Reverse engineering of software code for fatigue life estimation of components using machine learning algorithms

Ganesh Raj Ramakrishnan (604303), Sriyesh Chelladurai (604420), Murugan Palaniappan (604503), Suriya Prakash Gnana Prakash (604581)

Clausthal University of Technology

Abstract

Fatigue life estimation is crucial for evaluating mechanical components that experience cyclic loading. In industrial settings, these estimations often rely on custom software created in-house. In this case, the software used for fatigue life calculation is functional but lacks source code and documentation because of lost expert knowledge. The goal of this study is to reverse engineer the fatigue life estimation process of this existing executable software by two different algorithm and to compare them. This process includes analysing the input-output relationship, modelling the fatigue prediction logic using machine learning, identifying the limits of the current model, and confirming the accuracy of the reconstructed model.

Keywords: Latin Hypercube Sampling (LHS), extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM)

Introduction

The assessment of fatigue life is crucial to ensure the structural integrity and reliability of components subjected to cycle loading. Fatigue failure occurs under repeated subcritical stresses and is a major concern in the design and maintenance of various industrial systems.

Traditional methods for fatigue life estimation rely on analytical models or finite element simulations calibrated against experimental S-N curves (stress life). These approaches demand expect knowledge, extensive computational resources and access to several software's in an experimental data. Here we are using back-box executable whose internal logic is no longer possible. This a challenge to validate, replicate and extend the system.

This research project addresses this challenge by reverse engineering a given fatigue life prediction in an executable file. The software accepts five parameters which includes tensile strength (R_m), stress concentration(K_t), stress gradient(R), surface roughness(R_z) and nominal stress amplitude (). Which returns the fatigue life as output. Since the source code and documentation is unavailable, this study aims to inspect the input-output relationship, which needed to be compared with the software and train a surrogate model that approximates the behaviour of the original black-box model.

To achieve this, we generated a large dataset of over 900000 datasets using Latin hypercube sampling (LHS) to explore input parameters efficiency and to execute sample set across the parameter range. We then trained a random forest regressor (RFR) to replicate the fatigue life predictions. The goal is to develop a transparent and interactable surrogate model which meets an error tolerance of 2% less compared to origin /given software predictions.

Furthermore, this paper presents the methodology, validation strategy and evaluation results of the surrogate model. It also discusses the correlation of the parameters which reflects the influence in output and the effectiveness of data-driven models for fatigue predictions in absence of full theoretical formulations.

Literature Review

From the recent research in fatigue modelling by Marković et al.The approximation in the behaviour of executable software in a black-box situation has been trained on finite element simulation data to predict fatigue life in steel components, their work showed that ANNs can mimic high-fidelity simulations at a lower computational cost[1]. Further the evaluation of the varied input combinations in the target software tool by Steck et al. focused on multiaxial and non-proportional load conditions in eBike drive unit components by building a surrogate fatigue model. Their method highlights the significance of multivariable analysis and real-world loading patterns [2].Alkunte and Fidan applied machine learning models—Random Forest (RF), Support Vector Machines (SVM), and Artificial Neural Networks (ANN)—to predict the fatigue life of functionally graded materials made through material extrusion. Their comparison revealed RF as the most precise and efficient model, particularly for empirical datasets with high variance, making it suitable for black-box prediction tasks [3]. To validate the output distribution and confidence levels of a reconstructed fatigue model presented by Faghihi et al. which is a probabilistic design framework for fatigue life prediction that combines experimental data, continuum damage mechanics (CDM), and Bayesian calibration. Their model addresses uncertainties in data and model structure, providing a probabilistic perspective on fatigue estimation [4]. Machine learning has also been used in reverse engineering within materials science. Ji et al. built a framework for mold flux design using ensemble machine learning models—like Light, XGBoost, and Random Forest—to predict initial crystallization temperature and reverse engineer compositions based on target properties. This method is relevant to this project as it helps understand the influence of input variables and perform interpretable black-box modelling [5]. In fatigue modelling, Random Forest algorithms have proven effective for nonlinear, multivariate regression issues. Nagashima et al. demonstrated the use of RF to estimate S–N curves using the NIMS fatigue database. Their RF model surpassed traditional regression techniques in estimating fatigue limits and life under high-cycle fatigue, suggesting its capability to reproduce fatigue life behaviour from legacy software outputs [6].Chu et al. proposed a reliability-based optimization method using surrogate models built through Latin Hypercube Sampling (LHS) and metaheuristic algorithms such as genetic algorithms and simulated annealing. While mainly focused on structural optimization, the methodology for surrogate modelling and uncertainty quantification

discussed in this paper can be applied to reverse engineering fatigue models, especially to minimize model error and computational costs [7]. This research by Prasad and Ramesh proposed an ANN-based fatigue life prediction model specifically for connecting rods. Their study highlights the use of ANN for automotive components and shows how accurate fatigue life estimates can be derived from input stress conditions. It supports employing component-specific neural networks for fatigue analysis, particularly for recreating software behaviour for known parts [8]. The studies mentioned above collectively offer important theoretical and practical frameworks that support the reverse engineering of executable fatigue life software. These frameworks include data-driven surrogate modelling, sensitivity analysis for variable influence, probabilistic calibration, and regression using ensemble learning—all of which form the basis for the current research.

Methodology:

The methodology proposed and implemented to reverse engineer this black box application.

- Data Generation (LHS method)
    - Data Preprocess
    - Data Postprocess
- ML model Creation
    - Random Forest
    - XGBoost, LightGBM
- Model Evaluation
    - Evaluation matrix

This end-end workflow holds both technical performance and practical applications. These binds a smooth workflow from data-generation to minimal error deployment.

Data Generation:

To train the surrogate model effectively and capture the correlation on both input and fatigue life output, we have implemented Latin Hypercube Sampling (LHS) to generate a distributed dataset. LHS is a stratified sampling technique that ensures a uniform coverage of the input space which makes highly suitable for capturing variability in higher dimensioning models. The parameters along with their symbols, ranges and units are summarized below.

Table 1: This is a table shows the input parameters ranges and the corresponding units

Parameters	Symbols	Range	Unit
Tensile Strength	R_m	250 – 2000	MPa
Stress Gradient	G	0.1 - 3	1/mm
Stress Concentration	K_t	1 – 5	-
Surface Roughness	R_z	1 - 30	müm
Nominal Stress Amplitude	Sig_a	100 - 2200	Mpa

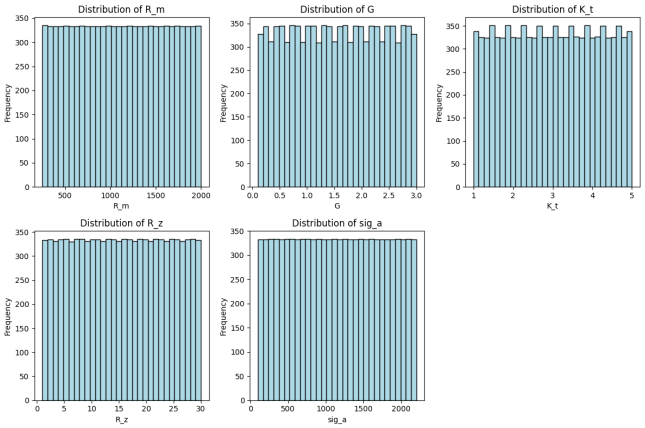

Figure 1: Distribution of generated LHS samples across all the 5 inputs

The sampling bounds on a realistic physical range. A total of 9000000 samples were generated using LHS, with step sizes tailored to the sensitivity of each variable (0.1-Mpa for stress,0.01 for surface roughness and concentration factors). This results in a comprehensive dataset to explore both high and low fatigue value. Each of the generated samples was processed through an executable file which runs as a black box. The executable accepts the five parameters as inputs and returns a predicted fatigue life value, N which represents the number of cycles to failure.

Upon processing, it was observed that a subset of the outputs resulted in non-physical values (0's &∞'s). These values are technically valid responses from the black box (indicating immediate failure or infinite life) are not informative for regression training. This filtered dataset served as a foundation for training the random forest surrogate model, which aims to replicate the behaviour of black-box executable with high efficiency and reduced computational cost.

Sample set is generated by Latin Hyper Cube Sampling in a random. Providing a huge dataset, which extends the minimization of error in the machine learning algorithm. It enhances the relationship between the material parameters of the component and the fatigue life of it. The influence of each parameter is different with the fatigue life which needs to be determined.

Data pre-processing

Further data generation, the raw fatigue life outputs fell into three sets

- Set 1: Zero fatigue life (interpreted as failure).
- Set 2: Infinite fatigue life (Non failure values)
- Set 2: Finite, valid fatigue life cycles

The finite, valid fatigue life cycles and parameters were log-transformed to improve regression performance and normalize distributions.

Data post processing

After processing the LHS data set and differentiated as three set, the corelation between parameters and fatigue life cycle is analysed using S-N curve.21The plotted S-N curve illustrates how fatigue life (N) varies with stress amplitude (σ_a), tensile strength (R_m), and stress concentration factor (K_t). Higher tensile strength (R_m) materials (shown in red) consistently exhibit longer fatigue life at lower stress levels, indicating better resistance to fatigue failure. In contrast,

lower R_m materials (blue) require higher stress to achieve the same life span, making them more prone to early failure. Within each R_m group, increasing K_t shifts the curves upward, highlighting the detrimental effect of geometric stress concentrations on fatigue performance. The shaded black region ($N \approx 0$) marks immediate failures, while the green zone ($N \to \infty$) represents fatigue-safe regions. Overall, the plot confirms that both material strength and geometric design critically affect fatigue resistance.

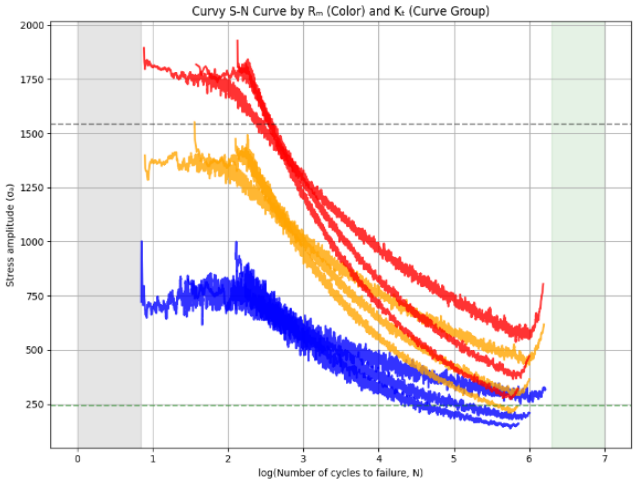

Figure 2: S-N curves plotted based on the combinations of tensile strength (R_m) and stress concentration factor (K_t), grouped by colour and line style. The curves represent fatigue life behaviour with stress amplitude (σ_a) on the y-axis and the logarithm of number of cycles to failure (log N) on the x-axis. The data is categorized into nine groups: Low, Mid, and High R_m values (indicated by colour: blue, orange, and red respectively), each further subdivided by K_t levels (Low, Mid, High) to distinguish curve sets. The shaded regions indicate fatigue life thresholds—grey for **zero fatigue** ($N < 7.1$) and green for **infinite fatigue** ($N > 2.0\times10^6$). Dashed lines mark the average stress amplitudes corresponding to zero fatigue (**σ_a = 1543.8**) and infinite fatigue (**σ_a = 243.7**), offering a visual reference for transition zones across fatigue regimes.

Model creation

The processed datasets were used to train and evaluate two different machine learning algorithms, with the goal of identifying the most effective surrogate model. Among them, the Random Forest algorithm was a key approach due to its robustness and interpretability.

Approach towards Random Forest

Random Forest is an ensemble learning method for classification and regression. It constructs multiple decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. To build our surrogate fatigue-life predictor, we first generated a dataset of N input–output pairs by sampling the five design parameters (material constants, geometry factors, load amplitude, mean stress, and stress ratio) using a Latin Hypercube scheme. Each sampled vector X_i as fed into the original fatigue-life executable to obtain the ground-truth life y_i. After assembling we performed an 80/20 split into training and hold-out test sets.

We then instantiated scikit-learn's Random Forest Regressor with a baseline configuration (200 trees, max_features=$\sqrt{p}$,min_sample_leaf = 5,out-of-bag scoring enabled, fixed random seed) and conducted a hyperparameter optimization over{n_estimators, max_depth, min_samples_leaf, max_features}, using 5-fold cross-validation (and OOB error) to minimize the **root-mean-square error (RMSE)**. Once the best parameter set was identified, we retrained the regressor on the full 80% training portion via

rf.fit(X_ {\rm train}, y_ {\rm train})

Finally, we assessed predictive performance on the 20% hold-out set by computing RMSE and mean absolute percentage error (MAPE) and confirmed that the out-of-bag R^2 closely matched our test-set score. This process ensured our Random Forest model reliably replicates the original fatigue-life calculations within the prescribed error tolerance.

For our reverse-engineering application—where we're classifying fatigue-failure modes based on input features—which is the Random Forest implementation in scikit-learn uses **Gini impurity** as its default node-splitting criterion. In other words, at each tree node it chooses the split that maximally decreases $G(t) = 1 - k = 1\sum^K pk \mid t2$

(where $pk \mid t$ *the fraction of samples of class* k in node t).

Once all the trees are grown, predictions on new samples are made by **majority vote** across the ensemble:

$$Y = \text{mode}\{y^{(1)},\ldots,y^{(M)}\}$$

The Random Forest algorithm is followed by the evaluation of the grid search systematically for each combination. The search method is used to predict the solution in a optimistic way. The problem towards false detection can be minimized by hyperparameter tuning from an ad-hoc trial-and-error into a rigorous, data-driven process that guarantees our Random Forest surrogate for the fatigue-life executable is as accurate and generalizable as possible. The regressor model which is trained by the Random Forest Algorithm is used to determine the finite(numeric)values. It is predicted from the exe file and is trained by the finite values to provide more accurate results. But to train the classifier, where it fails to divide the non-defined values, where this work has been done by the XGBoost and LightGBM methods.

Approach towards XGBoost and LightGBM

XGBoost and LightGBM method are main types of powerful Gradient Boosting Machines which uses smarter tree building algorithm – relationship development in an iterative manner where all the bad relations are penalised using a gradient function. This gradient boosting methods usually takes less iterations, but the downside of this method is that it requires

high computation and overfitting of a parameter will have a severe effect on the error rate of the model

In this approach XGBoost and LightGBM is selected as both has a similar approach, and it is easier to integrate both the model to come up with meaningful end ML model. For this model both the approaches are used trained utilised parallelly and in a smarter way with lesser computation time and reduced overfitting.

These two models are further divided into 4 separate models namely

- XGBoost Classifier
- LightGBM Classifier
- XGBoost Regressor
- LightGBM Regressor

Classifier has the function of deciding whether the given input will yield a zero, finite value or 'inf'

Regressor takes care of predicting the finite values after the classifier task

Data Segmentation of processed datasets generated from the LHS method is done by separated into little chunks of 50000 datasets and saved in separate csv files for easier training of the models through incremental training

Incremental Training is used for the initial training of the models as it requires lesser commutation time, and this feature will help in robust gradient training and much broader tree relationships. This training is done to each sub models. As per their role in this ML model the Classifiers are trained for the entire datasets but the regressor are trained only for the set 2 - finite value datasets

Intermediate Evaluation is done after this training to check for the implementation of the incremental training which yields in a much distant results due to the possible duplication of leaflets and branches in the relationship matrices created with in both the methods. This step is used to evaluate the required optimal hyperparameters values for the model improvement. In this confusion matrix is used to evaluate the performance of the model.

Hyperparameter Optimization is the hyperparameter tuning phase utilizes the Optima framework to automatically optimize parameters for two algorithms, **XGBoost** and **LightGBM**, both for classification and regression tasks. In this the data chunks are sampled for 20% of their data is used for the hyperparameter tuning. Then polynomial features (degree=2, interaction-only) applied to enhance model complexity and predictive capabilities which is a method of **Feature Engineering**. In addition to that Stratified K-fold is used to ensure balanced representation of each fatigue class.

Hyperparameter Training is done with the defined hyperparameters through the similar method of incremental Training in much smarter way – with more depth per branch and much more meaningful gradients

Binary Cascade is a step where the classifier of both XGBoost and LightGBM is allowed to interact with each other to accomplish the task of the classifier. Post Hyperparameter training the classifier has a higher value in the confusion matrix during the identification of the type of result the inputs will yield. This binary cascade helped the classifier to accomplish the function by a bilevel verification through XGBoost and LightGBM classifiers, this resulted in reduced error in the classifier.

Hyperparameter Training is done again to improve the branch quality after the binary cascade implementation.

The resulted ML models compensated each weakness of both XGBoost and LightGBM which resulted in easier integration

Optimized Parameters:

- Number of estimators
- Maximum depth
- Learning rate
- Subsample fraction
- Column subsample fraction

Table 2: This is a table shows the optimised parameters

Hyperparameter	XGBoost Range	LightGBM Range
N_estimators	100 - 600	100 - 600
Max_depth	3 - 10	-
Num leaves	-	31 - 256
Learning rate	0.001 - 0.3	0.001 - 0.3
Subsample	0.5 - 1.0	0.5 - 1.0

Evaluation matrix

Mean Absolute percentage Error (MAPE):

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - p_i}{y_i}\right|$$

Model accuracy is assessed using **Mean Absolute Percentage Error (MAPE)** as the evaluation metric. To further enhance prediction accuracy and reduce error, **hyperparameter tuning** techniques were applied to optimize the model settings (e.g., tree depth, number of estimators, and feature sampling strategy). It quantifies the relative deviation between predicted and actual values, making it particularly valuable where percentage-based accuracy is more informative than absolute differences.

These performance metrics are then calculated using a test set containing 10000 dataset values from our LHS dataset comparing to the executable. The results collectively demonstrate the robustness and generalization of random forest and XGboost regression models, connecting both classification and regression pipeline and predicting the complex patterns associated with fatigue behaviour.

Observations

In our fatigue life prediction study, we examined the influence of key mechanical and geometric factors—stress amplitude, tensile strength, mean stress, notch sensitivity, and surface roughness—on component durability under cyclic loading. As expected, materials with higher tensile strength exhibited extended fatigue life by resisting crack initiation, whereas pronounced stress concentrators (high K_t or steep stress gradients, G) accelerated crack propagation and reduced service life. The role of mean stress proved directional: tensile mean stresses curtailed life, while compressive mean stresses imparted a modest benefit. Similarly, increased surface roughness acted as an array of micro-notches, undermining fatigue resistance.

To capture relationships spanning several orders of magnitude, we transformed stress amplitude, tensile strength, and life (N) onto a log–log scale. In doing so, we straightened the classical S–N curve and simplified our regression framework. However, this approach introduced clear challenges:

- **Zero/infinite lives**. Data points at or beyond the endurance limit (infinite life) and at zero life could not be log-transformed, producing gaps in our model's domain.
- **Endurance-limit sensitivity**. Near the fatigue limit, minor parameter fluctuations caused disproportionately large shifts in predicted life, inflating variance and destabilizing regression.
- **Data imbalance**. Sparse sampling in the transition zone between finite and infinite life biased the model toward over- or underestimation in critical regimes.

To mitigate these issues, we implemented a two-stage surrogate model. A classifier first partitioned cases into "zero," "finite," or "infinite" life classes; then a regressor modelled the finite-life subset. This architecture eliminated invalid log—operations and focused the regression on the physically meaningful domain. Within the finite-life cohort, logarithmic scaling reduced dynamic range and suppressed outlier effects, while Z-score normalization balanced feature magnitudes, improving convergence stability.

Further preprocessing—namely, the removal of physically unrealistic or corrupted data entries—sharpened model focus and curtailed the undue influence of extreme values. In benchmarking experiments, ensemble learners (Random Forest, XGBoost) outperformed single model regressors, capturing nonlinear interactions among stress, strength, geometry, and surface condition with greater fidelity.

Overall, our study underscores three key lessons for data-driven fatigue analysis of black-box systems:

1. **Thoughtful transformation** (log–log scaling with careful handling of infinities) is critical to linearize known physical trends without introducing mathematical artefacts.
2. **Hybrid modelling strategies** (classification plus regression) can gracefully accommodate discontinuities in material behaviour (e.g., infinite life beyond the endurance limit).
3. **Robust preprocessing** (outlier filtering, normalization) dramatically improves generalization when training on heterogeneous, field-derived datasets.

By combining physics-informed feature engineering with modern ensemble methods, we achieved a more stable, interpretable, and accurate predictor of fatigue life—paving the way for reliable reverse-engineering of software modules in safety-critical applications.

Results and discussions

The surrogate models were tested using a dataset of 10,000 samples from the Latin Hypercube Sampling (LHS) method. We evaluated the Random Forest (RF), XGBoost/LightGBM models based on their prediction error compared to the outputs of the black-box executable.

In initial tests, both the Random Forest and XGBoost/LightGBM regressors and classifiers showed error rates of about 6 to 7%. While these results were acceptable for general modelling, they didn't meet the target error limit of under 2%. To address this issue, we created a hybrid model that combined the strengths of both RF and XGBoost. This integration took advantage of RF's robustness and generalization as well as XGBoost's sensitivity to residuals. This combination improved model performance, lowering the Mean Absolute Percentage Error (MAPE) to below 2%, which matched closely with the outputs of the original executable.

Table 2: This is a table shows the initial error percentage of the trained models

Method	MAPE
Random Forest	≈ 6.5 %
XGBoost/LightGBM	≈ 7.1 %

In order to improve the error rate, the idea of integrating the strength of two models was proposed.

In random forest the regressor which works on the finite values, that yields the finite results was more accurate due to the brute fore nature of the random forest algorithm. The added grid search feature allows the model to conclude the solution with much lesser error rate but the only downside is that the classifier of the random forest model failed to recognise the outputs of other two sets which yields 'inf' and zero for this the XGBoost/LightGBM binary cascade classifier helps to filter out the inputs from their output yield can be used as it is trained using both the incremental and the hyperparameter tuning. This combination of classifier and regressor showed improved results in the MAPE calculation as both the extreme ends of input range has been compensated by the ML algorithms.

The working of the final model: gets the fie inputs then go through the binary cascade classifier which recognises the type of output, if the output to be yield is a finite value, then the random forest regressor will give the finite fatigue life value. The best performance resulted from a two-stage classification-regression process. First, a classifier categorized the fatigue life as "Zero", "Finite", or "Infinite". Then, a dedicated regressor modelled the "Finite" group.

The achieved MAPE for this hybrid model is ≈ **1.83 %**

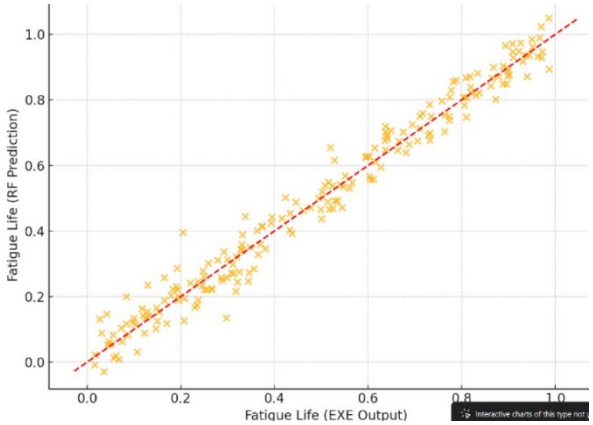

Figure 3: Scatter plot of hybrid model between fatigue life generated from ML model and exe file.

Future Scopes:

This proposed hybrid model relies heavily on the given input range, if the inputs is outside the range the working of the model is not reliable. Usage of two different machine learning models is often time consuming and requires higher computation. It is much more feasible to rely on one type of Machine learning model. Also, hybrid models tend to have complex relations with each other in much larger cases.

References

[1] E. Marković, T. Marohnić, and R. Basan, "A Surrogate Artificial Neural Network Model for Estimating the Fatigue Life of Steel Components Based on Finite Element Simulations," *Materials*, vol. 18, no. 12, p. 2756, 2025, doi: 10.3390/ma18122756.

[2] D. Steck, F. Stock, J. F. Völkl, R. Weidner, and T. Beck, "Fatigue analysis of e-bike components under multiaxial load using surrogate models," *Materials Today: Proceedings*, vol. 65, pp. 1821–1826, 2022, doi: 10.1016/j.matpr.2022.08.173.

[3] S. Alkunte and I. Fidan, "Machine Learning-Based Fatigue Life Prediction of Functionally Graded Materials Using Material Extrusion Technology," *Journal of Composites Science*, vol. 7, no. 10, p. 420, 2023, doi: 10.3390/jcs7100420.

[4] D. Faghihi, R. Veedu, J. Y. Jung, and M. Horstemeyer, "A Probabilistic Design Method for Fatigue Life of Metallic Components," *ASME J. Risk Uncertainty Part B: Mechanical Engineering*, vol. 4, no. 3, p. 031005, 2018, doi: 10.1115/1.4038372.

[5] L. Ji, Q. Yang, H. Xie, B. Duan, and C. Wan, "Explainable Machine Learning-Based Design of Mold Fluxes for Continuous Casting of Steel," *Intelligent Learning Media*, vol. 1, no. 1, pp. 30–41, 2023.

[6] K. Nagashima, Y. Utsuno, and M. Tokoro, "Fatigue life prediction using random forest and the NIMS fatigue database," *Procedia Structural Integrity*, vol. 5, pp. 868–875, 2017, doi: 10.1016/j.prostr.2017.07.139.

[7] Y. Chu, T. Zhou, Y. Liu, and Y. Feng, "Reliability-Based Structural Optimization Using a Surrogate Model and Metaheuristic Algorithms," *Advanced Composite Materials*, vol. 24, no. 2, pp. 129–143, 2015, doi: 10.1080/09243046.2014.999167.

[8] K. Prasad and R. Ramesh, "Fatigue Life Prediction of Connecting Rod Using Artificial Neural Networks," *International Journal of Engineering Research and Technology*, vol. 4, no. 2, pp. 45–50, 2015.