International Journal of Fatigue 179 (2024) 108029

Contents lists available at ScienceDirect

International Journal of Fatigue

journal homepage: www.elsevier.com/locate/ijfatigue

Estimating notch fatigue limits via a machine learning-based approach structured according to the classic K_f formulas

Luca Susmel

Department of Civil and Structural Engineering, The University of Sheffield, Mapping Street, Sheffield S1 3JD, UK

ARTICLE INFO

Keywords:
Notch fatigue limit
Machine learning
K_f
Critical distance

ABSTRACT

This paper deals with the problem of estimating notch fatigue limits via machine learning. The proposed strategy is based on those constitutive elements that were used by the pioneers like Peterson, Neuber, Heywood, and Topper to devise their well-known formulas. The machine learning algorithms being considered were trained and tested using a database containing 238 notch fatigue limits taken from the literature. The outcomes from this study confirm that machine learning is a promising approach for designing notched components against fatigue. In particular, the accuracy in the estimates can easily be increased by simply increasing size and quality of the calibration dataset. Further, since machine learning regression models are highly flexible and can handle high-dimensional datasets with many input features, they can capture complex relationships between input features and the target variable. This means that the accuracy in estimating notch fatigue limit can be increased by including in the analyses further input features like, for instance, grain size or hardness. Finally, machine learning's generalization ability is crucial for regression tasks where the goal is to predict values for new materials.

1. Introduction

The prediction of the fatigue behaviour of materials containing notches (such as keyseats, fillets, or holes) is a topic of enduring interest in engineering and materials science. In structural components, notches act as stress concentrators and they are known to significantly reduce the fatigue strength of materials. Since the estimation of the fatigue strength of notched components is of primary importance in ensuring the structural integrity in various engineering applications, the development of reliable design methodologies has been a challenge for a large number of researchers, resulting in decades of dedicated investigations, explorations and experimental analyses. Estimating fatigue strength in the presence of stress concentrators involves the assessment of various factors that include, amongst others, geometrical profiles of the notches, material mechanical properties, loading conditions, and environmental effects.

When dealing with ferrous metallic materials, the *fatigue limit* is a threshold stress that is associated with the presence of a non-propagating crack whose growth is blocked by the first microstructural barrier (such as, for instance, the first grain boundary) [1,2]. In theory, when a component is in the fatigue limit condition, fatigue failure should never occur, i.e. the component is supposed to withstand without breaking for a number of cycles equal to infinity. However, due to cyclic- and time-dependent phenomena (where the specific features of the applied load history and the environment play a role of primary importance), fatigue limits are seen to disappear [3,4]. This is a consequence of the fact that in the very high-cycle fatigue regime cracks no longer initiate on the surface. In contrast, they develop inside of the components, with these internal cracks ultimately governing fatigue failures [3].

Unlike ferrous metals, non-ferrous materials instead do not have a fatigue limit. This is way they are always designed by targeting a specific finite number of cycles to failure [5]. Aluminium alloys are a classic example of engineering materials displaying no fatigue limit.

As far as fatigue design is concerned, since the fatigue limit can disappear or it does not exist at all, it is preferable to refer to the so-called *endurance limit*. The endurance limit is nothing but a threshold stress which is extrapolated to a reference number of cycles to failure that usually ranges in the interval $5{\cdot}10^5$–10^8 [5,6].

Bearing in mind what has been said above regarding fatigue versus endurance limits, for the sake of simplicity in what fallows the term fatigue limit will be used to indicate both the fatigue and the endurance limit. In other words, the term fatigue limit will be used to denote a reference threshold stress defined in the high-cycle fatigue regime. In this setting, it is worth pointing out that those theories that were originally developed by strictly referring to fatigue limits can be applied also in terms of endurance limits. This can be done provided that, for a given material, plain and notch endurance limits are determined under the same experimental conditions (in particular, under the same load ratio) and defined by referring to the same reference number of cycles to failure [5].

https://doi.org/10.1016/j.ijfatigue.2023.108029
Received 4 August 2023; Received in revised form 5 October 2023; Accepted 29 October 2023
Available online 31 October 2023

Nomenclature

a	notch depth
a_H	material constants in Heywood's relationships
a_N	material constant in Neuber's equation
a_P	material length in Peterson's formula
d_n, d_g	net and gross diameter/width of the notched specimens
f, f_1, f_2	functions used to estimate K_f
n	sample size
y	target in a linear regression
E	error (in percentage)
F	constant in DuQuesnay, Topper and Yu's formula
K	number of "folds"
K_f	fatigue strength reduction factor referred to the net area
$K_{f,i}$, $K_{f,i\text{-}ext}$	experimental and estimated value of K_f (for the i-th sample)
K_t	stress concentration factor referred to the net area
L	material characteristic length
N_f	number of cycles of failure
Oxyz	system of coordinates at the notch tip
R	load ratio
RMSE	Root Mean Square Error
X_1, …, X_n	features in a linear regression
β	notch opening angle
β_0, …, β_n	linear regression coefficients
ε	error term in a linear regression
k	geometrical quantity in Glinka and Newport's equation
σ_{ep}	linear-elastic peak stress
σ_{net}	nominal stress referred to the net area
σ_{UTS}	ultimate tensile strength
σ_y	normal stress parallel to axis y
σ_{min}, σ_{max}	minimum and maximum value of the stress in the cycle
$\sigma_{net,\,min}$, $\sigma_{net,max}$	minimum and maximum value of the net stress in the cycle
ρ	notch root radius
ΔK_{th}	threshold value of the stress intensity factor range
$\Delta\sigma_0$	plain material fatigue limit range
$\Delta\sigma_{0n}$	notch fatigue limit range referred to the net area
$\Delta\sigma_{0n,est}$	estimated value of the net notch fatigue limit range
$\Delta\sigma_{net}$	range of the nominal net stress

At its core, Machine learning (ML) is a subset of artificial intelligence (AI) that enables computer systems to automatically learn and improve from experience without being explicitly programmed for a particular task. It relies on mathematical algorithms and statistical models to identify patterns and relationships within data. The process involves feeding vast amounts of data into algorithms, which then iteratively learn from the data, adjusting their internal parameters to improve their performance on a given task.

As ML continues to advance, it finds applications in various fields, including fatigue. For instance, ML algorithms have been extensively applied to predict fatigue life in metallic materials [7–10], additively manufactured metals [11–13] and composite materials [14–16]. Despite these promising findings, challenges remain in the application of ML to fatigue and fracture prediction. Limited availability of high-quality and diverse datasets, issues of data reliability, and the need for interpretable models continue to be areas of active research.

In the scenario briefly discussed above, the ultimate objective of the present study is to provide a comprehensive understanding of the way of using various state-of-the-art ML techniques to estimate the notch fatigue limit. In particular, by revisiting a wide array of studies from the past to the present, this work aims to facilitate further advancements in predicting the fatigue behaviour of notched components by making the most of ML. It is anticipated that the knowledge gained from this study will aid engineers and researchers in making informed decisions to design safer and more reliable notched components, contributing to the overall advancement of the structural integrity discipline.

2. Considered machine learning techniques

The flow chart reported in Fig. 1 shows in a schematic, simplified way how ML algorithms can be used to make predictions.

The first step is to collect relevant data that will be used to train and evaluate the ML model being used. Once the data are collected, they need to be processed to ensure they are in a suitable format for the adopted ML algorithm (Fig. 1a).

The subsequent step is to choose a suitable ML algorithm (Fig. 1b). Selecting an appropriate ML algorithm is crucial and depends on the nature of the prediction problem. Different types of problems (e.g., regression, classification, clustering) and data characteristics (e.g., linear, nonlinear) may require specific algorithms. The selected

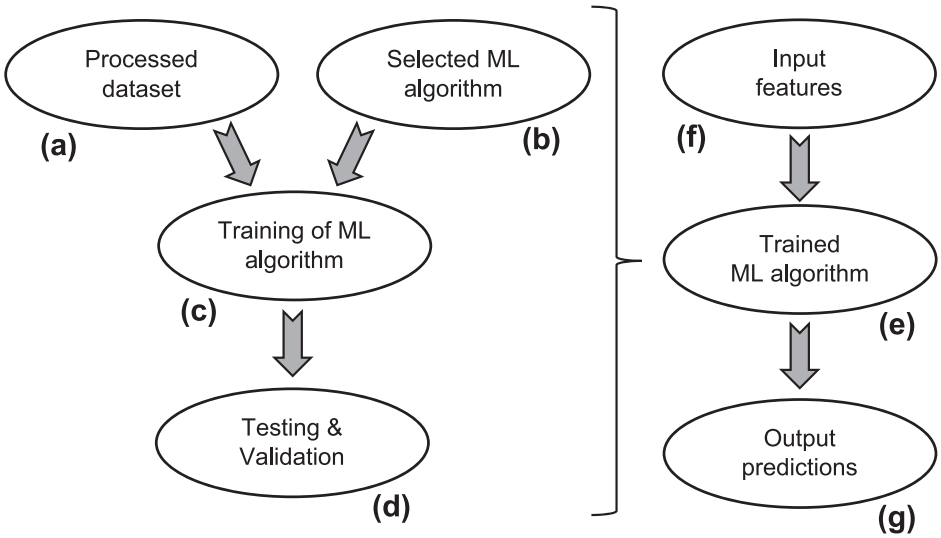

Fig. 1. Flow-chart summarising the procedure to use machine learning algorithms to make predictions.

algorithm is then trained on the training dataset (Fig. 1c), which involves providing the algorithm with the input features (X_i for i = 1,2. …, n) and the corresponding target (or output) variable (y_j for j = 1,2,…,k). The model learns from the data and adjusts its internal parameters iteratively to minimize the prediction error or loss function.

After training the model, it needs to be evaluated to assess its performance on unseen data (Fig. 1d). The model is tested on the testing dataset, and various performance metrics, such as accuracy, precision, recall, or mean squared error, are calculated to gauge its effectiveness in making predictions.

Once the model is trained and evaluated, it is ready to make predictions on new, unseen data (Fig. 1e). The model takes the input features of the new data as input (Fig. 1f) and produces the corresponding output predictions (Fig. 1g). These predictions can be used for various applications, such as prediction, classification, regression, anomaly detection, or clustering, depending on the nature of the problem.

In what follows, the ML algorithms used in the present investigation are briefly described.

2.1. Linear Regression

Linear Regression [17] is a widely used supervised ML algorithm that belongs to the family of regression models. It is designed to model the relationship between a dependent variable (target) and one or more independent variables (features) by fitting a linear equation to the data. The primary objective of linear regression is to find the best-fitting line that minimizes the difference between the predicted values and the actual target values, thus enabling the algorithm to make accurate predictions on new, unseen data.

Mathematically, a linear regression model can be represented as:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon \quad (1)$$

where y is the dependent variable (target) to be predicted, X_i (for i = 1, 2, …, n) are the independent variables (features) that influence the target, β_j (for i = 0, 1, …, n) are the regression coefficients that determine the relationship between the features and the target, and, finally, ε represents the error term, which accounts for the difference between the predicted values and the actual values.

The linear regression algorithm aims to estimate the regression coefficients β_j (for i = 0, 1, …, n) that best fit the given data. This estimation is often performed using the Ordinary Least Squares method, which minimizes the sum of the squared residuals (the differences between the actual target values and the predicted values).

Training a linear regression model involves feeding it with a labelled dataset, where both the independent variables (features) and the dependent variable (target) are known. The algorithm then iteratively adjusts the regression coefficients to minimize the error until convergence occurs, creating the best-fitting line that describes the relationship between the features and the target.

Linear regression is an interpretable and relatively simple algorithm, making it a popular choice for various applications, trend analysis included. However, it is essential to ensure that the data satisfies the assumptions of the linear regression model, such as linearity, independence of errors, and homoscedasticity, to obtain reliable and accurate predictions. In cases where the data exhibits non-linear relationships, more complex regression models or feature transformations may be required to achieve better results.

2.2. Support Vector Machines (SVM)

Support Vector Machines (SVM) [18] is a powerful and versatile supervised ML algorithm used for both classification and regression tasks. SVM is particularly effective in solving binary classification problems, where the goal is to separate data points into two classes. The algorithm operates by finding an optimal hyperplane in a high-dimensional feature space that best separates the data points belonging to different classes, while maximizing the margin (distance) between the closest data points of each class. This hyperplane is also referred to as the decision boundary.

The key concepts behind SVM involve: hyperplane, support vectors, margin, and kernel trick.

In a binary classification problem with n features, the hyperplane is an (n-1)-dimensional flat plane that separates the data points into two classes (e.g. in a 2D space, the hyperplane is a line, and in a 3D space, it is a plane, etc.).

Support vectors are the data points that lie closest to the decision boundary on either side and have the most influence on determining the position and orientation of the hyperplane. The support vectors are crucial in defining the maximum margin that SVM seeks to achieve.

In the SVM algorithm, the margin is the distance between the support vectors of different classes and the decision boundary. The primary objective of SVM is to maximize this margin, as it improves the algorithm's generalization ability and helps avoid overfitting. SVM can efficiently handle non-linearly separable data by transforming the input features into a higher-dimensional space using a kernel function. Common kernel functions include Polynomial, Radial Basis Function (RBF), and Sigmoid. The kernel trick allows SVM to implicitly compute the dot product between the transformed feature vectors without explicitly calculating the higher-dimensional coordinates, which can be computationally expensive.

Training an SVM involves finding the optimal hyperplane that maximizes the margin between classes. This process is formulated as a convex optimization problem, and various optimization techniques, such as the Sequential Minimal Optimization (SMO) algorithm, are commonly used to efficiently solve it. Once the hyperplane is determined, new data points can be classified by evaluating which side of the decision boundary they fall on.

SVM has proven to be a robust and effective algorithm for a wide range of applications, including image classification, text categorization, and bioinformatics. Its ability to handle high-dimensional data and nonlinear relationships, along with its strong theoretical foundation, makes SVM a popular choice for both academic research and real-world ML tasks. However, SVM's performance can be affected by the choice of the kernel and the appropriate regularization parameters, which should be carefully selected to achieve optimal results.

2.3. Gaussian Process Regression

Gaussian Process Regression (GPR) [19] is a powerful non-parametric ML algorithm used for regression tasks. It is a Bayesian probabilistic approach that allows for flexible modelling of complex relationships between input data and corresponding output values. GPR is particularly well-suited for scenarios where data points are sparse or noisy, and where the underlying function being modelled is unknown or difficult to define explicitly.

The fundamental concept of GPR revolves around modelling the relationship between input data points and output values as a distribution of functions rather than a single deterministic function. It assumes that any finite set of output values follows a joint multivariate Gaussian distribution. In simpler terms, GPR treats each prediction as a random variable with an associated mean and uncertainty.

A Gaussian process is defined by a mean function and a covariance function (also known as a kernel function). The mean function captures the overall trend in the data, while the covariance function defines the similarity between data points. Popular kernel functions include the Radial Basis Function (RBF), Matern, and Exponential kernels, among others. The choice of kernel determines the smoothness and complexity of the learned functions.

In GPR, prior beliefs about the relationship between input data and output values are represented by the mean and covariance functions. After observing data, the prior distribution is updated to a posterior

distribution, incorporating the newly acquired information. The posterior distribution represents the predictive distribution over functions, enabling uncertainty quantification in predictions.

GPR involves hyperparameters, such as the kernel parameters and noise variance, which need to be estimated from the data. The process of finding optimal hyperparameters often involves maximizing the likelihood of the observed data under the Gaussian process model.

The outcome of GPR is a predictive distribution over functions for new, unseen data points. The predictive distribution provides not only the point estimate of the output value but also the associated uncertainty (variance or confidence interval).

Training a Gaussian Process Regression model involves learning the hyperparameters and inferring the mean and covariance functions from the available data. The model can then make predictions for new data points by computing the predictive mean and variance using the trained Gaussian process.

Gaussian Process Regression excels in tasks with limited data, as it provides a principled approach to handle uncertainty, which is crucial when making predictions with sparse or noisy data. However, the algorithm's computational complexity increases significantly with the number of data points, which may limit its scalability for very large datasets. Nevertheless, GPR remains a popular choice in various fields, including robotics, finance, and engineering, where uncertainty estimation and high-quality predictions are essential.

2.4. Cross decomposition – Partial Least Squares (PLS) regression

Partial Least Squares (PLS) regression [20] is a powerful ML algorithm primarily used for multivariate regression tasks, especially when dealing with high-dimensional and collinear data. PLS regression is well-suited for scenarios where the number of features is much larger than the number of samples, and traditional linear regression techniques may suffer from overfitting or poor performance.

The PLS regression algorithm involves a process of extracting latent variables (also known as components) that capture the most relevant information from both the input features and the target variable. These latent variables are constructed in a way that maximizes the covariance between X_i and y_i, aiming to find the underlying relationships that explain the variability in both datasets.

In more detail, before applying PLS regression, data preprocessing steps such as mean centering and standardization are often performed. Mean centering ensures that the data has a zero mean, while standardization scales the data to have unit variance. These steps are crucial for the PLS algorithm to work effectively, especially when dealing with features of different scales.

PLS regression iteratively extracts a series of latent variables, each representing a linear combination of the original input features and target variable. These latent variables are derived to maximize the covariance between X_i and y_i, ensuring that the most relevant information from both datasets is captured.

The number of latent variables (components) to be extracted is a crucial hyperparameter in PLS regression. This number can be determined using various techniques, such as cross-validation or analyzing the explained variance. Selecting an appropriate number of components is essential to avoid overfitting and achieve a balance between model complexity and performance.

Once the latent variables are extracted, PLS regression performs a linear regression on these components to predict the target variable. The coefficients of the linear regression are computed during the modelling step, enabling predictions for new, unseen data.

PLS regression offers several advantages, including its ability to handle multicollinearity (correlation between input features) and its effectiveness in dealing with high-dimensional datasets. By capturing the most relevant information in the form of latent variables, PLS regression reduces the risk of overfitting, making it a valuable tool for tasks involving a large number of features.

However, PLS regression's performance heavily depends on the appropriate selection of the number of components and the quality of the extracted latent variables. Furthermore, it may not be the best choice for datasets with non-linear relationships or when dealing with a small number of samples.

2.5. Decision Tree Regression

Decision Tree Regression (DTR) [21] is a popular and versatile supervised ML algorithm used primarily for regression tasks. It employs a tree-like model to make predictions based on the input features, where each internal node represents a decision based on a specific feature, and each leaf node corresponds to a predicted output value.

The key components of DTR are briefly summarised in what follows. The DTR algorithm selects the best features and corresponding split points to create decision rules. The primary objective is to minimize the variance of the target variable within each subset, leading to more homogeneous subsets with respect to the predicted values.

The process of creating a decision tree involves recursively splitting the data into subsets based on the selected features and split points. Each subset represents a branch in the tree, and the process continues until a stopping criterion is met (e.g., reaching a predefined maximum tree depth or having a minimum number of samples in each leaf node).

The predicted output value at each leaf node in the tree is calculated as the average (or median) of the target variable values within that leaf node's subset. This prediction represents the regression model's estimate for the corresponding region of the feature space.

Decision trees may have a tendency to overfit the training data, capturing noise and small fluctuations in the data. To avoid overfitting and improve generalization, post-processing techniques like pruning can be applied. Pruning involves removing certain branches or nodes from the tree that do not contribute significantly to the overall predictive performance.

In a nutshell, training a DTR model involves finding the optimal split points and feature selection that minimize the variance within the resulting subsets. The process is performed recursively for each node, leading to the construction of the full tree. Optionally, pruning may be applied to refine the model and improve its performance on unseen data.

DTR offers several advantages, including its simplicity, interpretability, and ability to handle non-linear relationships between input features and the target variable. It can capture complex patterns and interactions in the data, making it particularly useful for tasks with non-linear behaviour.

DTR is a flexible and intuitive algorithm that effectively handles regression tasks with non-linear relationships. Its hierarchical decision-making process makes it easy to interpret and visualize, making it a valuable tool for both beginners and experienced data analysts. By controlling overfitting and leveraging ensemble methods, DTR can deliver accurate predictions in a wide range of real-world applications.

2.6. Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP) [22] is a widely used supervised ML algorithm belonging to the family of artificial neural networks. It is primarily used for various tasks, such as classification, regression, and pattern recognition, and is well-known for its ability to model complex relationships in data.

The fundamental building block of an MLP is the artificial neuron, also known as a node or perceptron. An MLP consists of multiple layers of interconnected neurons, each layer comprising an input layer, one or more hidden layers, and an output layer. The neurons in each layer are fully connected to the neurons in the subsequent layer, and each connection is associated with a weight.

The key components and operations of MLP involve activation function, forward propagation, loss function, and backpropagation.

Each neuron in an MLP applies an activation function to the

weighted sum of its inputs to introduce non-linearity into the model. Common activation functions include the sigmoid, Rectified Linear Unit, hyperbolic tangent, and softmax (used in the output layer for classification tasks).

To make predictions, data is fed into the input layer of the MLP. The input data is then passed through the network layer by layer, where each neuron computes its activation based on the weighted sum of its inputs and applies the activation function. The outputs of the neurons in one layer serve as inputs to the neurons in the next layer until the output layer is reached, which generates the final predictions.

The MLP's performance is evaluated using a loss function that measures the difference between the predicted values and the actual target values. Common loss functions for regression tasks include Mean Squared Error (MSE), while for classification tasks, Cross-Entropy Loss (log loss) is commonly used.

The process of training an MLP involves updating the weights of the connections to minimize the chosen loss function. This is achieved through an optimization algorithm known as backpropagation, which computes the gradients of the loss function with respect to the model's weights. The weights are then adjusted in the direction that reduces the loss, iteratively improving the model's performance during the training process.

MLP's strength lies in its ability to learn complex patterns and representations from data, making it a powerful tool for a wide range of applications, including image and speech recognition, natural language processing, and time series prediction. However, MLP can be sensitive to the choice of hyperparameters, such as the number of hidden layers, the number of neurons in each layer, and the learning rate. Tuning these hyperparameters and dealing with potential issues such as overfitting can be essential for achieving optimal performance.

MLP is then a versatile and powerful neural network architecture capable of handling various ML tasks. Its capacity to learn non-linear relationships makes it particularly well-suited for complex data modelling. By carefully tuning hyperparameters and managing training parameters, MLP can produce accurate predictions in a wide range of engineering applications.

2.7. Implementation of the selected machine learning algorithm

As far as open-source programming language Python is concerned, open-access software environments for ML include:

- Scikit-Learn: a widely used Python library for machine learning that provides a comprehensive set of tools for classification, regression, clustering, and more.
- TensorFlow: an open-source deep learning framework for building neural networks and deep learning models.
- PyTorch: an open-source deep learning framework known for its flexibility and dynamic computation graph.
- Keras: a high-level neural networks API that can run on top of TensorFlow, Theano, or CNTK, making it accessible and easy to use.
- XGBoost: an optimized gradient boosting library that is highly efficient and effective for a wide range of machine learning tasks.

The ML algorithms briefly described in Sections 2.1 to 2.6. were used to estimate notch fatigue limits by taking full advantage of the Scikit-Learn library [23]. This choice was dictated by the fact that Scikit-Learn is the most popular and widely used open-source ML library for Python. It provides a rich set of tools and algorithms for various ML tasks, making it an effective, easy-to-use tool for ML-based analyses. The Scikit-Learn library collects a very large numbers of ML algorithms that are subdivided into the following six groups: Classification, Regression, Clustering, Dimensionality Reduction, Model Selection and Preprocessing. The ML algorithms used in the present investigation belong to the Regression group. To date, this group contains 17 subgroups, allowing the user to employ more than 100 different ML algorithms. In the present investigation, we selected a limited number of ML techniques, focusing our attention on those algorithms that are most widely used to solve regression problems in different scientific disciplines. This being said, the goal of this paper is to investigate the accuracy and reliability of the ML-based approach in estimating notch fatigue limits, with this being done by using some well-known, representative algorithms. In this setting, it can be highlighted that other than Python/Scikit-Learn, there are some other vendors of commercial machine learning solutions such as, for instance, Microsoft Azure Machine Learning, Google Cloud AI Platform, SAS, MATLAB and many others. Each of these software packages offers a range of tools and services tailored to different needs. In the present investigation, we decided to take full advantage of Python and the Scikit-Learn library because it is a very popular/widely used open source/open access tool. Accordingly, those researchers and engineers who are interested in employing ML to predict notch fatigue limits can easily make direct use of the methodology discussed in the present paper. This also explains the reason why all the experimental data used in this investigation are reported in an explicit form: the data summarised in Table 1 and the open access Scikit-Learn library allow anyone either to replicate the calculations reported below or to use ML to estimate notch fatigue limits for different materials and/or for different notch geometries.

3. The high-cycle notch fatigue problem

Consider the notched component sketched in Fig. 2a. This component is subjected to either tension or bending and the material under investigation is assumed to be linear-elastic. The stress field in the vicinity of the notch tip can be determined either numerically or analytically. As to the latter strategy, for instance, Glinka and Newport [24] suggested estimating the distribution of normal stress σ_y along the notch bisector via the following well-known formulas:

Blunt notch under tension, $K_t \leq 4.5$

$$\sigma_y = K_t\sigma_{net}\left[1-2.33\left(\frac{x}{\rho}\right)+2.59\left(\frac{x}{\rho}\right)^{1.5}-0.907\left(\frac{x}{\rho}\right)^{2}+0.037\left(\frac{x}{\rho}\right)^{3}\right] \quad (2)$$

Sharp notch under tension, $K_t > 4.5$

$$\sigma_y = K_t\sigma_{net}\left[1-0.235\left(\frac{x}{\rho}\right)^{0.5}-1.33\left(\frac{x}{\rho}\right)+1.28\left(\frac{x}{\rho}\right)^{1.5}-0.337\left(\frac{x}{\rho}\right)^{2}\right] \quad (3)$$

Blunt notch under bending or bending and tension, $K_t \leq 4.5$

$$\sigma_y = K_t\sigma_{net}\left[1-2.33\left(\frac{x}{\rho}\right)+2.59\left(\frac{x}{\rho}\right)^{1.5}-0.907\left(\frac{x}{\rho}\right)^{2}+0.037\left(\frac{x}{\rho}\right)^{3}\right]\left(1-\frac{x}{\kappa}\right) \quad (4)$$

In Eqs (2) to (4) σ_{net} is the nominal net stress (defined as seen in Fig. 2), ρ is the notch root radius and, finally, κ is a geometrical quantity that depends on the stress gradient characterising the nominal net stress distribution (Fig. 2b). According to Neuber [25] and Peterson [26], the stress concentration factor referred to the net area, K_t, is defined as:

$$K_t = \frac{\sigma_{ep}}{\sigma_{net}} \quad (5)$$

where σ_{ep} is the elastic peak stress at the notch tip (Fig. 2).

Consider now the notched specimen shown in Fig. 3a. This specimen (containing a known geometrical feature) is subjected to a nominal uniaxial loading, i.e., either to a cyclic axial loading or to cyclic bending. The fatigue strength of both the notched specimen and the parent material is summarised in the SN log–log chart of Fig. 3b. This diagram plots the range of the nominal net stress, $\Delta\sigma_{net}$, vs. the number of cycles to failure, N_f. The upper fatigue curve refers to the plain material, whereas the lower one to the notched specimens. According to Fig. 3b, the reduction in fatigue strength due to the presence of the notch is quantified through the fatigue strength reduction factor, K_f, which is defined as [27]:

Table 1
Summary of the collected experimental results (CNB = circumferential notch cylindrical bar, CNP = centre notch in plate, DENP = double edge notch in plate, CBF = cylindrical bar with fillet, AX = push–pull, B = bending, RB = rotating bending).

Material	Ref.	σ_{UTS} [MPa]	Specimen Type	Load Type	$\Delta\sigma_0$ [MPa]	ΔK_{th} [MPa$\sqrt{m}$]	R	a [mm]	d_n [mm]	d_g [mm]	ρ [mm]	β [°]	K_t	$\Delta\sigma_{0n}$ [MPa]	K_f
AA356-T6	[40]	260	CNB	RB	231	4.4	−1	0.24	7.32	7.56	0.1	79.9	3.68	126.5	1.83
					231	4.4	−1	0.63	8.41	9.04	0.18	59.2	4.16	136.4	1.69
					231	4.4	−1	1.3	7.73	9.03	0.09	60.3	8.94	71.4	3.23
					231	4.4	−1	2.9	9.08	12	0.08	68.3	17.20	40.7	5.68
C45	[41]	632	CNB	RB	582	8.1	−1	0.01	5	5.01	0.05	60	1.67	550.0	1.06
					582	8.1	−1	0.01	5	5.01	0.02	60	2.06	550.0	1.06
					582	8.1	−1	0.01	5	5.01	0.01	60	2.52	560.0	1.04
					582	8.1	−1	0.01	5	5.02	0.05	60	1.95	490.0	1.19
					582	8.1	−1	0.01	5	5.02	0.02	60	2.52	500.0	1.16
					582	8.1	−1	0.01	5	5.02	0.01	60	3.19	490.0	1.19
					582	8.1	−1	0.1	5	5.2	0.6	60	1.58	420.0	1.39
					582	8.1	−1	0.1	5	5.2	0.3	60	1.89	380.0	1.53
					582	8.1	−1	0.1	5	5.2	0.1	60	2.72	360.2	1.62
					582	8.1	−1	0.1	5	5.2	0.05	60	3.54	360.0	1.62
					582	8.1	−1	0.1	5	5.2	0.02	60	5.21	360.0	1.62
					582	8.1	−1	0.5	5	6	0.6	60	1.86	360.0	1.62
					582	8.1	−1	0.5	5	6	0.3	60	2.39	300.0	1.94
					582	8.1	−1	0.5	5	6	0.1	60	3.80	280.0	2.08
					582	8.1	−1	0.5	5	6	0.05	60	5.19	280.0	2.08
					582	8.1	−1	0.5	5	6	0.02	60	7.94	290.0	2.01
					582	8.1	−1	0.5	5	6	0.01	60	11.00	290.0	2.01
					582	8.1	−1	1.5	5	8	0.6	60	1.91	350.0	1.66
					582	8.1	−1	1.5	5	8	0.3	60	2.52	280.0	2.08
					582	8.1	−1	1.5	5	8	0.1	60	4.09	250.0	2.33
					582	8.1	−1	1.5	5	8	0.05	60	5.66	250.0	2.33
					582	8.1	−1	1.5	5	8	0.02	60	8.78	250.0	2.33
					582	8.1	−1	1.5	5	8	0.01	60	12.30	260.0	2.24
C36	[42]	999	CNB	RB	450	4.6	−1	0.1	13	15	0.2	60	1.56	443.2	1.02
					450	4.6	−1	0.15	13	14.4	0.2	60	2.02	345.2	1.30
					450	4.6	−1	0.3	13	14	0.2	60	2.87	253.1	1.78
					450	4.6	−1	0.5	13	13.6	0.2	60	3.91	187.0	2.41
					450	4.6	−1	0.7	13	13.3	0.2	60	4.94	146.5	3.07
					450	4.6	−1	1	13	13.2	0.2	60	6.16	118.6	3.80
6060-T6	[43]	220	DENP	AX	110	6.1	0.1	2.5	45	50	1.25	0	3.22	61.1	1.79
					110	6.1	0.1	10	30	50	2	0	3.20	52.8	2.07
					110	6.1	0.1	2.5	45	50	0.2	0	7.00	52.9	2.07
					110	6.1	0.1	10	30	50	0.2	0	9.18	42.3	2.59
SM41B	[44,45]	423	CNP	AX	326	12.4	−1	3	39	45	0.16	–	8.48	110.0	2.96
					326	12.4	−1	3	39	45	0.39	–	5.72	120.0	2.72
					326	12.4	−1	3	39	45	0.83	–	4.23	110.0	2.96
					326	12.4	−1	3	39	45	3	–	2.60	148.0	2.20
					274	8.4	0	3	39	45	0.16	–	8.48	73.0	3.75
					244	6.4	0.4	3	39	45	0.16	–	8.48	84.2	2.90
Mild Steel (0.15 % C)	[46]	440	CNB	AX	420	12.8	−1	5.08	32.8	43	0.05	60	14.00	118.0	3.56
					420	12.8	−1	5.08	32.8	43	0.1	60	10.00	120.0	3.50
					420	12.8	−1	5.08	32.8	43	0.13	60	9.00	116.0	3.62
					420	12.8	−1	5.08	32.8	43	0.25	60	6.60	118.0	3.56
					420	12.8	−1	5.08	32.8	43	0.64	60	4.40	118.0	3.56
					420	12.8	−1	5.08	32.8	43	1.27	60	3.30	132.0	3.18
					420	12.8	−1	5.08	32.8	43	5.08	60	1.90	208.0	2.02
Mild Steel (0.15 % C)	[46]	440	DENP	AX	420	12.8	−1	5.08	53.8	64	0.1	0	12.50	100.0	4.20
					420	12.8	−1	5.08	53.8	64	0.25	0	8.20	108.0	3.89
					420	12.8	−1	5.08	53.8	64	0.5	0	6.10	100.0	4.20
					420	12.8	−1	5.08	53.8	64	1.27	0	4.00	124.0	3.39
					420	12.8	−1	5.08	53.8	64	7.62	0	2.10	186.0	2.26
Al-2024-T351	[35]	466	CNP	AX	248	5.0	−1	0.12	44.8	45	0.12	–	2.98	160.0	1.55
					248	5.0	−1	0.25	44.5	45	0.25	–	2.96	124.0	2.00
					248	5.0	−1	0.5	44	45	0.5	–	2.94	124.0	2.00
					248	5.0	−1	1.5	42	45	1.5	–	2.82	90.0	2.76
					172	4.0	0	0.12	44.8	45	0.12	–	2.98	172.9	0.99
					172	4.0	0	0.25	44.5	45	0.25	–	2.96	114.3	1.51
					172	4.0	0	0.5	44	45	0.5	–	2.94	109.4	1.57
					172	4.0	0	1.5	42	45	1.5	–	2.82	91.9	1.87
SAE 1045	[35]	745	CNP	AX	608	13.6	−1	0.12	44.8	45	0.12	–	2.98	358.9	1.69
					608	13.6	−1	0.25	44.5	45	0.25	–	2.96	310.0	1.96
					608	13.6	−1	0.5	44	45	0.5	–	2.94	279.2	2.18
					608	13.6	−1	1.5	42	45	1.5	–	2.82	248.0	2.45
					608	13.6	−1	2.5	40	45	2.5	–	2.70	261.0	2.33
					448	6.9	0	0.12	44.8	45	0.12	–	2.98	326.7	1.37
					448	6.9	0	0.25	44.5	45	0.25	–	2.96	311.5	1.44
					448	6.9	0	0.5	44	45	0.5	–	2.94	276.1	1.62
					448	6.9	0	1.5	42	45	1.5	–	2.82	227.1	1.97

(continued on next page)

Table 1 (*continued*)

Material	Ref.	σ_{UTS} [MPa]	Specimen Type	Load Type	$\Delta\sigma_0$ [MPa]	ΔK_{th} [MPa$\sqrt{m}$]	R	a [mm]	d_n [mm]	d_g [mm]	ρ [mm]	β [°]	K_t	$\Delta\sigma_{0n}$ [MPa]	K_f
					448	6.9	0	2.5	40	45	2.5	–	2.70	235.1	1.91
Al-Alloy BS L65	[47]	486	CNB	AX	300	4.2	−1	5.08	32.8	43	0.01	55	27.00	80.0	3.75
					300	4.2	−1	5.08	32.8	43	0.05	55	14.00	77.0	3.90
					300	4.2	−1	5.08	32.8	43	0.1	55	10.00	46.0	6.52
					300	4.2	−1	5.08	32.8	43	0.2	55	7.30	46.0	6.52
					300	4.2	−1	5.08	32.8	43	0.51	55	4.85	62.0	4.84
					300	4.2	−1	5.08	32.8	43	1.270	55	3.30	93.0	3.23
2.25 Cr − 1 Mo Steel	[48]	530	CNB	AX	440	12.0	−1	0.03	4.94	5	0.03	0	2.99	440.0	1.00
					440	12.0	−1	0.05	4.90	5	0.05	0	2.95	420.0	1.05
					440	12.0	−1	0.07	4.86	5	0.07	0	2.92	340.0	1.29
					440	12.0	−1	0.20	4.60	5	0.20	0	2.68	280.0	1.57
					440	12.0	−1	0.40	4.20	5	0.40	0	2.34	296.0	1.49
					440	12.0	−1	0.76	3.48	5	0.76	0	1.87	320.0	1.38
G40.11	[49]	538	CNP	AX	464	15.9	−1	0.20	69.60	70	0.20	–	2.98	338.0	1.37
					464	15.9	−1	0.48	69.04	70	0.48	–	2.96	242.0	1.92
					464	15.9	−1	4.80	60.40	70	4.80	–	2.59	238.0	1.95
AISI 304	[47]	505	CNB	AX	720	12.0	−1	5.08	32.8	43	0.05	60	14.00	124.0	5.81
Ni-Cr Steel	[47]	869	CNP	AX	1000	12.8	−1	0.51	21.6	22.6	0.13	60	4.82	207.4	4.82
					1000	12.8	−1	5.08	32.8	43	0.05	60	18.33	54.6	18.31
					1000	12.8	−1	5.08	21.6	31.8	0.13	60	11.77	85.1	11.75
EN-GJS-800–8	[43]	800	DENP	AX	440	8.1	0.1	0.4	19.2	20	0.04	90	7.52	144.7	3.04
					440	8.1	0.1	1	18	20	0.1	90	6.29	109.6	4.02
Grey Iron	[50]	249	CNB	AX	155	15.9	−1	3.18	23.6	30	0.3	90	5.60	146.6	1.06
					99	11.2	0.1	3.18	23.6	30	0.3	90	5.60	96.6	1.02
					68	8.0	0.5	3.18	23.6	30	0.3	90	5.60	70.9	0.96
					48	5.2	0.7	3.18	23.6	30	0.3	90	5.60	51.5	0.93
2024-T3	[51,52]	497	DENP	AX	304		−1	9.53	38.1	57.2	1.45	0	4.35	96.0	3.17
					304		−1	9.53	38.1	57.2	8.06	0	2.14	165.0	1.84
7075-T6	[51,52]	569	DENP	AX	414		−1	9.53	38.1	57.2	1.45	0	4.35	103.5	4.00
					414		−1	9.53	38.1	57.2	8.06	0	2.14	213.1	1.94
SAE 4130	[51,52]	817	DENP	AX	648	8.5	−1	9.53	38.1	57.2	1.45	0	4.35	193.6	3.35
					648	8.5	−1	9.53	38.1	57.2	8.06	0	2.14	345.1	1.88
HT60	[53]	590	DENP	AX	580	13.0	0	0.5	50	51	0.05	90	6.25	257.0	2.26
					580	13.0	0	1	50	52	0.05	90	7.82	183.0	3.17
					580	13.0	0	5	50	60	0.05	90	18.08	108.0	5.37
					580	13.0	0	12.5	50	75	0.05	90	22.60	80.9	7.17
					580	13.0	0	0.5	50	51	0.05	135	4.34	263.0	2.21
					580	13.0	0	5	50	60	0.05	135	10.87	142.0	4.08
					580	13.0	0	12.5	50	75	0.05	135	11.86	113.0	5.13
2024-T3	[54]	427	CNP	AX	231		0	1.59	98.4	102	1.5	–	2.91	110.5	2.09
					231		0	3.18	95.2	102	3.18	–	2.83	103.5	2.23
					231		0	6.35	88.9	102	6.35	–	2.67	102.9	2.25
					231		0	12.7	76.2	102	12.7	–	2.43	104.0	2.22
					231		0	25.4	50.8	102	25.4	–	2.16	128.0	1.80
					231		0	0.79	49.2	50.8	0.79	–	2.91	128.0	1.80
					231		0	1.59	47.6	50.8	1.59	–	2.83	117.3	1.97
					231		0	3.18	44.4	50.8	3.18	–	2.67	109.7	2.10
					231		0	6.35	38.1	50.8	6.35	–	2.43	108.0	2.14
					231		0	12.7	25.4	50.8	12.7	–	2.16	132.0	1.75
2024-T3	[54]	427	CNP	AX	290		−1	1.59	98.4	102	1.59	–	2.91	138.3	2.10
					290		−1	3.18	95.2	102	3.18	–	2.83	128.0	2.27
					290		−1	6.35	88.9	102	6.35	–	2.67	123.4	2.35
					290		−1	12.7	76.2	102	12.7	–	2.43	126.7	2.29
					290		−1	25.4	50.8	102	25.4	–	2.16	136.0	2.13
					290		−1	0.79	49.2	50.8	0.79	–	2.91	138.3	2.10
					290		−1	1.59	47.6	50.8	1.59	–	2.83	128.0	2.27
					290		−1	3.18	44.4	50.8	3.18	–	2.67	124.6	2.33
					290		−1	6.35	38.1	50.8	6.35	–	2.43	126.7	2.29
					290		−1	12.7	25.4	50.8	12.7	–	2.16	136.0	2.13
7075-T6	[54]	565	CNP	AX	243		0	1.59	98.42	102	1.59	–	2.91	113.6	2.14
					243		0	3.18	95.24	102	3.18	–	2.83	117.3	2.07
					243		0	25.40	50.80	102	25.40	–	2.16	142.0	1.71
					243		0	0.79	49.22	50.8	0.79	–	2.91	129.0	1.88
					243		0	1.59	47.62	50.8	1.59	–	2.83	123.7	1.96
					243		0	12.70	25.40	50.8	12.70	–	2.16	142.0	1.71
7075-T6	[54]	565	CNP	AX	290		−1	1.59	98.42	102	1.59	–	2.91	138.3	2.10
					290		−1	3.18	95.24	102	3.18	–	2.83	126.9	2.28
					290		−1	25.40	50.80	102	25.40	–	2.16	146.0	1.99
					290		−1	0.79	49.22	50.8	0.79	–	2.91	146.6	1.98
					290		−1	1.59	47.62	50.8	1.59	–	2.83	134.4	2.16
					290		−1	12.70	25.40	50.8	12.70	–	2.16	152.0	1.91
2024-T3	[51]	497	CNP	AX	304		−1	38.1	38.1	114	38.1	–	2.08	165.0	1.84
7075-T6	[51]	569	CNP	AX	414		−1	38.1	38.1	114	38.1	–	2.08	213.0	1.94
SAE 4130	[51]	817	CNP	AX	648	8.5	−1	38.1	38.1	114	38.1	–	2.08	345.0	1.88

(*continued on next page*)

Table 1 (*continued*)

Material	Ref.	σ_{UTS} [MPa]	Specimen Type	Load Type	$\Delta\sigma_0$ [MPa]	ΔK_{th} [MPa$\sqrt{m}$]	R	a [mm]	d_n [mm]	d_g [mm]	ρ [mm]	β [°]	K_t	$\Delta\sigma_{0n}$ [MPa]	K_f
0.4 % C steel	[55]	648	CNB	B	664		−1	0.51	7.62	8.64	0.01	55	18.00	358.2	1.85
3 % Ni steel	[55]	526	CNB	B	685		−1	0.51	7.62	8.64	0.01	55	18.00	419.8	1.63
3/3.5 % Ni steel	[55]	723	CNB	B	704		−1	0.51	7.62	8.64	0.01	55	13.30	605.2	1.16
Cr-Va steel	[55]	752	CNB	B	858		−1	0.51	7.62	8.64	0.01	55	12.10	432.2	1.99
3.5 % NiCr steel	[55]	895	CNB	B	1081		−1	0.51	7.62	8.64	0.02	55	8.70	537.2	2.01
3.5 % NiCr steel	[55]	897	CNB	B	1019		−1	0.51	7.62	8.64	0.02	55	8.70	494.0	2.06
NiCrMo steel	[55]	1000	CNB	B	1321		−1	0.51	7.62	8.64	0.03	55	7.50	543.4	2.43
Low Carbon Steel	[56]	500	CNB	AX	424.6		−1	2.54	7.62	12.7	0.2	35	4.61	137.0	3.10
					424.6		−1	2.54	7.62	12.7	0.4	35	3.38	182.8	2.32
C40 Steel	[57]	715	CNB	AX	528		−1	4	12	20	0.5	90	3.68	235.6	2.24
En3B	[58]	676	CNB	AX	668		−1	1.5	5	8	0.2	60	3.80	170.0	3.93
SAE 1045	[59]	621	CBF	B	392		−1	5	40	50	5	–	1.42	377.2	1.04
39NiCrMo3	[60]	995	CNB	AX	631		−1	4	12	20	0.1	90	7.46	314.2	2.01
AISI 416	[61]	700	CNB	AX	699		−1	4	12	20	0.1	90	7.46	195.0	3.59
					699		−1	2	16	20	0.1	90	7.62	190.2	3.68
					699		−1	0.5	19	20	0.1	90	5.35	363.2	1.93
EN-GJS400	[62]	378	CNB	AX	284		−1	4	12	20	0.1	90	7.46	168.0	1.69
C40 Steel	[63]	852	CNB	AX	544		−1	1.43	9.15	12	0.23	35	4.42	165.8	3.28
En6	[64]	701	CNB	AX	343		−1	10	18	38	1.5	0	2.69	116.1	2.95
SAE 1045	[65]	760	CNB	AX	146.0		0.8	3.3	6	12.6	2	0	1.65	220.0	0.66
					146.0		0.8	3.23	5.44	11.9	0.25	65	3.65	130.0	1.12
SAE 1045	[65]	1220	CNB	AX	240.0		0.8	3.3	6	12.6	2	0	1.65	280.0	0.86
					240.0		0.8	3.23	5.44	11.9	0.25	65	3.65	146.0	1.64
SAE 1045	[65]	2370	CNB	AX	380.0		0.8	3.3	6	12.6	2	0	1.65	360.0	1.06
					380.0		0.8	3.23	5.44	11.9	0.25	65	3.65	146.0	2.60
SAE 1045	[65]	760	CNB	AX	72.0		0.9	3.3	6	12.6	2	0	1.65	110.0	0.65
					72.0		0.9	3.23	5.44	11.9	0.25	65	3.65	100.0	0.72
SAE 1045	[65]	1220	CNB	AX	120.0		0.9	3.3	6	12.6	2	0	1.65	170.0	0.71
					120.0		0.9	3.23	5.44	11.9	0.25	65	3.65	120.0	1.00
SAE 1045	[65]	2370	CNB	AX	230.0		0.9	3.3	6	12.6	2	0	1.65	240.0	0.96
					230.0		0.9	3.23	5.44	11.9	0.25	65	3.65	140.0	1.64
Ti-6Al-4 V	[66–68]	978	CNB	AX	529	4.3	0.1	0.13	5.47	5.72	0.13	60	2.80	240.5	2.20
					529	4.3	0.1	0.25	5.21	5.72	0.2	60	2.80	258.0	2.05
					529	4.3	0.1	0.73	4.26	5.72	0.33	60	2.70	244.9	2.16
Ti-6Al-4 V	[66–68]	978	DENP	AX	529	4.3	0.1	0.13	5.25	5.51	0.15	60	2.72	235.1	2.25
					529	4.3	0.1	0.64	4.24	5.51	0.43	60	2.72	195.9	2.70
Ti-6Al-4 V	[66–68]	978	CNB	AX	362	2.9	0.5	0.13	5.47	5.72	0.13	60	2.80	202.1	1.79
					362	2.9	0.5	0.25	5.21	5.72	0.2	60	2.80	191.4	1.89
					362	2.9	0.5	0.73	4.26	5.72	0.33	60	2.70	194.5	1.86
Ti-6Al-4 V	[66–68]	978	DENP	AX	362	2.9	0.5	0.13	5.25	5.51	0.15	60	2.72	196.6	1.84
					362	2.9	0.5	0.64	4.24	5.51	0.43	60	2.72	165.2	2.19
Ti-6Al-4 V	[66–68]	978	CNB	AX	184	2.6	0.8	0.13	5.47	5.72	0.13	60	2.80	146.2	1.26
					184	2.6	0.8	0.25	5.21	5.72	0.2	60	2.80	145.0	1.27
					184	2.6	0.8	0.73	4.26	5.72	0.33	60	2.70	148.5	1.24
Ti-6Al-4 V	[66–68]	978	DENP	AX	184	2.6	0.8	0.13	5.25	5.51	0.15	60	2.72	164.5	1.12
					184	2.6	0.8	0.64	4.24	5.51	0.43	60	2.72	141.7	1.30
Ti-6Al-4 V	[66–68]	978	CNB	AX	529	4.3	0.1	0.13	5.47	5.72	0.13	60	2.85	264.3	2.00
					529	4.3	0.1	0.28	5.16	5.72	0.13	60	3.51	202.7	2.61
					529	4.3	0.1	0.64	4.45	5.72	0.13	60	4.07	191.3	2.77
					529	4.3	0.1	0.1	5.52	5.72	0.33	60	1.97	362.5	1.46
					529	4.3	0.1	0.2	5.31	5.72	0.33	60	2.30	320.8	1.65
					529	4.3	0.1	0.38	4.96	5.72	0.33	60	2.58	306.2	1.73
					529	4.3	0.1	0.73	4.26	5.72	0.33	60	2.72	248.7	2.13
Ti-6Al-4 V	[66–68]	978	CNB	AX	362	2.9	0.5	0.13	5.47	5.72	0.13	60	2.85	193.7	1.87
					362	2.9	0.5	0.28	5.16	5.72	0.13	60	3.51	160.4	2.25
					362	2.9	0.5	0.64	4.45	5.72	0.13	60	4.07	135.4	2.67
					362	2.9	0.5	0.1	5.52	5.72	0.33	60	1.97	250.6	1.44
					362	2.9	0.5	0.2	5.31	5.72	0.33	60	2.30	210.6	1.72
					362	2.9	0.5	0.38	4.96	5.72	0.33	60	2.58	174.6	2.07
					362	2.9	0.5	0.73	4.26	5.72	0.33	60	2.72	156.2	2.32
Low Carbon Steel	[69]	500	CNB	RB	436.1		−1	2.54	7.62	12.7	0.2	35	2.93	179.2	2.43
					436.1		−1	2.54	7.62	12.7	0.4	35	2.50	214.6	2.03
Ti-6Al-4 V	[70,71]	978	CNB	AX	522	3.9	0	0.1			0.05	45	3.92	522.0	1.00
					522	3.9	0	0.3			0.2	45	3.16	480.4	1.09
					522	3.9	0	0.3			0.05	45	5.71	347.6	1.50
					522	3.9	0	0.5			0.05	45	6.61	245.6	2.13
FeP04	[72]	310	DENP	AX	247	10.0	0.1	10	30	50	0.16	45	11.52	75.7	3.26
					247	10.0	0.1	10	30	50	0.16	135	6.54	81.3	3.04
					247	10.0	0.1	10	30	50	0.16	160	3.35	145.3	1.70
SS41	[73]	418	DENP	AX	231	6.4	0.05	10	30	50	0.1	90	18.13	43.2	5.35
					231	6.4	0.05	10	30	50	0.1	120	13.34	66.5	3.47
HT60	[73]	598	DENP	AX	425	6.6	0.05	10	30	50	0.1	90	18.13	51.7	8.23
					425	6.6	0.05	10	30	50	0.1	120	13.34	72.3	5.88
SAE 1010-HR	[74]	326	CNP	AX	320	11.8	−1	0.5	44	45	0.5	–	2.94	220.7	1.45

(*continued on next page*)

Table 1 (*continued*)

Material	Ref.	σ_{UTS} [MPa]	Specimen Type	Load Type	$\Delta\sigma_0$ [MPa]	ΔK_{th} [MPa$\sqrt{m}$]	R	a [mm]	d_n [mm]	d_g [mm]	ρ [mm]	β [°]	K_t	$\Delta\sigma_{0n}$ [MPa]	K_f
					320	11.8	−1	1.5	42	45	1.5	–	2.81	205.1	1.56
SAE 1010-CR22	[74]	476	CNP	AX	410	10.2	−1	0.12	44.8	45	0.12	–	2.98	308.3	1.33
					410	10.2	−1	0.5	44	45	0.5	–	2.94	280.8	1.46
					410	10.2	−1	1.5	42	45	1.5	–	2.81	251.5	1.63
SAE 1010-CR56	[74]	525	CNP	AX	546	8.4	−1	0.12	44.8	45	0.12	–	2.98	339.1	1.61
					546	8.4	−1	1.5	42	45	1.5	–	2.81	254.0	2.15
SAE101-CR76	[74]	689	CNP	AX	614	6.4	−1	0.12	44.8	45	0.12	–	2.98	363.3	1.69
					614	6.4	−1	0.5	44	45	0.5	–	2.94	321.5	1.91
					614	6.4	−1	1.5	42	45	1.5	–	2.81	265.8	2.31
SAE 945X-HR	[74]	558	CNP	AX	500	13.4	−1	0.12	44.8	45	0.12	–	2.98	390.6	1.28
					500	13.4	−1	0.5	44	45	0.5	–	2.94	301.2	1.66
					500	13.4	−1	1.5	42	45	1.5	–	2.81	290.7	1.72
SAE 945-CR30	[74]	621	CNP	AX	588	12.4	−1	0.12	44.8	45	0.12	–	2.98	374.5	1.57
					588	12.4	−1	0.5	44	45	0.5	–	2.94	321.3	1.83
					588	12.4	−1	1.5	42	45	1.5	–	2.81	307.9	1.91
SAE 945-CR61	[74]	752	CNP	AX	630	12.0	−1	0.12	44.8	45	0.12	–	2.98	406.5	1.55
					630	12.0	−1	0.5	44	45	0.5	–	2.94	324.7	1.94
					630	12.0	−1	1.5	42	45	1.5	–	2.81	310.3	2.03

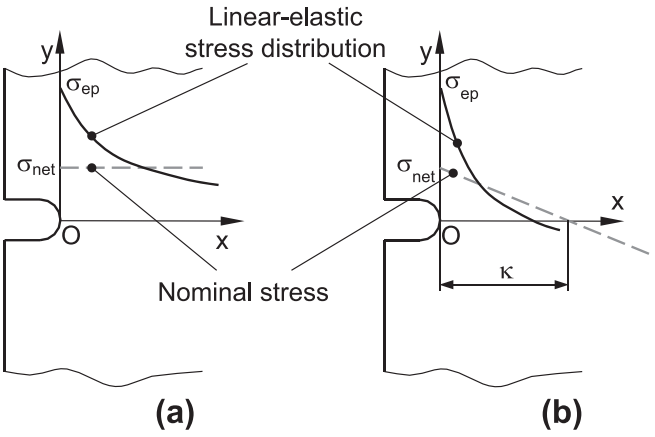

Fig. 2. Notched component loaded in tension (a) and in bending (b); definition of nominal net stress, σ_{net}, and elastic peak stress, σ_{ep}.

$$K_f = \frac{\Delta\sigma_0}{\Delta\sigma_{0n}} \tag{6}$$

In Eq. (6) $\Delta\sigma_0$ and $\Delta\sigma_{0n}$ are the range of the plain and notch fatigue limit, respectively. To compute K_f correctly, $\Delta\sigma_0$ and $\Delta\sigma_{0n}$ are to be determined either under the same load ratio or by setting the same mean stress value.

Definition (6) and Fig. 3b make it evident that the experimental approach is certainly the most accurate method for the quantification of the fatigue strength reduction factors. However, owing to the fact that this is not always feasible in practice, systematic research work has been carried out since the early 1900s to formulate specific approaches capable of estimating K_f for different materials and different notch profiles [28,29].

At the beginning of the 1900s Neuber [25] argued that the detrimental effect of geometrical features can be quantified by averaging the stress in the vicinity of the assessed notch over materials units such as grains or "arbitrary particles" of material (Fig. 4a). According to this idea, fatigue strength reduction factor can directly be estimated as follows:

$$K_f = 1 + \frac{K_t - 1}{1 + \sqrt{\frac{\alpha_N}{\rho}}} \tag{7}$$

In Eq. (7) α_N is a reference material length that is linked with the grain size or an "arbitrary particle" of material. As suggested in Refs [29–31], α_N can be estimated from the material ultimate tensile strength, σ_{UTS}, via the following empirical relationship (valid for $\sigma_{UTS} < 1520$ MPa):

$$\alpha_N = 10^{-\frac{\sigma_{UTS}-134}{586}} \tag{8}$$

where $\alpha_N = f(\sigma_{UTS})$ is measured in units of [mm] and σ_{UTS} in units of [MPa].

A few years later, the approach proposed by Neuber was further simplified by Peterson [3] who proposed to assess notch fatigue strength by referring to the stress determined at a given distance from the tip of the stress raiser being designed (Fig. 4b). Based on this idea, the fatigue strength reduction factor can then be estimated through the following

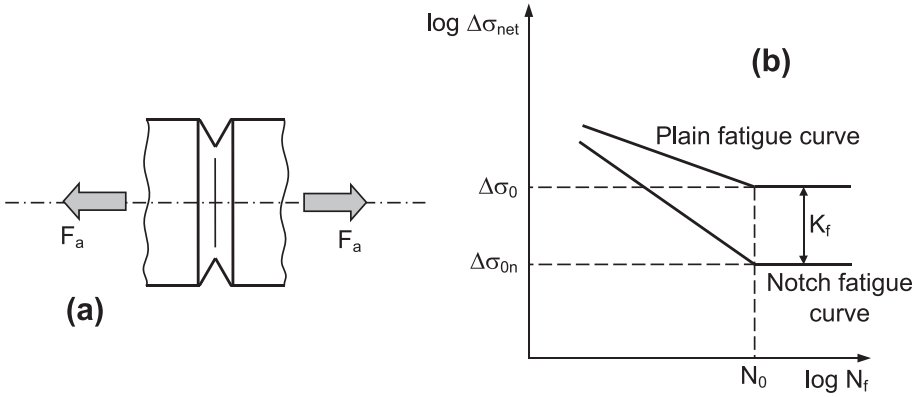

Fig. 3. Notched component subjected to fatigue loading (a) and SN diagram summarising the strength of the plain and notched material (b).

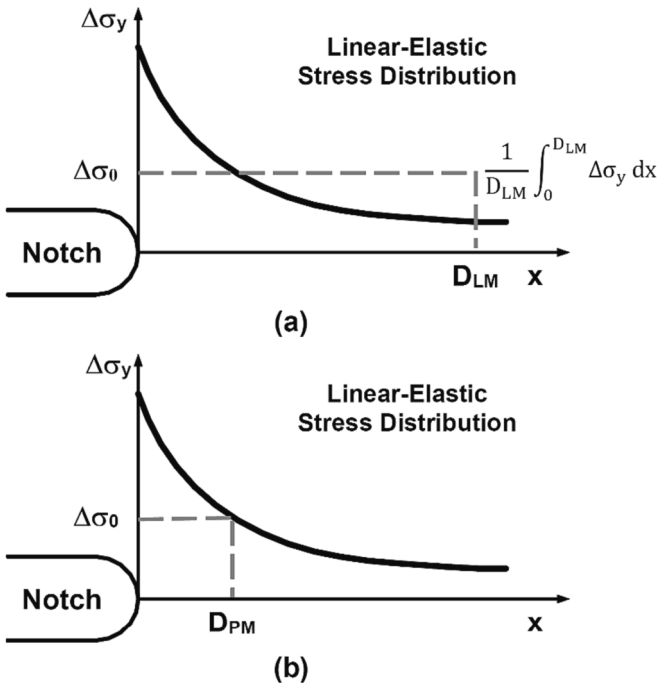

Fig. 4. Neuber's line method (a) and Peterson's point method (b).

relationship:

$$K_f = 1 + \frac{K_t - 1}{1 + \frac{\alpha_P}{\rho}} \tag{9}$$

where again α_P is a characteristic length which is different for different materials. For steels having $\sigma_{UTS} > 560$ MPa material constant α_P can be estimated as follows [6]:

$$\alpha_P = f(\sigma_{UTS}) = 0.0254\left(\frac{2079}{\sigma_{UTS}}\right)^{1.8} \text{[mm]} \tag{10}$$

Even if their derivation was based on a different reasoning, in the late 1950s Heywood proposed two formulas similar to those devised by Peterson and Neuber, i.e. [6,32]:

$$K_f = \frac{K_t}{1 + 2\sqrt{\frac{\alpha_H}{\rho}}} \tag{11}$$

$$K_f = \frac{K_t}{1 + 2\sqrt{\frac{\alpha_H}{\rho}}\left(\frac{K_t - 1}{K_t}\right)} \tag{12}$$

In Eqs (11) and (12) α_H is a material constant that depends on both the component geometry and the ultimate tensile strength. For instance, for cast iron with spheroid graphite $\sqrt{a_H}$ is equal to $\frac{173.6}{\sigma_{UTS}}$ [$mm^{1/2}$].

If attention is focused on Eqs (2) to (4), it is straightforward to observe that, according to the way Glinka and Newport structured their formulas, the shape of the linear-elastic stress field along the notch bisector depends on both the stress concentration factor, K_t, and the notch root radius, ρ. As mentioned earlier, Peterson and Neuber derived their formulas by assuming that K_f depends on the distribution of the local linear-elastic stress in the vicinity of the notch being assessed (Fig. 4). These two remarks explain why K_f calculated according to Eqs (7) and (9) depends on K_t and ρ. Further, the material constants in Eqs (7), (9), (11) and (12) can all be estimated from the ultimate tensile strength. Based on these considerations, the hypothesis can be formed that K_f can directly be estimated from σ_{UTS}, K_t and ρ as follow:

$$K_f = f(\sigma_{UTS}, K_t, \rho) \tag{12}$$

where f is a complex function that can take different forms.

If the threshold value of the stress intensity factor range, ΔK_{th}, is brought into play then the so-called material critical distance can be determined according to the following definition [33,34]:

$$L = \frac{1}{\pi}\left(\frac{\Delta K_{th}}{\Delta\sigma_0}\right)^2 \tag{13}$$

Critical length L can then be used as a further calibration information to estimate K_f. For instance, DuQuesnay, Topper and Yu [35] suggested estimating the fatigue strength reduction factor in the presence of sharp notches by using the following formula:

$$K_f = \frac{1}{F}\left(1 + \sqrt{\frac{a}{L}}\right) \tag{14}$$

where F is a geometric constant of the order of unity and a is the notch depth.

Similarly, by applying Neuber's line method (Fig. 4a) according to Tanaka's strategy [36], Atzori, Lazzarin, Meneghetti and Tovo devised the following formula to estimate K_f [37,38]:

$$K_f = \frac{K_t}{\sqrt{1 + \frac{4L}{\rho}}} \tag{15}$$

where again the fatigue strength reduction factor depends on critical distance L. As to Eq. (15), it is interesting to observe that the structure of this formula is very similar to the one proposed by Lukáš and Klesnil [39].

Eqs (13), (14) and (15) make it evident that, when critical length L is used, K_f can directly be estimated as follow:

$$K_f = f(\Delta\sigma_0, \Delta K_{th}, K_t, \rho) \tag{16}$$

where, again, f is a complex function that can take different forms.

Having briefly reviewed our understanding of the notch fatigue problem, the classic equations to estimate K_f that have been devised over the last century clearly suggest two possible strategies - i.e., Eq. (12) and (16) - to use ML to quantify the notch fatigue strength reduction factor. Accordingly, in what follows, these two strategies will be reformulated and expanded in order to use the ML algorithms reviewed in Section 2 to directly estimate K_f.

4. Formulation of a machine learning-based approach to estimate notch fatigue limits

When it comes to designing notched components against high-cycle fatigue, engineers are supposed to know the static strength (σ_{UTS}) of the material planned to be used. Further, independently of the strategy used to estimate K_f, according to Eq. (6) the notch fatigue limit ($\Delta\sigma_{0n}$) can be estimated only if the plain fatigue limit is known ($\Delta\sigma_0$). Thus, it is possible to form the hypothesis that both σ_{UTS} and $\Delta\sigma_0$ are available during the fatigue design process. According to this initial assumption, relationship (12) can then be expanded as follows:

$$K_f = f_1(\sigma_{UTS}, \Delta\sigma_0, K_t, \rho) - \textit{Strategy 1} \tag{17}$$

In other words, Eq. (17) suggests that, according to Peterson, Neuber and Heywood, K_f depends on two different pieces of information, i.e. the material mechanical behaviour and the distribution of the local linear-elastic stress field. The former is quantified via σ_{UTS} and $\Delta\sigma_0$, whereas the latter via K_t and ρ – see also Eqs (2) to (4). In what follows, this will be referred to as Strategy 1.

Assuming again that, for the same reasons as above, both σ_{UTS} and $\Delta\sigma_0$ are known during the design process, then relationship (16) based on the use of critical distance L, Eq. (13), can be rearranged as follows:

$$K_f = f_2(\sigma_{UTS}, \Delta\sigma_0, \Delta K_{th}, K_t, \rho) - \textit{Strategy 2} \tag{18}$$

As far this second strategy is concerned, while the profile of the linear-elastic stress field is still assumed to depend on K_t and ρ, this time the mechanical behaviour is modelled via three different material properties, i.e. σ_{UTS}, $\Delta\sigma_0$ and ΔK_{th}.

In the next section, the accuracy in estimating notch fatigue limits of Strategy 1, Eq. (17), and Strategy 2, Eq. (18), applied along with the MF algorithms briefly reviewed in Section 2 will be tested based on a large database of experimental results taken from the literature.

5. Validation by experimental data

5.1. The database

As per the flowchart seen in Fig. 1, the first step to use the ML approach is to build a coherent database suitable for training and testing the various algorithms being considered. As far as the notch fatigue problem is concerned, the data collected from the technical literature to create a suitable population of experimental results are summarised in Table 1. These results were generated by testing notched specimens having the geometries schematically sketched in Fig. 5. The technical drawings of Fig. 5 also explain the meaning of the symbols used in Table 1 to quantify the absolute dimensions of the various notched samples being considered. The results listed in Table 1 were generated under either axial loading (AX), bending (B) or rotating bending (RB), with the load ratio ($R = \sigma_{min}/\sigma_{max} = \sigma_{net,min}/\sigma_{net,max}$) ranging in the interval 1-0.9.

The values of the net stress concentration factor, K_t, listed in Table 1 were taken from the original sources and double-checked either by using Peterson's book [75] or via standard bi-dimensional linear-elastic Ansys® Finite Element (FE) models.

Finally, it is important to point out that the values of both $\Delta\sigma_0$ and $\Delta\sigma_{0n}$ were reported in the original sources either as fatigue limits or as endurance limits. The reader is referred to the original papers for detail descriptions of the various experimental strategies being followed in order to generate the individual experimental data listed in Table 1.

5.2. Selecting the most effective machine learning algorithm

As already mentioned in Section 2, the considered ML algorithms were applied by making the most of the Scikit-Learn library for Python [23]. In this regard, it is important to highlight here that the solvers coded to use the various ML algorithms being investigated were all characterised by the same level of programming complexity.

The accuracy of Strategy 1, Eq. (17), and Strategy 2, Eq. (18), was assessed via the Root Mean Square Error (RMSE) defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(K_{f,i} - K_{f,i-est}\right)^2} \tag{19}$$

In Eq. (19) n is the sample size, whereas $K_{f,i}$ and $K_{f,i\text{-}ext}$ are (for the i-th sample) the experimental and estimated value of the fatigue strength reduction factor, respectively.

To apply the GPR algorithm, the "length_scale" hyperparameter in the Radial Basis Function kernel was tuned using K-Fold Cross-Validation. In the GPR algorithm the kernel determines how much influence nearby data points have on the prediction for a given data point. Different kernels will lead to different regression behaviours. K-Fold Cross-Validation is a technique used to assess the performance of a ML algorithm. In particular, the dataset is divided into K roughly equal-sized "folds" or subsets and the ML algorithm under investigation is trained and evaluated K times. In each iteration, one fold is used as the test set, and the remaining K-1 folds are used for training. By so doing, the algorithm's performance is quantified K times, typically using metrics like accuracy or mean squared error. Subsequently, the K performance scores are averaged to provide a single, more reliable estimate of the model's performance. K-Fold Cross-Validation helps to assess a ML algorithm's performance while reducing the risk of overfitting or underfitting, as it tests the model on different subsets of the data. K-Fold Cross-Validation was used to set the "length_scale" hyperparameter by taking K equal to 10, with this process resulting in an optimal value of 0.06 for our specific datasets. This optimal value was determined by using the RMSE, Eq. (19), to quantify the performance of the Scikit-Learn GPR algorithm.

In the PLS regression algorithm, the number of components refers to the number of latent variables used to represent the relationships between the predictors and the response variables. Choosing the appropriate number of components is essential to strike a balance between model complexity and its ability to capture the underlying relationships in the data. Typically, the number of components is set by using cross-validation techniques or by examining the prediction performance on a validation dataset. As the number of components increases, the model can potentially fit the training data better, but it might also become more prone to overfitting. In this setting, the optimal choice may vary depending on the specific dataset and the problem one is trying to solve. Therefore, it is common practice to explore different values for the number of components and select the one that gives the best balance between model complexity and performance on new, unseen data. According to the above consideration, to apply the Scikit-Learn PLS regression algorithm, different values were explored and, via a simple "trial and error" procedure, the best predictions - assessed via the RMSE, Eq. (19) - were obtained by setting the number of components equal to 3.

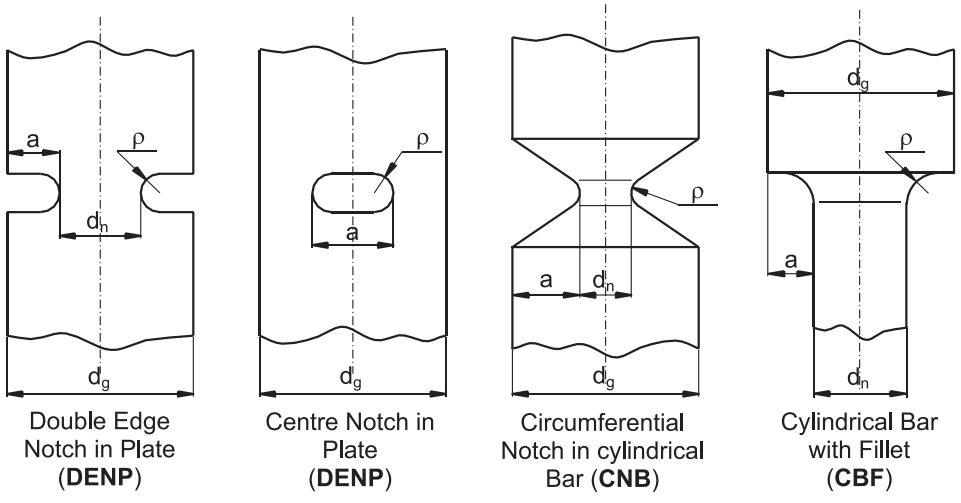

Fig. 5. Geometries of the notched specimens and definition of the adopted symbols.

To apply Scikit-Learn's MLP algorithm, the rectified linear unit function was employed as the activation function in the hidden layers. Further, the Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (which is suitable for small datasets) was used as optimization routine to train the neural network. Finally, to specify the architecture of the neural network, two hidden layers of sizes 100 and 50 neurons were employed.

In contrast, the use of the other three ML algorithms (i.e., Linear Regression, SVM and DTR) required no specific tuning, thus they were used by adopting the pre-set values for the hyperparameters.

The considerations reported above make it evident that when the Scikit-Learn ML algorithms being considered could not be employed by directly using the pre-set values, the values for the relevant hyperparameters were determined by following standard, simple optimisation procedures. However, the goal of the present work, certainly, is not to find the best values possible for the relevant hyperparameters. In contrast, this research aims at formulating and validating an ML-based strategy suitable for estimating notch fatigue limits. Here the ML process is informed through the notch fatigue knowledge gained over the last century and briefly summarised in Section 3. This aspect is very important because, given a specific ML algorithm, the optimal values for the various hyperparameters vary not only as the adopted library changes (for instance, moving from Scikit-Learn to TensorFlow or moving from SAS to MATLAB), but also as size and quality of the input dataset changes. As to the size of the input dataset, it is important to highlight that, from a fatigue viewpoint, the database summarised in Table 1 is certainly very large. However, as far as ML applications are concerned, a database containing less than 300 data is considered to be a relatively small dataset. All these key aspects should be borne in mind since they are behind the results from the analyses discussed in what follows.

Strategy 1 was trained and tested by referring to the entire population of the data summarised in Table 1 (i.e., 238 experimental results in total). In contrast, Strategy 2 was trained and tested by using solely those materials for which the experimental value of the threshold value of the stress intensity factor range, ΔK_{th}, was available (i.e., 167 experimental results in total).

Using the above two datasets, the accuracy of Strategy 1 and Strategy 2 applied along with the considered ML algorithms was assessed in terms of RMSE, Eq. (19), and the results are summarised in Table 2. The values for the average, variance, and the standard deviation (S.D.) reported in Table 2 were calculated (for any ML algorithm/strategy) from a population of 100 RMSEs coming from 100 independent testing trials. For any of these 100 testing trials, 85 % of the input experimental results was used to train the adopted ML algorithm, whereas the remaining 15 % was used as test data. The reason behind this modus operandi, which applies to any ML algorithm/strategy being investigated, is as follows. Given the input dataset, the test data are extracted randomly, with the remaining data being used to train the ML algorithm. Since every time the individual experimental results used for calibration change, the training process leads to slightly different values for the constants in the ML constitutive relationship. In parallel, as the training experimental data varies, the test data as well change. This makes it evident that any testing trial is characterised by a different set of calibration results as well as by a different set of validation data, with this obviously leading to different RMSE values.

Another important aspect associated with the training/testing approach described above is that increasing the number of testing trials would obviously result in (slightly) different values for the average, the variance, and the standard deviation of the RMSE. However, this is not an issue because the results reported in Table 2 were calculated to assess against each other the accuracy of the considered ML algorithms and not to evaluate their absolute accuracy.

In order to interpret the results summarised in Table 2 correctly, one-way analysis of variance (one-way ANOVA) was used to determine whether or not there is a statistically significant difference between the average values of the RMSE associated with the five ML algorithms being investigated. The hypothesis test was based on the following two standard hypotheses:

- Null Hypothesis (H_0): the RMSE average values associated with the five ML algorithms are equal;
- Research Hypothesis (H_a): at least one RMSE average value is significantly different.

The one-way ANOVA analyses run to test Strategy 1 and Strategy 2 returned a p-value equal to $2.5{\cdot}10^{-74}$ and to $3.6{\cdot}10^{-58}$, respectively. Since these statistical analyses were carried out by setting the α level equal to 0.05, the fact that both p-values are smaller than 0.01 means the results are highly statistically significant, so that H_0 can be rejected. This implies that at least one RMSE average value is significantly different from the other ones.

Having clarified these important aspects, Table 2 makes it evident that the most accurate results (i.e., the lowest values for the average, the variance, and the standard deviation of the RMSE) were obtained by applying DTR. Further, as far as the DTR algorithm is concerned, the use of Strategy 1 resulted in values of the average, the variance and the standard deviation of the RMSE that were slightly lower than those obtained by adopting Strategy 2. This can simply be ascribed to the fact that the population of calibration data used with Strategy 1 was more numerous that the one used with Strategy 2 (i.e., 202 vs. 142 calibration data).

5.3. Machine learning, Peterson's formula and the Theory of Critical Distances

The re-analyses discussed in the previous section suggest that, amongst the ML techniques being considered, the highest level of accuracy in estimating the fatigue strength reduction factor is reached (in relative terms) by using the DTR algorithm. Accordingly, in the present section the predictive capability of this ML technique applied along with Strategy 1, Eq. (17), and Strategy 2, Eq. (18), is assessed against the accuracy of Peterson's formula - i.e., Eq. (9) applied along with Eq. (10) – as well as of the Point Method [34,36]. To this end, two specific

Table 2
Accuracy of the considered ML algorithm in estimating the fatigue strength reduction factor based on Strategy 1, Eq. (17), and on Strategy 2, Eq. (18).

	Input Variables	**Strategy 1**			**Strategy 2**		
		σ_{UTS}, $\Delta\sigma_0$, ρ, K_t			σ_{UTS}, $\Delta\sigma_0$, **L**, ρ, K_t		
		K_f - RMSE			**K_f - RMSE**		
		Average	*Variance*	*S.D.*	*Average*	*Variance*	*S.D.*
Machine-Learning Algorithm	*Linear Regression*	1.31	0.39	0.62	1.32	0.28	0.52
	Support Vector Machines	1.42	0.46	0.68	1.65	0.62	0.78
	Gaussian Process Regression	2.63	0.31	0.56	2.97	0.64	0.80
	Cross decomposition – PLS regression	1.30	0.34	0.58	1.39	0.38	0.61
	Decision Tree Regression	1.02	0.25	0.50	1.21	0.39	0.62
	Multi-Layer Perceptron (MLP)	1.36	0.24	0.48	1.43	1.35	1.15

complete data sets were selected from the database reported in Table 1, i.e. the notch results by Nisitani and Endo [41] and by Kobayashi and Nakazawa [42] generated by testing, under rotating bending, C45 steel and C36 steel, respectively.

As formulated by Tanaka [36] and Taylor [34], Peterson's Point Method is applied by taking the critical distance (i.e., distance D_{PM} in Fig. 4b) equal to L/2, where material length L is estimated via definition (13). By so doing, a notched component is assumed to be in the fatigue limit condition when the range of the linear-elastic stress at a distance of L/2 from the apex of the assessed notch is equal to the plain material fatigue limit, $\Delta\sigma_0$. The local linear-elastic stress fields needed to apply the Point Method to estimate the selected notch results were determined from simple bi-dimensional linear-elastic models solved using commercial FE code Ansys®. In these models, the mesh in the notch region was gradually refined until convergence occurred.

To train the DTR algorithm, the experimental results referring to the notched specimens of C45 steel [41] and C36 steel [42] were removed from the dataset of Table 1. Since to apply Strategy 2, Eq. (18), the range of the threshold value of stress intensity factor range, ΔK_{th}, was required, clearly the population of training data used with this approach was markedly less numerous than the population of data used to train the DTR algorithm applied along with Strategy 1, Eq. (17).

The accuracy of the considered four approaches in estimating the results generated by testing notched specimens of C45 steel [41] and C36 steel [42] were quantified by defining the error, E, as follows:

$$E\,[\%] = \frac{\Delta\sigma_{0n} - \Delta\sigma_{0n,est}}{\Delta\sigma_{0n,est}} \bullet 100 \tag{20}$$

where $\Delta\sigma_{0n}$ and $\Delta\sigma_{0n,est}$ are the experimental and the estimated value of the notch fatigue limit, respectively.

The results of this accuracy assessment exercise are summarised in the $\Delta\sigma_{0n}$ vs. $\Delta\sigma_{0n,est}$ chart of Fig. 6 as well as in Table 3. Fig. 6 makes it evident that the use of the DTR algorithm applied along with Strategy 1, Peterson's formula and the Point Method resulted in a similar level of accuracy. This finding is further confirmed by Table 3 since the use of these three notch fatigue design strategies resulted in very similar values of both the average and standard deviation of error E calculated according to Eq. (20). In contrast, the estimates obtained by applying the DTR algorithm along with Strategy 2 were characterised by a larger level of scattering. Even if more data are needed to confirm this, this may be ascribed to the fact that the dataset used to train the ML algorithm contained a limited number of experimental results.

Turning back to Table 3 and Fig. 6, it is interesting to point out again that the use of the DTR algorithm applied along with Strategy 1, Peterson's formula and the Point Method returned the same level of accuracy. Both Peterson's method and the Theory of Critical Distances are approaches that are already used in industry on a daily bases. Thus, the fact the DTR-Strategy 1 method is characterised by the same level of accuracy as the other two standard methodologies suggests the ML-based philosophy proposed in the present paper can safely be used to address fatigue design problems of practical interest. This being said, the advantage of the ML-based design method over the other existing standard approaches is the intrinsic flexibility of ML algorithms. For instance, the accuracy in estimating notch fatigue limits can easily be increased by simply increasing size and quality of the dataset used to calibrate the ML algorithms being employed. Additionally, because ML regression models can handle high-dimensional datasets with numerous input features, they are able to capture complex correlations between

Table 3
Accuracy in predicting the notch fatigue strength of C45 [41] and C36 [42].

Approach	$\Delta\sigma_{0n}$ - E *[%]*	
	Average	*S.D.*
DTR - Strategy 1	0.4	18.4
DTR - Strategy 2	−4.4	35.4
Peterson's formula	5.2	18.7
Point Method	−5.2	11.1

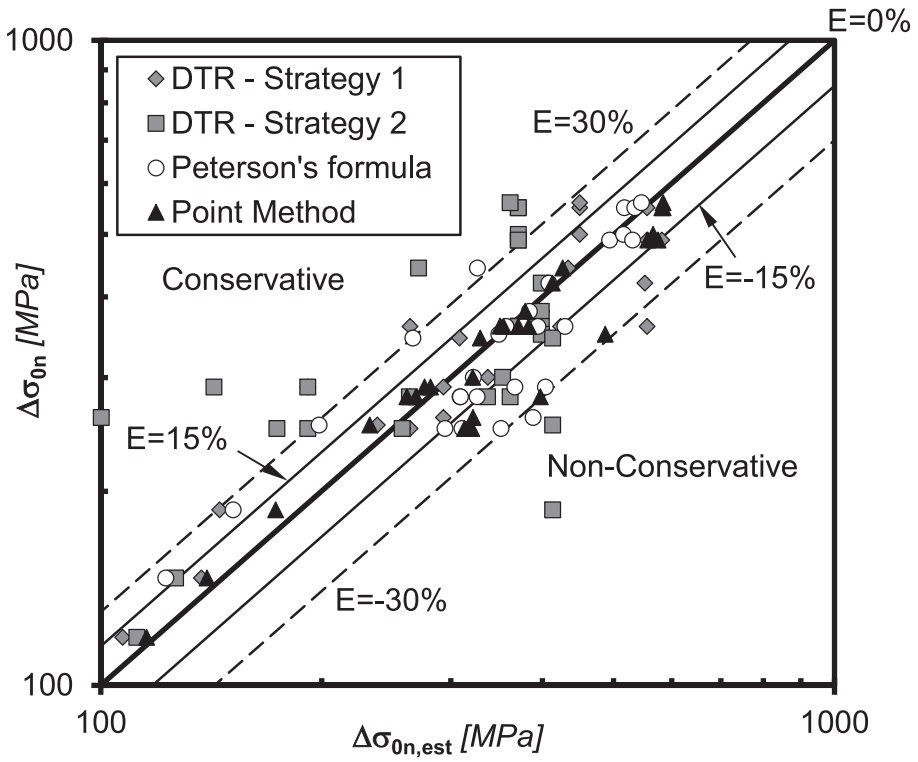

Fig. 6. Accuracy of DTR – Strategy 1, DTR – Strategy 2, Peterson's formula - - Eq. (9) applied along with Eq. (10) – as well as of the Point Method [34,36] in estimating the notch fatigue limits of C45 steel [41] and C36 steel [42].

input features and the target variable. This indicates that, by incorporating additional material property-related input variables in the ML analyses, the accuracy in estimating the notch fatigue limit can be improved markedly.

6. Conclusions

This investigation deals with the accuracy and reliability in estimating notch fatigue limits of a number of popular ML algorithms. The analyses discussed in the present paper were based on a large number of experimental results generated by testing under uniaxial loading notched specimens of various metallic materials, aluminium alloys and titanium alloys. These experimental results were all collected from the technical literature and then organised in a coherent database.

The strategies being proposed to use ML to assess notch high-cycle fatigue strength were informed by learning from a number of classic formulas specifically devised to estimate the fatigue strength reduction factor. Based on the findings reported in the present paper, as long as ML is applied in conjunction with the classic nominal stress approach, the most relevant conclusions are summarised in what follows.

- If K_f is the target, σ_{UTS}, $\Delta\sigma_0$, ΔK_{th}, K_t and the notch root radius, ρ, can be used as features to train standard ML algorithms.
- Amongst the considered ML algorithms, the highest level of accuracy (in relative terms) in estimating K_f was obtained by using the DTR algorithm.
- Amongst a number of other factors, the accuracy of a ML algorithm also depends on the size of the training dataset. This explains why the use of DTR – Strategy 1 resulted in more accurate estimates than those obtained via DTR – Strategy 2.
- The use of the DTR algorithm applied along with Strategy 1 was seen to lead to a level of accuracy similar to the one obtained by applying both Peterson's formula and the Point Method.
- More experimental work needs to be done in order to create a large database of notch fatigue results that can be used to train ML algorithms.

CRediT authorship contribution statement

Luca Susmel: Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Miller KJ. The two thresholds of fatigue behaviour. Fatigue Fract Eng Mater Struct 1993;16:931–9.
[2] Akiniwa Y, Tanaka K, Kimura H. Microstructural effects on crack closure and propagation thresholds of small fatigue cracks. Fatigue Fract Eng Mater Struct 2001;24:817–29.
[3] Miller KJ, O'Donnell WJ. The fatigue limit and its elimination. Fatigue Fract Engng Mater Struct 1999;22:545–57.
[4] Sonsino CM. Course of SN-curves especially in the high-cycle fatigue regime with regard to component design and safety. Int J Fatigue 2007;29(12):2246–58.
[5] Susmel L. Multiaxial Notch Fatigue: from nominal to local stress-strain quantities. Woodhead & CRC, Cambridge, UK, ISBN: 1 84569 582 8, 2009.
[6] Lee Y-L, Pan J, Hathaway RB, Barkey ME. Fatigue Testing and Analysis. Elsevier Butterworth-Heinemann; 2005.
[7] Ma X, He X, Tu ZC. Prediction of fatigue–crack growth with neural network-based increment learning scheme. Eng Fract Mech 2021;241:107402.
[8] Bartošák M. Using machine learning to predict lifetime under isothermal low-cycle fatigue and thermo-mechanical fatigue loading. Int J Fatigue 2022;163:107067.
[9] Hao WQ, Tan L, Yang XG, Shi DQ, Wang ML, Miao GL, et al. A physics-informed machine learning approach for notch fatigue evaluation of alloys used in aerospace. Int J Fatigue 2023;170:107536.
[10] Lian Z, Li M, Lu W. Fatigue life prediction of aluminum alloy via knowledge-based machine learning. Int J Fatigue 2022;157:106716.
[11] Horñas J, et al. Modelling fatigue life prediction of additively manufactured Ti-6Al-4V samples using machine learning approach. Int J Fatigue 2023;169:107483.
[12] Ciampaglia A, Tridello A, Paolino DS, Berto F. Data driven method for predicting the effect of process parameters on the fatigue response of additive manufactured AlSi10Mg parts. Int J Fatigue 2023;170:107500.
[13] Maleki E, Bagherifard S, Razavi N, Bandini M, du Plessis A, Berto F, et al. On the efficiency of machine learning for fatigue assessment of post-processed additively manufactured AlSi10Mg. Int J Fatigue 2022;160:106841.
[14] Bezazi A, Pierce SG, Worden K, Harkati EH. Fatigue life prediction of sandwich composite materials under flexural tests using a Bayesian trained artificial neural network. Int J Fatigue 2007;29(4):738–47.
[15] Fernández J, et al. Uncertainty quantification in Neural Networks by Approximate Bayesian Computation: Application to fatigue in composite materials. Eng Appl Artif Intell 2022;107:104511.
[16] Ben-Yelun I, et al. Self-learning locally-optimal hypertuning using maximum entropy, and comparison of machine learning approaches for estimating fatigue life in composite materials of the aerospace industry. Eng Struct 2023;283:115829.
[17] Matloff N. Statistical Regression and Classification - From Linear Models to Machine Learning. Taylor & Francis Group, Florida, USA: CRC Press; 2017.
[18] Campbell C, Ying Y. Learning with Support Vector Machines. Switzerland: Springer Nature; 2022.
[19] Rasmussen CE, Williams CKI. Gaussian Processes for Machine Learning. Cambridge, MA, USA: MIT Press; 2006.
[20] Wegelin JA. A survey of Partial Least Squares (PLS) methods, with emphasis on the two-block case. Department of Statistics, University of Washington, Seattle, Washington, 98195 USA, Technical Report No. 371, 2000.
[21] Rokach L, Maimon O. Data mining with decision trees – Theory and Applications. Singapore: World Scientific Publishing Co.; 2008.
[22] Marsland S. Machine Learning, 2nd Edition, CRC Press. Florida, USA: Taylor & Francis Group; 2015.
[23] Raschka S, Liu Y, Mirjalili V. Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python. Birmingham, UK: Pack Publishing; 2022.
[24] Glinka G, Newport A. Universal features of elastic notch-tip stress fields. Int J Fatigue 1987;9(3):143–50.
[25] Neuber H. Kerbspannungslehre. Berlin Heildelberg, Germany: Springer Verlag; 1958.
[26] Peterson RE. Stress Concentration Design Factors. New York, USA: John Wiley; 1953.
[27] Peterson RE. Notch sensitivity. In Metal Fatigue, Eds: G. Sines and J. L. Waisman, McGraw-Hill, New York, 1959, pp. 293-306.
[28] Weixing Y, Kaiquan X, Yi G. On the fatigue notch factor. K_f Int J Fatigue 1995;17(4):245–51.
[29] Ciavarella M, Meneghetti G. On fatigue limit in the presence of notches: classical vs. recent unified formulations. Int J Fatigue 2004;26(3):289–98.
[30] Kuhn P, Hardrath HF. An engineering method for estimating notch-size effect in fatigue tests on steel. NASA Tech Note 2805, 1952.
[31] Dowling NE. In: Mechanical behavior of materials. 2nd edition. New Jersey, USA: Prentice-Hall Inc; 1998.
[32] Heywood RB. Designing against fatigue. London, UK: Chapman & Hall; 1962.
[33] El-Haddad MH, Topper TH, Smith KN. Prediction of non-propagating cracks. Eng Fract Mech 1979;11:573–84.
[34] Taylor D. Geometrical effects in fatigue: a unifying theoretical model. Int J Fatigue 1999;21(5):413–20.
[35] DuQuesnay DL, Topper TH, Yu MT. The Effect of Notch Radius on the Fatigue Notch Factor and the Propagation of Short Cracks. In: The Behaviour of Short Fatigue Cracks, EGF Pub. 1 (Edited by K. J. Miller and E. R. de los Rios, 1986, Mechanical Engineering Pubblications, London, pp. 323-335.
[36] Tanaka K. Engineering formulae for fatigue strength reduction due to crack-like notches. Int J Fract 1983;22:R39–45.
[37] Atzori B, Lazzarin P, Tovo R. Evaluation of the fatigue strength of a deep drawing steel. Österreichische Ingenieur-und Architekten-Zeitschrift, Jg 1992;137:556–61.
[38] Lazzarin P, Tovo R, Meneghetti G. Fatigue crack initiation and propagation phases near notches in metals with low notch sensitivity. Int J Fatigue 1997;19(8–9):647–57.
[39] Lukáš P, Klesnil M. Fatigue limit of notched bodies. Mater Sci Eng 1978;34:61–9.
[40] Atzori B, Meneghetti G, Susmel L. Fatigue Behaviour of AA356-T6 cast aluminium alloy weakened by cracks and notches. Eng Fract Mech 2004;71(4–6):759–68.
[41] Nisitani H, Endo M. Unified treatment of Deep and Shallow Notches in Rotating Bending. In: Basic questions in fatigue: Vol. 1, ASTM STP 924, Edited by J. T. Fong and R. J. Fields, pp. 136-153, 1988.
[42] Kobayashi H, Nakazawa H. The Effects of Notch Depth on the Initiation Propagation and Non-propagation of Fatigue Cracks. Transactions of the Japan Society of Mechanical Engineers 1969;35(277):1856–63.
[43] Susmel L, Taylor D. Fatigue Design in the Presence of Stress Concentrations. J Strain Anal Eng 2003;38(5):443–52.
[44] Tanaka K, Nakai Y. Propagation and non-propagation of short fatigue cracks at a sharp notch. Fatigue Fract Engng Mater Struct 1983;6:315–27.

[45] Tanaka K, Akiniwa Y. Notch geometry effect on propagation threshold of short fatigue cracks in notched components. In: Fatigue '87, Vol. II, Edited by R. O. Ritchie and E. A. Starke Jr., 3th Int. Conf. On Fatigue and Fracture Thresholds, pp. 739-748, 1987.
[46] Frost NE. A relation between the critical alternating propagation stress and crack length for mild steel. Proc Inst Mech Engrs 1957;173:811–34.
[47] Frost, N. E. Non-propagating cracks in V-notched specimens subjected to fatigue loading. Aeoronaut Quart 1957;VIII:1-20.
[48] Lukáš P, Kunz L, Weiss B, Stickler R. Non-damaging notches in fatigue. Fatigue Fract Engng Mater Struct 1986;9:195–204.
[49] El Haddad MH. A study of the growth of short fatigue cracks based on fracture mechanics. Waterloo, Ontario, Canada: University of Waterloo; 1978. PhD Thesis.
[50] Taylor D, Hughest M, Allen D. Notch fatigue behaviour in cast irons explained using a fracture mechanics approach. Int J Fatigue 1996;18:439–45.
[51] Grover HJ, Bishop SM, Jackson LR. Axial-Load fatigue tests on notched sheet specimens of 24S-T3 and 75S-T6 aluminum alloys and of SAE 4130 steel with stress-concentration factors of 2.0 and 4.0. NACA report TN 2389, 1951.
[52] Grover HJ, Bishop SM, Jackson LR. Fatigue strengths of aircraft materials axial load fatigue tests on unnotched sheet specimens of 24S-T3 and 75S-T6 aluminum alloys and of SAE 4130 steel. NACA report TN 2324, 1951.
[53] Usami S. Short crack fatigue properties and component life estimation. In: Tanaka T, Jono M, Komai K, editors. Current research on fatigue cracks. Amsterdam: Elsevier; 1987. p. 119–47.
[54] Landers CB, Hardrath HF. Results of axial-load fatigue tests on electropolished 2024–T3 and 7075–T6 aluminum-alloy-sheet specimens with central holes. NACA report TN 3631, 1956.
[55] Gough HJ. Engineering steels under combined cyclic and static stresses. Proc Inst Mech Engrs 1949;160:417–40.
[56] Quilafku G, Kadi N, Dobranski J, Azari Z, Gjonaj M, Pluvinage G. Fatigue specimens subjected to combined loading. Role of hydrostatic pressure. Int J Fatigue 2001;23:689–701.
[57] Atzori B, Berto F, Lazzarin P, Quaresimin M. Multi-axial fatigue behaviour of a severely notched carbon steel. Int J Fatigue 2006;28:485–93.
[58] Susmel L, Taylor D. The Modified Wöhler Curve Method applied along with the Theory of Critical Distances to estimate finite life of notched components subjected to complex multiaxial loading paths. Fatigue Fract Eng Mater Struct 2008;31(12):1047–64.
[59] Kurath P, Downing SD, Galliart D. R. Summary of non-hardened notched shaft round robin program. In: Multiaxial Fatigue (edited by G. E. Leese and D. F. Socie), AE-14, Society of Automotive Engineers, pp. 13–32, 1989.
[60] Berto F, Lazzarin P, Yates JR. Multiaxial fatigue of V-notched steel specimens: a non-conventional application of the local energy method. Fatigue Fract Engng Mater Struct 2011;34:921–43.
[61] Berto F, Lazzarin P. Fatigue strength of structural components under multi-axial loading in terms of local energy density averaged on a control volume. Int J Fatigue 2011;33:1055–65.
[62] Tovo R, Lazzarin P, Berto F, Cova M, Maggiolini E. Experimental investigation of the multiaxial fatigue strength of ductile cast iron. Theor Appl Fract Mech 2014;73:60–7.
[63] Susmel L, Taylor D. A critical distance/plane method to estimate finite life of notched components under variable amplitude uniaxial/multiaxial fatigue loading. Int J Fatigue 2012;38:7–24.
[64] Namiq ZF, Susmel L. Proportional/non-proportional constant/variable amplitude multiaxial notch fatigue: cyclic plasticity, non-zero mean stresses, and critical distance/plane. Fatigue Fract Eng Mater Struct 2019;42(9):1849–73.
[65] Pals TG, Stephens RI. The influence of high R ratio on mild and sharp notched and unnotched fatigue behavior of 1045 steel with three different heat treatments. Int J Fatigue 2004;26:651–61.
[66] Lanning DB, Haritos GK, Nicholas T. Influence of stress state on high cycle fatigue of notched Ti–6Al–4V specimens. Int J Fatigue 1999;21:S87–95.
[67] Lanning DB, Nicholas T, Haritos GK. On the use of critical distance theories for the prediction of the high cycle fatigue limit stress in notched Ti–6Al–4V. Int J Fatigue 2005;27:45–57.
[68] Chiandussi G, Rossetto M. Evaluation of the fatigue strength of notched specimens by the point and line methods with high stress ratios. Int J Fatigue 2005;27:639–50.
[69] Qylafku G, Azari Z, Kadi N, Gjonaj M, Pluvinage G. Application of a new model proposal for fatigue life prediction on notches and key-seats. Int J Fatigue 1999;21:753–60.
[70] Yamashita Y, Ueda Y, Kuroki H, Shinozaki M. Fatigue life prediction of small notched Ti–6Al–4V specimens using critical distance. Eng Fract Mech 2010;77:1439–53.
[71] Wanga J, Yang X. HCF strength estimation of notched Ti–6Al–4V specimens considering the critical distance size effect. Int J Fatigue 2012;40:97–104.
[72] Atzori B, Lazzarin P, Meneghetti G. Estimation of fatigue limits of sharply notched components. In: Proc. of Fatigue 2006, Atlanta, Georgia, USA, 2006.
[73] Kihara S, Yoshii A. A strength evaluation method of a sharply notched structure by a new parameter, the equivalent stress intensity factor. JSME 1991;34:70–5.
[74] Yu MT, DuQuesnay DL, Topper TH. Notched fatigue behaviours of two cold rolled steels. Fatigue Fract Engng Mater Struct 1991;14(1):89–101.
[75] Pilkey WD, Pilkey DF, Bi Z. Peterson's stress concentration factors. 4th Edition. John Wiley & Sons Inc; 2020.