Machine Learning
https://doi.org/10.1007/s10994-023-06438-2

Neural network structure simplification by assessing evolution in node weight magnitude

Ralf Riedel[1] · **Aviv Segev**[1]

Received: 7 June 2023 / Revised: 15 August 2023 / Accepted: 7 October 2023

Abstract
The increasing complexity of artificial intelligence models has given rise to extensive work toward understanding the inner workings of neural networks. Much of that work, however, has focused on manipulating input data feeding the network to assess their effects on network output or pruning model components after the often-extensive time-consuming training. It is shown in this study that model simplification can benefit from investigating the network node, the most fundamental unit of neural networks, during training. Whereas studies on simplification of model structure have mostly required repeated model training, assessing evolving trends in node weights toward model stabilization may circumvent that requirement. Node magnitude stability, defined as the number of epochs where node weights retained their magnitude within a tolerance value, was the central construct in this study. To test evolving trends, a manipulated, a contrived, and two life science data sets were used. Data sets were run on convolutional and deep neural network models. Findings indicated that neural network progress toward stability differed by model, where CNNs tended to add influential nodes early during training. The magnitude stability approach of this study showed superior time efficiencies, which may assist in XAI research toward producing more transparent models and clear outcomes to technical and non-technical audiences.

Keywords Structure simplification · Node patterns · Transparency · Neural networks · Model structure · Pruning

Editors: Vu Nguyen, Dani Yogatama.

✉ Ralf Riedel
rr2023@jagmail.southalabama.edu

Aviv Segev
segev@southalabama.edu

[1] School of Computing, University of South Alabama, Mobile, AL 36688, USA

Published online: 20 December 2023

1 Introduction

The increasing accumulation of data today is leading us into a future where artificial intelligence (AI) is becoming ever more central. Self-driving cars, robots expressing emotions, real-time language translation, virtual teachers, and diagnoses in healthcare are examples where AI excels (Jordan and Mitchell, 2015; Hamet and Tremblay, 2017; Kaul et al., 2020; Martin et al., 2021). The outlook of AI on our everyday life is short of revolutionizing. The benefits brought to society by AI, unfortunately, do not come without hindrances. One consequence of the increasing complexity of AI models, of which neural networks are one of the most common flavors, is lack of understanding of their inner workings, especially on how the information from input data is used in producing model outcomes. One cause of such obscurity is that neural networks are increasingly becoming overparameterized (Denil et al., 2013; Sze et al., 2020; Yeom et al., 2021), which hinders their understanding. Better understanding of how models use information may lead to insights into human learning or uncover new knowledge in science (Raghu and Schmidt, 2020; Makino et al., 2022). Knowledge of the inner workings of neural network models may be advanced if a bottom-up, neuron-focused approach is undertaken, as the ultimate construct determining network outcomes is the neuron.

Understanding the neuronal patterns as a function of input may better assist in defining what neural networks learn and how. Knowledge of the relational link between model input and output may help in constructing simpler models, assisting in fine tuning hyperparameters, or in creating more efficient and safer algorithms (Bhatt et al., 2020; Ivanovs et al., 2021; Quinn et al., 2021), all in the interest of the continued improvements in the understanding of AI systems. Such understanding, however, is contingent on model complexity. Models of simpler structure, with fewer parameters, are more tractable, thus more transparent, than their full-parameter counterparts. The challenge in reducing structural complexity is centered on which components from models to eliminate without compromising predictive power or unduly adding time to model training.

This work offers an approach on how simplification in the structure of neural network models can be accomplished in a time-efficient manner, without incurring the time costs from repeated training during model pruning tasks with different sparsity parameters. Because the time effort associated with model training will only increase as models become more complex, a means to simplify such models while selecting sparsity parameters without requiring model retraining may not only eliminate or diminish time constraints, but also more significantly simplify model structure upon training completion. To that effect, this study proposes the use of a construct to find influential model nodes with the objective of simplifying model structure in a time-effective fashion without compromising model predictive ability. Node magnitude stability was the central construct to realize the objective of this study. Magnitude stability was defined as the number of epochs where node weight values retained their magnitude within a tolerance from that of the stable, trained model, defined in this study as models with predictive accuracy ≥ 0.90.

To achieve the goal of this work, the remainder of this paper is organized as follows. The next section will cover studies attempting to shed light on model simplification through pruning or manipulation of input data. The third section will offer the theory behind the analytical construct of this study. The fourth section will cover this study's experiments and describe a comparison between the method proposed in this study and a popular model simplification approach. The fifth section is reserved for findings. The

last two sections will bring together this study by discussing the relevance of its findings and concluding with suggestions for future research.

2 Related work

Studies toward AI model simplification may consist of removing neural network components, either from the model, such as neurons, or features from the input data. Selecting which components need removal is mostly done heuristically by selecting nodes with low weights due to their small influence on model outcomes (Han et al., 2016; Iandola et al., 2016). Perhaps one of the earliest work in neural network model simplification is the study by Glorfeld (1996), which used back-propagation to eliminate input features from a classifier not contributing to prediction accuracy. In a study removing one neuron at a time from a neural network during model training, Srinivas and Babu (2015) show the redundancy of similar neurons, providing a methodology for their removal based on knowledge distillation (Hinton et al., 2015). Simplification has also been attempted by network compression, with redundant connections identified during training, followed by pruning unimportant connections, and retraining the model on the remaining connections (Han et al., 2016). Focusing on the pooling layers of convolutional neural networks (CNN) has also been used as an approach to model simplification. A simplified convolution operation by learning a pooling operator from image feature channels has reduced model complexity without compromising performance (Sun et al., 2017). Still on input feature, Zou et al. (2018) proposed an iterative method for input feature pruning during training for CNN by eliminating features with low discrimination power in producing model outcomes.

More recent work on network simplification has been added as neural network complexity is increasing, showing the concerns with understanding how AI models work in light of more complex models. In a study attempting to make neural network models more transparent, Singla et al. (2020) generated progressive perturbations to the input image and observed a correspondence in the output, thus synthesizing the input to a selection of most important features to be retained by the final model. Simplification techniques have also been used in AI applications in the medical field. Choudhary et al. (2021) used transfer learning in combination with structured filtered pruning for histopathology classification, showing that the reduced model outperformed the full model version. In attempting to simplify CNN model topology, Hajabdollahi et al. (2019) used hierarchical pruning, gradually removing components, for diabetic retinopathy applications. A lightweight model for lumbar spondylolisthesis diagnosis has been developed for small-scale devices, demonstrating the importance of model simplification for widespread use on various applications (Saravagi et al., 2022). Using a monotonic decreasing parameter to decay neural network filter weights, Cai et al. (2021) was able to achieve a 40% pruning on ResNet-34 without loss in accuracy. In comparing linear, oscillating, and exponential decay rules for weight pruning of a multi-layer perception model, Chouliaras et al. (2022) showed faster convergence when using exponential decay, with an additional benefit of an upward of 95% memory footprint reduction of the final pruned model version compared to the full model. Lastly, input simplification by minimizing image bit size, thus removing redundant features, has been used in simplifying neural network models (Schirrmeister et al., 2022).

The studies above show a significant potential for model simplification, without compromising accuracy, promoting model understanding. Those studies, however, required additional time from post-training processing tasks, adding to the already lengthy model

development task. Investigating how neural network weights evolve from random to stable values may provide indications on how to reduce complexity during training. By investigating node weight patterns according to the methodology below, this work aims to further our understanding of the processes occurring between data input and model outcomes of AI applications.

3 Analytical construct formalization

Analyses in this study were based on the construct of node magnitude stability, herein defined as the number of epochs where node weights within a layer retained their value from that of the stable model within a given tolerance. Node weight magnitude was used as a construct for this study due to weights potentially predicting model accuracy and serving as a base for optimizing model hyperparameters (Eilertsen et al., 2020; Martin et al., 2021; Unterthiner et al., 2021). Additionally, considering how weights stabilize during training may provide further benefits in understanding the operation of neural networks.

The underlying analytical flow for magnitude stability to neural network model pruning (Fig. 1) relies on stable nodes holding their position throughout training. Which nodes become stable may be completely random, determined somewhere during training, but once fixed, a node value should not vary widely, going from significant to unimportant in determining model outcome. Moreover, accretion of magnitude stability must be a function of input data because data properties, such as their size or within-class variance, are what determine model predictive performance. What is less clear is the

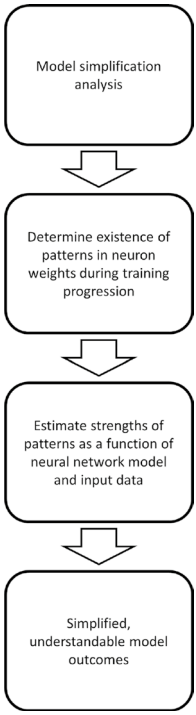

Fig. 1 Analytical flow for the magnitude stability approach to neural network model pruning

influence of model choice on the accretion of magnitude-stable node positions during training. What can also be of value is the relative importance of input data and model in determining magnitude stability. One may not have a choice as to data quality, but choosing models is less of an impediment.

Node magnitude stability is formally defined as

$$\lim_{t\to\infty} |W_{ijk,t} - W_{ijk,t-1}| \leq \epsilon$$

where W = weight, i = row, j = column, k = depth (color channel), t = epoch, and ϵ = tolerance. To determine magnitude stability, all nodes from a model layer at each epoch are sorted by their weight absolute value. The sorted nodes of the last epoch are used as reference. The position of reference nodes is noted and followed backward to the first epoch. The number of consecutive epochs from the reference nodes where node weights in the same position retain a value within a tolerance is defined as the magnitude stability for that node position.

To assess the underlying mechanism that governs model pruning using the construct of magnitude stability may be assisted by a formulation of the process through training time describing accretion of magnitude-stable node positions. It is postulated that the accretion will start slow due to the randomness or quasi-randomness of model weight initial conditions. As training progresses the most influential nodes for model prediction are fixed early, followed by less influential nodes late during training. The process herein proposed for describing node accretion depicts cases ranging from few influential nodes at one extreme to a large number of nodes contributing to model predictive abilities (Fig. 2).

The functional relationship detailed above may best be described by a biological growth curve, commonly used in theoretical population ecology (Kingsland, 1982; Hall, 1988; Kareva and Karev, 2018). Growth in a natural population tends to start slow, accelerate shortly after, when resources abound, and level off toward the end, when those same resources are over-exploited. The same pattern might be observed during the addition of influential nodes in neural networks, where few are added at first but accumulate as training progresses. The biological growth curve adapted to this study takes the form

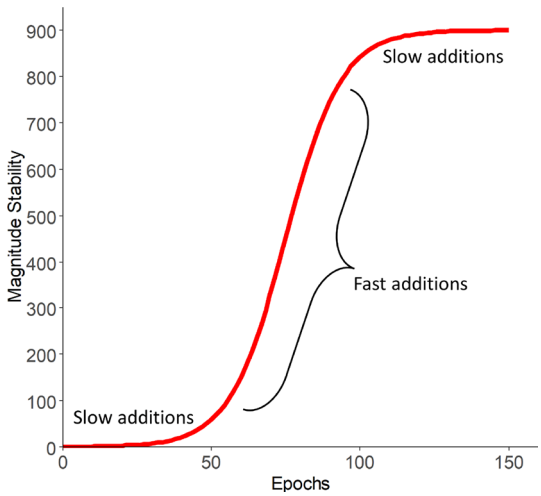

Fig. 2 Theoretical relationship between the mean number of magnitude-stable nodes and training epoch

$$N_t = \frac{K}{1 + \frac{(K-N_0)}{N_0} e^{-rt}}$$

where t is as above, and N_t is the number of magnitude-stable nodes at epoch t, K = maximum number of magnitude-stable nodes, N_0 = number of stable nodes at the first epoch, and r = the rate of magnitude-stable node accretion during model training. This relationship is able to capture both accretion rate extremes above for number of influential nodes determining model predictive power. Moreover, due to its closed form, it may be readily solved after each epoch and used in conjunction with algorithms for model training to estimate the rate of magnitude stability accretion for deciding on possible early stopping of training or pruning less significant nodes.

Early stopping, a compromise approach between training time and model accuracy (Montesinos et al., 2022), has been used in response to the need of improved time-efficiencies in gradient boosting, natural language process, and facial recognition applications by means of feature extraction, manipulations of learning rates, or token skipping Zeng et al. (2022), Ye et al. (2021), Kaya et al. (2019), Guan et al. (2022). Using the approach proposed in this study in conjunction with existing early stopping methods might result in improved training efficiencies due to elimination of non-influential nodes while training.

4 Experiments

4.1 Data sets

Experiments were done to assess randomness of magnitude stability in neural networks. If the process of magnitude-stable node accretion is completely random, independent of models and input data, there is little point in attempting to understand networks in an effort to produce time-effective pruning mechanisms. Toward that effect, a processed, contrived, and two data sets from life science were used in this study's experiments and a popular, unmodified data set for the simplification approach demonstration. The processed data set (PR) was a reduced version of the MNIST handwritten data set. One-hundred images per category were randomly selected from the MNIST data set to keep accuracies from reaching high values early during training. Avoidance of high early accuracy was to enable investigation of patterns in weight value progression during training, as per experiments below. The first life science data set (SC) was of Gulf menhaden (*Brevoortia patronus*) scale black-and-white images, used in estimation of fish age. Classes for this data set were 0–4, the possible observed ages for the target fish. The number of images within each class was 530, of 100 by 100 pixels each. The second life science data set (XR) was of pneumonia x-ray images. This latter data set was of 500 100 by 100 pixel grayscale images, with a category for pneumonia presence and one for absence.

The contrived data set (CT) was generated to have better control and understanding of model output, especially the number of epochs before stabilization. Having better control of model output might enable finer and more accurate study findings and interpretation. The contrived data set comprised of grayscale images with 10 shades of gray, producing 10 classes, which formed the inputs to neural network models. The classes for this data set were generated by sampling from a Gaussian distribution with means of 20, 50, 75, 100, 125, 150, 175, 200, 225, and 240 to mimic pixel depth for grayscale images. The means selected for classes were chosen to ensure an even spread of shades of gray among classes.

The standard deviation was 1,000 across all classes, which allowed model convergence within an adequate number of epochs. Data points that were below zero or above 255, the range for grayscale image shades of gray, were forced to take the minimum or maximum value, respectively, of that range. The number of generated images within each class was 50, each of size 100 by 100 pixels.

A demonstration of the effectiveness of this study's approach to simplification was conducted to show how the findings of this study can be applied to further research in XAI by providing an alternative method to achieve simpler, more interpretable, neural network models. The demonstration was based on the full MNIST data set. The full MNIST data set was used in the interest of research reproducibility and because of its frequent use in research, enabling more ready comparisons with existing research studies in XAI and model simplification.

4.2 Models and runs

To address randomness in magnitude stability, experiments were conducted to investigate patterns in the progression during training of node weights according to AI models, data sets, and epochs. Epochs were used as the time surrogate in assessing training weight progression. The models tested were deep neural networks (DNN) and convolutional neural networks. Models were chosen because both, CNN and DNN share a fully connected layer architectural component, making result comparisons more robust. The DNN model architecture consisted of a flattened input layer, six fully connected layers of 30 x 30 nodes, and the output softmax layer. The CNN model comprised of an input, a fully connected, and an output set of layers. The fully connected and output layers were as in the DNN model. The CNN input layer consisted of a sequence of three convolution layer (3 x 3 kernel) and max pooling layers, followed by another convolution layer, and ending with a flattened layer. The flattened CNN layer served as the input to the fully connected layer. To keep comparisons consistent, a ReLU activation function, Adam optimizer, He-Normal initializers, batch size of 32, and the learning rate of 0.001 were used throughout.

Images in the data sets above were fed one at a time into the input layer of the above models. Images were flattened and padded to keep size consistence prior to model input. Padding was done by addition of extra pixels of value 0, mimicking an empty, black background. Each image pixel corresponded to a node in the input layer to the model tested.

Both, DNN and CNN models were run toward stabilization on the four data sets above in replicates of 50. Each replicate was run for 150 epochs. Model runs that did not converge to at least 0.90% accuracy were re-run for inclusion in analysis. After each epoch, the current model accuracy and the node weights from the layer before that for the softmax activation were extracted for node pattern analysis (focus layer henceforth). The last layer was used for analysis due to its highest stability in representation as data propagates through the network Goodfellow et al. (2009), Montavon et al. (2011). Node analysis consisted of an assessment of whether the progression of node weights from the initial run to stabilization could be predicted as to the stability of their value, potentially identifying the most significant nodes in determining model accuracy early in the training process.

4.3 Data analysis

To investigate magnitude stability, a visualization approach was developed for the focus layer according to the above experiment. A biological growth curve was fit to and

associated parameters shown for the magnitude stability data as an aid for visualizing the shape of the relationship between epoch and magnitude stability. A growth curve was fit for each combination of the models and data sets discussed above for a better understanding of the differences in growth characteristics according to those factors.

Statistical analyses in this work to assess randomness were based on the construct of node magnitude stability, defined and determined as above. To test for neural network node patterns in weight evolution through time, the statistical significance of magnitude stability was assessed. The tolerance for node magnitude stability was taken to be 3%. Only nodes from the focus layer were tested. Stability patterns were assessed with a generalized linear model (GLM) using magnitude stability as the response variable with a quasi-Poisson link function. The predictors for the GLM were the model type, of levels DNN or CNN, and the data sets, of levels CT, PR, SC, and XR. The Tukey's Honest Significance post-hoc GLM was used to infer within-mean differences for statistically significant factors.

An additional test was performed for assessing the relationship between the number of stable nodes and training time. The relationship was tested in two steps. The first step consisted of using a linear regression to estimate the slope of the relationship between epochs, the independent variable surrogate for training time, and the number of stable nodes in the focus layer for each replicate as above. Stable nodes were defined as nodes that were magnitude-stable at a given epoch, i.e., nodes that retained their value within the above tolerance from that of the last epoch. For each epoch, stable nodes were counted and used as the response variable in the regression analysis. The response variable was logarithm-transformed prior to estimating regression parameters to ensure assumption meeting of homoscedasticity. The second step in analyzing node progression was to examine the slopes of the regression analysis obtained in the first step. An analysis of variance (ANOVA) using as factors model and data set, and the regression slopes as the response variable was conducted.

4.4 Comparison of methods

Comparison of model simplification approaches were done contrasting the exponential decay (ED) method (Chouliaras et al., 2022), one of the most popular and efficient (Liang et al., 2021), against the node magnitude stability approach proposed in this study. For magnitude stability, tolerance values of 0.01, 0.03, 0.05 were used in comparisons. The ED method consists of removing neural network nodes by identifying node weights that gradually decay to zero, instead of the more common pruning alternative of removing nodes of weights below a given threshold. A gradual, smooth decay tends to preserve trained information in nodes (Chouliaras et al., 2022).

Node magnitude stability was determined by extracting the weights of the focus layer for each epoch from the trained model. Stability value was calculated by first sorting the weights from each epoch. Second, starting from the last epoch, counting nodes that retained their value within the specified magnitude tolerance was done. Finally, nodes to eliminate were selected from the focus layer. Node elimination was done by choosing a stability value (surrogate for sparsity) within the number of epochs used to train the model. Nodes of magnitude stability smaller than the chosen value were eliminated (assigned zero weight value). Note that choice of stability value subsequent from the first does not require model retraining.

Method comparisons were done using elimination of nodes from the focus layer. For the ED method, nodes were eliminated until a threshold sparsity was achieved. Sparsity

threshold values ranging from zero, no nodes eliminated, to 100% were examined. After each threshold choice, model retraining was required for re-estimation of accuracy. Alternatively, when testing for magnitude stability, their stability values as defined above were used as a surrogate threshold for sparsity. Magnitude stabilities ranging from one to the maximum possible value, the number of epochs, were used. Models for the demonstration of this study were of the same structure and trained using identical parameters as above, except for epochs. Epochs for testing the ED model were only three, the minimum to achieve model convergence. To allow for magnitude stability to reach values for adequate comparisons, 100 epochs were used for testing the method proposed herein.

Five model runs for CNN and five for DNN were conducted for replication. Model predictive accuracy, number of nodes with non-zero weights, and time to pruning were recorded for both methods tested.

5 Results

The visualization approach conducted as above showed that the number of magnitude-stable nodes at the focus layer as a function of epoch tended to fit a biological growth function shown in Fig. 2. Runs for the DNN models tended to show a weak early growth with a rapid growth toward the end. The reverse was observed for CNN models. The timing of the accelerated growth for CNN models was different for the data sets tested, with the PR starting at the earliest. The CNN runs generally followed a biological growth curve covering an entire life of an organism or population, in that it grew fast early and levelled off at later epochs. For the DNN, the growth curve shape was more characteristic of a population unconstrained by resources, where growth tends to be exponential (Hall, 1988). Additionally, DNN runs on the PR, SC, and XR data sets showed unrealistically high values for the theoretical maximum number of magnitude-stable nodes, characteristic of a population with unlimited resources (Fig. 3).

All experimental runs for testing magnitude stability randomness reached stability after 150 epochs, many of which achieved accuracies well above 0.90. Exceptions worth mentioning were models run on the CT data set. For CNN models, an additional 42 runs and for DNN models, an additional 6 runs had to be conducted to reach 50 replicates for that data set.

Magnitude stability averages within model and data set were always highest for CNN models. Also, stability values for CNN models tended to be more consistent, as indicated by the smaller standard deviations. Average magnitude stability for factor model over all data sets was 133.7 $\pm$ 32.20 standard deviations (SD) for level CNN and 78.1 $\pm$ 53.46 SD for level DNN. For factor data, average magnitude stability over all models was 71.8 $\pm$ 61.57 SD for level PR, 121.8 $\pm$ 35.90 SD for level CT, 118.3 $\pm$ 41.76 SD for level SC, and 111.8 $\pm$ 49.59 SD for level XR. The lowest magnitude stability for the CNN model was observed for the PR data set. The other data sets for that model were of similar magnitude stability. Magnitude stabilities for data sets run on the DNN models were more variable, perhaps an indication of sudden additions of magnitude-stable nodes during training. The higher standard deviations for the data sets run on the DNN model are also an indication that magnitude-stable nodes were added at irregular times during training (Table 1).

The GLM results showed factors model ($p \leq 0.01$, F-value = 587,949), and data set ($p \leq 0.01$, F-value = 3,908) to be highly significant predictors, indicating that magnitude stability was dependent on those factors. The model factor had the highest influence on

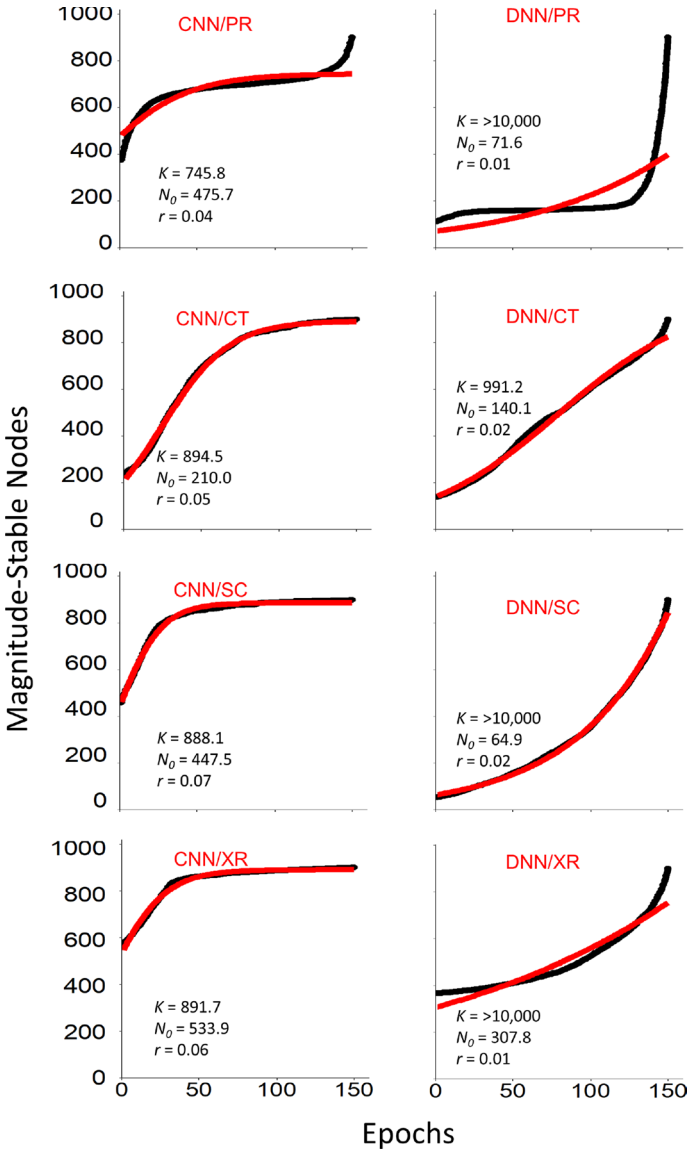

Fig. 3 Relationship between the mean number of mangitude-stable nodes in the focus layer across replicates and epoch; red lines are the fit to the growth function, black lines are the observed data; *PR* processed, *CT* contrived, *SC* scale, *XR* x-ray data sets

the GLM results (as per F-values), indicating it had a stronger effect on the response variable than did the factor data set (Table 2). This is encouraging results because models, unlike data sets, can be adjusted and modified toward successful pruning based on the concept of magnitude stability. Post-hoc analysis also indicated highly significant differences in mean magnitude stability among all the data sets (Table 3).

Table 1 Summary statistics of node magnitude stability for experiments using CNN and DNN models on a contrived (CT), processed (PR), fish scale (SC), and x-ray (XR) data sets, *MRS* mean magnitude stability, *SSD* magnitude stability standard deviation

Model	Data set	MRS	SSD
CNN	CT	130.3	26.76
CNN	PR	118.9	48.78
CNN	SC	142.8	17.79
CNN	XR	142.9	18.39
DNN	CT	113.2	41.42
DNN	PR	24.8	27.81
DNN	SC	93.8	44.44
DNN	XR	80.6	51.37

Table 2 Results for the generalized linear model testing node magnitude stability for experiments using CNN and DNN models on a contrived (CT), processed (PR), fish scale (SC), and x-ray (XR) data set, *DF* degrees of freedom, *SSQ* sums-of-squares

Factors	DF	SSQ	F-value	*p*-value
Data set	3	15,202,416	3,908	< 0.01
Model	1	762,470,774	587,949	< 0.01
Residuals	359,995	466,852,693		

Table 3 Tukey's honest significance difference post-hoc results of node magnitude stability for experiments using CNN and DNN models on contrived (CT), processed (PR), fish scale (SC), and x-ray (XR) data sets

	Difference	Lower	Upper	*p*-adjusted
Data sets				
PR-CT	− 49.9	− 50.4	− 49.5	< 0.01
SC-CT	− 3.5	− 3.9	− 3.0	< 0.01
XR-CT	− 10.0	− 10.5	− 9.5	< 0.01
SC-PR	46.5	46.0	47.0	< 0.01
XR-PR	39.9	39.5	40.4	< 0.01
XR-SC	− 6.5	− 7.0	− 6.1	< 0.01
Models				
DNN-CNN	− 55.6	− 55.9	− 55.4	< 0.01

The results following ANOVA analysis on the slope of the regression of epochs on the logarithm of magnitude-stable node counts also showed high significance for the factors model and data set. As per the F-value, the factor model ($p \leq 0.01$, F-value = 110.7) was more influential in determining the slope than was the factor data set ($p \leq 0.01$, F-value = 15.6), a parallel result from the GLM analysis for the higher influence of models over that of input data (Table 4). The factor model, therefore, was the most important in defining the rate of addition with epoch of magnitude-stable nodes. The benefit of this finding is as above reported for the GLM analysis. Post-hoc analysis for the ANOVA indicated statistical significance for mean magnitude stability differences between the XR and CT, XR and PR, and XR and SC data sets, with only a marginal significance between the SC and PR data set (Table 5).

Comparisons of model simplification approaches consistently showed over one order magnitude larger time efficiencies for magnitude stability than the ED model simplification

Table 4 Results for the AOV model testing addition rate of magnitude-stable nodes for experiments using CNN and DNN models run on a contrived (CT), processed (PR), fish scale (SC), and x-ray (XR) data sets, *DF* degrees of freedom, *SSQ* sums-of-squares

Factors	DF	SSQ	F-value	*p*-value
Data set	3	< 0.01	15.6	< 0.01
Model	1	< 0.01	110.7	< 0.01
Residuals	395	< 0.01		

Table 5 Tukey's honest significance difference post-hoc results of node magnitude stability addition rate for experiments using CNN and DNN models on contrived (CT), processed (PR), fish scale (SC), and x-ray (XR) data sets

	Difference	Lower	Upper	*p*-adjusted
Data sets				
PR-CT	− 0.001	− 0.002	0.001	0.712
SC-CT	0.001	− 0.001	0.003	0.487
XR-CT	− 0.003	− 0.005	− 0.002	< 0.01
SC-PR	0.002	< 0.001	0.003	0.064
XR-PR	− 0.003	− 0.004	− 0.001	< 0.01
XR-SC	− 0.004	− 0.006	− 0.003	< 0.01
Models				
DNN-CNN	0.005	0.004	0.006	< 0.01

method. Both methods showed similar accuracies roughly until 25–30% sparsity values. The magnitude stability method tended to decay in accuracy more gradually. Higher tolerance values for magnitude stability tended to hold higher accuracies for longer (Fig. 4).

6 Discussion

This study shows that magnitude stability is a function of model and data, discarding the contention that neural network learning is a completely random process. Model stabilization tended to occur later for DNN, rather than early during training, as for CNN models. The timing when those weights and their stability was established might have varied among individual runs, but did show a clear pattern according to model and data set. This lends support to the existence of a critical period when training performance is determined (Golatkar et al., 2019). Establishing that critical period may be useful in further advancing network simplification as proposed in this study by combining simplification with early stopping. If the critical period matches or is close to the inflection point along the growth curve, training may be stopped then, resulting in a model with simpler structure without lowering its predictive power.

The GLM and ANOVA results showed that the factor model not only is important, but also that it had the highest influence in determining magnitude stability and at which point during training stability is reached. One explanation may be that CNN and DNN models tend to have distinct predictive powers. The growth of magnitude-stable nodes with epochs indeed followed a biological population growth process. Growth with epochs showed that CNN models added significant nodes in determining stabilization early during training, given the general shape of the relationship between stable nodes and epochs. It is also worth noticing that the early addition of stable nodes implies that more influential nodes for CNN models are observed than for

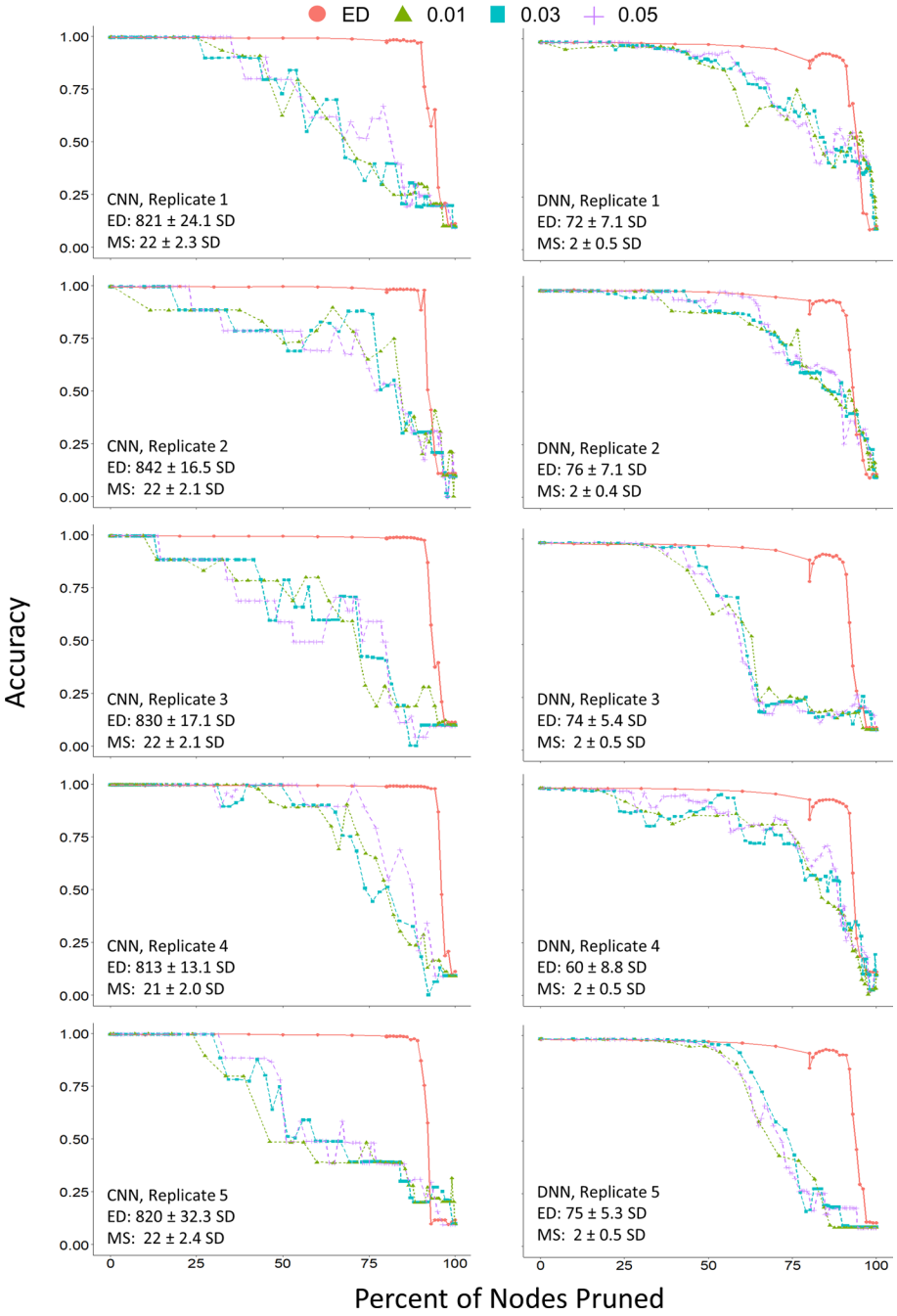

Fig. 4 Comparison of model simplification methods; ED is exponential decay, MS is magnitude stability; values are mean algorithm run times ± standard deviations between sparsity values; 0.01, 0.03, and 0.05 are tolerances used for magnitude stability model simplification runs

DNN models at early epochs, when pruning might be conducted. This finding may be critical in understanding model output. If more nodes are significant, linkages between model weight patterns and patterns of model input data can be simplified and possibly made more apparent when training is stopped at early epochs. This finding, therefore, implies that the final model will comprise of fewer nodes, producing a simpler model more conducive to transparency of outcomes, provided early stopping still produces acceptable model accuracies.

Perhaps the most practical finding from this study is the overwhelming gain in time using magnitude stability when simplifying neural network model structure. The ED method requires model retraining after each selection for sparsity value. Conversely, magnitude stability requires only a count of node weights within a tolerance range prior to running the simplification algorithm. The time gain using magnitude stability may even be an underestimation. The MNIST data set with only three epochs was used for testing the ED method in this study. The choice of epochs was adequate for the data set used, but may be an unrealistic selection for most practical applications. Neural network models in today's applications mostly require many more epochs to reach stability (Aggarwal, 2018). Additionally, models may be trained for several days and even weeks. Under such scenarios, model retraining for simplification is impractical. Using an alternative simplification method that only requires a few matrix operations is a preferred choice, especially with the increasing trends in model deployment to devices of limited processing and storage (Malik et al., 2022; Yuan and Agaian, 2023).

Model simplification using magnitude stability, however, has its limitation. The rapid loss in accuracy values from models simplified using magnitude stability may perhaps be met with discouragement. That loss, however, must be taken lightly. At the initial stages of simplification, losses are negligible, making the magnitude stability method a preferred choice in situations where multiple choices for sparsity used in simplification are required. One such use-case might be in research, where one might want to investigate data representation robustness with varying degrees of simplification. Using magnitude stability in this latter scenario allows a researcher to test several versions of simplified models without the added time burden imposed by alternative methods.

The concept of value stability in this study hinges on the choice of the tolerance value. The larger the tolerance, the more stable a value will be. The choice of that value, however, is less arbitrary than appears on the surface. The time cost of model training, especially when complexity is high, will mostly be large. That cost, however, is low compared to cases where misclassification is high, such as in many healthcare AI applications. Due to training time, popular simplification methods may impede fine adjustments of model sparsity to minimize misclassifications. In case of applications in need of light-weight models, repeatedly adjusting the tolerance when using the magnitude stability concept may be a strategy without incurring prohibitively high time costs.

As suggestions depend on each application, making strong cases for or against the choice of tolerance values might be unwise. Starting with a high tolerance value and observing the effects on model performance and sparsity might be a better approach than initially considering low tolerances, which might over prune models and compromise performance.

7 Conclusions and future work

In general, this work suggested that the progression toward model stabilization using magnitude stability as a metric was not linear. The evidence for an exponential relationship between magnitude stability and epoch was particularly appealing for DNN models, where

magnitude stability more evenly grew toward the end of training, without plateauing afterward. Conversely, for CNN models, the fit to the biological growth curve indicated that addition of stable nodes leveled off after some point during training. Particularly for CNN models, using the findings of this study to lower training time through tasks such as pruning may be attempted with success. Training may stop when reaching the plateau, saving training time, along with simplifying the model by eliminating non-stable nodes. Regardless of findings, however, this study points to the possibility of making training more efficient and models more transparent. Expanding on this study's findings, therefore, may only further add to the potential benefits of this work.

This study's experiments show that randomness does not dominate neural network learning. To capitalize on this, the concept of magnitude stability for model structure simplification was offered. Additional work in line with this study's approach might be investigations on spatial patterns according to magnitude stability values. Examining if a spatial structure exists when viewing a layer as a 2-D or 3-D array may assist in better understanding how the network represents input data. More on further studies, the results of this study apply only to the last layer before the softmax of CNN and DNN models. Assessing other layers might provide a better picture of how a network can be further simplified. Adding additional layers from the fully connected layers of CNN and DNN models might be challenging because of the complex dependencies of weights from different layers. A weight from a given layer is dependent on the weights of all previous layers, making selection on which weights to eliminate from previous layers a challenge. Future work extending the results of this study, such as multilayer model simplification using magnitude stability, may not only offer better, more effective models, but also, possibly more importantly, aid in closing the gap between model complexity and understanding, hopefully matching the speed in model advances with how fast we understand their inner workings.

Author Contributions RR provided the initial draft and analyses; AS provided editing and analytical suggestions.

Funding No funding was provided for this study.

Data availability The data will be available upon publication of this manuscript.

Code availability The code for this work will be available upon publication of this manuscript.

Declarations

Conflict of interest No conflict of interests that are relevant to the content of this manuscript.

Ethics approval Not Applicable.

Consent to participate Not Applicable.

Consent for publication Not Applicable.

References

Aggarwal, C. C. (2018). *Neural networks and deep learning: A textbook* (1st ed.). Springer.

Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M. F., & Eckersley, P. (2020). Explainable machine learning in deployment. In *Proceedings of the 2020*

conference on fairness, accountability, and transparency. FAT* '20, pp. 648–657. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3351095.3375624. Accessed 2022–09–01.

Cai, L., An, Z., Yang, C., & Xu, Y. (2021). Softer pruning, incremental regularization. In *2020 25th international conference on pattern recognition (ICPR)*, pp. 224–230. https://doi.org/10.1109/ICPR48806.2021.9412993

Choudhary, T., Mishra, V., Goswami, A., & Sarangapani, J. (2021). A transfer learning with structured filter pruning approach for improved breast cancer classification on point-of-care devices. *Computers in Biology and Medicine, 134*, 104432. https://doi.org/10.1016/j.compbiomed.2021.104432

Chouliaras, A., Fragkou, E., & Katsaros, D. (2022). Feed forward neural network sparsification with dynamic pruning. In *Proceedings of the 25th pan-hellenic conference on informatics*. PCI '21, pp. 12–17. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3503823.3503826. Accessed 2023–05–21.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M. A., & de Freitas, N. (2013). Predicting parameters in deep learning. In *Advances in neural information processing systems*, vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2013/hash/7fec306d1e665bc9c748b5d2b99a6e97-Abstract.html. Accessed 2022–08–31.

Eilertsen, G., Jönsson, D., Ropinski, T., Unger, J., & Ynnerman, A. (2020). Classifying the classifier: Dissecting the weight space of neural networks. arXiv. arXiv:2002.05688 [cs]. https://doi.org/10.48550/arXiv.2002.05688. Accessed 2023–08–07.

Glorfeld, L. W. (1996). A Methodology for simplification and interpretation of backpropagation-based neural network models. *Expert Systems with Applications, 10*(1), 37–54. https://doi.org/10.1016/0957-4174(95)00032-1

Golatkar, A., Achille, A., & Soatto, S. (2019). Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. *Advances in Neural Information Processing Systems*. https://doi.org/10.48550/arXiv.1905.13277. arXiv. arXiv:1905.13277 [cs, stat].

Goodfellow, I., Lee, H., Le, Q., Saxe, A., & Ng, A. (2009). Measuring Invariances in Deep Networks. In *Advances in neural information processing systems*, vol. 22. Curran Associates, Inc., https://proceedings.neurips.cc/paper/2009/hash/428fca9bc1921c25c5121f9da7815cde-Abstract.html. Accessed 2022–08–30.

Guan, Y., Li, Z., Leng, J., Lin, Z., & Guo, M. (2022). Transkimmer: Transformer Learns to Layer-wise Skim. In *Proceedings of the 60th annual meeting of the association for computational linguistics* (Volume 1: Long Papers), pp. 7275–7286. Association for Computational Linguistics, Dublin, Ireland. https://doi.org/10.18653/v1/2022.acl-long.502. Accessed 2023–05–28.

Hajabdollahi, M., Esfandiarpoor, R., Najarian, K., Karimi, N., Samavi, S., & Reza Soroushmehr, S.M. (2019). Hierarchical pruning for simplification of convolutional neural networks in diabetic retinopathy classification. In *2019 41st annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pp. 970–973. https://doi.org/10.1109/EMBC.2019.8857769 ISSN: 1558-4615.

Hall, C. A. S. (1988). An assessment of several of the historically most influential theoretical models used in ecology and of the data provided in their support. *Ecological Modelling, 43*(1), 5–31. https://doi.org/10.1016/0304-3800(88)90070-1

Hamet, P., & Tremblay, J. (2017). Artificial intelligence in medicine. *Metabolism, 69*, 36–40. https://doi.org/10.1016/j.metabol.2017.01.011

Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv:1510.00149 [cs]. https://doi.org/10.48550/arXiv.1510.00149. Accessed 2023–01–18.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv. arXiv:1503.02531 [cs, stat]. https://doi.org/10.48550/arXiv.1503.02531 Accessed 2023–01–19.

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv:1602.07360 [cs]. https://doi.org/10.48550/arXiv.1602.07360. Accessed 2023-01-18.

Ivanovs, M., Kadikis, R., & Ozols, K. (2021). Perturbation-based methods for explaining deep neural networks: A survey. *Pattern Recognition Letters, 150*, 228–234. https://doi.org/10.1016/j.patrec.2021.06.030

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science, 349*(6245), 255–260. https://doi.org/10.1126/science.aaa8415

Kareva, I., & Karev, G. (2018). from experiment to theory: What can we learn from growth curves? *Bulletin of Mathematical Biology, 80*(1), 151–174. https://doi.org/10.1007/s11538-017-0347-5

Kaul, V., Enslin, S., & Gross, S. A. (2020). History of artificial intelligence in medicine. *Gastrointestinal Endoscopy, 92*(4), 807–812. https://doi.org/10.1016/j.gie.2020.06.040. Accessed 2021-12-21.

Kaya, Y., Hong, S., & Dumitras, T. (2019). Shallow-deep networks: Understanding and mitigating network overthinking. arXiv:1810.07052 [cs, stat]. https://doi.org/10.48550/arXiv.1810.07052. Accessed 2023-05-28.

Kingsland, S. (1982). The refractory model: The logistic curve and the history of population ecology. *The Quarterly Review of Biology, 57*(1), 29–52. https://doi.org/10.1086/412574

Liang, T., Glossner, J., Wang, L., Shi, S., & Zhang, X. (2021). Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing, 461*, 370–403. https://doi.org/10.1016/j.neucom.2021.07.045

Makino, T., Jastrzębski, S., Oleszkiewicz, W., Chacko, C., Ehrenpreis, R., Samreen, N., Chhor, C., Kim, E., Lee, J., Pysarenko, K., Reig, B., Toth, H., Awal, D., Du, L., Kim, A., Park, J., Sodickson, D. K., Heacock, L., Moy, L., … Geras, K. J. (2022). Differences between human and machine perception in medical diagnosis. *Scientific Reports, 12*(1), 6877. https://doi.org/10.1038/s41598-022-10526-z

Malik, S., Tyagi, A. K., & Mahajan, S. (2022). Architecture, generative model, and deep reinforcement learning for IoT applications: Deep learning perspective. In S. Pal, D. De, & R. Buyya (Eds.), *Artificial intelligence-based internet of things systems. Internet of things* (pp. 243–265). Springer. https://doi.org/10.1007/978-3-030-87059-1_9

Martin, S.M., Casey, J. R., & Kane, S. (2021). History of artificial intelligence and personalized learning. In *Serious games in personalized learning*. Routledge.

Martin, C. H., Peng, T. S., & Mahoney, M. W. (2021). Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications, 12*(1), 4122. https://doi.org/10.1038/s41467-021-24025-8

Montavon, G., Braun, M. L., & Mueller, K.-R. (2011). Kernel analysis of deep networks. *Journal of Machine Learning Research, 12*, 19.

Montesinos López, O. A., Montesinos López, A., & Crossa, J. (2022). Fundamentals of artificial neural networks and deep learning. In O. A. Montesinos López, A. Montesinos López, & J. Crossa (Eds.), *Multivariate statistical machine learning methods for genomic prediction* (pp. 379–425). Springer. https://doi.org/10.1007/978-3-030-89010-0_10

Quinn, T. P., Gupta, S., Venkatesh, S., & Le, V. (2021). A field guide to scientific XAI: Transparent and interpretable deep learning for bioinformatics research. arXiv. arXiv:2110.08253 [cs, q-bio]. Accessed 2022–09–01.

Raghu, M., & Schmidt, E. (2020). A survey of deep learning for scientific discovery. arXiv. arXiv:2003.11755 [cs, stat]. https://doi.org/10.48550/2003.11755. Accessed 2023–01–19.

Saravagi, D., Agrawal, S., Saravagi, M., & Rahman, M. H. (2022). Diagnosis of lumbar spondylolisthesis using a pruned CNN model. *Computational and Mathematical Methods in Medicine, 2022*, 2722315. https://doi.org/10.1155/2022/2722315

Schirrmeister, R. T., Liu, R., Hooker, S., & Ball, T. (2022). When less is more: Simplifying inputs aids neural network understanding. arXiv. arXiv:2201.05610 [cs]. https://doi.org/10.48550/arXiv.2201.05610. Accessed 2023–01–18.

Singla, S., Pollack, B., Chen, J., & Batmanghelich, K. (2020). Explanation by Progressive Exaggeration. https://openreview.net/forum?id=H1xFWgrFPS Accessed 2023–01–19.

Srinivas, S., & Babu, R.V. (2015). Data-free parameter pruning for Deep Neural Networks. arXiv:1507.06149 [cs]. https://doi.org/10.48550/arXiv.1507.06149. Accessed 2023-01-19.

Sun, M., Song, Z., Jiang, X., Pan, J., & Pang, Y. (2017). Learning Pooling for Convolutional Neural Network. *Neurocomputing, 224*, 96–104. https://doi.org/10.1016/j.neucom.2016.10.049

Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. (2020). *Efficient processing of deep neural networks*. Morgan & Claypool Publishers.

Unterthiner, T., Keysers, D., Gelly, S., Bousquet, O., & Tolstikhin, I. (2021). Predicting neural network accuracy from weights. arXiv. arXiv:2002.11448 [cs, stat]. https://doi.org/10.48550/arXiv.2002.11448. Accessed 2023–08–07.

Ye, D., Lin, Y., Huang, Y., & Sun, M. (2021). TR-BERT: Dynamic token reduction for accelerating BERT inference. arXiv:2105.11618 [cs]. https://doi.org/10.48550/arXiv.2105.11618. Accessed 2023–05–28.

Yeom, S.-K., Seegerer, P., Lapuschkin, S., Binder, A., Wiedemann, S., Müller, K.-R., & Samek, W. (2021). Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition, 115*, 107899. https://doi.org/10.1016/j.patcog.2021.107899

Yuan, C., & Agaian, S. S. (2023). A comprehensive review of binary neural network. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-023-10464-w

Zeng, J., Zhang, M., & Lin, S.-B. (2022). Fully corrective gradient boosting with squared hinge: Fast learning rates and early stopping. *Neural Networks, 147*, 136–151. https://doi.org/10.1016/j.neunet.2021.12.016

Zou, J., Rui, T., Zhou, Y., Yang, C., & Zhang, S. (2018). Convolutional neural network simplification via feature map pruning. *Computers and Electrical Engineering, 70*, 950–958. https://doi.org/10.1016/j.compeleceng.2018.01.036