Forschung im Ingenieurwesen (2024)88:11
https://doi.org/10.1007/s10010-024-00730-w

ORIGINALARBEITEN/ORIGINALS

Development of a workflow to build optimal machine learning models for stress concentration factor regression

Paul Tunsch[1] · **Nils Becker**[1] · **Berthold Schlecht**[1]

Received: 13 November 2023 / Accepted: 28 February 2024

Abstract

In the design of shafts for drivetrains, it is important to have precise knowledge of the effective stress in critical notches. In nominal stress approaches, stress concentration factors are used to estimate the stress in the notch based on geometric properties of the shaft. They can be calculated using numerical methods like finite element method, which can be time consuming. Analytical equations have been developed for simple geometries, like shaft shoulders and round grooves, they are less accurate but much faster than numerical solutions. In this paper, machine learning is used to combine the advantages of both solutions. A process chain to develop models for the calculation of stress concentration factors is presented. It consists of methods to process data, creation and training of regression models and evaluation of the results. This toolbox allows different regression models to be used for different tasks without the need for major changes to the source code. The process is illustrated for shaft shoulders under tension and compression, bending and torsion. The resulting model is capable of calculating stress concentration factors with better accuracy than common analytical approaches while having comparable computation time.

Entwicklung einer Prozesskette zur Erstellung optimaler Machine-Learning-Modelle für die Regression von Formzahlen

Zusammenfassung

Bei der Auslegung von Wellen für Antriebsstränge ist es von großer Bedeutung, die effektive Spannung in kritischen Kerben genau zu kennen. Bei Nennspannungsansätzen werden Formzahlen verwendet, um die Spannung in der Kerbe auf der Grundlage der geometrischen Eigenschaften der Welle abzuschätzen. Sie können mit numerischen Verfahren wie der Finite-Elemente-Methode berechnet werden, was sehr zeitaufwändig sein kann. Für einfache Geometrien wie Absätze und Rundnuten wurden analytische Gleichungen entwickelt, die zwar weniger genau, aber wesentlich schneller als numerische Lösungen sind. In dieser Arbeit wird maschinelles Lernen eingesetzt, um die Vorteile beider Lösungen zu kombinieren. Es wird eine Prozesskette zur Entwicklung von Modellen für die Berechnung von Formzahlen vorgestellt. Sie besteht aus Verfahren zur Datenverarbeitung, der Erstellung und dem Training von Regressionsmodellen und der Auswertung der Ergebnisse. Mit dieser Toolbox können verschiedene Regressionsmodelle für unterschiedliche Aufgaben verwendet werden, ohne dass größere Änderungen am Quellcode notwendig sind. Der Prozess wird für Wellenabsätze unter Zug/Druck, Biegung und Torsion veranschaulicht. Das resultierende Modell ist in der Lage, Formzahlen mit höherer Genauigkeit zu berechnen als herkömmliche analytische Ansätze bei vergleichbarer Rechenzeit.

The authors contributed equally to this work.

✉ Paul Tunsch
paul.tunsch@tu-dresden.de

Nils Becker
nils.becker@tu-dresden.de

Berthold Schlecht
berthold.schlecht@tu-dresden.de

[1] Chair of Machine Elements, TUD Dresden University of Technology, 01062 Dresden, Germany

Published online: 03 April 2024

1 Introduction

When designing shafts for drivetrains, it is of great importance to be able to correctly assess the influence of notches on the strength. Notches can be, for example, shaft shoulders or round grooves. At these locations, there is an increase in stress compared to the unnotched component. For component design according to nominal stress approaches, this stress concentration is expressed by a stress concentration factor K_t, which is defined as follows [1]:

$$K_t = \frac{\sigma_{max}}{\sigma_n} \quad \text{or} \quad K_t = \frac{\tau_{max}}{\tau_n}. \tag{1}$$

It indicates the ratio of the maximum stress at the notched cross-section to the nominal stress at the unnotched cross-section. Its value depends on the geometry of the component and the load type. There are several methods to determine the stress concentration factor.

Approximate equations for simple geometries like shaft shoulders are given by the calculation specifications of DIN 743 [2]. These are based on the fundamental work on notch stress theory by Neuber, who gives the stress concentration factors for real notches by averaging the theoretically determined stress concentration factors for the limiting cases of an "infinitely shallow" and an "infinitely deep" notch. Further work by Petersen and Peterson extended and simplified Neuber's equation and adjusted the constants to empirical data [1]. This process resulted in Eq. (2) with the coefficients given in Table 1. The meaning of the symbols D, d, t and r is given in Fig. 1.

$$K_t = 1 + \frac{1}{\sqrt{A \cdot \frac{r}{t} + 2 \cdot B \cdot \frac{r}{d} \cdot \left(1 + 2 \cdot \frac{r}{d}\right)^2 + C \cdot \left(\frac{r}{t}\right)^z \cdot \frac{d}{D}}} \tag{2}$$

Alternatively, the stress concentration factor can be determined numerically using the finite element method (FEM) [3, 4]. In this case, the geometry to be investigated is simulated in a calculation program. By applying an external load, the maximum stress at the notch can be found. Using Eq. (1), the stress concentration factor is obtained.

Table 1 Coefficients for the determination of the stress concentration factor for shaft shoulders according to [2]

Load type	Tension/compression	Bending	Torsion
A	0.62	0.62	3.4
B	3.5	5.8	19
C	–	0.2	1
z	–	3	2

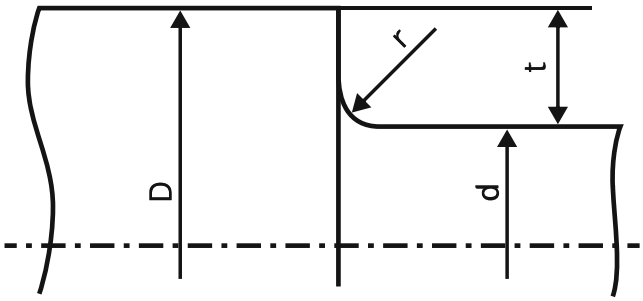

Fig. 1 Dimensions at the shaft shoulder

The application of the approximate equations can be done very quickly and with low resource requirements. However, the accuracy of this method is limited. In contrast, the determination of the stress concentration factors by means of the FEM can be very accurate, depending on the mesh quality. This is offset by the effort required to create the model and the significantly longer calculation time.

A determination method that combines the accuracy of FEM simulations with the speed of applying approximate equations is desirable. To some extent, this has been investigated by Wendler [5] who developed approximate equations for selected spline shafts based on extensive FEM studies with traditional regression methods. However, they are of limited precision with an error of up to 10%. In the recent past, machine learning (ML) methods have proven to be a promising technique for precisely representing complex relationships [6, 7]. In the present work, therefore, their suitability for the regression of stress concentration factors will be investigated. From the results of a parameter variation study performed by FEM, a regression analysis is to be carried out using machine learning methods. Thereby, the accuracy advantages of the FEM shall be combined with the speed of the application of approximate equations.

2 Basics of machine learning

Supervised learning constitutes one category of machine learning methods and can be used for regression. In supervised learning, a function f of the type $\hat{y} = f(\vec{x})$ is optimized in such a way that the deviation between the predictions $\hat{y}$ and the true labels y becomes minimal. [8] For this purpose, the prediction of the model is compared with the actual value for each data set from the training data. Depending on the deviation, the model parameters are adjusted so that the prediction error is minimized. This process is repeated until a termination criterion is reached. The termination criterion can be a prediction error limit or a maximum number of iterations. [9] In this particular case, $\vec{x}$ is the geometric quantities of the notch and $\vec{y}$ is the

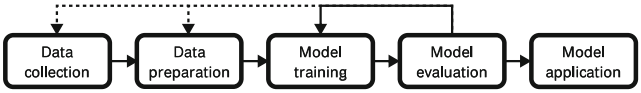

Fig. 2 Machine learning Workflow

stress concentration factors for the three load cases tension/compression, bending and torsion. After training, the model should be able to correctly estimate the stress concentration factors even for geometries that were not part of the training data.

The machine learning process can vary in detail depending on the application, but generally proceeds as depicted in Fig. 2. An extensive overview of this process can be found in [10].

In this work, the holdout approach and cross-validation are used to assess the performance of the trained models. This technique is also employed to optimise the adjustment options of the models in a process called hyperparameter tuning. These procedures are described in more detail in [11].

The function of artificial neural networks (ANN) as described in [12–14] is based on the structure of the human brain, where a large number of nerve cells (neurons) are interconnected. This structure enables information to be passed on and get processed. Similarly, an ANN is composed of several layers, which are called input layer, hidden layer or output layer according to their position. Each layer consists of several neurons that are connected to those of the adjacent layers, see Fig. 3.

These neurons each accept a number of input values x_i with $i = 1...n$, where n is the number of inputs. A certain weighting factor w_i is assigned to each input. In the neuron, each input is multiplied by the associated weighting factor and the result is summed up. Additionally, a bias input value w_0 is added. The neuron applies an activation function φ to this network input. [15]

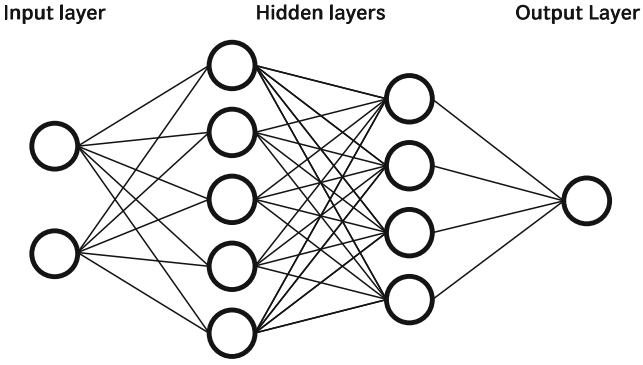

Fig. 3 Structure of an ANN

The output of the artificial neuron is thus calculated as follows:

$$\widehat{y} = \varphi\left(\sum_{i=1}^{n} x_i \cdot w_i + w_0\right). \tag{3}$$

φ is usually a nonlinear function, for example the sigmoid function [16].

Depending on the position of the neuron in the ANN, $\widehat{y}$ could either be the model's output or one input to the next layer of neurons. During the training process, the weights of the neurons are repeatedly adjusted according to the magnitude and direction of the prediction error using a method called backpropagation [14]. The number of layers and neurons, the learning rate, the activation function, and the number of epochs are hyperparameters of an ANN.

3 Model development process

To implement the workflow described in Sect. 2, an automated process chain for creating and evaluating ML models using Python is developed. The objective is to automate the entire procedure of model creation and assessment to the extent that it runs without further intervention and can be applied to new geometries without major changes. This also includes the process of model optimization through hyperparameter tuning. The goal is to be able to examine a wide variety of model architectures without making significant changes to the source code.

For this purpose, widely used software libraries are utilized. To prepare the data, perform the splits and evaluate the models, scikit-learn [17] is used. The ANNs are created using Tensorflow with the Keras interface [18, 19].

The process chain is divided into three Python scripts, as shown in Fig. 4. The main script is the central interface that initializes the process chain. It defines the input and output directories and creates a specific folder structure to manage the extensive data that accumulates in the model creation process. The experiment's main folder stores general information, such as data set analyses, log files of the training process, and graphical comparisons of the model results. The training and testing data sets are stored in a subfolder for reproducible evaluation. Each model architecture to be tested has a separate subfolder that contains the fully trained and stored model, as well as diagrams and data for assessing the model quality.

The training data are collected in a CSV file. Through the main script, a visualization of the distributions and relationships in the dataset is performed using the pairplot function of the seaborn python library [20]. In the next step, the data is split into a training and test dataset. For supervised learn-

ing, the data is then divided into features (input variables) and labels (output variables).

This is followed by the actual creation of the models. For this purpose, the main script calls functions implemented by another Python file (the model script). These are used to create different model architectures with their respective hyperparameters. The ANNs are created using Keras Functional Application Programming Interface (API). Input layer size is automatically determined based on input data structure. Size and number of hidden layers are user-defined. Output layer size is determined by number of output quantities. Training is done with cross-validation using scikit-learn function "KFold" to split training data into parts. Subsequently, one model instance is trained on each sub-dataset to determine error metrics using the evaluation function described below. The metrics are averaged and standard deviation is calculated. To prevent overfitting, early stopping provided by Keras API is used. This stops the training procedure if the validation error does not improve over several epochs. The validation dataset is split off the training data of the current cross validation run.

The evaluation is performed using the functions of a third Python file. It compares the trained model against the separate test data and determines standard error metrics such as mean squared error (MSE) or mean absolute percentage error (MAPE) [21]. In addition, plots comparing predicted to true stress concentration factor and residuals are automatically generated. Furthermore, graphs of the progression of the prediction error during training are generated. These allow an assessment of the learning progress. Finally, the trained model is stored in serialized form for later use.

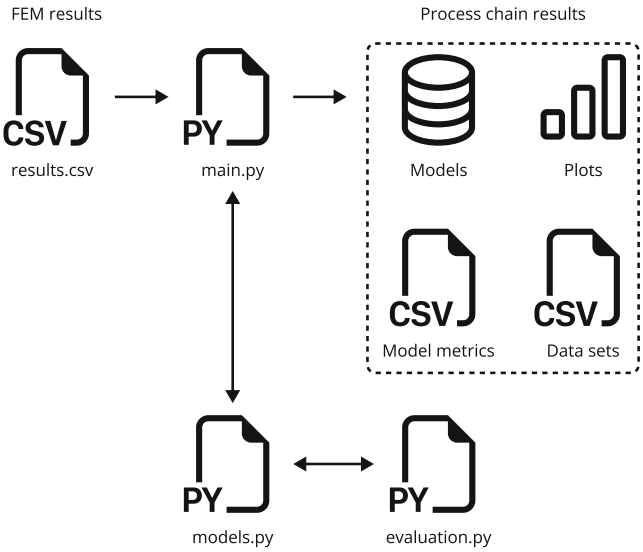

Fig. 4 Process chain and developed Python scripts

4 Application on shaft shoulders

4.1 Data generation

To illustrate the developed method, it is applied to determine the stress concentration factors for shaft shoulders under tension/compression, bending and torsion. The training data is generated by FEM simulation.

However, the developed workflow including the insights described in the following sections can be adapted to all types of notch geometries. The agnostic approach to the nature of the input data is a key advantage of the process chain. The only limitation lies in the ability to generate a large number of differently parameterized notch geometries. In general, this can be done with any FEM software. Alternatively automated FEM workflows can be applied, reducing the amount of manual effort. This approach is used in this work, using the program Kerbert [22]. Based on the definition of the notch geometry, the program automatically generates a model, meshes it and performs the FE calculation using the Ansys solver. Stress concentration factors can be automatically calculated and returned by the program in Reusable Engineering Exchange Standard (REXS) format [23]. To automate this process, a calculation workflow is developed. It first creates the geometry files based on a given parameter field. Afterwards, it starts the serialized calculation for all input files. It extracts the stress concentration factors from the resulting files and stores them together with the corresponding geometry data in a CSV file. This file constitutes the training data. The parameter field for the geometries to be examined is defined in accordance with DIN 743, which gives the following limits for the validity of the approximate equations [2]:

$$r/t \geq 0.03, \quad d/D \leq 0.98, \quad K_{\mathrm{t}} \leq 6. \tag{4}$$

In the following, the geometry parameter r/t is referred to as (notch) sharpness and d/D is called (notch) depth.

Within these limits, training data needs to be generated covering the range of valid stress concentration factors as homogeneously distributed as possible. For this purpose, the geometry parameters have to be chosen appropriately. Therefore, different distributions of notch sharpness and notch depth and their effects on the resulting stress concentration factors are investigated. The actual geometry parameters of each shaft shoulder are determined by randomly drawing from the distribution. With these values, the estimation of the expected stress concentration factors is carried out according to Eq. (2).

A normal distribution with mean 0.9 and standard deviation 0.3 is selected for the notch depth. According to Eq. (4), it is trimmed to a range of values from 0.05 to 0.98.

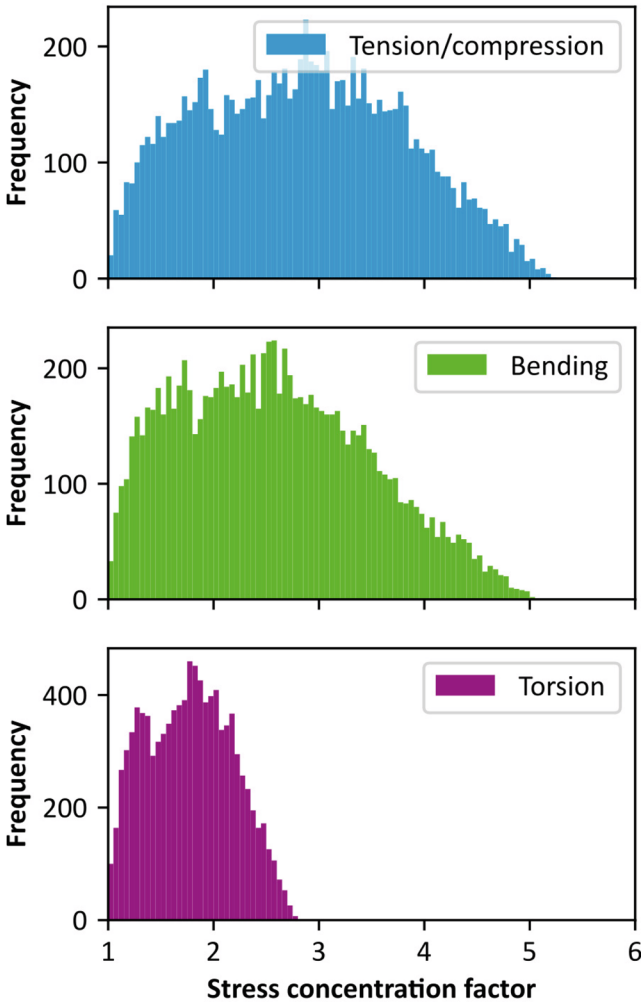

Fig. 5 Distributions of the expected stress concentration factors (calculated via approximate equations of DIN 743) for tension/compression, bending and torsion

For notch sharpness an exponential distribution is chosen, which produces predominantly small values of notch sharpness. These are necessary to provide a sufficient amount of large stress concentration factors for the training process. To ensure that larger values of the notch sharpness are also present in the training data, a normal distribution with a mean value of 1 is superimposed. The value range of this composite distribution is limited to 0.05–2. The deviation to the definition range of Eq. (4) is due to computational constraints. Preliminary tests showed that for very sharp notches with a notch sharpness < 0.05 no stable meshing and calculation is possible.

A total of 10.000 shaft shoulder geometries are created. With the described settings, the distributions of the stress concentration factors for each load case are shown in Fig. 5. A largely uniform distribution is obtained, which covers the range of practically relevant stress concentration factors very well.

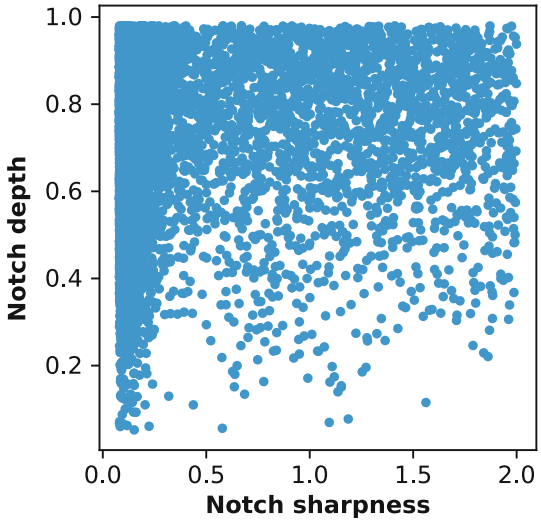

Fig. 6 Scatter plot of notch sharpness and notch depth

The scatter plot in Fig. 6 shows the realised combinations of notch sharpness and depth. The advantage of random sampling can be recognized clearly. In contrast to a full factorial combination, there are no discrete steps. Instead, the quantities can be considered as continuous.

4.2 Initial model training

In the following, three different regression models will be investigated. For reference, linear regression and a third-degree polynomial regression are considered. Their performance is compared to that of an ANN.

As explained in Sect. 3, the ANN is created using the Keras library. Initially, a small network shall be investigated. It consists of two hidden layers with five neurons each. Since two input parameters (notch sharpness and depth) and three output values (stress concentration factor for each load case) are given, the overall architecture of the network is 2-5-5-3. In the first layer, an additional normalization of the input data is performed. The learning rate is chosen to be 0.001 and 500 epochs of training are performed with a batch size of 32.

For the reference models the scikit-learn library is used. Normalization is carried out similarly to the ANN.

Table 2 Model metrics

Model	MSE	MAPE	R^2
Linear regression	0.1251	12.15%	0.8183
Polynomial regression	0.0178	4.59%	0.9753
Initial ANN	0.0010	1.07%	0.9984
Optimized ANN	0.0001	0.37%	0.9998
Regularized ANN	0.0004	0.68%	0.9993

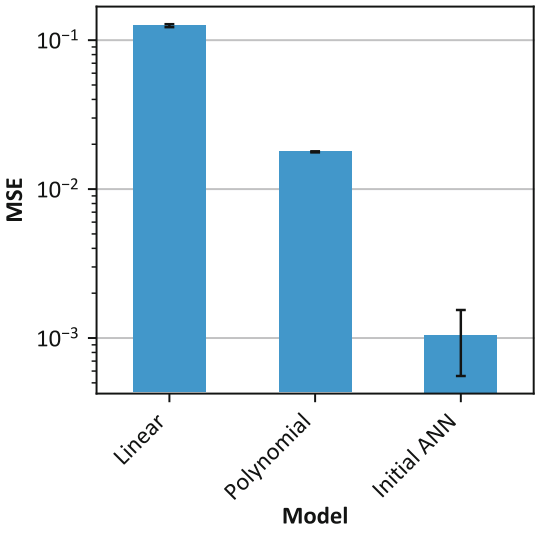

Fig. 7 MSE of different models for shaft shoulder stress concentration factor regression

With these settings, the results shown in Table 2 and Fig. 7 are obtained. The models are assessed using triple cross-validation.

The ANN shows a significantly better fit than linear or polynomial regression. While the MAPE of the classical regression techniques gets no better than 4,6%, it is possible to reduce it to 1.1% using the ANN.

To better assess the performance of each model, the prediction behavior on the test data is examined. For this purpose, the residuals are investigated in relation to the output values, see Fig. 8. They shall be homoscedastic, i.e. the residual scatter should be approximately equal and symmetrical across all stress concentration factors. For linear regression, this is obviously not the case. The residuals show a parabolic pattern. This is an indicator that the model cannot capture the complexity of the data sufficiently well. A similar pattern emerges with polynomial regression. Despite the quantitatively much better fit, the ANN also shows heteroskedastic behavior with slightly larger residuals in the range of higher stress concentration factors.

4.3 Model tuning

The pattern in the residuals plot of the ANN indicates that the model is not complex enough to represent the relationship. For this reason, its architecture shall be optimized. The other hyperparameters (learning rate, batch size) will also be tuned.

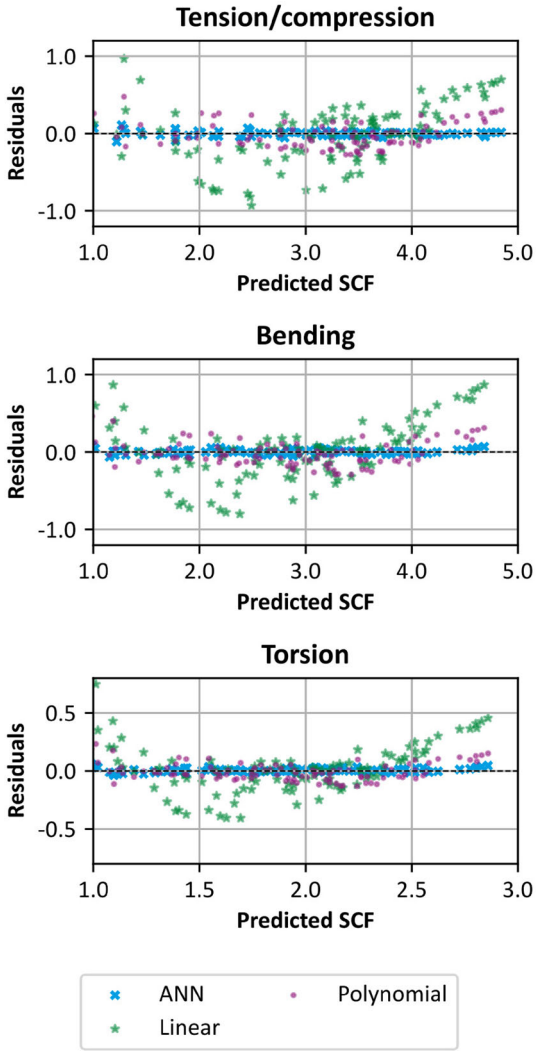

Fig. 8 Residuals of the stress concentration factors predicted by the models

Keras provides different tuners for this process. In this work the Hyperband tuner is used. It is a multi-fidelity approach that implements a successive halving algorithm [24]. Given a range of possible hyperparameters, a large number of model instances with different hyperparameter sets are trained over a few epochs. The best parameter combinations

Table 3 ANN hyperparameter ranges and optimized values

Hyperparameter	Range	Optimized result	Regularized
Neurons per hidden layer	4, 8, 12, 16, 20, 24, 28, 32	32 and 32	32 and 32
Learning rate	0.00005–0.005	0.001	0.001
Batch size	16, 32, 64	16	16
Dropout?	Yes, No	No	Yes
Dropout value	2–20%	–	2%

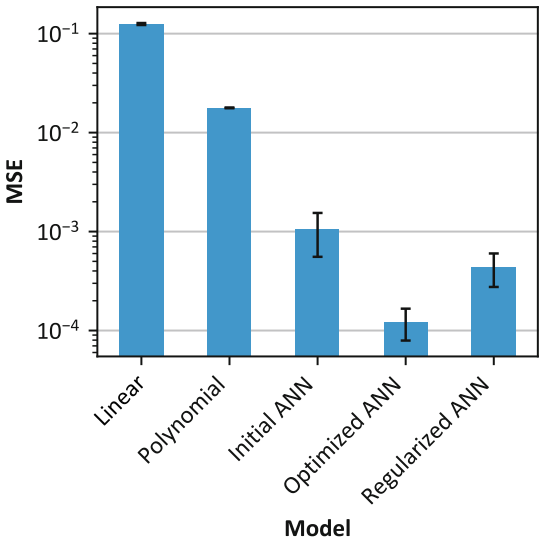

Fig. 9 MSE of the tuned models compared to initial results

are then trained further, while the weaker half are discarded. This process continues until only one model instance with the optimal hyperparameters remains.

The selected parameter ranges and the settings found by the tuning process are listed in Table 3.

Fig. 9 shows the mean errors of the tuned ANN compared to those with default settings. A significant improvement was achieved. The MAPE is minimized to 0.37%, which is a further reduction of 65% compared to the initial ANN. To rule out the possibility that this is a case of overfitting, the prediction results are examined in more detail. For this purpose, the predicted stress concentration factors are visualized as a function of the geometry parameters notch sharpness and depth, see Fig. 10. The resulting solution surface should preferably be as smooth as possible. Major unevenness is equivalent to overfitting individual training points, for which there is no justification in terms of the actual physical context.

The optimized ANN shows individual convolutions that indicate overfitting. This behavior is particularly evident in the plots in Figs. 11 and 12. These show the predicted stress concentration factors for a notch sharpness of $r/t = 0.4$, respectively $r/t = 1.3$ across all notch depths. While the stress concentration factors show a smooth trend for the notch with sharpness 0.4, a discontinuity can be seen for the smoother notch with sharpness 1.3. At a notch depth of around $d/D = 0.5$, the steepness changes abruptly, which seems physically not justified and might be an imperfection in the simulated data.

Regularization methods are used to correct this behavior. These limit the maximum complexity of the model already in the training process and thus reduce the risk of overfitting. For the ANN, this is achieved by dropout regularization. In this approach, a random fraction of neurons is deac-

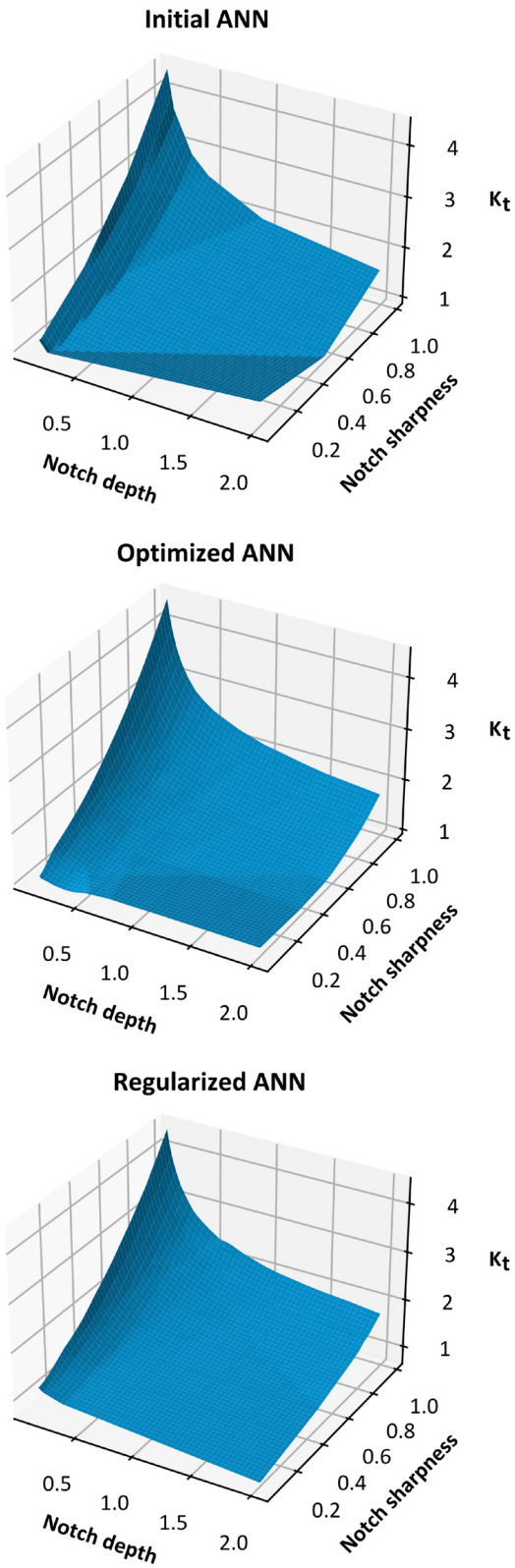

Fig. 10 Stress concentration factors for tension/compression as functions of notch sharpness and depth for different ANNs

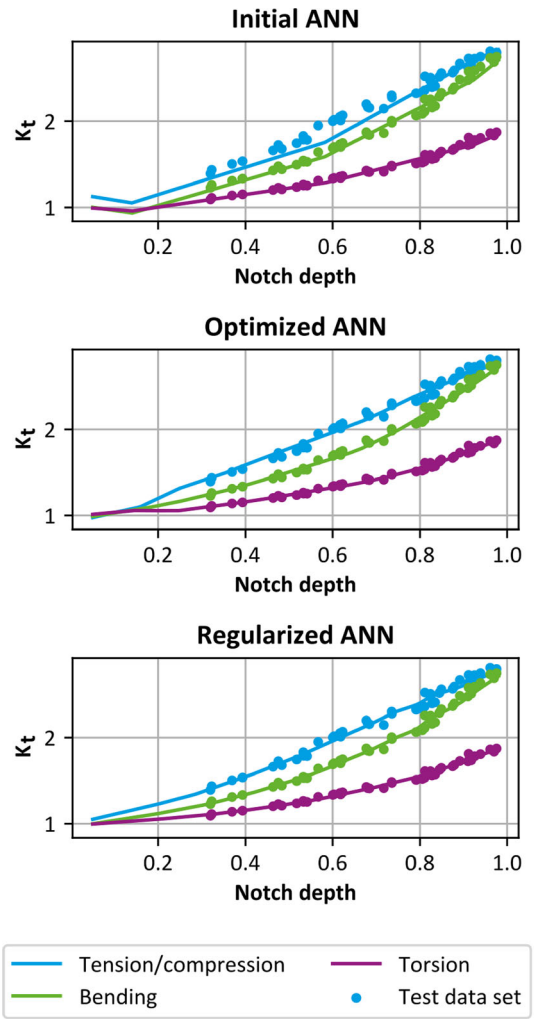

Fig. 11 Regression result of different ANNs for notch sharpness 0.4

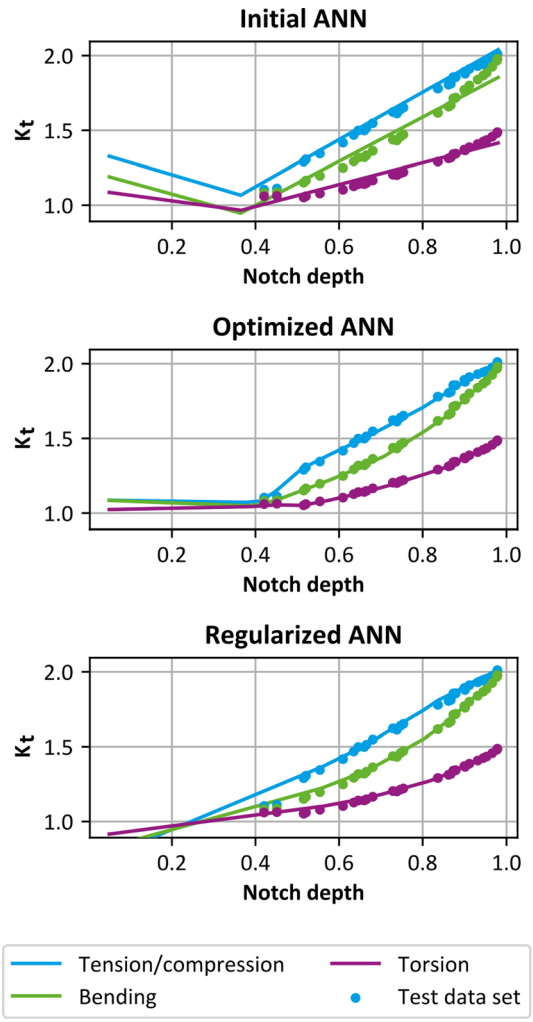

Fig. 12 Regression result of different ANNs for notch sharpness 1.3

tivated in each run during the training process. The network is forced to form alternative connections and excessively large weights leading to overfitting are prevented. [25]

Table 3 lists the setting of the regularized model. Particullary, a dropout of 2% is applied after every hidden layer. The result surface of the regularized ANN in Fig. 10 has no more convolutions and shows a smooth behavior. This observation is also shown in the plots in Figs. 11 and 12. The regularized neural network compensates the fluctuations in the data better than using the hyperparameters from tuning and is not influenced by single outlier data points.

Fig. 9 shows a compilation of the errors of the initial, optimized, and regularized ANN compared to those of the reference models. The improved generalization capability of the regularized ANN simultaneously causes individual diverging data points to be captured less accurately. As a result, the prediction error is increased, with an MAPE of 0.68%. This is referred to as the bias-variance trade-off [26]. A stable model capable of generalization shows a low variance, but has a larger bias (a larger systematic error). Conversely, overfitted models are able to represent the relationship between input and output very well and consequently have a low bias. At the same time, however, they are very sensitive to small variations in their training data and thus show a larger variance.

It can be stated that the used tuning algorithm prefers hyperparameter combinations with low bias. This results in a tendency to overfit. This is somewhat unexpected since the Hyperband tuning algorithm uses a separate validation dataset and multiple iterations to evaluate the achieved goodness of fit. However, since the numerical deviations

Table 4 Model metrics

Model	MSE	MAPE
DIN 743	0.0547	6.68%
Melzer	0.0039	1.85%
ANN	0.0005	0.79%

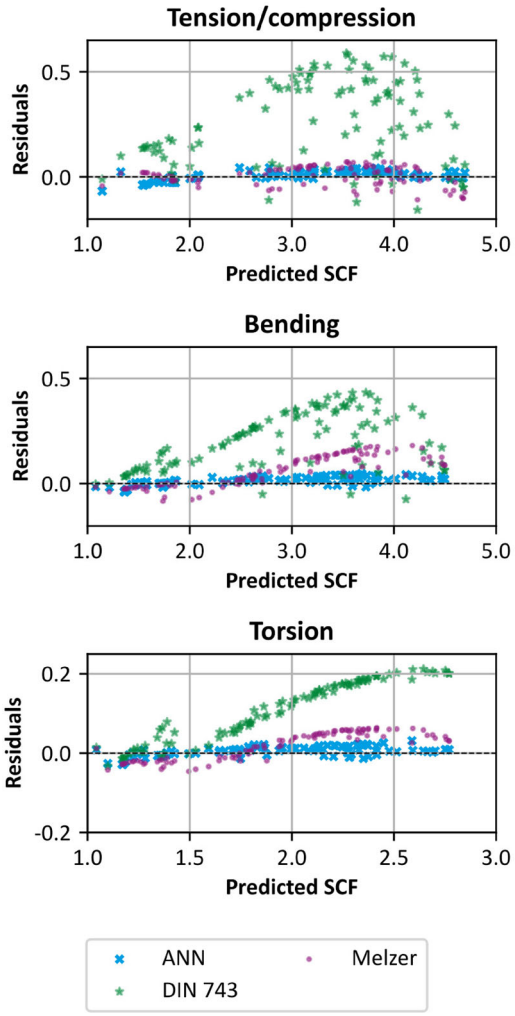

Fig. 13 Comparison of the residuals of different models

in the FEM results are present in the entire data set, the overfitting can only be detected to a limited extent with this method. A manual check of the optimization results is therefore recommended.

Furthermore, it can bee seen in Fig. 12 that the regularized ANN predicts stress concentration factors $K_t < 1$, which are not physically reasonable for a real shaft shoulder. This is due to the characteristics of the data creation process which has the objective of an even stress concentration factor distribution. This leads to the result that only few data points cover the range of small values of notch depth. However, since this area of extreme changes in shaft diameter is of little relevance in a technical context, no extension of training data was carried out for this work. Nevertheless, this emphasises the importance of the data collection process in machine learning. Since the extrapolation capability of regression models is always limited, it is of vital essence to cover a wide range of data points during the training process.

4.4 Comparison to other methods

In order to finally assess the quality of the models, a comparison with other calculation methods is performed. The first reference are approximate equations according to DIN 743 [2]. In addition, the modified equations according to Melzer [27] are also taken into account. They are of particular interest because these approximation equations were adjusted by regression to FEM results carried out by Melzer. This approach is therefore essentially comparable to the method presented here, although different regression techniques were used.

The error metrics for the different calculation methods are shown in Table 4. By using machine learning, the mean absolute percentage error on the test data set is reduced to 0.79%. This is an improvement of 57% compared to the Melzer equations. This corresponds to a very accurate representation of the stress concentration factors determined by the FEM simulation while maintaining the high calculation speed of using approximate equations.

In the residuals plot in Fig. 13, it can be clearly seen that the ANN provides more accurate results than the calculation according to DIN 743 across the entire stress concentration factor range investigated. Even in comparison to the calculation according to Melzer, the ANN shows better results. Nevertheless, it should be considered that the precision of the stress concentration factor prediction depends on the quality of the training data, in this case the FEM simulations. The determined error metrics focus solely on the models' capability to capture the rules inherent to the simulation results. Therefore, the results can be further improved by refined simulations.

5 Conclusion

Machine learning models possess great potential in quick and accurate solutions for complex problems. This work demonstrates the potential for new approaches to strength calculation. As they describe the complex relationship between notch geometry and stress concentration, the calculation of stress concentration factors is used to implement machine learning approaches for regression of these factors. The model development workflow allows for automatic calculation of FE-models for a given range of geometric parameters and notch geometry using the software Kerbert and Ansys. Afterwards, machine learning models can be automatically trained and optimized for the specific problem.

This process can be adapted to a multitude of application scenarios, allowing users with different backgrounds to benefit from the precision of FEM calculations without having to create the models manually. Since the entire pro-

cess from FEM modeling to training to output of usable models is fully automated, the workflow can be used with little prior knowledge of these topics, making the benefits of machine learning available to a wider range of users.

Results for the shaft shoulder, investigated in this work, show a good accuracy of the stress concentration factor prediction by the machine learning models, even surpassing common approximation equations like DIN 743 or Melzer. Once the training process is completed, the machine learning models combine a short computation time with high accuracy.

This proves most beneficial for complex geometries such as splined shafts. In the case of complex spline geometries, universally applicable approximation equations do not exist or are not as precise as desired, resulting in the need for complex FE models, which require high modelling effort and computational power, resulting in long computation times. This hindrance may be surmounted by the application of machine learning models trained on precise finite element outcomes for these complex geometries. This study presents the necessary tools to develop these models and enable fast yet accurate calculation of stress concentration factors for complex notch geometries.

Due to the model and data agnostic approach and the automatic adaption to the given structure of the input and output data, the developed workflow may even be applied to regression problems beyond the scope of stress concentration factors. With a sufficient amount of high quality training data, the potential use cases are almost unlimited.

Funding Open Access funding enabled and organized by Projekt DEAL.

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Pilkey WD, Pilkey DF, Bi Z (2020) Peterson's Stress Concentration Factors. Wiley, Hoboken
2. DIN 743-2:2012-12: Calculation of load capacity of shafts and axles – Part 2: Theoretical stress concentration factors and fatigue notch factors
3. Mayr CM, Rother K (2019) Improved stress concentration factors for circular shafts for uniaxial and combined loading. Mater Test 61(3):193–203. https://doi.org/10.3139/120.111305
4. Ozkan MT, Erdemir F (2020) Determination of stress concentration factors for shafts under tension. Mater Test 62(4):413–421. https://doi.org/10.3139/120.111500
5. Wendler J (2019) Tragfähigkeitsberechnung von Bauteilen mit Mehrfachkerben im Nennspannungskonzept. Technische Universität Dresden, Dresden (PhD thesis)
6. Urbas U, Zorko D, Vukašinović N (2021) Machine learning based nominal root stress calculation model for gears with a progressive curved path of contact. Mech Mach Theory 165:104430. https://doi.org/10.1016/j.mechmachtheory.2021.104430
7. Liang L, Liu M, Martin C, Sun W (2018) A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. J R Soc Interface 15(138):20170844. https://doi.org/10.1098/rsif.2017.0844
8. Li J (2019) Regression and classification in supervised learning. In: Proceedings of the 2nd international conference on computing and big data ICCBD 2019. Association for Computing Machinery, New York, pp 99–104 https://doi.org/10.1145/3366650.3366675
9. Brumen B, Hölbl M, Harej Pulko K, Welzer T, Hericko M, Juric M, Jaakkola H (2012) Learning process termination criteria. Informatica 23:521–536. https://doi.org/10.15388/Informatica.2012.373
10. Wang M, Cui Y, Wang X, Xiao S, Jiang J (2018) Machine learning for networking: workflow, advances and opportunities. IEEE Netw 32(2):92–99. https://doi.org/10.1109/MNET.2017.1700200
11. Raschka S (2018) Model evaluation, model selection, and algorithm selection in machine learning. CoRR. arXiv: 1811.12808
12. McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5(4):115–133. https://doi.org/10.1007/BF02478259
13. Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65(6):386–408. https://doi.org/10.1037/h0042519
14. Rumelhart DE, McClelland JL (1987) Learning internal representations by error propagation. In: Parallel distributed processing: explorations in the microstructure of cognition: foundations, pp 318–362
15. Bhadeshia HKDH (1999) Neural networks in materials science. ISIJ Int 39(10):966–979. https://doi.org/10.2355/isijinternational.39.966
16. Nwankpa C, Ijomah W, Gachagan A, Marshall S (2018) Activation functions: comparison of trends in practice and research for deep learning. CoRR. arXiv: 1811.03378
17. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830
18. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/. Accessed 2023-11-13
19. Chollet F et al (2015) Keras. https://keras.io. Accessed 2023-11-13
20. Waskom ML (2021) seaborn: statistical data visualization. JOSS 6(60):3021. https://doi.org/10.21105/joss.03021
21. Botchkarev A (2018) Evaluating performance of regression machine learning models using multiple error metrics in Azure machine learning studio. SSRN Electron J. https://doi.org/10.2139/ssrn.3177507

22. Ulrich C, Glamsch J, Zimmermann M (2023) Automatisierte Tragfähigkeitsberechnung von Mehrfachkerben durch parametrische FE-Modelle. FVA 700 III
23. FVA (2017) REXS – reusable engineering exchange standard. https://rexs.info/. Accessed 2023-11-13
24. Li L, Jamieson K, DeSalvo G, Rostamizadeh A, Talwalkar A (2018) Hyperband: a novel bandit-based approach to hyperparameter optimization
25. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(56):1929–1958
26. Geman S, Bienenstock E, Doursat R (1992) Neural networks and the bias/variance dilemma. Neural Comput 4(1):1–58. https://doi.org/10.1162/neco.1992.4.1.1
27. Melzer (2000) Numerische Ermittlung von Spannungsformzahlen für Hohlwellen und deren Aufbereitung für die Anwendung in der Tragfähigkeitsberechnung. FVA 353