Article:
Proudlove, E., Thompson, H.M., Woollam, R. C. et al. (2026) Machine Learning Based Surrogate Modelling of High-fidelity Multiphysics CO_2 Corrosion Model Predictions. Corrosion. ISSN: 0010-9312

https://doi.org/10.5006/4888

Machine Learning Based Surrogate Modelling of High-fidelity Multiphysics CO_2 Corrosion Model Prediction

Ethan Proudlove[1], Harvey Thompson[1], Richard C. Woollam[1], Richard Barker[1], Micheal Jones[2]

[1] University of Leeds, Faculty of Engineering and Physical Sciences, Woodhouse Lane, Leeds, LS2 9JT, United Kingdom
[2] Deepwater EU Ltd. Unit 4.8 Frimley Business Park, Camberley, Surrey, GU16 7SG, United Kingdom

Key words: Machine Learning, Carbon Dioxide, Corrosion Rate, Modelling, Simulation.

1. Abstract

The University of Leeds has developed a high-fidelity mechanistic CO_2 corrosion model that simulates the underlying bulk equilibrium, mass transport, and electrochemical processes using multiphysics simulation software. This high-fidelity model (the Leeds Model) provides accurate forecasting of corrosion rate (CR), surface pH, and the precipitation of protective films within mild steel pipelines. However, there are a number of barriers hindering the effective use of the outputs from this model, including the requirement for trained users, licensing costs and the timescale and complexity of achieving satisfactory convergence when performing extended run-time, large scale parametric sweeps.

In this work we investigate the use of Machine Learning (ML) based surrogate modelling to combat these limitations, with the aim of developing a surrogate model to emulate Leeds Model predictions in a much more convenient format. Six ML models were trained on over 160,000 CO_2 CR predictions made by the Leeds Model, each with hyperparameters optimised accordingly. Analysis indicates test set predictions by the Deep Neural Network (DNN) and Extra Trees Regressor (ETR) models are most accurate. Further trade-offs between these surrogates are discussed, highlighting significant improvements in simulation time, computational cost, and usability compared to the original model. Additionally, the DNN proves effective in detecting convergence issues in the Leeds Model's CR predictions, therefore can improve confidence in source model output data.

2. Introduction

The effects of corrosion have enormous costs, with the 2016 NACE IMPACT report [1] estimating that corrosion costs the global economy $2.5 trillion annually, representing 3.4% of the global GDP. These costs stem from a failure to manage and predict corrosion, resulting in losses in efficiency, damage to equipment, and overly conservative designs [2]. Within the energy sector, CO_2 corrosion is considered the largest threat to carbon steel pipeline integrity, causing approximately 60% of failures [3,4].

There is increasing interest in developing accurate models for CO_2 corrosion in harsher, more corrosive environments and to use these in application areas such as geothermal energy and carbon capture and storage [5]. Empirical models are the current industry standard; however, these are essentially a 'black box' function based on the best fit to experimental data. These are simple and easy to use but have the significant disadvantage that they are unreliable outside the range of their training data. Mechanistic models address this limitation by simulating the underlying complex physio-chemical processes occurring in CO_2 corrosion. These provide greater physical insight but with higher complexity and computational costs. Such mechanistic models have been reviewed comprehensively by Kahyarian et al. [6] and Jones et al. [7].

Researchers at the University of Leeds have developed a state-of-the-art mechanistic model capable of predicting CO_2 corrosion using the COMSOL Multiphysics simulation software [8], building on the work of Norsdveen *et al*. [9]. This model is referred to throughout this paper as the Leeds Model [10,11]. In principle, this model is applicable to a broader range of experimental conditions than purely empirical models, however, there are practical challenges that inhibit its wider adoption by industry. These include: (i) the need for specialist knowledge to use the COMSOL commercial package effectively and the cost of commercial licenses; (ii) simulations over a large range of input parameters can be computationally expensive and time-consuming and (iii) the need to account for convergence errors causing anomalous predictions and missing output data - increasingly difficult to manage in large datasets. Hence the necessity for surrogate modelling.

Ultimately, simulation models are used to explore the design space, predict the relationship between model inputs and outputs and use these to identify optimal design variables. Simulation-based design optimisation is used routinely in the aerospace and automotive industries to solve complex design problems, and their usage is spreading rapidly to many other application areas [12]. When a large number of design variables are involved, it is impractical to run the simulation model at all possible design scenarios and surrogate models can be extremely beneficial. Surrogate models aim to approximate the

relationship between model inputs and outputs by creating surrogates that can be evaluated much more quickly than the original simulation model. In surrogate-assisted optimisation, the surrogates replace the simulations in the optimisation process, making optimisation much faster [13]. Numerous surrogate modelling methods have been developed, including Moving Least Squares, which is effective at minimising the deleterious effects of numerical noise [14], and Gaussian Process Regression, which can determine confidence intervals around predictions and deal effectively with mixed continuous/categorical variables [15].

Machine Learning (ML) algorithms are well-suited for surrogate modelling due to significantly greater speed, efficiency, and cost-effectiveness compared to high-fidelity mechanistic models. A number of recent studies have employed ML methods to learn the complex, nonlinear relationships between input conditions, such as temperature, flow speed and pH, and the resultant corrosion rate (CR). These have concluded that ML can be an extremely useful means of predicting CR, but that it is not possible to predict in advance which methods will prove the most effective.

Aghaaminiha *et al.* [16], for example, used ML to model experimental measurements of CR of carbon steel as a function of the time after corrosion inhibitors are added in different dosages and dose schedules. They found that Random Forests predicted the entire profile of CRs accurately. They also found that CR sensitivities are also well predicted by trained Random Forests models. Yang et al. [17] used five different ML methods, including Neural Networks, Random Forests and Extreme Gradient Boosting (XGB), to reveal the contribution of molecular features to the inhibition performance of amino acids. They found that XGB performed best for this application.

Feng *et al.* [18] compared the performance of the Random Forests and CatBoost ML methods, trained on thousands of CR predictions from the OLI Studio Corrosion Analyser. Using temperature, molar concentrations, velocity and pH as inputs, they found that both ML methods performed well for predicting CRs in both crude oil and natural gas pipelines. Seto et al. [19] recently used ML to predict corrosion caused by supercritical CO_2 and found that ensemble-based methods, such as Random Forests, worked best for this particular application.

The present study explores the use of six different ML algorithms as a surrogate model trained on the largest dataset reported so far with > 160,000 CR predictions from the Leeds Model. The accuracy, efficiency and robustness of the surrogates are considered, as is the need to account for inaccurate modelling predictions in challenging regions of parameter space.

3. The Leeds Model

This section offers a brief description of the Leeds Model and underlying CO_2 corrosion mechanisms. The underlying coupled chemical, electrochemical and fluid mechanical equations are discussed in greater detail in references [9–11].

3.1 Transportation of Species

The characteristics of the flow in the boundary layer are representative of CO_2-saturated water flowing through a straight a pipe under constant, turbulent hydrodynamic conditions. A modified version of the Nernst-Planck equation is used to describe the transportation of chemical species through the model. Due to the no-slip condition at the surface, convection is also negligible within this region, however turbulent eddies may still permeate through the boundary layer. Therefore, a turbulent diffusivity D_T is included to account for this effect:

$$D_T = 0.18 \cdot \frac{\mu}{\rho}\left(\frac{x}{\delta}\right)^3 \quad (1)$$

where μ is the dynamic viscosity, ρ is the fluid density, x is the perpendicular distance from the steel surface, and δ is the boundary layer height.

The flux of species through the boundary is determined through the following modified equation:

$$N_j = -(D_{M,j} + D_{T,j})\frac{\partial c_j}{\partial x} - z_j u_j F c_j \frac{\partial \Phi_l}{\partial x} \quad (2)$$

where N_j is the flux of species j, D_M is the molecular diffusion coefficient, D_T is the turbulent diffusivity, c_j is the species concentration, x is the distance perpendicular to the surface, z_j is the electrical charge, u_j is the mobility of species, F is the Faraday constant, and Φ_l is the electrolyte potential.

3.2 Bulk Equilibrium Chemistry

The bulk solution is assumed to maintain a constant chemical composition, serving as one of the boundary conditions in the model. The concentrations of the relevant species in the bulk phase are calculated based on the equilibrium constants corresponding to the reactions listed in Table 1. These constants represent the standard equilibrium processes that occur when CO_2 dissolves in water; specifically, the hydration to form carbonic acid (Equation 3), followed by its two dissociation steps (Equations 4 and 5). The model also accounts for the dissociation of water (Equation 6).

$$CO_{2_{(aq)}} + H_2O_{(l)} \rightleftharpoons H_2CO_{3_{(aq)}} \quad (3)$$

$$H_2CO_{3(aq)} \rightleftharpoons HCO^-_{3(aq)} + H^+_{(aq)} \quad (4)$$

$$HCO^-_{3_{(aq)}} \rightleftharpoons CO^{2-}_{3_{(aq)}} + H^+_{(aq)} \quad (5)$$

$$H_2O_{(l)} \rightleftharpoons OH^-_{(aq)} + H^+_{(aq)} \quad (6)$$

Table 1: Equilibrium constants, definition, and formulation for bulk chemistry boundary condition

Constant	Definition	Formula	Units
K_{H,CO_2}	$\frac{[CO_2]}{P_{CO2}}$	$\frac{14.5}{1.00258} \cdot 10^{-\left(2.27+5.65\times10^{-3}\cdot T_f - 8.06\times10^{-6}\cdot T_f^2 + 0.075\cdot I\right)}$	M·bar^{-1}
K_{Hyd}	$\frac{[H_2CO_3]}{[CO_2]}$	2.58×10^{-3}	-
K_{ca}	$\frac{[H^+][HCO_3^-]}{[H_2CO_3]}$	$387.6 \times 10^{-\left(6.41-1.594\times10^{-3}\cdot T_f + 8.52\times10^{-6}\cdot T_f^2 - 3.07\times10^{-5}\cdot P - 0.4772\cdot I^{0.5} + 0.1180\cdot I\right)}$	M
K_{bi}	$\frac{[H^+][CO_3^{2-}]}{[HCO_3^-]}$	$10^{-\left(10.61-4.97\times10^{-3}\cdot T_f + 1.331\times10^{-5}\cdot T_f^2 - 2.624\times10^{-5}\cdot P - 1.166 I^{0.5} + 0.3466\cdot I\right)}$	M
K_w	$[H^+][OH^-]$	$10^{-\left(29.3868-0.0737549\cdot T_K + 7.47881\times10^{-5}\cdot T_K^2\right)}$	M^2

where T_K and T_f are the fluid temperature in Kelvin and Fahrenheit respectively, I is the ionic strength in M, and P is the operating pressure in psi.

3.3 Bulk Specie Reaction Kinetics

Within the boundary layer, species concentrations deviate from equilibrium due to ion fluxes driven by electrochemical surface reactions. These shifts alter the rates of the reversible bulk reactions (Equations 3 to 6), with either the forward or reverse direction becoming dominant depending on the local concentration of each species. To account for these dynamics, the reaction kinetics are modelled using a system of ordinary differential equations (Equation 7) comprising the equilibrium constant previously

referred to in Table 1, in conjunction with forward (k_f) and backward (k_b) rate constants, whose relationships are defined in Table 2:

$$\begin{bmatrix} R_{CO_{2(aq)}} \\ R_{H_2CO_3} \\ R_{HCO_3^-} \\ R_{CO_3^{2-}} \\ R_{H^+} \\ R_{OH^-} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} k_{f,Hyd}\left[CO_{2(aq)}\right] - k_{b,Hyd}[H_2CO_3] \\ k_{f,ca}[H_2CO_3] - k_{b,ca}[HCO_3^-][H^+] \\ k_{f,bi}[HCO_3^-] - k_{b,bi}[CO_3^{2-}][H^+] \\ k_{f,w}[H_2O] - k_{b,w}[OH^-][H^+] \end{bmatrix} \quad (7)$$

where R_j is the rate of change of species j.

Table 2: Kinetic reaction rate constants for reversible reactions in solution (where T_K and T_C are the fluid temperature in Kelvin and Celsius respectively)

Constant	Definition	Formula	Units
$k_{f,Hyd}$	$K_{Hyd} \cdot k_{b,Hyd}$	$10^{329.85-110.541\times\log(T_K)-\frac{17265.4}{T_K}}$	s^{-1}
$k_{f,ca}$	$K_{ca} \cdot k_{b,ca}$	$10^{5.71+0.0526\times T_C-2.94\times10^{-4}\times T_C^2+7.91\times10^{-7}\times T_C^3}$	s^{-1}
$k_{f,bi}$	$K_{bi} \cdot k_{b,bi}$	10^9	s^{-1}
$k_{b,w}$	$\frac{k_{f,w}}{K_w}$	7.85×10^{10}	$M^{-1}s^{-1}$

3.4 Electrochemical Surface Reactions

Three electrochemical reactions are modelled at the steel surface; one anodic reaction being the dissolution of iron (Equation 8), and two cathodic reactions being the reduction of hydrogen (Equation 9) and the reduction of carbonic acid (Equation 10).

$$Fe_{(s)} \rightarrow Fe^{2+}_{(aq)} + 2e^- \quad (8)$$

$$2H^+_{(aq)} + 2e^- \rightarrow H_{2(g)} \quad (9)$$

$$2H_2CO_{3(aq)} + 2e^- \rightarrow H_{2(g)} + 2HCO_{3(aq)}^- \quad (10)$$

The local current density (i_{loc}) for each reaction can be determined from the exchange current density ((i_0) - determined using the relationships outlined in Table 3), the overpotential (η), and either the anodic or cathodic Tafel slope values (b_a and b_c, respectively) via Equations 11 and 12.

$$i_{loc} = i_0 \times 10^{\frac{\eta}{b_a}} \quad (11)$$

$$i_{loc} = -i_0 \times 10^{\frac{\eta}{b_c}} \quad (12)$$

The overpotential (η) is given by:

$$\eta = E - E_{rev} \quad (13)$$

Where E is the electric potential and E_{rev} is the reversible potential. The electric potential is calculated as the difference between the electrode potential (Φ_s) and the electrolyte potential (Φ_l) at the interface:

$$E = \Phi_s - \Phi_l \quad (14)$$

Table 3: Table of values for surface electrochemical reaction modelling

$$\boldsymbol{i_0 = i_{0,ref}\left(\frac{c_H^+}{c_{H^+,ref}}\right)^{a_1}\left(\frac{c_{CO_2}}{c_{CO_2,ref}}\right)^{a_2}\left(\frac{c_{H_2CO_3}}{c_{H_2CO_3,ref}}\right)^{a_3} e^{-\frac{\Delta H}{R}\left(\frac{1}{T}-\frac{1}{T_{ref}}\right)}}$$

Reaction	$i_{0,ref}$ $\frac{A}{m^2}$	a_1	$c_{H^+,ref}$ M	a_2	$c_{CO_2,ref}$ M	a_3	$c_{H_2CO_3,ref}$ M	ΔH $\frac{kJ}{mol}$	T_{ref} $°C$	E_{rev} V	b V
$\mathbf{2H^+ + 2e^- \rightarrow H_2}$	0.05	0.5	10^{-4}	0	N/A	0	N/A	30	25	$-2.303\frac{RT}{F}$	$2.303\frac{2RT}{F}$
$\mathbf{2H_2CO_3 + 2e^- \rightarrow H_2 + 2HCO_3^-}$	0.06	−0.5	10^{-5}	0	N/A	1	10^{-4}	50	20	$-2.303\frac{RT}{F}$	$2.303\frac{2RT}{F}$
$\mathbf{Fe_{(s)} \rightarrow Fe^{2+}_{(aq)} + 2e^-}$	1	2 for pH < 4 1 for 4 ≤ pH < 5 0 for pH ≤ 5	10^{-4}	1 for $P_{CO_2} < 1$ 0 for $P_{CO_2} \geq 1$	0.0366	0	N/A	37.5	25	−0.488	0.03 for pH < 4 0.08 for 4 ≤ pH < 5 0.12 for pH ≥ 4

CR is calculated from the molar flux of Fe from the pipe surface. Leeds Model hydrodynamic parameters, domain size, reaction equilibrium constants, and electrochemical parameters, etc., are all heavily influenced by 5 key Leeds Model input parameters: fluid temperature (T), bulk solution pH (pH), CO_2 partial pressure (pCO_2), flow velocity (V), and pipe diameter (D). A parametric sweep of these 5 parameters was run on the Leeds Model to generate a large dataset of CR predictions.

4. Description of the Computational Dataset

The Leeds Model simulated CO_2 corrosion in 161,051 unique cases, by varying the 5 input parameters identified previously (T, pH, pCO_2, V, and D). Each of the 5 input parameters have 11 unique values and the dataset has 11^5 = 161,051 datapoints. A statistical summary of the simulation input values is provided in Table 4. Full explanations of parameter distributions are detailed in the Data Processing section.

For each 5-dimentional input (or design) point, the Leeds Model simulation should yield a CR prediction which a surrogate will aim to emulate. However, an important issue in generating this dataset is that the Leeds Model requires user-specified initial conditions. These act as a starting point for each simulation, which the model uses to solve the nonlinear equations and, hopefully, converge to a meaningful solution. When running a sweep on the scale of Table 4, preferable initial conditions for one end of the sweep may become inadequate at the other, and consequently the model may fail to converge, resulting in a missing dataset value. In exceptional cases, the model converges to a local minimum, resulting in an anomalous CR prediction referred to here as an *outlier*. Two datasets have been used during this project. An original Low Convergence (**LC**) success rate dataset was obtained, simulating all values with a single initial condition. This led to 17,079 missing values, 10.6% of all input cases. The LC dataset was used during the data preprocessing, training, and hyperparameter optimisation phases of this study. However, upon the detection of outliers within the LC dataset, a second dataset was simulated. This second simulation of identical input cases was obtained with each pH run separately, under optimised initial conditions. This second dataset is referred to as the High Convergence (**HC**) success rate dataset. The HC dataset saw significant improvement in convergence, reducing the number of missing values to 127, just 0.0789% of the 161,051 cases.

The Leeds Model requires approximately 48 hours of computational time to obtain each full dataset. However, significantly more time and attention are required to adjust initial conditions and process the outputs of the HC simulation. Table 4 also details a comparison of statistical properties of CR in both the LC and HC datasets.

Table 4: Statistical summary of input conditions used in Leeds Model simulations (both LC and HC datasets) and the CR (output) in the LC and HC datasets after removal of missing values.

Parameter [units]	Min	Mean	Median	Max	Distribution
Input					
pH	5	5.5	5.5	6	Linear
T [K]	273.15	323.15	323.15	373.15	Linear
D [m]	0.01	0.248	0.1	1	Exponential
V [m/s]	0.1	2.448	1	10	Exponential
pCO_2 [Bar]	0.1	2.448	1	10	Exponential
Output					
CR - LC dataset [mm/year]	0.00614	4.082	1.982	43.447	-
CR - HC dataset [mm/year]	0.0789	4.099	2.018	43.447	-

The similarity between maximum, mean, and median of LC and HC corrosion rates in Table 4 indicate consistency of predictions in each simulation, and that the additional ~17,000 values in the HC dataset do not significantly change the profile of the data. However, a key difference is that the minimum value of the LC dataset is an order of magnitude below that of the HC dataset - the first indication of the potential existence of outliers.

5. Data Processing

The data processing stage improves the accuracy of the trained ML models and speeds up training by processing the input data into a more accessible format. This typically includes identifying anomalous datapoints, or using mathematical transformations. There is no optimal processing technique or 'one size fits all' method so several different approaches were trialled, with methods yielding the highest accuracy trained ML models adopted.

5.1 Input Distribution

In Table 4, a linear input distribution indicates a uniform gap between inputs (E.g. pH input values are 5, 5.1, 5.2, 5.3, … 6, with all values separated by 0.1). An input with an exponential distribution means the gap between successive values increases by a constant factor (1.585 or $10^{0.2}$ in this dataset). For example, V input values are 0.10, 0.16, 0.25, 0.40, …, 10 m/s. A Log10 operation is applied to convert exponentially distributed input variables (D, V and pCO_2) to linearly distributed ones, so that each input parameter has 11 linearly distributed values. Koukaras et al., [20] suggests that using a Log10 operation to transform data improves model accuracy because it reduces skew within inputs (in this case removes it entirely) and reduces the effects of outliers. The values of each input after a Log10 operation are shown in Table 5. This ensures all inputs have the same range after feature scaling.

Table 5: Statistical summary of exponential inputs after being processed with Log10.

Parameter [units]	Min	Mean	Median	Max	Distribution

LogD [m]	-2	-1	-1	0	Linear
LogV [m/s]	-1	0	0	1	Linear
LogpCO_2 [Bar]	-1	0	0	1	Linear

5.2 Feature Scaling

Once all input features are linearly distributed, a scaling method is required to ensure all input parameters are scaled to similar ranges [20]. Some models, such as neural networks and those that are distance-based, are known to perform optimally with feature scaling [20]. Feature scaling can also improve training speed and algorithmic convergence [21]. The feature scaling method used in this study is standardisation: each input parameter is non-dimensionalized into a distribution with a mean of 0, and a standard deviation of 1. An input parameter (X), with a training set mean of μ, and training set standard deviation of σ, is standardised with Equation 15.

$$X_{standardized} = \frac{(X - \mu)}{\sigma} \quad (15\text{~~64~~})$$

Min-max scaling was also trailed; however, standardisation was generally found to be more effective. All 5 inputs have the same range after both processing steps: between -1.507 and 1.507. The CR values are not scaled or processed.

6. Machine Learning Models

ML describes algorithms used to generalise parameter relationships learnt from training data, to new, unseen data. In this study, a ML model infers a function that maps the 5 inputs to a CR throughout the design space in the form $CR = f(pH, T, D, V, pCO_2)$~~, where f is the ML model)~~.

6.1 General training procedure for a ML model

This study uses a supervised learning approach, where the models learn from labelled training data. After data processing, all data is randomly allocated into one of three sets, the training, test and validation sets, in the ratio 70/10/20. The purpose of each set is as follows:

Training set – 70% of processed LC data. This set is labelled: a ML model will see the CR value alongside each input condition in this set. During training, each model iteratively adjusts internal weights to minimise a loss function between its predictions and training set labels. Mean Squared Error (MSE) is typically chosen as the loss function for computational efficiency [21]. Each model has different internal weights and algorithms employed to optimise them; further details are provided in Table 6.

Test set – 20% of processed LC data. This set is unlabelled: a ML model does not see the correct CR value associated with a datapoint in this set and therefore cannot learn from this data. After training, the models are evaluated on predictions of this unseen set. This allows a comparison of each model's ability to generalise to new, unseen data.

Validation set – 10% of processed LC data, also unlabelled. This set is used to periodically assess the performance of selected models during training and prevent overfitting. The purpose of this set is described further in Section 6.4.

In subsequent sections the sets are grouped into two categories. The labelled training set is referred to as the '**seen**' data (as the models have seen and learnt from the CR labels for those datapoints during training). The unlabelled test and validation sets are combined into '**unseen**' data. Overfitting and

underfitting must be mitigated during the training, optimisation, and selection of a ML surrogate model. **Overfitting** occurs when a model shows significantly higher predictive accuracy in training than on unseen data, implying it has learnt noise or correlations unique to the training set, that do not generalise to the overall problem. **Underfitting** describes poor predictive accuracy on both training and test set, typically due to an inadequate model or selection of model settings.

6.2 A brief description of chosen models

Preliminary testing was conducted to shortlist six regression algorithms that have been used successfully in previous corrosion studies [22]. Models are implemented in Python, using the libraries Scikit-learn [23] and TensorFlow [24]. The ML algorithms examined in this paper are described in Table 6. Decision trees are also described as they are a key constituent of several of the ML models used. Further details can be found in references [21,23–30] and in Table 6 below.

Table 6: Machine Learning Algorithm Descriptions

Model	Description	Used in / Further Reading
LR - Linear Regressor	A weighted sum of each of the 5 input parameters and a constant, resulting in a linear line of best fit through the data [21]. LR is the simplest and most constrained algorithm used in this study, often performing poorly in associated literature as the parameter relationships being modelled are highly non-linear. However, it is used in the studies of Seghier et al. [31] and Peng et al. [32] as a comparative baseline metric or to indicate feature importance.	[17,31,32] / [21]
KNN - K Nearest Neighbors	A KNN regressor is an interpolation function, storing the entire training dataset and estimating the output for a new datapoint by calculating the mean of the k nearest labels [21,25]. Configurable parameters include k, and the distance metric used (typically Euclidian) [16,25]. Aghaaminiha et al. [16] find a KNN to yield lower a MSE than Neural Networks, while Seto et al. [19] find the KNN yields lower predictive accuracy than an RFR when predicting supercritical CO_2 corrosion.	[16,19] / [21,25]
Decision Tree	The following three algorithms are ensemble methods, composed of hundreds of decision trees. A Decision tree is a structure of branch and leaf nodes. Branch nodes contain a binary decision, which splits the training data into two subsets: whether an input is above and below a threshold, i.e. "*is pH > 5*?". Each leaf node simply contains the average CR of the remaining datapoints [21]. To make a CR prediction, a datapoint will traverse the tree along the branches until eventually reaching a leaf node/CR prediction [21]. During training, branches (and leaves) are added until either the remaining subset is too small, or a maximum depth is reached. Training is the process of selecting branch decisions and threshold values.	- / [16,21]
RFR - Random Forest Regressor	Individual decision trees are highly susceptible to overfitting and overly sensitive to training data [21,31]. Tree-based ensemble methods average the predictions of many decision trees, with the philosophy that aggregating a group of predictors will outperform any single individual predictor overall [21,33]. Coelho et al. [22] document tree-based ensemble methods as the most popular within the literature predicting corrosion.	[16–19,31,33,34] /19/05/2026 14:35:00[21,26,27]

	The RFR trains multiple decision trees on random subsets of the data (random subsets of features and datapoints), averaging their predictions to obtain a final output [21,26,27]. In a RFR, decision tree branches (decisions) are tuned to select features and thresholds that minimise MSE within the remaining subsets [21]. Fang et al. [18] use a RFR to demonstrate improved corrosion rate predictions with boosting and to inform feature selection Aghaaminiha et al. [16] demonstrate RFR to outperform Neural Networks and KNN algorithms when modelling the CR of carbon steel as a function of time.	19/05/2026 14:35:00
GBR - Gradient Boosted Regressor	The GBR is another tree-based ensemble method. However, it trains extra decision trees in areas of underperformance in training iterations known as epochs, sequentially correcting its errors. [21,28]. This process is known as boosting. Ossai [34] found a GBR to outperform a Deep Neural Network and RFR when predicting corrosion defect depth growth.	[31,33,34] / [21,28]
ETR - Extra (Extremely Randomised) Trees Regressor	Like the RFR and GBR, the ETR computes the average prediction of several decision trees. However, instead of searching for optimal features and thresholds to split nodes, an ETR splits nodes using random thresholds. This accelerates training and reduces the risk of overfitting through improving tree diversity [21,29]. There is limited established literature applying specifically the ETR to corrosion prediction, however tree-based ensembles have shown to be popular and effective within relevant studies [22]. Feng et al. [33] find four ensemble learning models to outperform Neural Networks, and established mechanistic models on predictive accuracy.	- / [21,29]
DNN - Deep Neural Network	A DNN is a computational representation of the brain, with multiple interconnected layers of neurons. Each connection has an associated weight, training involves iteratively tweaking weights between neurons and bias terms to minimise the MSE of predictions, using backpropagation and gradient descent [21,30]. DNN sophistication lends itself to the complexity of CO_2 corrosion rate forecasting but also increases the risk of overfitting. Coelho et al. [22] document the widespread use of DNNs within the literature with Nesic et al. [35] and De Masi et al. [36] finding the DNN to outperform empirical models and other ML models.	[16,17,32,34–36] / [21,30]

6.3 Hyperparameter optimisation

Hyperparameters are ML model parameters which are predefined before training. Examples include the number of decision trees in an ETR, or number of neurons and layers in a DNN. Each model (Except the LR which has zero configurable parameters) comes with a default configuration of hyperparameters which requires optimisation for any specific dataset. Grid search is a popular means of hyperparameter optimisation, see e.g. Feng et al [33] and Seghier et al [31], and is utilised in this study. In grid search optimisation, for each hyperparameter being optimised several candidate values are chosen. For example, for a DNN the number of hidden layers could take the values [2, 3, 4], and the number of neurons per layer [2, 20, 200]. Grid search simply trains a DNN with every permutation of hyperparameters, returning the combination yielding optimal predictive accuracy on an unseen testing dataset, to mitigate overfitting. Since grid search is time consuming and computationally expensive, many studies [16,31,33] use *k*-fold cross validation to improve efficiency, whereby the models are trained and

tested on subsections of the dataset in parallel. This study uses 5-fold cross validation. A full breakdown of each model's hyperparameters can be found in the Scikit-learn [23] and TensorFlow [24] documentation. Hyperparameter values before and after optimisation are included in Appendix 1.

6.4 Nuances in DNN, and GBR implementation

The basic training procedures of all six implemented models is detailed in Table 6, however, nuances in the training methods of the GBR and DNN require further expansion. DNN and GBR training is done in iterations known as epochs. After each epoch, a loss function is calculated, and the model incrementally improves its performance on the training data. The simplest approach, which recognises the danger of over-fitting to the training data, is to terminate the fitting process after a predetermined number of epochs. A more sophisticated approach, known as early stopping, is demonstrated in the work of Fang et al. [18] whereby after each epoch, models are evaluated on the validation set. Training ends when performance on the validation set does not improve for a specified number of epochs, indicating it is only learning information relevant to the training set. The present study uses a 20-epoch stopping threshold.

Learning rate is another key parameter requiring optimisation in the GBR, and DNN methods, which specifies the weight given to new information. Learning rate scheduling was introduced by Smith, [37] allowing a model to begin training with a high learning rate, gradually reducing it as performance plateaus. This method reduces training time and improves model performance, as initially the model learns quickly, then the rate of learning reduces significantly. A learning rate threshold of 8 and reduction factor of 0.8 are employed during training of the DNN, meaning if validation performance does not improve over 8 epochs, learning rate is reduced by a factor of 0.8.

6.5 Model evaluation metrics

To evaluate a model's predictive performance, the following metrics have been chosen: Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE) and Coefficient of Determination (R^2). Formulae for each are shown below, where $C_{Actual,\ i}$ is the *i-th* actual corrosion rate, $C_{Pred,\ i}$ is the *i-th* predicted corrosion rate, n is the number of samples, and $C_{Average}$ is the average corrosion rate across the dataset [31].

RMSE
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(C_{Actual,\ i} - C_{Pred,\ i})^2}$$

MAPE
$$100 * \frac{1}{n}\sum_{i=1}^{n}\left|\frac{C_{Actual,\ i} - C_{Pred,\ i}}{C_{Actual,\ i}}\right|$$

R^2
$$1 - \frac{\sum_{i=1}^{n}\left(C_{Actual,\ i} - C_{Pred,\ i}\right)^2}{\sum_{i=1}^{n}\left(C_{Average} - C_{Pred,\ i}\right)^2}$$

A model's predictions are considered more accurate when they exhibit a lower RMSE, lower MAPE, and a R^2 closer to 1. RMSE quantifies the difference between two sets of corrosion rate values, measured in mm/year. RMSE is the standard error metric in the literature due to its sensitivity to large prediction errors. [18,31–34,36]. MAPE is a relative measure of model performance frequently utilised within the literature, particularly when the target variable spans several orders of magnitude [18,33,36]. R^2 measures the correlation between actual and predicted values and ranges between 1 and -1. The benefits of using R^2 are highlighted in a study by Feng *et al.* [33]; its defined range provides more insight as a standalone metric than RMSE and MAPE, which necessitate a comparative value. However, R^2 is only interpretable in the context of a dataset, whereas RMSE is quoted in mm/year, therefore more relevant to an industry user.

7. Preliminary Evaluation

In this section, each model's test set predictions are evaluated before and after hyperparameter optimisation. After comparison, two candidate models are shortlisted and their predictions investigated. This begins with a holistic analysis of model performance, moves to analysis of performance of different data subsets, and culminates in comparisons of individual datapoints.

7.1 Hyperparameter optimisation results

Figures 1 and 2 show evaluation metrics of each surrogate's test set predictions before and after hyperparameter optimisation, illustrating its effect on performance and facilitating assessment of optimised models. Note, the LR has no configurable parameters, hence metrics remain unchanged after optimisation.

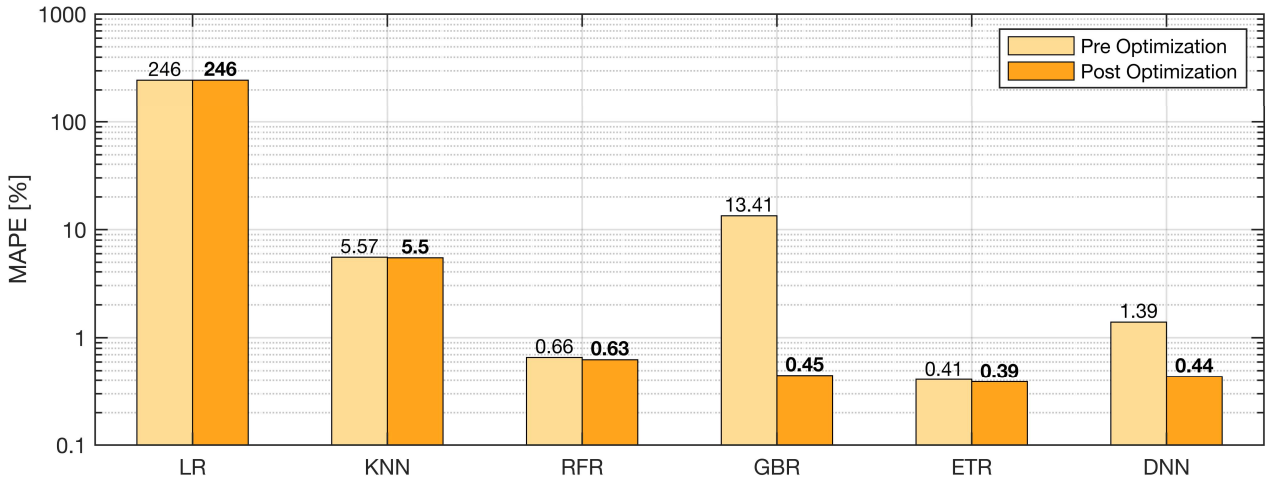


Figure 1: Effect of hyperparameter optimisation on MAPE of surrogate test set predictions

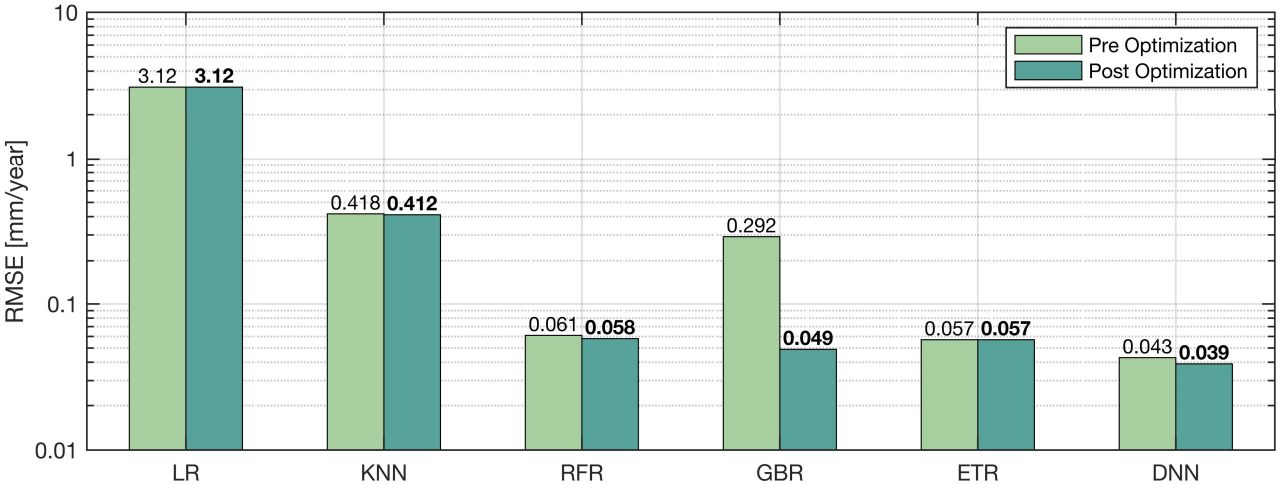


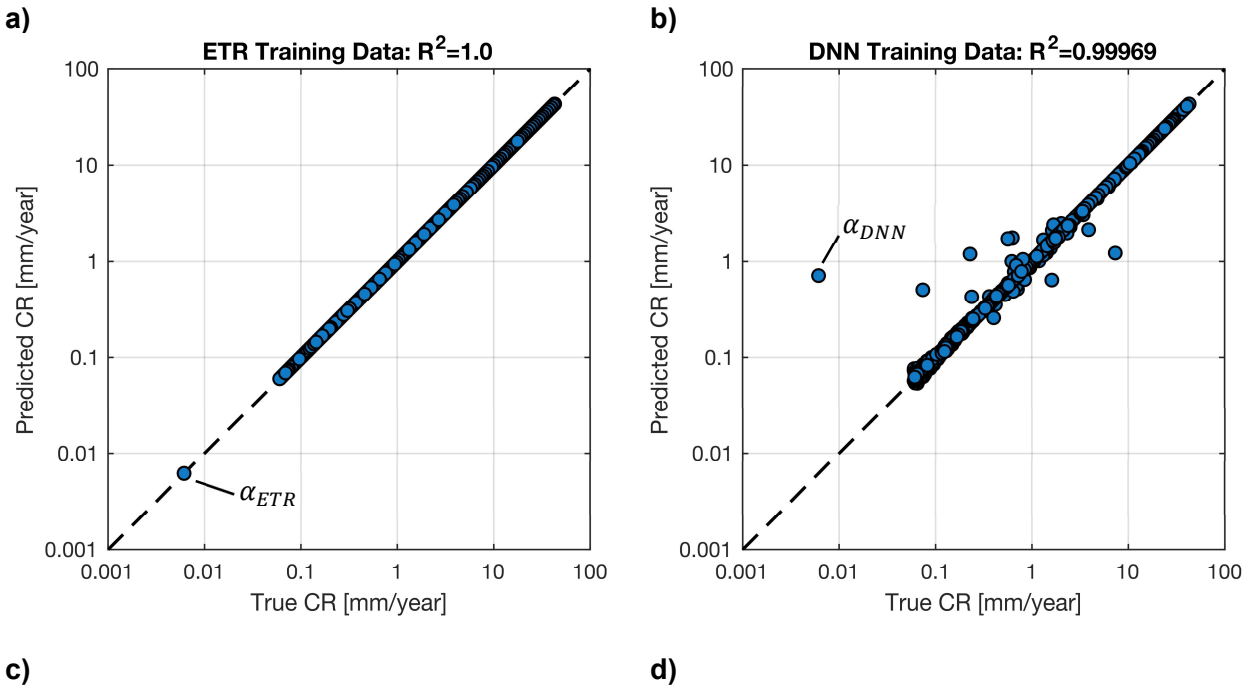
Figure 2: Effect of hyperparameter optimisation on RMSE of surrogate test set predictions

Results show the performance of the KNN and LR models are the poorest for this dataset. LR has the highest RMSE and MAPE, hence the least accurate CR predictions. This finding is consistent with literature [17,31,32] which finds that linear fitting is ineffective for the non-linear parameter relationships found in CO_2 corrosion. Similarly, both figures show the KNN to be underfitting, with RMSE and MAPE significantly greater than other models. This indicates KNN model architecture is too simple to accurately model CO_2 corrosion.

At a first assessment each of the four remaining models appear viable, adequately fitting the dataset. Pre-optimisation, the GBR appears a poor choice of model yielding a RMSE of 0.292 mm/year, which is several times that of the ETR, RFR, and DNN. Optimisation improves this metric to 0.049 mm/year, the second best of all models. Similar observations can be made in the MAPE results of the DNN. These highlight the importance comparing models after hyperparameters optimisation. In terms of overall test set predictions, the ETR achieves the lowest MAPE whereas the DNN achieves the lowest RMSE. Therefore, selecting the optimal model based on overall test set performance alone, is dictated by a choice between MAPE and RMSE. To identify an overall optimal surrogate, further investigation into DNN and ETR predictions is required.

7.2 ETR and DNN initial comparison

Figure 3 shows parity plots of the ETR and DNN predictions against the true dataset values. Every CR value in the LC dataset is plotted. The seen and unseen sets are plotted separately for each model (the unseen points are just the test and validation sets combined, recall the DNN and ETR have never seen the true CR for these points). Accuracy of a prediction in either figure is illustrated by proximity to the dashed line y=x and explicitly given by the R^2 metric in each plot's title. Such visualisations are common within the literature [16–18,31,33–36], however a logarithmic scale is required here as the target variable spans several orders of magnitude.



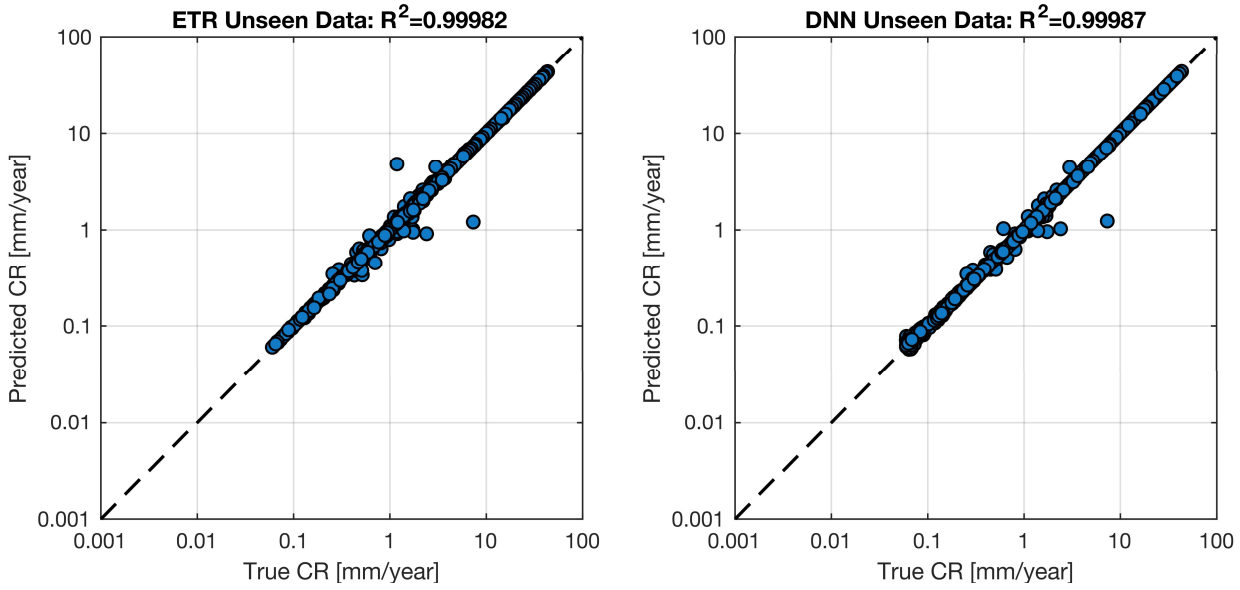


Figure 3: Predicted against true CR. Training set plotted for the (a) ETR and (b) DNN. Unseen datapoints for the (c) ETR and (d) DNN.

Both models exhibit excellent accuracy. Results in Figure 3 indicate the ETR is overfitting; points in Figure 3a lie entirely on y=x, whereas there is more deviation from y=x in Figure 3c. The R^2 value achieved by the ETR when predicting the training set is 1 (to 5 decimal places). This indicates excellent predictive accuracy of the training data, with many datapoints predicted correctly to within 6 decimal places. Whereas the R^2 over ETR predictions of test and validation data is marginally worse. Geron [21] suggests tree based ensembles are particularly prone to overfitting, and that this tendency can be mitigated by limiting the number of trees or the maximum tree depth. In contrast, the DNN predictions do not show evidence of overfitting. In fact, DNN predictions of the unseen data are more accurate than for the training data. Additionally, there are several large deviations from y=x in the DNN predictions of the training set in Figure 3b. Over the whole LC dataset (train, test and validation sets combined), the ETR obtains higher predictive accuracy. This is reflected most clearly in the contrast between Figures 3a and 3b.

The DNN and ETR predictions for the input condition α=[pH=5.2, T=283.15 K, V=3.98 m/s, D=0.63 m, pCO_2=0.63 bar] are shown in Figures 3a/b by the points α_{DNN} and α_{ETR} respectively. The DNN's prediction of these specific input conditions (α_{DNN}) yields the highest percentage error of any prediction made by either model. However, the ETR predicts these input conditions accurately and α_{ETR} lies on y=x. Both models' predictions and true CR (dataset value) for α are shown in Table 7.

Table 7: DNN prediction of CR at α with corresponding dataset value and the ETR's prediction. All values are in mm/year.

Dataset Value	**ETR Prediction (α_{ETR})**	**DNN Prediction (α_{DNN})**
0.006141	0.006141	0.713158

The ETR's prediction (α_{ETR}) is correct to 6 decimal places; it has correctly learnt the dataset value. An error by the DNN of the magnitude observed in Table 7 is considered exceptional. Considering the DNN achieves excellent accuracy over the dataset as a whole and that the ETR and DNN are trained on identical data, why is α_{DNN} so different from α_{ETR}? The DNN's predictions of CR at input conditions around α are investigated in Table 8, where pH is varied while the other four inputs remain constant.

Table 8: Dataset values and DNN predictions of CR at pH values surrounding α (highlighted grey). All values are in mm/year.

	pH				
	5.0	**5.1**	**5.2**	**5.3**	**5.4**
Dataset Value	0.772	0.741	0.00614	0.694	0.677
DNN Prediction	0.767	0.738	0.713	0.693	0.680

The CR data in Table 8 shows the value recorded by the Leeds Model (the 'true' dataset value) at a pH of 5.2 is dubious. All recorded CRs in this region of the design space vary around 0.7 mm/year so the dataset value appears anomalous. This is later confirmed during the second HC simulation of all input conditions. Furthermore, the DNN's prediction of 0.713 mm/year appears reasonable when inserted into Table 8. Consequently, a second dataset was obtained using the Leeds Model to confirm and locate outliers in the existing LC dataset.

8. Outlier Confirmation and Model Selection

Upon the suspicion of outliers, a second **HC** dataset was simulated to locate their existence in the original **LC** dataset, that was used to train the models. Inconsistencies between the HC and LC dataset CR values at identical input conditions indicate at least one dataset is incorrect. The absolute percentage difference between the HC and LC dataset CR values was calculated for every input condition.

8.1 Outlier identification strategy

The 50 datapoints yielding the largest percentage difference between LC and HC value were re-run a third time on the Leeds Model. Agreement between the final re-run values, and the prediction of the HC dataset indicates the HC values are correct and the corresponding LC value is an outlier (or vice versa). Of the 50 cases re-run, this strategy identified 47 outliers in the original LC dataset, Figure 4 is a graphical example. This strategy also identified 3 outliers within the HC dataset. Hence, while convergence has improved and the severity of outliers has decreased in the HC dataset, they have not been eliminated. At present, there is no method of efficiently running Leeds Model parametric sweeps on the scale of this study without convergence issues.

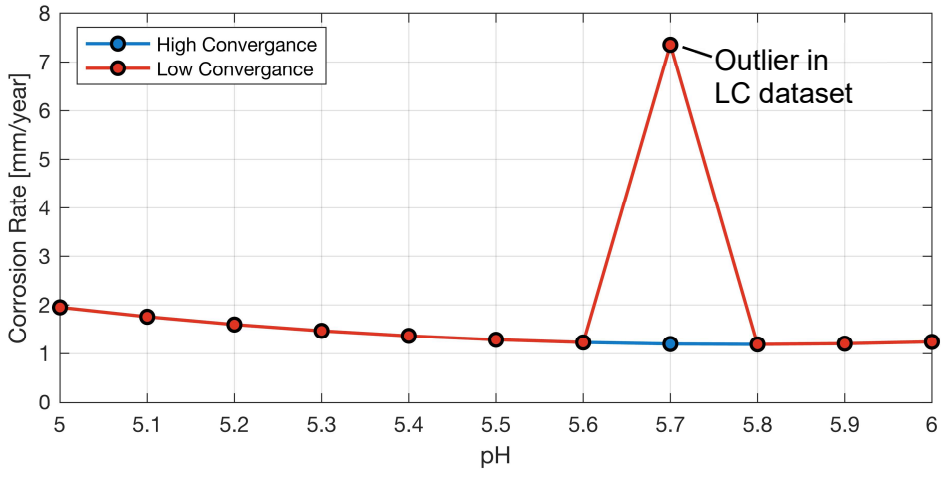
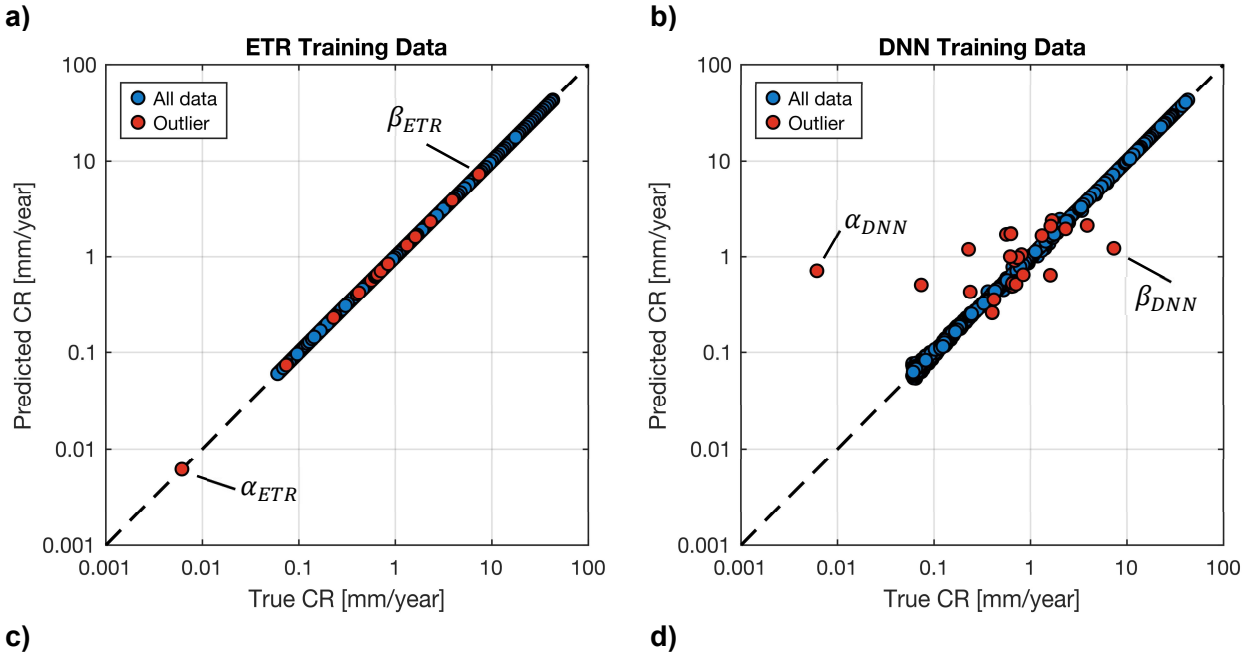


Figure 4: CR values in HC and LC dataset as a function of pH, other inputs are constant: T=353.15 K, V=1.59 m/s, pCO_2 = 0.25 bar, D=0.40 m.

For this sweep, at pH = 5.7 the CR recorded in the LC dataset is 7.35 mm/year. However, the HC dataset records it to be 1.20 mm/year. For the other ten input conditions plotted, both datasets are in agreement up to 4 decimal places for CR. From observing the trend of the LC data in Figure 4, the recorded CR of 7.35 mm/year is clearly incorrect. This is confirmed by the re-run simulation which predicts 1.20 mm/year, identical to the HC dataset. This method has identified 47 similar outliers within the LC dataset, of which, 30 occur in the training set. Hence, both the DNN and ETR have been trained on at least 30 incorrect values, the location of which has been documented.

8.2 Outlier impact on ETR and DNN predictions

Figure 5 is an identical illustration to Figure 3, depicting the ETR and DNN predictions against actual corrosion rate for every value in the LC dataset, except, in Figure 5 the 47 confirmed outliers in the LC dataset are now highlighted red. Therefore, when interpreting Figure 5, red points occurring on, or near, y=x are undesirable as it indicates the surrogate has learnt an incorrect outlier value. However, blue points occurring near, and on y=x are a good sign; the model is correctly predicting a valid CR.



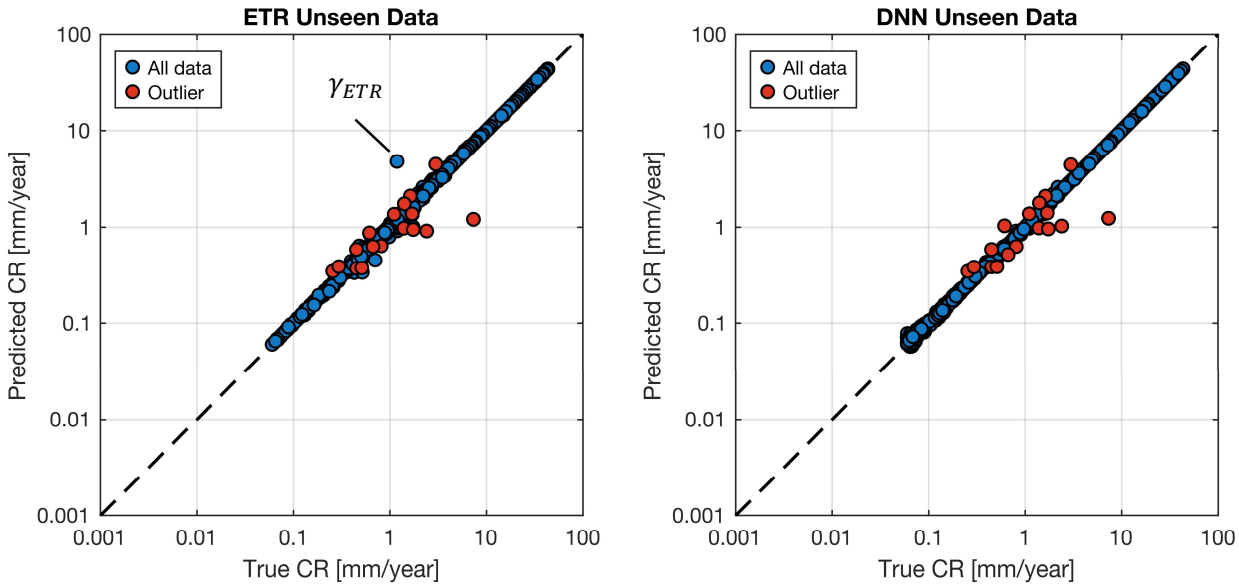


Figure 5: Predicted against true CR. Training set plotted for the (a) ETR and (b) DNN. Unseen datapoints for the (c) ETR and (d) DNN. The 47 LC outliers are depicted in red.

Results in Figure 5a indicate the ETR has learnt the exact CR label of all known outliers within the training data, often to several decimal places, including α_{ETR}. Perfect predictions of ETR outliers have contributed to its marginal superior predictive performance over the DNN overall. Additionally, close examination of Figure 5c shows that the ETR predicts several non-outlier CR values within the dataset less accurately than known outliers. Thus, although the ETR is slightly more accurate, it is also influenced strongly by outliers.

Conversely, DNN results in Figure 5 show the 47 known outliers are the furthest from y=x. This is most evident in Figure 5b, where all outlier predictions within the training set are clearly the 'least accurate'. This implies the DNN is less susceptible to being over-trained on outliers. Several points on Figure 5 have been labelled to highlight specific outliers for further investigation. Both α and β are specific input conditions corresponding to outliers in the LC dataset. As previously established, α_{DNN} and α_{ETR} are the CR predictions of input conditions α by the DNN and ETR (values given in Table 7). Similarly, β_{DNN} and β_{ETR} are the DNN's and ETR's predictions of β=[pH=5.7, T=353.15 K, V=1.59 m/s, D=0.40 m, pCO_2=0.25 bar]. The ETR has learnt the LC dataset label, hence β_{ETR} is a red point occurring on y=x. The input condition labelled γ=[pH=5.7, T=353.15 K, V=1.59 m/s, D=0.63 m, pCO_2=0.25 bar]. The CR label in the LC dataset is not an outlier. γ_{ETR} is the worst non-outlier prediction by either model, occurring in the predictions by the ETR (i.e. the actual CR recorded at γ is accurate in the LC dataset, hence γ_{ETR} the blue point furthest from y=x in Figure 5).

8.3 Analysis of individual outliers

Figures 6 and 7 illustrate the DNN and ETR's predictions around known outliers. This demonstrates how the models have interpreted outliers within the training set, and the effects they have on predictions of designs in their vicinity. Figure 6 shows CR predictions by the DNN and ETR as a function of T around α. The other four inputs were kept constant at pH=5.2, pCO_2=0.63 bar, D=0.63 m, and V=3.98 m/s. The DNN and ETR were used to predict the CR at temperatures varying from 273.15 K to 373.15 K, in 0.1 K increments. Note, this is a temperature resolution ~100 times finer than the training data (training data temperature has a datapoint every 10 K, this sweep has surrogate predictions every 0.1 K).

The red points plotted on Figure 6 are the training data. These are the values both models have learnt from during training. The yellow and green lines are predictions by the DNN and ETR respectively, hence, α_{DNN} and α_{ETR} occur on those lines (labelled on Figure 6). The accepted HC dataset values have also

been plotted on Figure 6, as black crosses plotted beneath the training data. In most cases, the LC and HC values are identical, hence the HC data is not visible beneath the training data.

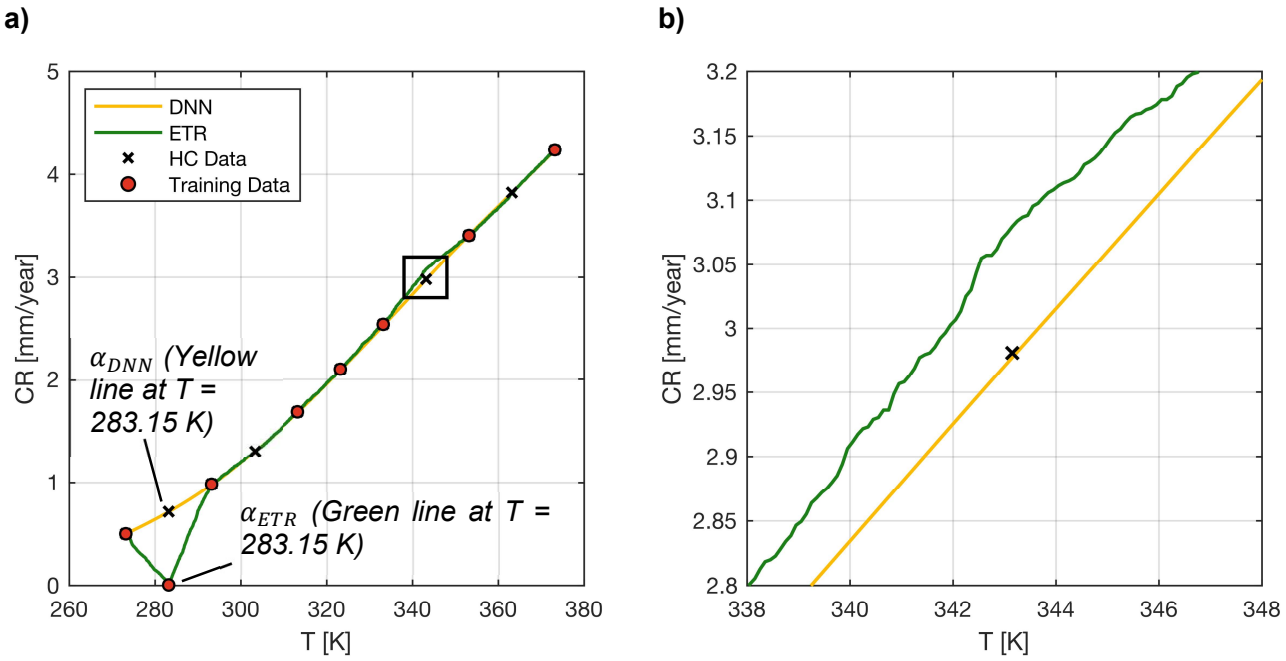


Figure 6: (a) DNN and ETR predictions around α, with a (b) magnified region: 338 < T < 348 K.

Clearly, there is an outlier both models have been trained on. This corresponds to the previously established α input conditions, occurring at T = 283.15 K, with a CR in the training set of 0.006141 mm/year. Figure 6a demonstrates the DNN's ability to ignore outliers in the training data. The DNN predicts α_{DNN} to be 0.713 mm/year. This prediction agrees with the HC dataset value of 0.714 mm/year and the trend in the training data, as shown in Figure 6a. This indicates the DNN is mapping a smooth, and broadly generalisable underlying function of CR, rather than learning precise training set values. With regards to the ETR, Figure 6a highlights an inability to distinguish between outliers and valid training data. As has already been shown, the ETR has learnt the exact training label of α. However, results in Figure 6a indicate the ETR has distorted its predictions of neighbouring points down to incorporate the anomalous CR at T=283.15 K. The ETR corrects itself on LC dataset values at 273.15 K and 293.15 K, but this nevertheless degrades the accuracy of ETR predictions between these points.

The magnified region in Figure 6b highlights an important contrast in the predictions of the ETR and DNN: the DNN predicts CR as a smooth function of T, whereas the ETR predictions are noisy. The HC dataset value in Figure 6b is in the test set, neither model having been trained using that value. Despite this, the DNN predicts much closer to the true HC value than the ETR. Issues with the high-resolution ETR predictions is caused, in part, by a sensitivity to noise, outliers, and missing values in neighbouring training points. Additionally its tree-based ensemble architecture makes it sensitive to variation in the training data [21]. During training, the ETR splits the data into subsets, and predictions are based on the average value within the subset. Therefore, the CR function of the ETR is a set of discrete bins of CR values, explaining the step changes observed in ETR predictions of neighbouring points in Figure 6b. Conversely, DNN architecture facilitates prediction of continuous outputs when using continuous activation functions [38], hence the smooth DNN output observed in Figure 6. Several methods of quantifying design sensitivity and robustness require reliable gradient calculations [39]. Such calculations would be challenging when using the ETR predictions near outliers.

ETR and DNN predictions around the outlier β are shown in Figure 7. The DNN and ETR were used to predict 1000 exponential increments of D between 0.01 m and 1 m. In this instance, pH = 5.7, pCO_2 =

0.25 bar, T = 353.15 K, and V = 1.59 m/s. Training datapoints and HC dataset values are plotted in the same format as Figure 6. Note the logarithmic scale of D on the x-axis.

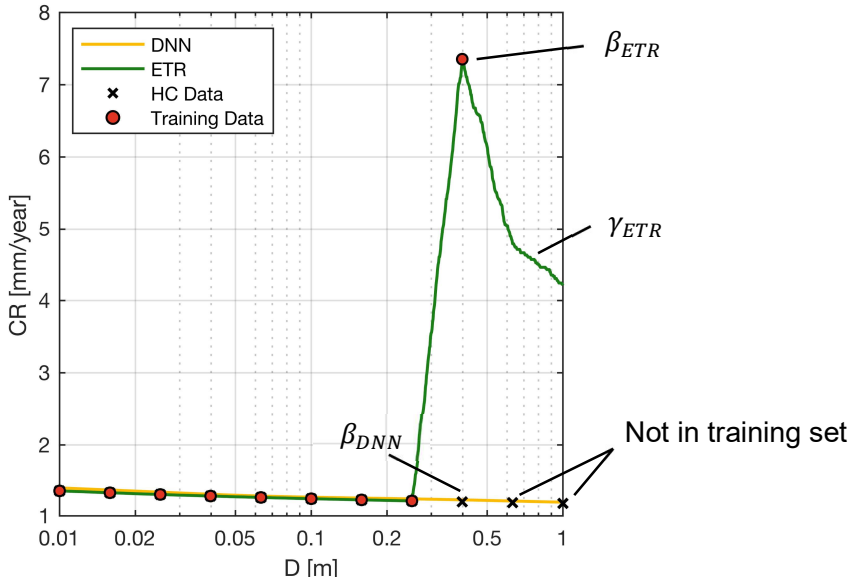


Figure 7: DNN and ETR predictions surrounding outlier β

The training data depicted in Figure 7 illustrates a unique scenario. The CR data appears consistent around 1.2 mm/year until a significant spike at β, occouring at D = 0.40 m. As established, the training label associated with β is an outlier, reporting a CR of 7.35 mm/year. At diameters beneath this outlier, predictions by both models match that within the training and HC data. Clearly, the ETR has learnt the outlier value, whereas the DNN predicts a CR of 1.22 mm/year (B_{DNN}) a 2.34% error from the accepted HC value. In this instance datapoints at both D = 0.63 m and 1 m are missing from the training set. Hence, neither model is exposed to a D above 0.40 m during training with the result that the ETR cannot rectify its predictions. It inaccurately predicts points a pipe diameter of 0.63 m (γ_{ETR}) to have a corrosion rate of 4.73 mm/year, despite the HC dataset recording this value as 1.18 mm/year. This datapoint is in the test set and is not an outlier. Hence, all ETR predictions of CR above approximately D = 0.25 m are corrupted, this represents ~75% of the range for D. In contrast, the DNN predicts CR values in Figure 7 that are consistent with trends in the HC dataset, recording a maximum deviation of 2.34%.

8.4 Using HC data as the true value

It is clear the DNN has predicted CR values consistent with the HC dataset in the two cases presented in figures 6 and 7, while the ETR has overfitted to the outlier value. A parity plot of the 47 confirmed outliers is given in Figure 8. The red points take the true CR value to be the outliers in the LC dataset, whereas blue points take the true value to be the accepted HC dataset value. Blue points in close proximity to the line y=x is desirable in both figures, whereas red points on y=x indicate the model has been overly-influenced by an outlier.

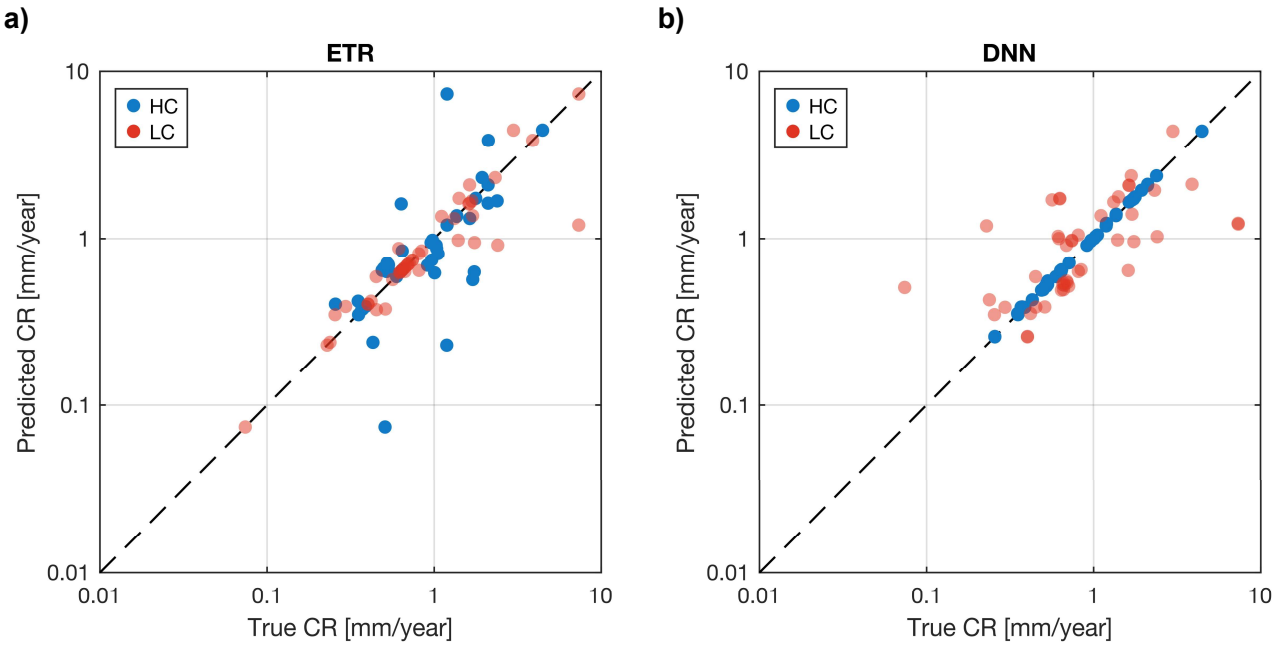


Figure 8: (a) ETR and (b) DNN predictions of the 47 confirmed outliers evaluated against LC dataset values in red, and original HC values in blue

Observing the results in Figure 8, it is clear the ETR is significantly less accurate than the DNN when evaluating the predictions of outliers against their corresponding HC dataset values. Blue points in Figure 8a obtain a negative R^2 value of -0.960, indicating ETR predictions of outliers are inconsistent with HC values. The significantly higher R^2 value of DNN predictions of 0.999, is reflected in Figure 8a, as DNN predictions consistently lie near y=x when the HC value is taken as the true value. Overall, the results confirm that the ETR is learning exact values in the LC dataset, whereas the DNN is rejecting outliers and fitting an underlying function which accurately represents the accepted HC dataset trends.

8.5 Practical utility

The practical advantages of the ML surrogates are now considered. ETR and DNN benchmark times were obtained on an Apple MacBook with an M1 chip (8-core CPU, 3.2 GHz, 16 GB of RAM) while the 48-hour Leeds Model simulation was run on a workstation with an AMD Ryzen Threadripper 3960X (24 cores, 48 threads, 3.8 - 4.5 GHz, 128 GB of RAM). The DNN requires 2.86 seconds to predict the CR at all 161,051 input conditions, whereas the ETR takes 7.39 seconds. Both surrogates are therefore substantially faster than the 48-hour compute time of the original Leeds Model. The ETR does not require a validation set so can be trained on a larger portion of the data than the DNN. This would likely improve its performance, which is already superior to the DNN when considering MAPE and R^2 over the entire dataset. All models are implemented using Python, a popular and widely-adopted programming language with many useful open-source libraries (e.g. Tkinter [40], Streamlit [41],…) which support the easy creation of User Interfaces.

In terms of outliers, selecting either the ETR or DNN reflects a trade-off between dataset quality and model robustness. The presented results indicate the trained ETR cannot distinguish between an outlier and valid datapoint within the training set. Hence, while the ETR yields marginally higher predictive accuracy over the training data, its adoption as a surrogate model requires outliers are removed before training. Conversely, the DNN is exceedingly resilient to the outliers identified in the training data. The ability to reject outliers during training lends itself to a novel outlier detection procedure to clean the model output data. Therefore, as well as its good accuracy, a surrogate DNN model also builds confidence in the output of the original source model.

9. Conclusion

Six ML algorithms have been trained on the outputs of a mechanistic CO_2 corrosion model, using the largest dataset of its kind to date. ML methods have been demonstrated to offer effective and efficient surrogate modelling capabilities for the Leeds Model. Trained surrogates are capable of predicting CR accurately, while significantly reducing computational and time demands of a simulation on the scale of this study. Additionally, their development in open source Python libraries and ease of integration with open-source UI libraries facilitate easy access to industry, academics and other end-users.

ML model hyperparameters have been optimised effectively, with the GBR yielding the most significant performance improvements. A preliminary evaluation indicated the ETR and DNN algorithms to have optimal test set predictive performance, however the choice of evaluation metric dictates which model is superior. The influence of outliers on ETR and DNN has proved an important differentiator. Results have shown that the trained ETR is sensitive to outliers, learning the exact value of all outliers documented in the training dataset. However, the DNN is significantly more robust against their influence, rejecting outliers in the training set, and predicting outlier values in line with the accepted HC simulation values. The DNN can therefore be used as a novel outlier detection method, to clean the output data of the original model. At the undertaking of this study, the existence of large outliers within the Leeds Model output data was unknown, adopting the DNN as a surrogate offers a reliable method of detecting and removing them from the source model output data.

The present work can be expanded in several directions, namely to corrosion systems with scaling products or by amalgamating the numerical simulation dataset with reliable experimental measurements. Additionally, the incorporation of additional Leeds Model outputs such as Saturation Index (akin to $FeCO_3$ scaling tendency) or surface pH as a second output of the DNN would enable interesting multi-objective optimisations to be explored.

Appendices

Appendix 1

Table A1: *Hyperparameter values before and after optimisation. Consult documentation in [23,24] for more information on specific parameters.*

Model	Parameter	Default value	Optimised value
LR	N/A	-	-
KNN	Weights	'uniform'	'distance'
RFR	Max Features	1	3
	Number of Estimators	100	800
GBR	Subsample	1.0	0.4
	Min Samples per Leaf	1	2
	Number of Estimators	100	400
	Max Depth	3	None
	Min Samples Split	2	3
ETR	Number of Estimators	100	1600
	Max Features	1	5
DNN	Number of Layers	2	4
	Neurons [Layer 1, Layer 2 …]	[100, 1]	[128, 128, 64, 1]
	Momentum	None	0.9
	Nesterov acceleration	False	True
	Activation Function	'relu'	'elu'
	Kernel Initialiser	'random_normal'	'he_nromal'

References

[1] G. Koch, J. Varney, N. Thompson, NACE IMPACT International Study – International Measures of Prevention, Application, and Economics of Corrosion Technologies Study., NACE International, Houston, TX, 2016.

[2] M.A. Jafar Mazumder, Global Impact of Corrosion: Occurrence, Cost and Mitigation, Global Journal of Engineering Sciences 5 (2020). https://doi.org/10.33552/GJES.2020.05.000618.

[3] Y. Bai, Q. Bai, Chapter 17 - Subsea Corrosion and Scale, in: Y. Bai, Q. Bai (Eds.), Subsea Engineering Handbook, Gulf Professional Publishing, Boston, 2010: pp. 505–540. https://doi.org/10.1016/B978-1-85617-689-7.10017-2.

[4] A.H. Al-Moubaraki, I.B. Obot, Corrosion challenges in petroleum refinery operations: Sources, mechanisms, mitigation, and future outlook, Journal of Saudi Chemical Society 25 (2021) 101370. https://doi.org/10.1016/j.jscs.2021.101370.

[5] M. Jones, J. Owen, G. De Boer, R.C. Woollam, M.C. Folena, H. Farhat, R. Barker, Numerical exploration of a fully mechanistic mathematical model of aqueous CO2 corrosion in steel pipelines, Corrosion Science 236 (2024) 112235. https://doi.org/10.1016/j.corsci.2024.112235.

[6] A. Kahyarian, M. Singer, S. Nesic, Modeling of uniform CO2 corrosion of mild steel in gas transportation systems: A review, Journal of Natural Gas Science and Engineering 29 (2016) 530–549. https://doi.org/10.1016/j.jngse.2015.12.052.

[7] M.P. Jones, Implementation and Numerical Exploration of Advanced Aqueous Carbon Dioxide Corrosion Models, phd, University of Leeds, 2024. https://etheses.whiterose.ac.uk/id/eprint/36915/ (accessed July 8, 2025).

[8] COMSOL Multiphysics, (2023). www.comsol.com.

[9] M. Nordsveen, S. Nešić, R. Nyborg, A. Stangeland, A Mechanistic Model for Carbon Dioxide Corrosion of Mild Steel in the Presence of Protective Iron Carbonate Films—Part 1: Theory and Verification, CORROSION 59 (2003) 443–456. https://doi.org/10.5006/1.3277576.

[10] M. Jones, G. de Boer, R. Woollam, J. Owen, R. Barker, Numerical Exploration of a Comprehensive Mechanistic CO2 Corrosion Model Part I: Influence of Temperature, Pressure and Bulk pH, in: Association for Materials Protection and Performance, 2023: pp. 1–12. https://doi.org/10.5006/C2023-19304.

[11] M. Jones, G. de Boer, R. Woollam, J. Owen, R. Barker, Numerical Exploration of a Comprehensive Mechanistic CO2 Corrosion Model Part II: Influence of Hydrodynamics and Bulk pH, in: Association for Materials Protection and Performance, 2023: pp. 1–10. https://doi.org/10.5006/C2023-19305.

[12] Z. Khatir, H. Thompson, CFD-enabled design optimisation of industrial flows–theory and practise, International Journal of Computational Fluid Dynamics 33 (2019) 235–236. https://doi.org/10.1080/10618562.2019.1674503.

[13] J.V. Soares Do Amaral, J.A.B. Montevechi, R.D.C. Miranda, W.T.D.S. Junior, Metamodel-based simulation optimization: A systematic literature review, Simulation Modelling Practice and Theory 114 (2022) 102403. https://doi.org/10.1016/j.simpat.2021.102403.

[14] C.S. Aydin, S. Ozgurler, M.B. Durmusoglu, M. Ozgurler, Response surface approach to robust design of assembly cells through simulation, AA 38 (2018) 450–464. https://doi.org/10.1108/aa-08-2017-093.

[15] J.N. Fuhg, A. Fau, U. Nackenhorst, State-of-the-Art and Comparative Review of Adaptive Sampling Methods for Kriging, Arch Computat Methods Eng 28 (2021) 2689–2747. https://doi.org/10.1007/s11831-020-09474-6.

[16] M. Aghaaminiha, R. Mehrani, M. Colahan, B. Brown, M. Singer, S. Nesic, S.M. Vargas, S. Sharma, Machine learning modeling of time-dependent corrosion rates of carbon steel in presence of corrosion inhibitors, Corrosion Science 193 (2021) 109904. https://doi.org/10.1016/j.corsci.2021.109904.

[17] J. Yang, J. Zhao, X. Guo, Y. Ran, Z. Fu, H. Qian, L. Ma, P. Keil, A. Mol, D. Zhang, Combinatorial discovery and investigation of the synergism of green amino acid corrosion inhibitors: Integrating high-throughput experiments and interpretable machine learning approach, Corrosion Science 245 (2025) 112675. https://doi.org/10.1016/j.corsci.2025.112675.
[18] J. Fang, X. Cheng, H. Gai, S. Lin, H. Lou, Development of machine learning algorithms for predicting internal corrosion of crude oil and natural gas pipelines, Computers & Chemical Engineering 177 (2023) 108358. https://doi.org/10.1016/j.compchemeng.2023.108358.
[19] E. Seto, M. Li, J. Liu, Predicting Corrosion Severity of Pipeline Steels in Supercritical CO2 Environments Using Supervised Machine Learning, in: Association for Materials Protection and Performance, 2024: pp. 1–15. https://doi.org/10.5006/C2024-20803.
[20] P. Koukaras, C. Tjortjis, Data Preprocessing and Feature Engineering for Data Mining: Techniques, Tools, and Best Practices, AI 6 (2025) 257. https://doi.org/10.3390/ai6100257.
[21] A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd ed., O'Reilly Media, Inc., 2019.
[22] L.B. Coelho, D. Zhang, Y. Van Ingelgem, D. Steckelmacher, A. Nowé, H. Terryn, Reviewing machine learning of corrosion prediction in a data-oriented perspective, Npj Mater Degrad 6 (2022) 8. https://doi.org/10.1038/s41529-022-00218-4.
[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay, Scikit-learn: Machine Learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, TensorFlow: a system for large-scale machine learning, in: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, USENIX Association, USA, 2016: pp. 265–283.
[25] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1967) 21–27. https://doi.org/10.1109/TIT.1967.1053964.
[26] Tin Kam Ho, Random decision forests, in: Proceedings of 3rd International Conference on Document Analysis and Recognition, IEEE Comput. Soc. Press, Montreal, Que., Canada, 1995: pp. 278–282. https://doi.org/10.1109/icdar.1995.598994.
[27] L. Breiman, Random Forests, Machine Learning 45 (2001) 5–32. https://doi.org/10.1023/A:1010933404324.
[28] J.H. Friedman, Greedy function approximation: A gradient boosting machine., Ann. Statist. 29 (2001). https://doi.org/10.1214/aos/1013203451.
[29] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach Learn 63 (2006) 3–42. https://doi.org/10.1007/s10994-006-6226-1.
[30] D.E. Rumelhart, J.L. McClelland, Learning Internal Representations by Error Propagation, in: Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations, MIT Press, 1987: pp. 318–362. https://ieeexplore.ieee.org/document/6302929 (accessed July 8, 2025).
[31] M.E.A. Ben Seghier, D. Höche, M. Zheludkevich, Prediction of the internal corrosion rate for oil and gas pipeline: Implementation of ensemble learning techniques, Journal of Natural Gas Science and Engineering 99 (2022) 104425. https://doi.org/10.1016/j.jngse.2022.104425.
[32] S. Peng, Z. Zhang, E. Liu, W. Liu, W. Qiao, A new hybrid algorithm model for prediction of internal corrosion rate of multiphase pipeline, Journal of Natural Gas Science and Engineering 85 (2021) 103716. https://doi.org/10.1016/j.jngse.2020.103716.
[33] D.-C. Feng, W.-J. Wang, S. Mangalathu, G. Hu, T. Wu, Implementing ensemble learning methods to predict the shear strength of RC deep beams with/without web reinforcements, Engineering Structures 235 (2021) 111979. https://doi.org/10.1016/j.engstruct.2021.111979.

[34] C.I. Ossai, A Data-Driven Machine Learning Approach for Corrosion Risk Assessment—A Comparative Study, BDCC 3 (2019) 28. https://doi.org/10.3390/bdcc3020028.
[35] S. Nesić, J. Postlethwaite, M. Vrhovac, CO2 CORROSION OF CARBON STEEL - FROM MECHANISTIC TO EMPIRICAL MODELLING, Corrosion Reviews 15 (1997) 211–240. https://doi.org/10.1515/CORRREV.1997.15.1-2.211.
[36] G. De Masi, M. Gentile, R. Vichi, R. Bruschi, G. Gabetta, Machine learning approach to corrosion assessment in subsea pipelines, in: OCEANS 2015 - Genova, IEEE, Genova, Italy, 2015: pp. 1–6. https://doi.org/10.1109/OCEANS-Genova.2015.7271592.
[37] L.N. Smith, A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay, (2018). http://arxiv.org/abs/1803.09820 (accessed April 15, 2024).
[38] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks 4 (1991) 251–257. https://doi.org/10.1016/0893-6080(91)90009-T.
[39] N.-K. Kim, D.-H. Kim, D.-W. Kim, H.-G. Kim, D.A. Lowther, J.K. Sykulski, Robust Optimization Utilizing the Second-Order Design Sensitivity Information, IEEE Trans. Magn. 46 (2010) 3117–3120. https://doi.org/10.1109/tmag.2010.2043719.
[40] tkinter — Python interface to Tcl/Tk, Python Documentation (n.d.). https://docs.python.org/3/library/tkinter.html (accessed November 18, 2025).
[41] streamlit/streamlit, (2025). https://github.com/streamlit/streamlit (accessed November 18, 2025).