Proceedings of the 35th European Safety and Reliability & the 33rd Society for Risk Analysis Europe Conference
Edited by Eirik Bjorheim Abrahamsen, Terje Aven, Frederic Bouder, Roger Flage, Marja Ylönen
 Published by Research Publishing, Singapore.
doi: 10.3850/978-981-94-3281-3_ESREL-SRA-E2025-P0152-cd

A Sobolev trained neural network surrogate with residual weighting scheme for computational mechanics

A. O. M. Kilicsoy, M. A. Valdebenito, M. G. R. Faes

Chair for Reliability Engineering, Fakultät Maschinenbau, TU Dortmund University, Leonhard-Euler-Str. 5, 44227 Dortmund, Germany E-mail: ali.kilicsoy@tu-dortmund.de

J. Liedmann

Previously at Lehrstuhl Baumechanik, Fakultät Architektur und Bauingenieurwesen, TU Dortmund University, August-Schmidt-Str. 8, 44227 Dortmund, Germany
Now at Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, Pfaffenwaldring 27, 70569 Stuttgart, Germany

F.-J. Barthold

Lehrstuhl Baumechanik, Fakultät Architektur und Bauingenieurwesen, TU Dortmund University, August-Schmidt-Str. 8, 44227 Dortmund, Germany

Repeated evaluation of system responses through models become necessary when quantifying uncertainty or optimizing such system. This task can accurately be done through use of complex numerical models such as finite elements. However these models bring with them high computational cost which scales with the complexity of the observed system. Therefore, the use of surrogate models is very practical as they can provide a feasible accuracy for less computational cost. Neural networks represent one type of such surrogate models, whereby a set of data is used to train the neural network model on. The incorporation of sensitivity data, called Sobolev training, can elevate the model performance in accuracy and training time by expanding the loss with additional terms. Each term is pondered with a coefficient weight, which are optimized in parallel training through an adaptive scheme. We use this neural network model in a case study of computational mechanics with regards to its performance.

Keywords: computational mechanics, machine learning, neural network, sobolev training, residual weighting, surrogate model, finite element model, nonlinear mechanics

1. Introduction

Surrogate modeling is a well known topic in engineering problems, where fidelity is exchanged with reduction of computational cost. This is quite crucial in highly complex systems where high fidelity and low processing time are difficult to scale together. Considering simple regression, neural networks represent a surrogate model trained on available data from numerical models such as, for example, finite element models, allowing efficient repeated evaluations of a given system, see Hornik et al. (1989); Grossmann et al. (2024). Incorporation of various conditions and constraints further improves accuracy or training time of these models, such as with Physics informed and Sobolev trained neural networks, see Czarnecki et al. (2017); J. Yu and Karniadakis (2022). Indeed, for the latter two schemes gradient information improves the accuracy of the underlying neural network. For the case of the Sobolev training, the gradients of the neural network are directly compared to the true gradients. To expand further, the information of function and gradient values must be properly weighted for optimal training. These weights can then also have their very own algorithm, ranging from simple to dynamic, to find an optimal ratio of loss terms to improve the training of its respective neural network model. In this paper we compare multiple Sobolev models, more specifically, a base model with no weighting, a model with an alternative weighting method, similar to Liu and Wang (2021) and a model with our new adaptive weighting scheme. Hereby, our method is an adaptive scheme which optimizes

residual weights for each training step for optimal training convergence. We apply the Sobolev models on a nonlinear material of a hook system, whereby training responses and sensitivities are generated via a finite element model using variational sensitivity analysis, see Barthold et al. (2016); Liedmann and Barthold (2020). The response of interest is the stress triaxiality which can give insight into material damage, i.e. void fraction area of the hook. We show that our applied residual weighting optimization leads to a reduction in variance of the training accuracy of the model and an improvement in model accuracy compared to the base Sobolev model as well as the alternative weighting method.

2. Methodology

This section will describe the methodology used in this work. For additional details, see Kilicsoy et al. (2024).

2.1. *Finite element model*

The behavior of a mechanical system is represented through the finite element method (FEM). Moreover, the sensitivity of that behavior with respect to certain parameters associated with the system's geometry (denoted in the following as $\boldsymbol{x}$) is calculated using variational sensitivity analysis. Adhering to principles of the finite element method, the solution to the weak equilibrium equation for nonlinear elasticity in the reference configuration $\mathcal{B}_0$ is

$$\begin{aligned} R(\boldsymbol{u},\boldsymbol{v}) = &\int_{\mathcal{B}_0} \boldsymbol{P}^K(\boldsymbol{u}) : \nabla \boldsymbol{v}\, dV \\ &- \int_{\mathcal{B}_0} \boldsymbol{b}_0 \cdot \boldsymbol{v}\, dV \\ &- \int_{\mathcal{B}_0} \boldsymbol{t}_0 \cdot \boldsymbol{v}\, dA = 0, \end{aligned} \tag{1}$$

where $\boldsymbol{t}_0$ and $\boldsymbol{b}_0$ denote traction and body forces, $\boldsymbol{P}^K$ represents the first Piola-Kirchhoff stress tensor. Additionally, the vector $\boldsymbol{v}$ consists of test functions and $\boldsymbol{u}$ is defined as the vector of primary displacements. The system response is solved iteratively using a Newton-Raphson scheme due to its nonlinear behavior, thereby implying a linearization

$$R(\boldsymbol{u},\boldsymbol{v}) + \delta_u R(\boldsymbol{u},\boldsymbol{v};\boldsymbol{\Delta u}) = 0. \tag{2}$$

As with general FEM, the system is discretized and the weak form of equilibrium is solved numerically for each finite element. Considering a response f, such as the stress triaxiality, which depends on the displacement vector $\boldsymbol{u}$ and design parameters $\boldsymbol{x}$, the FEM allows the response approximation. For the case of design sensitivity analysis, the change of a response function $f(\boldsymbol{u}(\boldsymbol{x}),\boldsymbol{x})$ w.r.t. a change in a chosen model parameter $\boldsymbol{x}$ can be quantified through variational sensitivity analysis, whereby said change is expressed in variational form

$$\delta f = \delta_u f + \delta_x f = \left[\frac{\partial f}{\partial \boldsymbol{u}}\right] \delta \boldsymbol{u} + \left[\frac{\partial f}{\partial \boldsymbol{x}}\right] \delta \boldsymbol{x}. \tag{3}$$

If, for example, our response of interest of the FEM is the ratio of the mean stress σ_{Mean} and equivalent von Mises stress σ_{Mises}, defined as the stress triaxiality T

$$T = \frac{\sigma_{\mathsf{Mean}}}{\sigma_{\mathsf{Mises}}} = \frac{\operatorname{tr}\boldsymbol{\sigma}}{3\sqrt{3J_2}}, \tag{4}$$

with the stress tensor $\boldsymbol{\sigma}$ and the second deviatoric stress invariant J_2. Finally, we derive the discrete sensitivity relation of the stress triaxiality as

$$\delta T = \left[\frac{\partial T}{\partial \sigma_{\mathsf{Mean}}}\frac{\partial \sigma_{\mathsf{Mean}}}{\partial \boldsymbol{\sigma}} + \frac{\partial T}{\partial \sigma_{\mathsf{Mises}}}\frac{\partial \sigma_{\mathsf{Mises}}}{\partial \boldsymbol{\sigma}}\right] \delta \boldsymbol{\sigma}. \tag{5}$$

2.2. *Neural network model*

As general approximators, a neural network is employed in a regression task to approximate the numerical model. With a general feedforward multilayer neural network, the given design parameter $\boldsymbol{x}$ propagates through the neural model layers of various weights, biases and activation functions, culminating in a scalar response. The models approximation is evaluated via a metric defined as the loss $\mathcal{L}$

$$\mathcal{L} = \sum_{i=1}^{n_y} \mathcal{E}_{\mathsf{Response},i} = \sum_{i=1}^{n_y} \frac{1}{2} \cdot (\hat{y}_i - y_i)^2, \tag{6}$$

which is made up of the residual of responses $\mathcal{E}_{\mathsf{Response},i}$, formulated as each mean-squared error (MSE) that is composed of the true responses

$\boldsymbol{y}$, which in this case is the numerical model's responses, and the neural model responses $\hat{\boldsymbol{y}}$ for the number of responses n_y. The optimization of the neural model then follows according to an optimization algorithm, in this case ADAM, see Kingma and Ba (2017), whereby gradients of this loss w.r.t. every model parameter $\boldsymbol{\theta}$ (where $\boldsymbol{\theta}$ contains the weights and biases of the neural network) are computed and applied in a gradient-based method to update said model parameters with the goal of minimizing the loss. This base neural model can be expanded with gradients of the response w.r.t. to the inputs and is the first model type that is trained. It will be used as a baseline for later results to compare with. During general neural network training, backpropagation provides the models gradients w.r.t. to various model parameters, which can then be extended for Sobolev training, whereby gradients w.r.t. inputs are obtained at near zero computational cost. These are subsequently used to compare to the true gradients in the redefined loss. Similarly to the response, additional MSE terms $\mathcal{E}_{\mathsf{Sensitivity},i,j}$ are formulated for each partial derivative w.r.t. inputs to expand the current total loss equation $\mathcal{L}$

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{n_y} \mathcal{E}_{\mathsf{Response},i} + \sum_{i=1}^{n_y}\sum_{j=1}^{n_x} \mathcal{E}_{\mathsf{Sensitivity},i,j} \\ &= \sum_{i=1}^{n_y} \frac{1}{2}\cdot(\hat{y}_i - y_i)^2 \\ &+ \sum_{i=1}^{n_y}\sum_{j=1}^{n_x} \frac{1}{2}\cdot\left(\frac{\partial \hat{y}_i}{\partial x_j} - \frac{\partial y_i}{\partial x_j}\right)^2,\end{aligned} \tag{7}$$

where we now consider inputs $\boldsymbol{x}$ and the number of inputs n_x. With multiple individual loss terms, we can now apply residual weights $\boldsymbol{\lambda}$ to the terms, redefining the loss $\mathcal{L}$

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{n_y} \boldsymbol{\lambda}_{\mathsf{Response},i}\cdot\mathcal{E}_{\mathsf{Response},i} \\ &+ \sum_{i=1}^{n_y}\sum_{j=1}^{n_x} \boldsymbol{\lambda}_{\mathsf{Sensitivity},i,j}\cdot\mathcal{E}_{\mathsf{Sensitivity},i,j}.\end{aligned} \tag{8}$$

Here we apply the residual weights $\boldsymbol{\lambda}_{\mathsf{Response},i}$ and $\boldsymbol{\lambda}_{\mathsf{Sensitivity},i,j}$ to their respective MSE term, where i references the i-th output and j references the j-th input. In general, we have n_y responses and $n_y \cdot n_x$ sensitivities, thus the number of residual weights is $n_\lambda = n_y \cdot (1 + n_x)$. These residual weights, for ease of readability referred to as $\boldsymbol{\lambda}$, are treated as trainable parameters in addition to the trainable neural network model parameters $\boldsymbol{\theta}$. For each residual weight in $\boldsymbol{\lambda}$ an optimization loop is executed, where all run in parallel with the general model parameter optimization loop. For each iteration t we fix parameters $\boldsymbol{\lambda}_t, \boldsymbol{\theta}_t$, and then obtain the next iterations $t+1$ model parameters and residual weights by optimizing a given function for fixed parameters at iteration t. Commonly, for the model parameters $\boldsymbol{\theta}$ we minimize the defined loss function $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})$. The second type of model which we train will make use of the established method of maximizing the loss $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda})$ in order to find the optimal $\boldsymbol{\lambda}$. The first and second model type provide a good backdrop for performance comparison of our weighting method, since we seek to improve Sobolev training, but also expand on established weighting methods. The third type of model, which uses our weighting method, will minimize the cosine distance $1 - \frac{\nabla\mathcal{L}\cdot\nabla\mathcal{L}_i}{|\nabla\mathcal{L}|\cdot|\nabla\mathcal{L}_i|}$ to find the optimal $\boldsymbol{\lambda}$. More specifically, we will observe the cosine distance between the gradient vector of the full loss equation $\nabla\mathcal{L}$ and a gradient vector of a single weighted MSE $\nabla\mathcal{L}_i$ for each λ_i, with the goal of aligning the gradient directions for better training convergence.

3. Experimental Setup

The FEM evaluates a mechanical system, whereby a hook shaped object is fixed to a wall and a force is applied to the hook. The numerical model simulates the two dimensional system for a nonlinear elasticity material behavior of the hook, which is St. Venant-Kirchoff material, in order to obtain the response and sensitivities of the stress triaxiality. The model has two defined design parameters, which affect the geometry of the hook mesh used for evaluation, see Fig. 1, which essentially captures the uncertainty in regards to the two nodes and the subsequent hook geometry. The two parameters shift the coordinates of the two selected nodal points radially at the middle of the hook, with their values limited to $[-1, 1]$. The system is

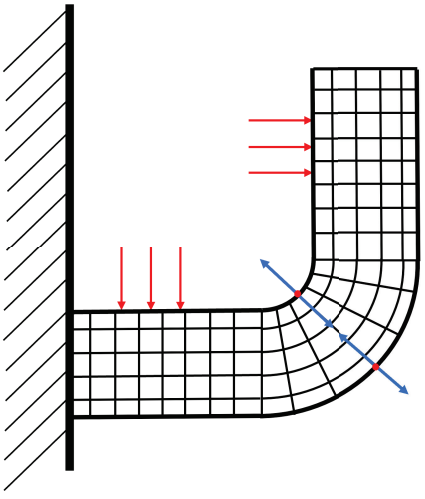

Fig. 1. Drawing of the mechanical system which the FEM evaluates: A hook is fixed to a wall and a force is applied. The design parameters reference two nodes, marked in red on the mesh, whereby their position is varied along the blue arrows to observe responses in regards to geometric uncertainty. The mesh is remade for each new node configuration.

simulated with regards to its stress state, which is the plane stress state, repeatedly in order to provide the necessary training data for the neural network models. There are a total of 625 evaluations by the FEM, which are split into 320 training points and 305 validation points for the neural models. The data points are computated per FEM within the defined design parameter range with a uniform and linear step size. The neural models are all trained on the same dataset, however the neural models are approximating only a single mesh node's triaxiality for simplicity. The neural models are programmed with the MATLAB deep learning toolbox within MATLAB R2023b. Their architecture follows a small and simple design with the layers defined as $[2, 10, 5, 5, 3, 1]$, with the first and last values signifying the input and output layer size, see Fig. 2. Additionally, the training occurs through the ADAM algorithm using standard parameter values, minibatches of size 64, starting learnrate 10^{-3} and 1000 epochs. While the training occurs with focus on the loss, the results are visualized through the relative l_2 error based on the validation data points, as comparison of the loss between models is not indicative of performance due to the residual weights scaling the losses. Furthermore, the results represent those of 100 trained neural models of each type, in order to represent the variability in accuracy of trained models given random factors during training, i.e. parameter initialization, batch order and more.

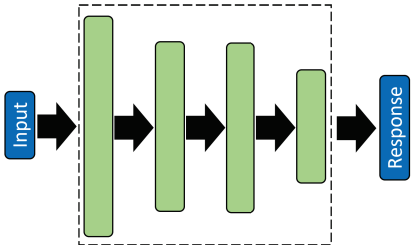

Fig. 2. Sketch of the used neural network architecture: In order, the hidden layers consist of 10, 5, 5 and 3 neurons in the shape of a feedforward multilayer neural model.

4. Discussion

With all three model types trained on the same data, we select the relative l_2 error of the best performing iteration across all 100 trained models. We then evaluate various data metrics and produce their boxplot for comparison. These boxplots can be seen in Fig. 3. We can observe that both

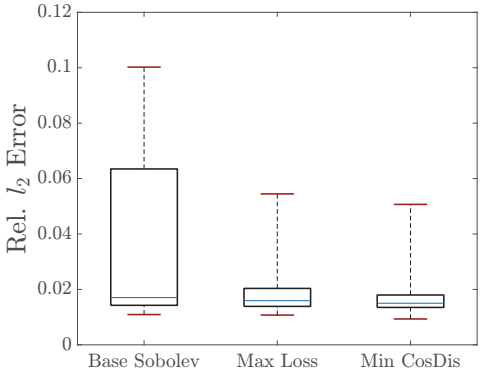

Fig. 3. Boxplots of the best rel. l_2 error across 100 trained models for the three model types: Base Sobolev without residual weights, Sobolev model with residual weights optimized for Loss maximization, Sobolev model with residual weights optimized for Cosine Distance minimization

weighted Sobolev models outperform the base Sobolev model. Notably, the median of the rel. l_2 error is smaller when we use residual weights, but further the model accuracy generally populates a lower bound meaning in brief, we have considerably less variance from training a neural network. While the base Sobolev median is not far off from the weighted model types, of the 100 trained models a subset seems to be stuck in an upper range of accuracy. This behavior can similarly be seen with the weighted model types, where the upper whiskers are quite large in contrast to the lower whiskers. Additionally, when comparing the two weighting methods, we can see the median of the cosine distance method to be better. Likewise, this slight performance edge for residual weight optimization by cosine distance minimization versus loss maximization extends to the general interval which the rel. l_2 error falls to.

5. Conclusion

With this work we have applied neural networks as surrogate models for a finite element model, where the numerical model simulates nonlinear elasticity material behavior. We make use of gradients w.r.t. design parameters in order to expand the neural network training with Sobolev training. From here we successfully introduce residual weights and a first-order optimization target to dynamically optimize these during training in order to improve convergence. Subsequently we show that general residual weighting improves gradient-enhanced training, with our weighting method outperforming the shown alternatives on average in accuracy and in general training in robustness. We observe a reduction of variance in the accuracy of the neural network training with a bias to lower bounds, with a reduced relative l_2 error median across multiple trained models. The results prove that there exists an optimal ratio for the defined loss of a neural network, whereby each term is weighted by its own residual weight. Moreover, a dynamical optimization of these residual weights, that takes each training step into consideration, improves training convergence. However, the optimal ratio is highly dependent on multiple factors, such as the selected training data and the current state of training, which is why our adaptive optimization scheme for the residual weights was chosen. While the residual weights tend to change similarly across the various trained models, their range of behavior is very large, which can make fixed residual weight tuning tedious. The gradient-based optimization of the residual weights via the defined cosine distance introduces a parallel computation to the main model training, that has a high computational cost. Furthermore, our weighting scheme only considers the magnitudes of gradient vectors for the alignment, limiting its ability to choose the optimal ratio of gradient vectors. The cosine distance could be altered to include a more finely comparison of gradient vectors by replacing each residual weight λ_i with a vector of residual weights $\boldsymbol{\lambda_{ij}}$ to control each element of the gradient vectors individually, which are the partial derivatives of the respective loss w.r.t. the model parameters. This expansion of residual weights would however amplify the already high computational cost. Therefore, a key issue to tackle in future research is the optimization of the method for lower computational cost. In addition, we have applied our weighting method to a small dimensioned case study of nonlinear mechanics. Further research should focus on greater neural network architectures in combination with benchmark datasets, for quick and easy comparison with contemporary methods of residual weighting. This should then ideally include a comparison to other contemporary state of the art methods not considered in this paper, such as in Liu et al. (2025).

Acknowledgement

Financial support from the German Research Foundation (DFG) via SFB/TRR 188 (project number 278868966), subproject C06, is gratefully acknowledged.

References

Barthold, F.-J., N. Gerzen, W. Kijanski, and D. Materna (2016). *Efficient Variational Design Sensitivity Analysis*, Volume 40 of *Computational Methods in Applied Sciences*. Switzerland: Springer International Publishing.

Czarnecki, W. M., S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu (2017). Sobolev train-

ing for neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 30. Curran Associates, Inc.

Grossmann, T. G., U. J. Komorowska, J. Latz, and C.-B. Schönlieb (2024, 05). Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics 89*(1), 143–174.

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural networks 2*(5), 359–366.

J. Yu, L. Lu, X. M. and G. Karniadakis (2022). Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering 393*, 114823.

Kilicsoy, A. O. M., J. Liedmann, M. A. Valdebenito, F.-J. Barthold, and M. G. R. Faes (2024). Sobolev neural network with residual weighting as a surrogate in linear and non-linear mechanics. *IEEE Access 12*, 137144–137161.

Kingma, D. P. and J. Ba (2017). Adam: A method for stochastic optimization.

Liedmann, J. and F.-J. Barthold (2020, 04). Variational sensitivity analysis of elastoplastic structures applied to optimal shape of specimens. *Structural and Multidisciplinary Optimization 61*(6), 2237–2251.

Liu, D. and Y. Wang (2021). A dual-dimer method for training physics-constrained neural networks with minimax architecture. *Neural Networks 136*, 112–125.

Liu, Q., M. Chu, and N. Thuerey (2025). Config: Towards conflict-free training of physics informed neural networks.