UDC 004.852
https://doi.org/10.31891/csit-2022-3-5

OLEKSANDR GURBYCH, MAKSYM PRYMACHENKO
Lviv Polytechnic National University

METHOD FOR REDUCTIVE PRUNING OF NEURAL NETWORKS AND ITS APPLICATIONS

Trained neural networks usually contain redundant neurons that do not affect or degrade the quality of target identification. The redundancy of models possesses an excessive load on computational resources, leading to high electricity consumption. Further, the deployment and operation of such models in resource-constrained environments such as mobile phones or edge devices are either complicated or impossible. There are various methods for the neural networks pruning - but all of them lead to decreased target identification rates, reducing the business value of the resulting models. Therefore, there is a need to create a method that would combine the removal of neural network weights with increasing the ability of the model to generalize target identification. This work presents such a method. Simultaneously with fast detection and removing redundant weights of fully connected neural networks, the method increases the generalized efficiency of simplified models. The essence of the methodology is as follows: first, small perturbations are made to the target variable; then, two neural networks with the same architecture and initial parameter values are trained: first (control) - on the original data set, second (experiment) - on data with disturbances; then, the weights with the largest relative deviations from the weights of the control model are determined and removed in the trained "experiment" network; the resulting "simplified" set of weights is tested on a separate independent data set; the training, detection, and removal of weights are repeated if the generalized accuracy of target identification increases. In addition, the logic of automatically determining the optimal number of remaining "significant" weights is implemented. The aforementioned features speed up the detection and removal of excess weights, reducing the time and resources required for computations and automating the identification of essential neurons. The method's effectiveness was demonstrated in two applied problems: predicting the quantitative yield of chemical reactions and molecular affinity. The molecular affinity prediction showed an impressive 86.88% decrease in the initial number of weights and a 5.16% decrease in the loss function value. A simplified model for estimating the yield percentage of chemical reactions showed a decrease in the value of the loss function by 3.26% from 70.79% of the initial number of network parameters. The conducted studies prove that the method is applicable for simultaneously reducing the number of weights and increasing the generalization of fully connected neural networks.

Keywords: machine learning, deep neural networks, molecular affinity, chemical reaction yield.

ОЛЕКСАНДР ГУРБИЧ, МАКСИМ ПРИМАЧЕНКО
Національний університет «Львівська політехніка»

МЕТОД РЕДУКЦІЙНОГО СПРОЩЕННЯ НЕЙРОННИХ МЕРЕЖ ТА ЙОГО ЗАСТОСУВАННЯ

Натреновані нейронні мережі зазвичай містять надлишкові нейрони, які не впливають або погіршують якість цільової ідентифікації. Надлишковість моделей призводить до надмірного навантаження обчислювальних потужностей та споживання зайвої електроенергії. Розгортання та експлуатація таких моделей у середовищах з обмеженими ресурсами, таких як мобільні телефони чи крайові пристрої, ускладнюється або унеможливлюється. Існують різноманітні методи спрощення нейронних мереж - але всі вони призводять до зниження показників цільової ідентифікації, знижуючи цінність результуючих моделей для бізнесу. Отож, гостро постає потреба у створенні методу, який би поєднував видалення ваг нейронних мереж із підвищенням здатності моделі до узагальнення цільової ідентифікації. У цій роботі представлено такий метод. Одночасно із швидким виявленням та видаленням надлишкових ваг повнозв'язних нейронних мереж, метод підвищує узагальнену ефективність спрощених моделей. Суть методології полягає у наступному: спочатку у цільову змінну вносяться невеликі збурення; потім тренуються дві нейронні мережі із однаковою архітектурою та початковими величинами параметрів: одна (контроль) - на оригінальному наборі даних, інша (експеримент) - на даних зі збуреннями; далі у натренованій мережі "експеримент" визначаються та видаляються ваги із найбільшими відносними відхиленнями від ваг контрольної моделі; утворений таким чином "спрощений" набір ваг тестується на окремому незалежному наборі даних; тренування, виявлення та видалення ваг повторюється доки генералізована точність цільової ідентифікації зростає. Додатково реалізована логіка автоматичного визначення оптимальної кількості залишкових «значущих» ваг. Згадані функції прискорюють виявлення та усунення зайвих ваг, скорочуючи час і ресурси, необхідні для обчислень, і автоматизуючи ідентифікацію основних нейронів. Ефективність методу продемонстровано на двох прикладних задачах: прогнозування кількісного виходу хімічних реакцій та молекулярної спорідненості. Передбачення молекулярної спорідненості продемонструвало вражаюче зменшення початкової кількості ваг на 86,88% і зменшення значення функції втрат на 5,16%. Спрощена модель для оцінки відсотку виходу хімічних реакцій показала зменшення значення функції втрат на 3,26% із 70,79% від початкової кількості параметрів мережі. Проведені дослідження доводять, що метод є застосовним для одночасного зменшення кількості ваг і підвищення генералізації повнозв'язних нейронних мереж.

Ключові слова: машинне навчання, глибокі нейронні мережі, молекулярна спорідненість, вихід хімічної реакції.

Introduction

Neural networks paved the way for modern machine learning. Most of the key achievements in natural language processing [1], computer vision [2, 3], and other applied tasks were obtained by constructing increasingly complex and deep neural networks (DNN). The most effective models can easily cross the threshold of hundreds of billions of parameters [4, 5]. As a result, the networks are immensely demanding computing resources. Therefore, the DNNs' drawbacks include high infrastructure costs, excessive electricity consumption, and constrained

operationalization on hardware-limited devices [6, 7]. However, trained networks usually contain neurons that decrease the model's generalization performance due to overfitting effects [8]. Reportedly, large networks can have 85% of redundant weights; 92% of them can be removed during fine-tuning of the pre-trained model [9].

Removing excessive weights of neural networks is called "pruning." A typical drawback of pruning methods is the deterioration of target identification due to the removal of parameters. In this paper, we try to address this shortcoming. We propose a method that simultaneously prunes the model and improves its generalization abilities. Additionally, we prove the method's efficiency in two applied tasks. Hence, our contributions are as follows:

- A universal automated ANN optimization method that makes the network lighter, faster, and increases its generalization performance.
- Verification of the effectiveness of the method on two regression targets: molecular affinity [24] and chemical yield [25].
- An open-source library for neural network reduction that is available at https://github.com/ogurbych/ann-reduction

The remainder of the paper is organized as follows: in Section 2 we briefly overview related works; Section 3 describes the data preprocessing and feature engineering approach; Section 4 details the reductive pruning methodology designed in this study; and Section 5 reports the results of application of the method to two regression objectives - molecular affinity and chemical yield.

Related works

Artificial neural network (ANN) pruning is a process of systematically detecting and removing redundant parameters while maintaining trained model performance. One of the first works in the field of neural network pruning belongs to Yann LeCun [8]. In his paper "Optimal brain damage", LeCun introduced the idea that some of the network's parameters are redundant and don't contribute to the output. Typically, neurons are ranked according to how much they contribute to the output. Then the low-ranking neurons are removed from the network, resulting in a smaller and faster network [10, 11, 12, 13]. Usually, the simplified neural network has comparable performance to its origin.

The general pruning strategy starts from training the network to convergence and issuing a score for each structural element in the network. Low-scoring elements are dropped then [6]. Pruning reduces the network's performance, so it has to be fine-tuned to regain the original's accuracy. The pruning and fine-tuning are repeated, reducing the network's size until a convergence limit is met [14].

Most of the works in the field develop minor modifications of the algorithm. For instance, some authors insert excessive parameters into the network to facilitate sparsity and enable scoring the trained network [15]. Others propose periodical [17] or initialization-time [18] pruning. Most of the methods terminate individual weights [13, 6], while others consider pruning groups of parameters, removing entire neurons, filters, or channels [19, 20]. Some works replace fine-tuning of the trained weights before the pruning with rewinding the network to an earlier state [21, 22] or reinitializing the entire network [23].

Nonetheless, any pruning method inflicts a tradeoff between the size and performance of the network, advancing the former while reducing the latter [14]. Thus, 97% of initial performance with 10% of the original neurons was reported for transformer nets [9]. A slight drop in accuracy was observed after a 5x practical reduction in adapted 3D-convolutional filters of a recurrent gesture classifier [16]. A "close to original" accuracy on CIFAR10 was regained after 34%, and 38% of the filters were removed from the convolutional nets [19]. Therefore, it is important to develop a method that simultaneously decreases the number of parameters and improves model generalization capacity.

Data preprocessing and feature engineering

We evaluated the performance of the method against two regression targets: predicting the molecular affinity of ligands to human thrombin [24] and predicting the yield of a chemical reaction [25]. This section details the data preparation and feature engineering procedures for each objective.

Molecular affinity

Molecular affinity is a measure of the strength of chemical association between a ligand, usually a small molecule, and a receptor, usually a large biomolecule. Typical receptors are biopolymers such as enzymes, DNA and RNA, ion channels, etc. Affinity can be expressed numerically as inhibition constant (K_i) - the concentration of the ligand in the solution, which is required to reduce the activity of the target receptor by 50%. K_i reflects the inhibitor's strength: the lower the K_i, the more active the inhibitor, and vice versa.

A dataset of 12,351 entries was created by combining human thrombin ligands from three open sources: BindingDB [26], DUD-E [27], and ChEMBL [28]. Ligands were represented using the SMILES strings [29]. During preprocessing, the SMILES were canonized using RDKit [30] to eliminate ambiguity in the representation of a molecule. Duplicates were discarded. Outliers were removed using the interquartile range (*IQR*) of the molecular weights of the ligands. We removed molecules with molecular weights less than *Q1 - 1.5 * IQR* and greater than *Q3 + 1.5 * IQR*. Since the concentration measurement error increases proportionally to its value, we used a logarithm of

the ligand concentration ($log10K_i$). A histogram showing the distribution of samples by $log10K_i$ values is shown in Figure 1, where K_i is expressed in nanomoles.

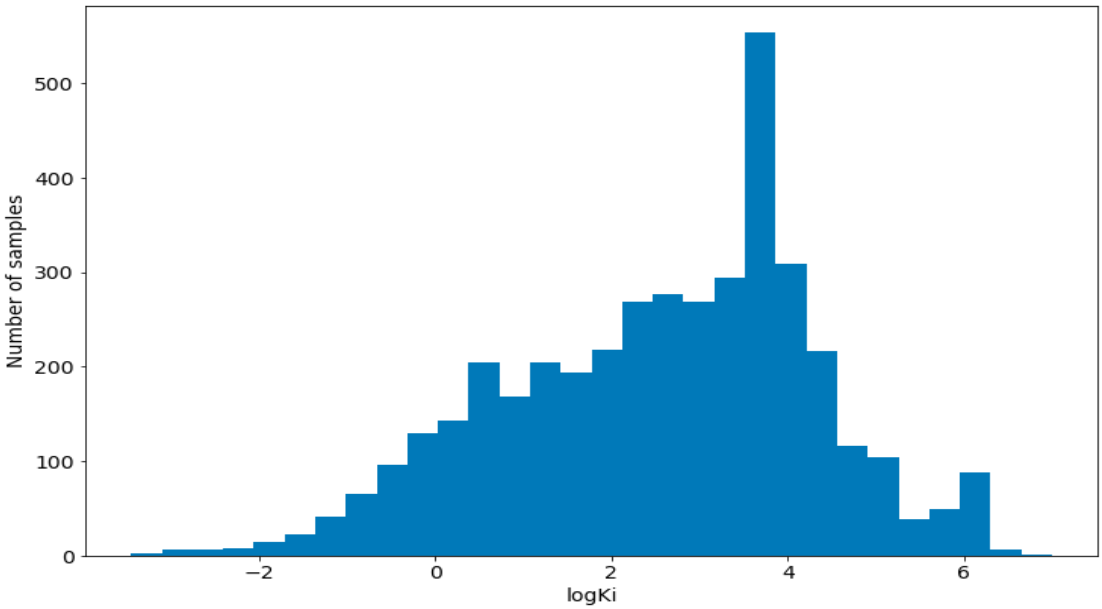

Fig. 1. Distribution of the inhibition constant (K_i) values in a decimal logarithm form.

Ligands, represented by SMILES strings, were encoded as binary ECFP4 fingerprints [31] with a radius of 4 and a length of 2048 bits. We split the dataset into five folds with an 80/20 train-to-test sample ratio for cross-validation.

Chemical reaction yield

The yield of a chemical reaction is the ratio of the number of moles of the product formed to the reagents used. Chemists use this indicator to select highly productive reactions during the design of multistage syntheses. The initial data consisted of 80,014 chemical reactions widely adopted in organic synthesis. Each reaction falls into one of the following broad classes: alkylation, heterocyclization, acylation, sulfanylation, and coupling. Reactions were divided into ten classes based on mechanism, reactant family, products, and reaction conditions. Original data fields included SMILES and InChI codes of the target product; percentage of chemical reaction yield; reaction class identifier; reagents' SMILES and identifiers; catalysts and additives in the form of SMILES strings. We removed the columns with InChI codes and reagent IDs as duplicate information. We also removed reactions where the target product was not obtained (0% yield). The total number of samples after the preparatory steps was 59,291. We used the percentage value of a reaction yield as the target variable. The spread of the studied value was from 0.1% to 100%. The distribution of the target variable is shown in Figure 2.

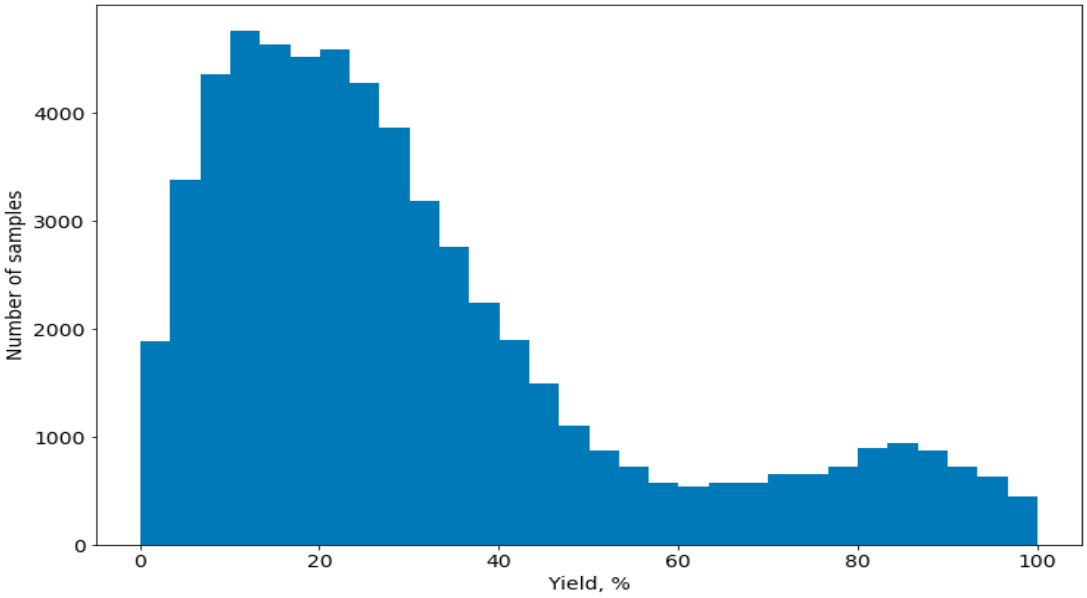

Fig. 2. Distribution of chemical reaction yield values

SMILES of reactants and reaction products were encoded into binary ECFP4 fingerprints, categorical features (reaction class, catalyst, and additives) - as one-hot vectors. As with affinity, this dataset was divided into five 80/20 splits for cross-validation.

Reductive pruning methodology

This section describes our methodology for reductive neural network pruning.

The core concept of the methodology is to iteratively identify and remove ANN's weights that respond the most to small perturbations introduced into the target variable. For this, the ANN was trained in two variants: control model (*CM*) - with default target values and reduction experiment (*RE*) - using the same features but with noise introduced into the target variable. Later, the weights of the trained *CM* and *RE* networks were compared to identify the weights of the *RE* ANN with the largest relative deviations from the corresponding weights of the *CM*. These weights were considered "redundant" and removed. Then, the generalization performance of the *RE* network was evaluated. The pruning was considered successful if the cross-validation loss had reduced. In this case, the next iteration was performed. Otherwise - if the metrics had degraded - the last successful "simplified" set of weights was loaded, and the removal of weights was stopped.

We split each of the datasets into two independent parts (80% and 20%) to avoid overfitting the algorithm to any specific data split. We trained and evaluated the networks using the five-fold cross-validation method on the first part (80%) of the data. Each of the five folds within an 80% split was created with the ratio of training samples to validation samples of 80:20. The iterative reduction of weights was stopped at the first decline of the cross-validation loss (Root Mean Squared Error or RMSE) averaged over 5 folds. Additionally, the generalization performance of the pruned models were evaluated on the second split (20%) of the data - as shown in Figure 3.

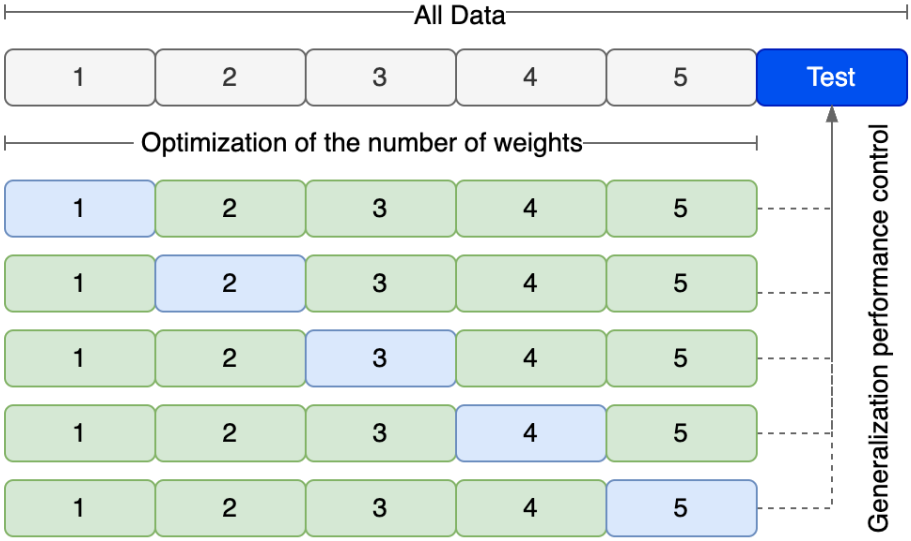

Fig. 3. Scheme for five-fold cross-validation using 80% of the data (in gray). Later, pruned models' generalization performance was evaluated on a 20% holdout test set (in blue).

We trained one *CM* model and two *RE* models on each of the five data partitions. To diversify the disturbances, one of the two *RE* networks was trained on the initial dataset where target values were multiplied by 0.9, while the other one was trained on the original dataset with target values multiplied by 1.1 (see Figure 4). All 15 neural networks (3 experiments with 3 networks on 5 partitions each) were initialized with the same default weights.

The algorithm of the reductive pruning is the following:

(1) Initialization stage

1. *CM* and *RE* models are initialized with the same default weights.
2. A binary mask *M* of the same shape as the models' weights is initialized. The mask is multiplied element-wisely with *CM* and *RE* weights at the training and testing stages. The initial mask $M \in \{1\}$ indicates that all weights are active. Later, identified redundant weights are marked with zeros at the appropriate indexes within the *M* mask.

(2) Epochs loop

3. At the beginning of each new epoch, initialize a vector of two disturbance coefficients $disturb = [d_1, d_2]$. d_1 is chosen from the range [0.9, 1.0) and d_2 - from (1.0, 1.1] so that d_1 and d_2 have equal absolute deviations from 1.0, strictly greater than zero.
4. Multiply the d_1 coefficient with target values of ${RE^1}_{0.9}$, ${RE^2}_{0.9}$, ${RE^3}_{0.9}$, ${RE^4}_{0.9}$, and ${RE^5}_{0.9}$ datasets, i.e., generate 5 datasets with modified target values.

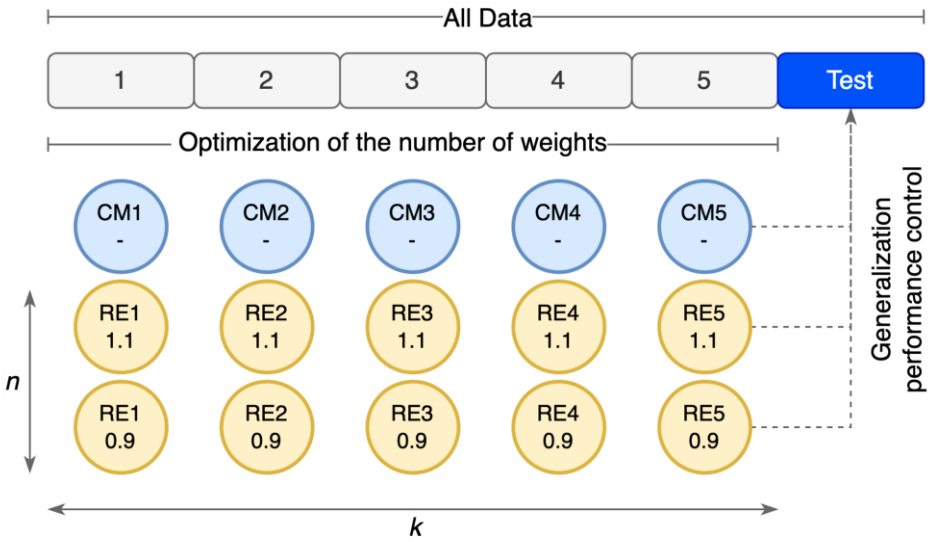

Fig. 4. Control (CM^1...CM^5) and reduction experiment (RE^1…RE^5) models responding to two levels of target variable disturbances (0.9 and 1.1). k - cross-validation fold index, $k \in \{1, 2, 3, 4, 5\}$; n - *RE* group index, $n \in \{1, 2\}$.

5. Multiply the d_2 coefficient with target values of $RE^1_{1.1}$, $RE^2_{1.1}$, $RE^3_{1.1}$, $RE^4_{1.1}$, and $RE^5_{1.1}$ datasets, i.e., generate 5 datasets with modified target values.

6. Leave target values of the *CM* datasets intact for comparison reasons, i.e., hold 5 original datasets.

7. Training and 5-fold cross-validation of the *CM* and *RE* models, resulting in 15 trained models in total. Redundant weights, identified in previous epochs, are turned off by the element-wise multiplication of the *M* mask.

8. Calculate average relative deviations of $RE_{0.9}$ ($n = 1$) and $RE_{1.1}$ ($n = 2$) model weights (trained on the datasets with disturbances) from the *CM* weights (trained on the original dataset) on each of the k cross-validation folds:

$$\delta_k = \frac{1}{n}\sum_{re=1}^{n}\frac{|w_{re} - w_{cm}|}{w_{cm}}, \quad (1)$$

where δ_k- a matrix of average relative deviations of RE^k model weights from the control CM^k weights within a fold k, $k \in \{1, 2, 3, 4, 5\}$; w_{re} - weights of n-th *RE* model within the fold k, $w_{re} \in \{RE^k_{0.9}, RE^k_{1.1}\}$; w_{cm} - weights of the *CM* model within the fold k, $w_{cm} \in \{CM^k\}$.

9. Calculate average relative deviations amongst all k folds:

$$\bar{\delta} = \frac{1}{k}\sum_{i=1}^{k}\delta_k, \quad (2)$$

where δ - a matrix of average relative deviations across all folds;

10. Store current mask M as M_{prev}.

11. Identify the N[1] largest deviations within the matrix δ and update the mask M by zeroing out the values at the appropriate indices. These "zeroed" indices indicate redundant weights that will be "excluded" at all subsequent epochs.

12. Apply the updated mask M to the *CM* model and run its inference using out-of-sample *Test* fold (see the blue partition in Fig. 4).

13. If the generalization performance of the resulting model becomes better, go to step 3 with the last actual N; else, if N is larger than 0.01%, substitute M with M_{prev} and go to step 3 with $N=N/2$; otherwise, interrupt the weights reduction and return M_{prev}.

(3) Finalization

14. Drop all *CM* model weights at the “zeroed” indices in the resulting mask. Save the architecture and the weights of the reduced model.

[1] The initial value of N was 10%. We tested 1%, 5%, 10%, 15%, and 20% to verify that 10% is the optimal speed-performance trade-off for both datasets.

Results and discussion

As already mentioned in Section 3, we evaluated the performance of the reduction algorithm against two regression targets: molecular affinity of ligands to human thrombin [24] and chemical reaction yields [25]. This section details the results for each objective.

Molecular affinity results

We tested the reduction algorithm described in Section 4 for the continuous molecular affinity target ($logK_i$) prediction using a neural network with one hidden layer. The initial network had 2,098,176 weights. We recorded such metrics as R^2, MSE, MAE, Max error, Explained variance as implemented in [sklearn] for each epoch using the out-of-sample *Test* partition (see Fig. 3 and Fig. 4). The evolution of the RMSE loss function and metrics is shown in Fig. 5.

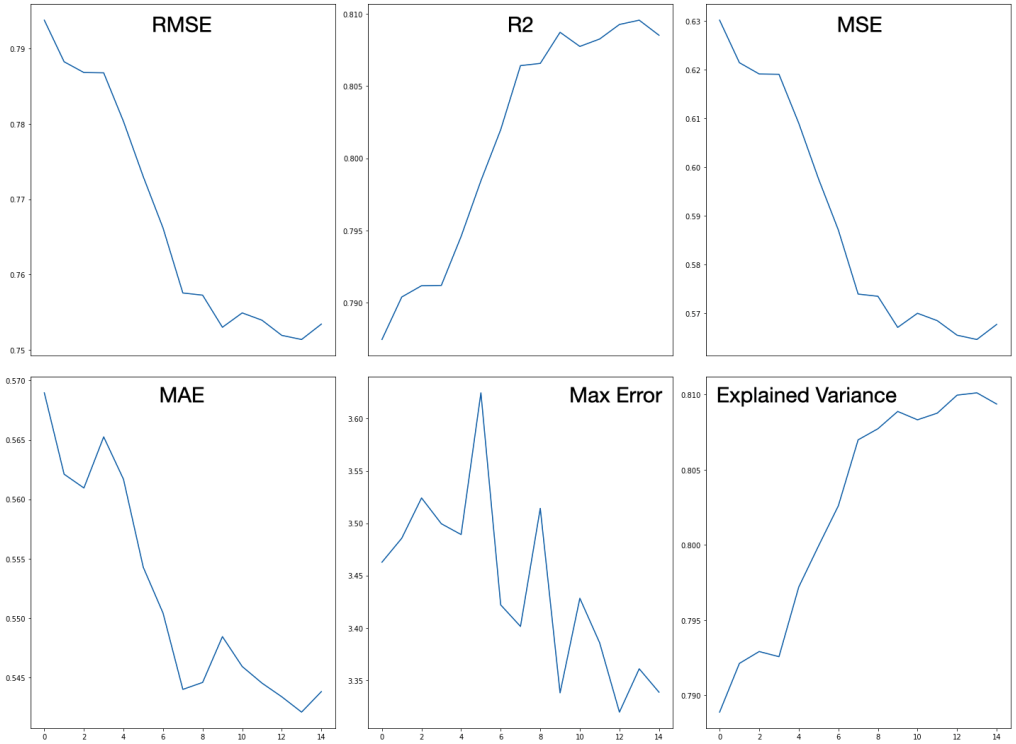

Fig. 5. Evolution of the *CM* model inference metrics on out-of-sample *Test* fold during the weights reduction for the molecular binding affinity prediction.

The reduction was interrupted at the 14th epoch due to the deterioration of the *Test* loss with *N* less than 0.01%. Thus, the final mask was reverted to the 13th epoch mask. Table 1 compares the initial and final models' performance and the number of weights. Easy to see that in this case, the reduction method not only allowed the removal of nearly 90% of the network weights but also improved the loss function by 5.16%. These results are comparable to the most efficient pruning methods mentioned in Section 2 in terms of the amount of the weights removed, but - and this is the most important feature of the method - the simplified network has increased generalization capacity instead of damaged performance.

Table 1

Comparison of the initial and reduced models for the affinity regression

Metric	Initial model	Reduced model	Delta, %
RMSE (loss function)	0.794	0.753	-5.16
R^2	0.787	0.809	+2.80
MSE	0.63	0.568	-9.84
MAE	0.569	0.544	-4.39
Max error	3.463	3.338	-3.61
Explained variance	0.789	0.809	+2.53
Active weights	2,098,176	275,091	**-86.88**

Chemical reaction yield results

We tested the reduction algorithm described in Section 4 for the continuous chemical reaction yield prediction using a neural network with one hidden layer. The initial network had 16,799,744 weights. The metrics and the loss function were the same as in the previous subsection. The evolution of the loss function and metrics for the yield objective is shown in Fig. 6.

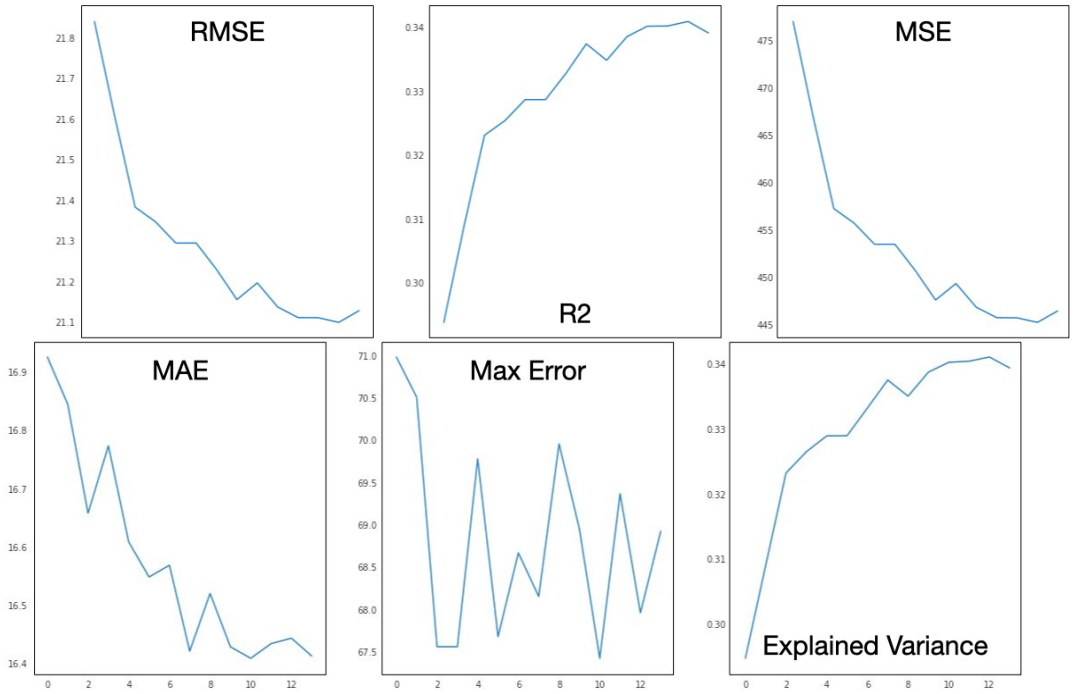

Fig. 5. Evolution of the *CM* model inference metrics on out-of-sample *Test* fold during the weights reduction for the chemical yield prediction

The reduction was interrupted at the 13th epoch due to the deterioration of the *Test* loss with *N* less than 0.01%. Thus, the final mask was reverted to the 12th epoch mask. Table 2 compares the initial and final models' performance and the number of weights.

Table 2

Comparison of the initial and reduced models for the yield regression

Metric	Initial model	Reduced model	Delta, %
RMSE (loss function)	21.83843	21.12756	-3.26
R^2	0.29373	0.33907	+15.43
MSE	477.0122	446.3936	-6.42
MAE	16.92516	16.41234	-3.03
Max error	70.97618	68.92058	-2.9
Explained variance	0.29471	0.33930	+15.3
Active weights	16,799,744	11,891,065	**-29.21**

Easy to observe that the simplified network (~70% of weights remaining) has increased generalization capacity (R^2 +15.4%, RMSE -3.26%) instead of damaged performance as it is typically observed in other pruning methods (see Section 2).

Conclusions

This work introduces a method for simultaneous weight reduction and performance improvement of dense neural networks that develops the reduction concept [32]. We demonstrated the method's effectiveness on two regression objectives. The molecular affinity prediction exhibited an impressive 5,16% loss reduction with only 13,12% of the initial weights. The chemical yield task showed a 3,26% loss reduction with 70,79% of the initial weights.

The method is applicable only to dense neural networks in its current implementation.

The method is comparable to the state-of-the-art pruning methods in terms of the number of removed weights. However, it outperforms other methods by improving the generalization capacity of the resulting models instead of reducing it.

Further development opportunities include comparing the speed and performance of the method with other pruning techniques using benchmark datasets; adaptation of the method to convolutional and recurrent neural networks; implementing a universal statistical computation for the number of muted weights (N) on each epoch; applications of the method for large transformer networks consisting of billions of weights.

Supplementary materials

The implementation and applications of the method have been deposited in the public GitHub repository, https://github.com/ogurbych/ann-reduction

References

1. Costa-Jussà M. R., Cross J., Çelebi O., Elbayad M., Heafield K., et.al. No Language Left Behind: Scaling Human-Centered Machine Translation. *arXiv*. 2022. URL: https://arxiv.org/abs/2207.04672.
2. Wang C.-Y., Bochkovskiy A., Liao M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv*. 2022. DOI: https://doi.org/10.48550/arXiv.2207.02696
3. Xian Y., Lampert C.H., Schiele B., Akata Z. Zero-Shot Learning – A Comprehensive Evaluation of the Good, the Bad, and the Ugly. *arXiv*. 2020. URL: https://arxiv.org/abs/1707.00600v4.
4. Fedus W., Zoph B., Shazeer N. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *Journal of Machine Learning Research*. 2022. Vol. 23 P. 1-40
5. Brown T.B., Mann B., Ryder N., Subbiah M., et.al. Language Models are Few-Shot Learners. *arXiv*. 2020. URL: https://arxiv.org/abs/2005.14165.
6. Han, S., Pool, J., Tran, J., and Dally, W. *Learning both weights and connections for efficient neural networks.* In Proceedings of the NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems. 2015. Vol.1. P. 1135–1143.
7. Sze V., Chen Y.-H., Yang T.-J., Emer J. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *arXiv*. 2017. URL: https://arxiv.org/abs/1703.09039.
8. LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. *Advances in neural information processing systems*. 1990. P. 598–605.
9. Dalvi F., Sajjad H., Durrani N., Belinkov Y.. Analyzing Redundancy in Pretrained Transformer Models. *arXiv*. 2020. URL: https://arxiv.org/abs/2004.04010
10. Janowsky, S. A. Pruning versus clipping in neural networks. *Physical Review A*. 1989. Vol. 39, No 12. P. 6600–6603.
11. Mozer, M. C., and Smolensky, P. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in neural information processing systems*. 1989. P. 107–115.
12. Mozer, M. C., and Smolensky, P. Using Relevance to Reduce Network Size Automatically. *Connection Science*. 1989. Vol. 1, No 1. P. 3–16.
13. Karnin, E. D. A simple procedure for pruning backpropagation trained neural networks. *IEEE transactions on neural networks*. 1990. Vol. 1, No 2. P. 239–242.
14. Blalock, D., Madden, S., and Guttag, J. Sprintz. Time series compression for the internet of things. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*. 2018. Vol. 2, No 3. P. 93.
15. Molchanov, D., Ashukha, A., and Vetrov, D. Variational dropout sparsifies deep neural networks. *Proceedings of the 34th International Conference on Machine Learning*. 2017. Vol. 70. P. 2498–2507.
16. Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient inference. *arXiv*. 2016. URL: https://arxiv.org/abs/1611.06440.
17. Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv.* 2019. URL: https://arxiv.org/abs/1902.09574.
18. Lee, N., Ajanthan, T., Gould, S., and Torr, P. H. S. A Signal Propagation Perspective for Pruning Neural Networks at Initialization. *arXiv*. 2019. URL: http://arxiv.org/abs/1906.06307.
19. Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *arXiv*. 2016. URL: https://arxiv.org/abs/1608.08710.
20. He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. *Proceedings of the IEEE International Conference on Computer Vision*. 2017. P. 1389–1397.
21. Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Stabilizing the lottery ticket hypothesis. *arXiv*. 2019. URL: https://arxiv.org/abs/1903.01611.
22. Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *7th International Conference on Learning Representations, ICLR 2019*, New Orleans, LA, USA, May 6-9, 2019.
23. Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. *7th International Conference on Learning Representations, ICLR 2019*, New Orleans, LA, USA, May 6-9, 2019.
24. Druchok M., Yarish D., Garkot S., Nikolaienko T., Gurbych O. Ensembling machine learning models to boost molecular affinity prediction. *Computational Biology and Chemistry*. 2021. Vol. 93.
25. Yarish D., Garkot S., Grygorenko O.O., Radchenko D.S., Moroz Y.S., Gurbych O. Advancing molecular graphs with descriptors for the prediction of chemical reaction yields. *Journal of Computational Chemistry*. 2022. Vol. 43, No 22.
26. Gilson, M., Liu, T., Baitaluk, M., Nicola, G., Hwang, L., Chong, J., BindingDB in 2015: a public database for medicinal chemistry, computational chemistry and systems pharmacology. *Nucleic Acids Res.* 2016. Vol. 44, P. D1045–D1053.
27. Mysinger, M., Carchia, M., Irwin, J., Shoichet, B. Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking. *J. Med. Chem.* 2012. Vol. 55, P. 6582–6594.
28. Mendez, D., Gaulton, A., Bento, A., Chambers, J., Veij, M., Felix, E., Magariños, M., Mosquera, J., Mutowo-Meullenet, P., Nowotka, M., Gordillo-Maranón, M., Hunter, F., Junco, L., Mugumbate, G., Rodriguez-Lopez, M., Atkinson, F., Bosc, N., Radoux, C., Segura-Cabrera, A., Hersey, A., Leach, A. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Res.* 2019. Vol. 47, P. D930–D940.
29. Weininger D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*. 1988. Vol. 28, No 1. P. 31–6.
30. Greg L., Paolo T., Brian K. et. al. RDKit: Open-Source Cheminformatics and Machine Learning. URL: http://www.rdkit.org

31. Rogers, D., Hahn, M. Extended-connectivity fingerprints. *J. Chem. Inf. Model.* 2010. Vol. 50, No 5. P. 742–754.
32. Matviychuk Y. Improvement of Mathematical Models by Applying the Reduction Principle. *Proceedings of the International Scientific Conference "Information Technologies and Computer Modeling"*, Ivano-Frankivsk, 2019. P. 284–288.

Oleksandr Gurbych **Олександр Гурбич**	Postgraduate Student, Department of Artificial Intelligence Systems, Lviv Polytechnic National University, Ukraine. e-mail: oleksandr.v.hurbych@lpnu.ua https://orcid.org/0000-0002-6821-3390	Аспірант кафедри систем штучного інтелекту, Національний університет "Львівська політехніка", Львів, Україна.
Maksym Prymachenko **Примаченко Максим**	Student, Department of Artificial Intelligence Systems, Lviv Polytechnic National University, Ukraine. e-mail: maksym.prymachenko.knm.2019@lpnu.ua	Студент кафедри систем штучного інтелекту, Національний університет "Львівська політехніка", Львів, Україна