN) methods such as *NEAT* [22, 23, 24].

III. Methods

A. Weighted l_1 Regularization

DAS requires a structured-sparsity-inducing regularizer for limiting the size of the model during training. For this purpose, we introduce *weighted l_1 regularization*. The principle is as follows: the units (in this paper neurons or filters) in every hidden layer are numbered from first to last, and a higher strength of l_1 regularization is set to the parameters belonging to all incoming connections of units with a higher index. Units with a sufficiently high index are then typically regularized so much that all of their parameters obtain near-zero values, and these units can thus be pruned, i.e. removed from the weight tensors, without any relevant changes in the model's outputs[1].

The general formula for the weighted l_1 regularization term is

$$\alpha\Omega(\mathbf{W}) = \alpha \sum_{i=1}^{N_l} \sum_{j=1}^{N_{l+1}} \sum_{k=1}^{P_l} f(j)|\mathrm{W}_{ijk}|, \tag{1}$$

where W_{ijk} is the k-th parameter of the connection from the i-th unit (neuron or filter) in the l-th layer to the j-th unit in the $(l+1)$-th layer, N_l is the number of units in the l-th layer, P_l is the number of parameters per outgoing connection from l-th layer, α is a hyperparameter controlling the emphasis on the regularization term in contrast to the rest of the cost function, and f is an arbitrary non-decreasing function that sets regularization strength for the unit with index j. Therefore in every hidden layer, the parameters of every connection leading to the j-th unit are l_1-regularized with the coefficient of $\alpha f(j)$.

In the experiments presented in this paper, an identity $f(j) = j$ is used for the function f, as this typically results in a roughly linear relationship between the hyperparameter α and the sizes of the resulting models. However, f can in general take the form of any non-decreasing function passing through origin. Regarding the bias terms, in this work we consider them to be parameters belonging to special connections and regularize them as well, as this is necessary for the pruning of whole units. Weighted l_1 regularization should not be applied to the output layer.

B. Dynamic Auto-Sizing

The training of a neural network regularized with weighted l_1 regularization typically leads to the state in which for units with a high-enough index, the optimizer has set the parameters of all of the incoming connections to values very close to zero. At the end of training, such units can be pruned, and the resulting model can be used for inference. The initial layer sizes usually have little influence over the layer sizes of the final model, provided that they were large enough. This is the essential principle of the original auto-sizing method as described in section II-B1, which however used the $l_{2,1}$ and $l_{\infty,1}$ regularizers.

DAS brings two major improvements. The first enhancement is that units that do not contribute to the model outputs are pruned periodically instead of only at the end of training,

[1] We assume the use of activation functions that satisfy $f(0) = 0$.

Algorithm 1 Dynamic auto-sizing

$model \leftarrow$ randomly initialized model, regularized with suitable sparsity inducing regularizer
$\tau \leftarrow$ pruning threshold
$\gamma_{perc} \leftarrow$ growth percentage
$\gamma_{min} \leftarrow$ minimum growth
for each epoch **do**
  for each dense or convolutional hidden layer l **do**
    $n \leftarrow$ number of units in l
    $\gamma \leftarrow max(n\gamma_{perc}, \gamma_{min})$
    Grow l by adding γ new units with parameters randomly initialized and multiplied by τ
  end for
  Train $model$ for one epoch on the train set
  for each dense or convolutional hidden layer l **do**
    Prune l by removing units with all parameters smaller in absolute value than τ
  end for
end for
return $model$

which improves performance as calculations are performed with smaller tensors. The second enhancement, which addresses the problem of original auto-sizing that training has to start with large layer sizes, is to periodically "grow" new units by adding them to the corresponding weight tensors with small initial random values of parameters. The parameter values are initially so small that the outputs of the model are virtually not influenced, but can be progressively increased by the optimizer; otherwise, the parameters are pruned in the next pruning step. Therefore, it is possible to start the training with a small model as it can automatically increase its size if necessary, leading to computational efficiency and relaxed requirements on the initial layer sizes. The complete pseudocode of DAS is presented in Algorithm 1.

In practice, a model trained using DAS starts with the initially set layer sizes and gradually keeps growing or shrinking. The size of the model typically stabilizes after some number of epochs, in a state near to an equilibrium in which all of the units added during the last growing step are pruned during the subsequent pruning step. Utilizing larger regularization strength α typically leads to a smaller resulting model.

IV. Experiments and Results

A. Setup

For the presented experiments, our implementation of DAS[2] was configured as follows: growth percentage λ_{perc} and minimum growth λ_{min} were set to 0.2 and 20, respectively, while the pruning threshold τ was set to 10^{-3}. Unless stated otherwise, weighted l_1 regularization was used, the regularization strength α was set to 2×10^{-5}, and the initial number of units was set to 100 for each hidden layer[3]. The number of training epochs was set so that the size of the model as well as the validation metric would be stabilized in all experiments. In some experiments, after the model was trained using the DAS approach, "fine-tuning" in a static manner (without regularization, growth, or pruning) was performed, as will be noted in the corresponding sections.

All models in the experiments are composed of five hidden layers, usually with the first four being convolutional and the final one being fully-connected. In case of convolutional models, the second and fourth convolutional layer used strides of 2 and were each followed by a dropout layer with dropout rates of 0.2 and 0.5, respectively. Self-normalization [26] was used to mitigate the problem of unstable gradients by standardizing the input data, using LeCun normal initialization for all parameters, and using the SELU activation function for all hidden layers. Training was performed using Adam optimizer with the categorical cross-entropy cost in classification tasks and the mean squared error cost in regression tasks.

B. Dependency of Discovered Layer Sizes on Various Factors

Training a DAS model with higher regularization strengths α typically leads to smaller resulting layer sizes, as can be seen in Fig. 1. On the other hand, DAS models tend to increase in size with the growing complexity of the problem domain, as can be seen in Table I, which lists the numbers of parameters, layer sizes, and accuracies of convolutional models trained using DAS on the MNIST, Fashion MNIST, SVHN (Street View House Numbers) [27], CIFAR-10, CIFAR-100, and Tiny ImageNet [28] datasets. This is not a surprising phenomenon, as the training of a small model on a complex task typically leads to a large training loss, which in turn leads to the regularization term not having as much importance in the cost function, allowing the model to grow. Further experiments

[2]The TensorFlow implementation, together with all of the presented experiments, is available under the MIT license at https://github.com/vcahlik/dynamic-auto-sizing.

[3]The values of DAS hyperparameters were mostly obtained by manual tuning and seem to work well for many problem domains. The pruning threshold τ was set heuristically so that the changes in the model outputs are negligible when the parameters are pruned.

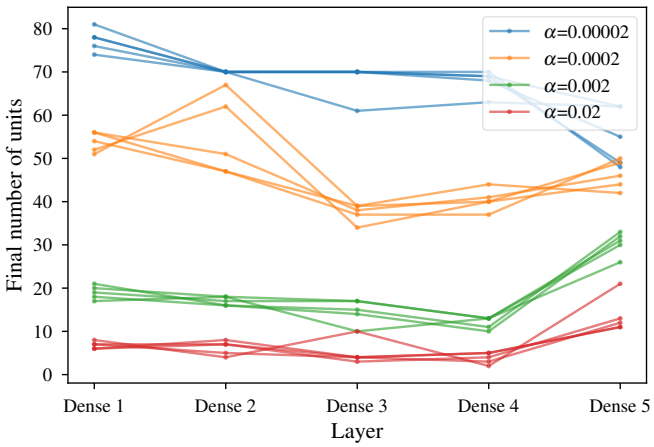

Fig. 1. Final layer sizes of fully-connected DAS models trained on the 15-puzzle dataset [25] with various regularization strengths α. Each curve represents one model.

TABLE I

AVERAGE SIZES (TOTAL PARAMETER COUNTS AND NUMBERS OF UNITS IN HIDDEN LAYERS) OF CONVOLUTIONAL MODELS TRAINED USING DAS ON VARIOUS COMPUTER VISION DATASETS, TOGETHER WITH CROSS-VALIDATED ACCURACIES AFTER STATIC "FINE-TUNING".

Dataset	**Parameters**	**Hidden layer sizes**					**Cross-val. accuracy**
		conv1	**conv2**	**conv3**	**conv4**	**dense**	$(mean \pm 2SD)$
MNIST	166K	19	14	18	34	93	99.4 % ± 0.1
Fashion MNIST	233K	22	15	21	36	124	93.3 % ± 0.5
SVHN	471K	17	16	21	45	158	93.1 % ± 0.3
CIFAR-10	665K	39	18	26	57	175	77.0 % ± 0.7
CIFAR-100	851K	66	20	30	64	194	45.1 % ± 1.1
Tiny ImageNet	5.32M	47	13	48	54	377	18.7 % ± 2.2

TABLE II

CROSS-VALIDATED ACCURACIES OF DAS MODELS TRAINED WITH WEIGHTED l_1 REGULARIZATION AGAINST THE FOLLOWING BASELINES: *Static A* (STATIC MODEL WITH IDENTICAL LAYER SIZES), *Static B* (STATIC MODEL WITH EQUIVALENT NUMBER OF PARAMETERS BUT UNIFORM SIZES OF CONVOLUTIONAL LAYERS), AND *Dynamic,* $l_{2,1}$ (DYNAMIC MODEL WITH COMPARABLE NUMBER OF PARAMETERS, TRAINED WITH $l_{2,1}$ REGULARIZATION).

Dataset	**Cross-val. accuracy** $(mean \pm 2SD)$			
	Dynamic, weighted l_1	**Static A**	**Static B**	**Dynamic,** $l_{2,1}$
MNIST	99.4 % ± 0.1	99.3 % ± 0.2	99.3 % ± 0.2	99.2 % ± 0.2
Fashion MNIST	93.3 % ± 0.5	92.4 % ± 0.3	92.8 % ± 0.4	92.1 % ± 0.3
SVHN	93.1 % ± 0.3	91.4 % ± 0.8	92.4 % ± 0.3	92.3 % ± 0.5
CIFAR-10	77.0 % ± 0.7	72.9 % ± 1.3	75.6 % ± 1.1	72.3 % ± 1.4
CIFAR-100	45.1 % ± 1.1	32.4 % ± 1.4	35.4 % ± 1.5	39.5 % ± 1.8
Tiny ImageNet	18.7 % ± 2.2	11.3 % ± 0.8	11.3 % ± 0.7	17.1 % ± 1.4

TABLE III

CROSS-VALIDATED ACCURACIES OF VARIOUS TYPES OF MODELS ON THE CIFAR-100 DATASET.

Type	**Growth**	**Pruning**	**Regularization**	**Fine-Tuning**	**Cross-val. accuracy** $(mean \pm 2SD)$
Dynamic	Yes	Yes	weighted l_1	Yes	45.1 % ± 1.1
Dynamic	No	Yes	weighted l_1	Yes	45.0 % ± 1.0
Static	-	-	weighted l_1	Yes	43.8 % ± 1.5
Dynamic	No	Yes	weighted l_1	No	43.7 % ± 0.8
Static	-	-	weighted l_1	No	43.0 % ± 1.8
Static	-	-	l_1	No	43.8 % ± 1.5
Static	-	-	None	-	32.4 % ± 1.4

showed that final layer sizes are largely independent of the initial layer sizes, as can be seen in Fig. 2.

C. Predictive Performance of Dynamic Auto-Sizing Models

We tested the predictive capabilities of DAS models against "static" models that do not utilize auto-sizing, as well as against DAS models trained with the group sparsity $l_{2,1}$ regularization, which roughly corresponds to the original auto-sizing method. On each dataset, first we trained a convolutional DAS model (utilizing weighted l_1 regularization) in a cross-validation setting, and used the mean resulting hidden layer sizes (measured in the number of filters) as the architecture of a baseline static model, labeled *A*. In order to determine how suitable the layer sizes discovered by DAS are for a static model, we trained another baseline static model (labeled *B*), this time with an equal number of filters in each convolutional layer. The size of the dense hidden layer as well as the total number of parameters was preserved. Finally, we trained a dynamic model utilizing the structured sparsity $l_{2,1}$ regularization for each dataset, with a regularization strength α set so that the resulting model would have comparable number of parameters to the other models. All models were trained for 40 epochs, with the DAS models using static "fine-tuning" for the second half of training. For each model, the results corresponding to the best epoch were recorded. The learning rates were set independently for each model and dataset using grid search.

The cross-validated results can be found in Table II. For each dataset, the model trained using DAS with weighted l_1 regularization significantly surpassed the accuracy of each baseline model, as verified by chi-square tests with significance level of 10^{-5}. The largest differences were present on the most difficult datasets. Notably, the static model B always matched or surpassed the accuracy of static model A, indicating that the layer sizes obtained using DAS are not optimal for use in static models.

V. ABLATION STUDY

In order to explain the observed accuracy benefit of DAS, we performed further tests on the CIFAR-100 dataset. First

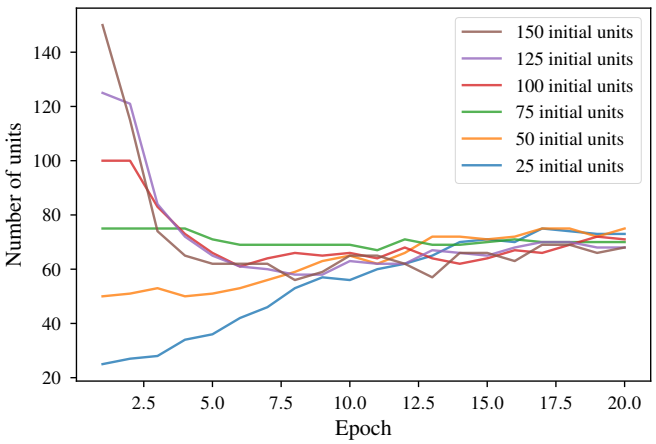

Fig. 2. Evolution of the number of filters in the last convolutional layer of DAS models over the course of training on the CIFAR-100 dataset. All models converge similarly regardless of the initial model size.

we trained a DAS model utilizing weighted l_1 regularization in a cross-validation setting. We then used the mean resulting layer sizes as the architecture for analogous models with deactivated growth of new units, which further differed by the utilized regularization type, whether or not pruning of units was activated, and whether or not the last 20 epochs were run in the static "fine-tuning" setting (with regularization, growth, and pruning deactivated). The learning rates as well as the strength of l_1 regularization (where utilized) were set independently for each model using grid search. The cross-validated results in Table III show that most of the predictive advantage of DAS models comes from the use of weighted l_1 regularization, which alone in this experiment caused a 10.6 % accuracy boost over an unregularized model. The results also show that an accuracy boost may be obtained using a static model with l_1 regularization, whose strength α must however be tuned in order to find the optimal value.

VI. Conclusion

In this work we introduced *dynamic auto-sizing*, a technique for training deep learning models that can dynamically shrink or grow according to the difficulty of the problem domain. We also introduced *weighted l_1 regularization*, a novel structured sparsity regularizer. The experiments show that DAS models regularized with weighted l_1 regularization can surpass the predictive performance of models trained with $l_{2,1}$ regularization as well as traditional approaches. As the size of DAS models grows with increasing difficulty of the problem domain, we hypothesize the potential of the method for adaptation of size to concept drift in on-line or reinforcement learning tasks, or for growth or pruning of models of unsuitable size by manipulation of a single hyperparameter.

References

1. Cahlik, V.: Automatické nastavování velikosti neuronových sítí v omezeném čase [Anytime Learning with Auto-Sizing Neural Networks]. MA thesis, České vysoké učení technické v Praze. Výpočetní a informační centrum. (2022).
2. Madala, H.R.: Inductive learning algorithms for complex systems modeling. CRC press (2019)
3. Kordik, P.: Fully automated knowledge extraction using group of adaptive models evolution. Czech Tech Univ Prague Fac Electr Eng (2006)
4. Murray, K., and Chiang, D.: Auto-sizing neural networks: With applications to n-gram language models. arXiv preprint arXiv:1508.05051 (2015)
5. Alvarez, J.M., and Salzmann, M.: Learning the number of neurons in deep networks. Advances in Neural Information Processing Systems 29 (2016)
6. Zhou, H., Alvarez, J.M., and Porikli, F.: Less is more: Towards compact cnns. In: European conference on computer vision, pp. 662–677 (2016)
7. Murray, K., Kinnison, J., Nguyen, T.Q., Scheirer, W., and Chiang, D.: Auto-sizing the transformer network: Improving speed, efficiency, and performance for low-resource machine translation. arXiv preprint arXiv:1910.06717 (2019)
8. Gordon, A., *et al.*: Morphnet: Fast & simple resource-constrained structure learning of deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1586–1595 (2018)
9. Yuan, X., Savarese, P., and Maire, M.: Growing efficient deep networks by structured continuous sparsification. arXiv preprint arXiv:2007.15353 (2020)
10. LeCun, Y., Denker, J., and Solla, S.: Optimal brain damage. Advances in neural information processing systems 2 (1989)
11. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) 58(1), 267–288 (1996)
12. Han, S., Pool, J., Tran, J., and Dally, W.: Learning both weights and connections for efficient neural network. Advances in neural information processing systems 28 (2015)
13. Han, S., Mao, H., and Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
14. Yuan, M., and Lin, Y.: Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68(1), 49–67 (2006)
15. Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H.: Learning structured sparsity in deep neural networks. Advances in neural information processing systems 29 (2016)
16. Ioffe, S., and Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, pp. 448–456 (2015)
17. Liu, H., Simonyan, K., and Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
18. Tan, M., and Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning, pp. 6105–6114 (2019)
19. Fahlman, S., and Lebiere, C.: The cascade-correlation learning architecture. Advances in neural information processing systems 2 (1989)
20. Frean, M.: The upstart algorithm: A method for constructing and training feedforward neural networks. Neural computation 2(2), 198–209 (1990)
21. Bengio, Y., Roux, N., Vincent, P., Delalleau, O., and Marcotte, P.: Convex neural networks. Advances in neural information processing systems 18 (2005)
22. Stanley, K.O., and Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation 10(2), 99–127 (2002)
23. Stanley, K.O., D'Ambrosio, D.B., and Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks. Artificial life 15(2), 185–212 (2009)
24. Miikkulainen, R., *et al.*: Evolving deep neural networks. In: Artificial intelligence in the age of neural networks and brain computing, pp. 293–312. Elsevier (2019)
25. Cahlik, V., and Surynek, P.: Near Optimal Solving of the (N2-1)-puzzle Using Heuristics Based on Artificial Neural Networks. In: International Joint Conference on Computational Intelligence, pp. 291–312 (2019)
26. Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S.: Self-normalizing neural networks. Advances in neural information processing systems 30 (2017)
27. Netzer, Y., Wang, T., Coates, Bissacco, A., Wu, B., and Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. (2011)
28. Le, Y., and Yang, X.: Tiny imagenet visual recognition challenge. CS 231N 7(7), 3 (2015)