Optimizing deep neural networks through hierarchical multiscale parameter tuning

Bogdan Dorneanua, Sushen Zhangb, Vassilios S. Vassiliadisb, Harvey Arellano-Garciaa

aLS Prozess- und Anlagentechnik, Brandenburgische Technische Universität Cottbus-Senftenberg, Cottbus, 03044, Germany

Department of Chemical Engineering and Biotechnology, University of Cambridge, Cambridge, CB3 0AS, United Kingdom

arellano@b-tu.de

Abstract

Deep neural networks (DNNs) are frequently employed for information extraction in big data applications across various domains; however, their application in real-time industrial systems is hindered by constraints such as limited computational, storage capacity, energy availability, and time constraints. This contribution introduces the development of a novel hierarchical multiscale framework for the training of DNNs that incorporates neural sensitivity analysis for the automatic and selective training of neurons evaluated to be the most effective. This alternative training methodology generates local minima that closely match or surpass those achieved by traditional approaches, such as the backpropagation method, utilizing identical starting points for comparative purposes.

**Keywords**: Deep neural networks, Hierarchical multiscale search, Sensitivity analysis.

* 1. Introduction

Deep neural networks (DNNs) are widely employed across various domains, including manufacturing, waste valorization, or Internet of Things (Samek et al., 2021) owing to their distinctive ability to extract information from large datasets. State-of-the-art frameworks for their training have remained stable, involving neuron optimization through backpropagation, often combined with well-known algorithms such as gradient descent (Shrestha & Mahmood, 2019). However, their deployment in real-time applications faces challenges due to limited computational resources. Solutions involve hardware optimization for data flow and memory, as well as software methods like pruning (Choudhary et al., 2020). A powerful technique for reducing the requirements of DNNs, pruning removes redundant parameters and connections while preserving essential features of the system. Optimizing the network partially during each iteration raises the question of determining the optimal layer for the initial step. A suitable criterion for this is neural sensitivity analysis, widely used in understanding and demystifying the black-box nature of DNNs (Zhang et al., 2021).

Initially focused on how an output or objective changes with input perturbations, neural sensitivity analysis has evolved to consider the relative values of weights or inputs as sensitivity measures (Montaño & Palmer, 2003).

This work introduces a novel hierarchical multiscale framework for training of DNNs integrating neural sensitivity analysis to selectively train neurons identified as most effective. Key contributions include the employment of automatic differentiation and the inclusion of first- and second-order information for sensitivity evaluation. Furthermore, a new definition of sensitivity measures is introduced, utilizing a scaling factor.

The paper is structured in the following way: Section 2 introduces the neural sensitivity analysis procedure based on scaling factors, while Section 3 describes the proposed hierarchical multiscale parameter tuning. Results from the application of the procedure on a case study are presented and discussed in Section 4, while Section 5 summarizes the conclusions of this work.

* 1. Scaling factors-based neural sensitivity analysis

The concept of utilizing a scaling factor in the training of a DNN is derived from Conejeros & Vassiliadis (2000) and enables efficient sensitivity ranking of each process step, simplifying the selection and quickly determining a minimal number of critical stages.

For the purpose of the analysis, a neural network is defined as the following process:

 (1)

Where = output from a neuron, = input to the neuron, is the layer index, while is the neuron index within the layer, and is the data point index.

In the proposed framework, the scaling factor plays a crucial role in the identification of the most sensitive pathways for optimizing the overall performance of the DNN. In the formulation of the network, this factor pre-multiplies the output value of a neuron as follows:

 (2)

Where are the scaling factors.

The sensitivity analysis procedure implies the solution of an optimization problem involving the set of parameters :

 (3)

Where the function defines the neuron fitting constraints adopting the scaling factor as:

 (4)

For the solution of the optimization problem defined in Eq. (3), the Lagrangian function is considered as:

 (5)

Where = the set of tuning parameters of the optimization problem, = the set of Lagrange multipliers associated with the constraints .

Finally, the total gradient of the objective function with respect to at the optimal point, () is calculated as follows:

 (6)

Other equations that are considered define the influence of the node at data point :

 (7)

With = the bias term.

Based on the discussion above, the sensitivity of the neuron layer is computed by evaluating the partial derivatives of the objective function with respect to the scaling factor. The mean square of errors (MSE) is utilized as an objective function for the purpose of this analysis. The sensitivity value is determined through automatic differentiation.

* 1. Hierarchical multiscale parameter tuning of DNNs

The concept of the proposed hierarchical multiscale analysis approach (Fig. 1) involves the analysis at the highest level of abstraction, i.e., the entire network structured visualised as an Input/Output (I/O) system. Employing a set bisection strategy, the procedure is moving downward through an expanding binary search tree (Fig. 2), iteratively refining the level of detail considered for optimization, until user-defined criteria are satisfied or the final level of detail is reached. To illustrate the procedure, a DNN with one Input, one Output and seven Hidden layers is considered in Fig. 2.



Figure 1: Hierarchical multiscale training of DNNs

At the first level of division, the network is split into two parts, and the focus is on the part of the tree with the highest sensitivity. This leads to subsequent divisions until the most sensitive layer is found. At each iteration, optimization is carried out for a single layer using the backpropagation method, and the process is reiterated through additional binary tree searches, until the convergence criterion is met.

To address the issue of layers being consistently trapped in a loop during selection, a randomized algorithm is introduced based on a probabilistic random factor. This approach introduces a trade-off between exploration and exploitation, allowing the neurons with lower sensitivity the opportunity for optimization.



Figure 2: Binary tree partitioning of a DNN with sensitivity parameters

* 1. Results and discussion

The proposed hierarchical multiscale search algorithm for the tuning of the DNNs is implemented to a network with 20 and 50 hidden layers of 5 neurons each, respectively. The hyperparameters used for the performance comparison between the different training methods are summarised in Table 1.

Table 1: Hyperparameters used for the training of the DNN

|  |  |
| --- | --- |
| Hyperparameter | Value |
| Trust region bound,  | 0.001 |
| Learning rate | 0.0001 |
| Number of epochs in each iteration | 1,000 |
| Upper limit on number of iterations | 200 |
| Tolerance | 0.001 |
| Number of data points | 10,000 |



Figure 3: Chemical process with 4 inputs and 2 outputs

The dataset utilised for the training comes from a highly nonlinear chemical process (Fig. 3) that includes a first order chemical reaction, with the reaction rate given by an Arrhenius-type equation. In the figure, F represents flowrate, V the reactor volume, T the reactor temperature, C the concentration of A and B, while P is the productivity.

The algorithm is implemented in Python and run on a macOS Big Sur version 11.0.1 (20B29) with a 2.3GHz Quad-Core Intel Core i5 processor and a memory of 8GB 2133 MHz LPDDR3.

Table 2: Comparison results for a network with 20 hidden layers

|  |  |  |  |
| --- | --- | --- | --- |
| Order | Backpropagation | First | Second |
| Training set MSE | 0.93835 | 0.94697 | 0.94631 |
| Testing set MSE | 0.86888 | 0.97121 | 0.93880 |
| Average epochs | 200.00 | 397.26 | 397.26 |
| Total iterations | 97 | 50 | 50 |
| Total CPU Time (s) | 912.7 | 1,022.4 | 1,153.5 |
| Average CPU time (s) | 9.376 | 18.681 | 18.402 |

Table 3: Comparison results for a network with 50 hidden layers

|  |  |  |  |
| --- | --- | --- | --- |
| Order | Backpropagation | First | Second |
| Training set MSE | 1.00001 | 1.00000 | 1.00000 |
| Testing set MSE | 1.04468 | 1.00000 | 1.00000 |
| Average epochs | 200.00 | 208.83 | 416.67 |
| Total iterations | 100 | 100 | 50 |
| Total CPU Time (s) | 2,529.4 | 3,271.9 | 3,746.6 |
| Average CPU time (s) | 25.206 | 24.519 | 48.912 |



Figure 4: Convergence rate of optimization using first- and second-order information for the 50-layer network: Infinity norm of gradients (top); CPU time across iterations (bottom)

Tables 2 and 3 present the results obtained utilizing the end-to-end backpropagation, as well as the hierarchical approach based on first- and second-order information, for the 20 and 50-layer DNNs, respectively. The average values are obtained for each iteration, while the total values referring to all iterations.

From these results it can be observed that in the case of the 20-layer network, the classical approach is the fastest in terms of total CPU time and CPU time per iteration. Moreover, the approach considering second-order information is slightly faster than the first for the proposed multiscale search method. The number of iterations is the same for the two hierarchical approaches, and significantly lower compared to backpropagation. When looking at the results for the 50-layer network, the first-order approach is the fastest in terms of average CPU time, followed by the classical backpropagation method, demonstrating the advantage of the proposed search algorithm for large scale networks. Nonetheless, the second-order method requires a lower number of iterations. Furthermore, the approach utilizing the second-order information seems to converge slower initially compared to the first-order case, and reaching similar speeds of convergence after several iterations. As the convergence speed is different from the 20-layer case, it can be concluded that there is no fixed dominance of the first- over the second-order information approaches.

Finally, from the results in Fig. 4 it can be concluded that there is a polarization in terms of data points, indicating either very slow or immediate convergence of a particular network layer. These findings suggest that the sluggish convergence may be attributed to layers that exert a substantial influence on the overall performance of the model, implying uneven importance among the different layers within the network.

* 1. Conclusions

The novel hierarchical multiscale search algorithm for DNN tuning employs a levelled approach where the network undergoes successive divisions, evaluating layer sensitivity using automatic differentiation. The algorithm selects and optimizes one side of a binary tree at each level until a single layer remains. This process repeats iteratively until a convergence criterion is met, offering efficient optimization with complexity for large DNNs. The innovation lies in selective tuning, enabling rapid convergence and equilibration of sensitivity values, particularly for large-scale networks. Moreover, the algorithm incorporates randomization through a binary selector, enhancing performance by reducing repetition during the optimization. Case studies demonstrate the efficiency of the proposed method, producing solutions comparable to or better than conventional end-to-end backpropagation. Future research will explore leveraging the method for the structure evolution of the DNNs based on sensitivity values.

References

T. Choudhary et al., 2020, A comprehensive survey on model compression and acceleration, Artificial Intelligence Review, vol. 53, pp. 5133-5155

R. Conejeros & V.S. Vassiliadis, 2000, Dynamic biochemical reaction processs analysis and pathway modificaiton predictions, Biotechnol Bioeng, vol. 68, pp. 285-297

J. Montaño & A. Palmer, 2003, Numeric sensitivity analysis applied to feedforward neural networks, Neural Comput Appl, vol. 12, pp. 119-125

W. Samek et al., 2021, Explaining deep neural networks and beyond: A review of methods and applications, Proceedings of the IEEE, vol. 109, no. 3, pp. 247-278

A. Shrestha & A. Mahmood, 2019, Review of deep learning algorithms and architectures, IEEE Access, vol. 7, pp. 53040-53065

T. Zhang et al., 2021, A survey on neural network interpretability, IEEE Trans Emerg Topics Comput Intell, vol. 5, pp. 726-741