applied sciences

MDPI

Article

Improving Hardenability Modeling: A Bayesian Optimization Approach to Tuning Hyperparameters for Neural Network Regression

Wendimu Fanta Gemechu [1], Wojciech Sitek [1,*] and Gilmar Ferreira Batalha [2]

[1] Scientific and Didactic Laboratory of Nanotechnology and Materials Technologies, Silesian University of Technology, 44-100 Gliwice, Poland; wendimu.gemechu@polsl.pl
[2] Polytechnic School, University of Sao Paulo, Sao Paulo 05508-010, Brazil; gilmar.batalha@poli.usp.br
* Correspondence: wojciech.sitek@polsl.pl

Abstract: This study investigates the application of regression neural networks, particularly the *fitrnet* model, in predicting the hardness of steels. The experiments involve extensive tuning of hyperparameters using Bayesian optimization and employ 5-fold and 10-fold cross-validation schemes. The trained models are rigorously evaluated, and their performances are compared using various metrics, such as mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE), and coefficient of determination (R^2). The results provide valuable insights into the models' effectiveness and their ability to generalize to unseen data. In particular, Model 4208 (8-85-141-1) emerges as the top performer with an impressive RMSE of 1.0790 and an R^2 of 0.9900. The model, which was trained with different datasets for nearly 40 steel grades, enables the prediction of hardenability curves, but is limited to the range of the training dataset. The research paper contains an illustrative example that demonstrates the practical application of the developed model in determining the hardenability band for a specific steel grade and shows the effectiveness of the model in predicting and optimizing heat treatment results.

Keywords: steel hardenability; Bayesian optimization; k-fold cross-validation; hyperparameter tuning; neural network regression; steel alloy; predictive model; heat treatment; hardenability band; Jominy end-quench

Citation: Gemechu, W.F.; Sitek, W.; Batalha, G.F. Improving Hardenability Modeling: A Bayesian Optimization Approach to Tuning Hyperparameters for Neural Network Regression. *Appl. Sci.* **2024**, *14*, 2554. https://doi.org/10.3390/app14062554

Academic Editors: Chia-Ming Fan, Jakub Krzysztof Grabski and Po-Wei Li

Received: 4 February 2024
Revised: 11 March 2024
Accepted: 15 March 2024
Published: 18 March 2024

1. Introduction

Remarkable progress has been made in the development of methods and tools for modeling and simulating the production, processing, and structural properties of steels and metal alloys. Computational modeling, a cost-effective approach to optimize certain factors, such as chemical composition and process conditions, is widely used in scientific and industrial research and helps to achieve the desired properties in metal materials [1–4].

A more cost-effective approach for evaluating the hardness of continuously cooled steel from the austenitizing temperature is the Jominy end-quench test. Integrating the results of this test into models used for heat treatment simulations requires the computation of cooling rates at specific locations on the cooled object, associating them with respective distances from the quenched end of the sample [5]. The techniques for computing Jominy hardenability curves are described in detail in various studies, including [6–11].

Modern steelmaking techniques allow precise regulation of the chemical composition and hardenability, with some manufacturers advocating strict limitations on hardenability. The widely used Jominy end-quench hardenability test is an important tool in the production, specification, procurement, and application of heat-treatable structural steels, which are critical to modern transportation, construction, and agricultural machinery. The delineation of hardenability bands and the associated metallurgical methods, which link

hardenability to heat treatment response, microstructure evolution, and eventual mechanical properties, form the basis for the economically viable selection of steels and the design of components [12].

The hardenability of steel depends on its composition, subject to specific limits defined for each steel grade. Despite compliance with the compositional specifications, a typical variation in hardenability can occur for each steel grade. In some instances, tighter compositional control is essential for the applications for which the steel is used. Therefore, several steels are offered in H-grade variants with tighter compositional control. The extent of control is precisely defined by the maximum and minimum hardenability limits [13].

In the field of materials engineering, there is a growing interest in using artificial and computational intelligence [4,14,15]. The increasing accessibility of material databases and advances in machine learning open up new possibilities for the prediction of material properties and the development of next-generation materials [16–19]. One of the best-known methods of computational intelligence is the use of artificial neural networks (ANN) [20–22]. Artificial neural networks (ANNs) are invaluable for practical applications as they overcome the hurdles associated with formulating mathematical models. Their ability to establish connections between studied variables without the need for explicit mathematical descriptions is a distinctive feature that enables ANNs to learn solutions to problems based on identified patterns from the given experimentally collected data [1,23–25].

Artificial neural networks (ANNs), especially multilayer perceptron networks (MLPs), are widely used in modeling steel and metal alloy problems due to their efficiency in handling classification and regression tasks. ANNs are characterized by their ability to learn from labeled dataset and are, therefore, well suited for supervised learning applications [26–28]. The critical requirement for the development of an effective neural model is the creation of a representative dataset. This requires quantifying various requirements, such as data availability, labelling accuracy, and clustering adequacy. For example, the dataset must cover a comprehensive range of variables, and their statistical distribution must be thoroughly evaluated. In addition, the dataset must be standardized (centering and scaling to have zero mean and unit standard deviation) before training the neural network to ensure consistent scales for all variables [29,30]. Proper training of the neural network requires the representation of patterns that evenly cover the entire range of variables. To ensure this, the value range of the variables must be determined, and their statistical distribution must be evaluated. The analysis emphasizes the importance of precisely defining the range of independent variables for neural models to avoid errors when extrapolating beyond the range of the training data, especially in multidimensional input domains. Selecting the optimal number of neurons in the hidden layer of an MLP network is about finding a balance between approximation and generalization, with overfitting being a well-studied problem. There are different approaches to determine the optimal number of neurons, often favoring the lowest error value, although the arbitrary application of this criterion may increase the risk of overfitting. The evaluation of neural models relies heavily on a comprehensive test set that adequately represents the full range of the model. Statistical values for the test set, such as the mean absolute error and the correlation coefficient, should match those of the training set and, thus, provide a crucial insight into the quality of the model [1,31].

Splitting a dataset into different subsets for training and validation is a fundamental aspect of machine learning. It plays a crucial role in various tasks, such as model evaluation, model comparison, and hyperparameter tuning. Common methods, such as holdout, bootstrap, and cross-validation (CV), are often used [32]. In this method, the available dataset is divided into two different subsets: a training set, which is used to determine the model parameters, and a separate validation set (test set), also known as the hold-out set or development set. The training set is crucial for model parameterization, while the validation set serves as an independent dataset for evaluating the model's performance. The selection criterion for the final model is to select the one with the fewest errors in

the validation set. This approach ensures that the selected model generalizes well to new, unseen data [24].

Cross-validation (CV) is a well-known resampling method that is widely used in statistical learning methods. It serves as an essential tool for estimating the test error associated with a particular statistical learning method and makes an important contribution to model evaluation and selection. Evaluating the overall performance of a model and selecting an appropriate degree of flexibility are integral aspects of cross-validation that are essential for refining and optimizing statistical learning models [33].

The selection of k in k-fold cross-validation involves a trade-off between bias and variance. The parameter k determines the number of folds into which the dataset is divided and influences the estimation of the model's test error. It is expected to use k = 5 or k = 10, as empirical evidence suggests that these values lead to a balance yielding test error rate estimates with moderate bias and variance. This choice helps to reduce the risk of underfitting (high bias) or overfitting (high variance) in the model evaluation process [33]. Furthermore, a comparison between 5-fold cross-validation and 10-fold cross-validation shows that 10-fold cross-validation generally performs better in regression scenarios [34].

Bayesian optimization (BO), known for its efficiency in optimizing expensive black-box functions, has attracted considerable attention, especially in the field of hyperparameter optimization. The basic concept involves approximating the unknown function, typically using Gaussian processes, with initial random estimates and iterative updates with each new observation. Success in navigating a multidimensional space with numerous hyperparameters depends on an appropriate number of iterations and careful selection of hyperparameter ranges, which are influenced by such factors as the size of the dataset, expert judgment, intuition, and computational resources. Since most machine learning algorithms require the configuration of a range of hyperparameters, the careful selection of their values has a significant impact on performance. To streamline the time-consuming and non-reproducible manual trial-and-error process of determining the optimal hyperparameter configurations, automatic hyperparameter optimization through Bayesian optimization can be used [35,36].

With limited datasets, BO proves to be more advantageous than a grid search for tuning hyperparameters. The effectiveness of BO results from the use of a probabilistic model, which enables a more efficient search for optimal hyperparameters at a lower computational cost. Bayesian tuning of hyperparameters is excellent for exploring the hyperparameter space and strengthening the robustness of machine learning models, especially in scenarios with small datasets. Initially, the BO approach searches for the initial set of hyperparameters through various combinations, guided by considerations of the problem space and presumptions about the potential impact of the model hyperparameters. After determining the initial hyperparameters, BO employs a probabilistic model to construct a surrogate model that facilitates the estimation of model performance in a large hyperparameter space. The model is then trained with the initial set of hyperparameters. This iterative process of model training and evaluation with different hyperparameters allows BO to accumulate more data points and, thus, improves the performance and accuracy of the surrogate model. As the surrogate model evolves, the algorithm becomes more adept at making informed decisions about where to look for the best hyperparameters. This iterative refinement process proves invaluable in identifying optimal hyperparameters within limited datasets, leading to improved model performance [36–38].

During the execution of an experiment, the Experiment Manager actively searches for the optimal combination of hyperparameters. Using a trial-and-error approach, a new set of hyperparameters is tested at each iteration of the experiment, considering the results of the previous experiments. Bayesian optimization plays a central role in this process by gradually building a probabilistic model of the objective function. This iterative method guides the selection of subsequent sets of hyperparameters, continuously refining the model's estimate of the function's behavior. Ultimately, this iterative strategy facilitates

the identification of the most advantageous hyperparameter configuration for a machine learning model.

2. Materials and Methods

Feed-forward neural networks, a subtype of artificial neural networks, are used for modeling classification and regression problems. A two-layer feed-forward network, often referred to as a shallow neural network, consists of an input layer, a hidden layer, and an output layer. In this architecture, the input layer receives data and the hidden layer, utilizing an activation function, applies weights and biases to the inputs. The final output is generated by the output layer. The hidden layer is critical for capturing complex patterns in the data and the network refines its understanding by adjusting the weights and biases during the training process. Despite their relative simplicity compared to deeper architectures, two-layer feed-forward networks can prove effective for certain problems, especially when dealing with less complex data or limited computational resources.

A *RegressionNeuralNetwork* object is a trained, feed-forward, and fully connected neural network for regression. The first fully connected layer of the neural network has a connection from the network input (predictors), and each subsequent layer has a connection from the previous layer, see Figure 1. Each fully connected layer multiplies the input by a weight matrix (*LayerWeights*) and then adds a bias vector (*LayerBiases*). An activation function follows each fully connected layer, except for the last layer (activations and *OutputLayerActivation*). The last fully connected layer generates the output of the network, namely the predicted response values. The solver utilized for training the neural network model is referred to as "LBFGS". In the context of creating the *RegressionNeuralNetwork* model, *fitrnet* employs the limited-memory Broyden–Fletcher–Goldfarb–Shanno quasi-Newton algorithm (LBFGS) [39] as an optimization algorithm to minimize the loss function. Specifically, the software aims to minimize the mean squared error (MSE) during the training process [29].

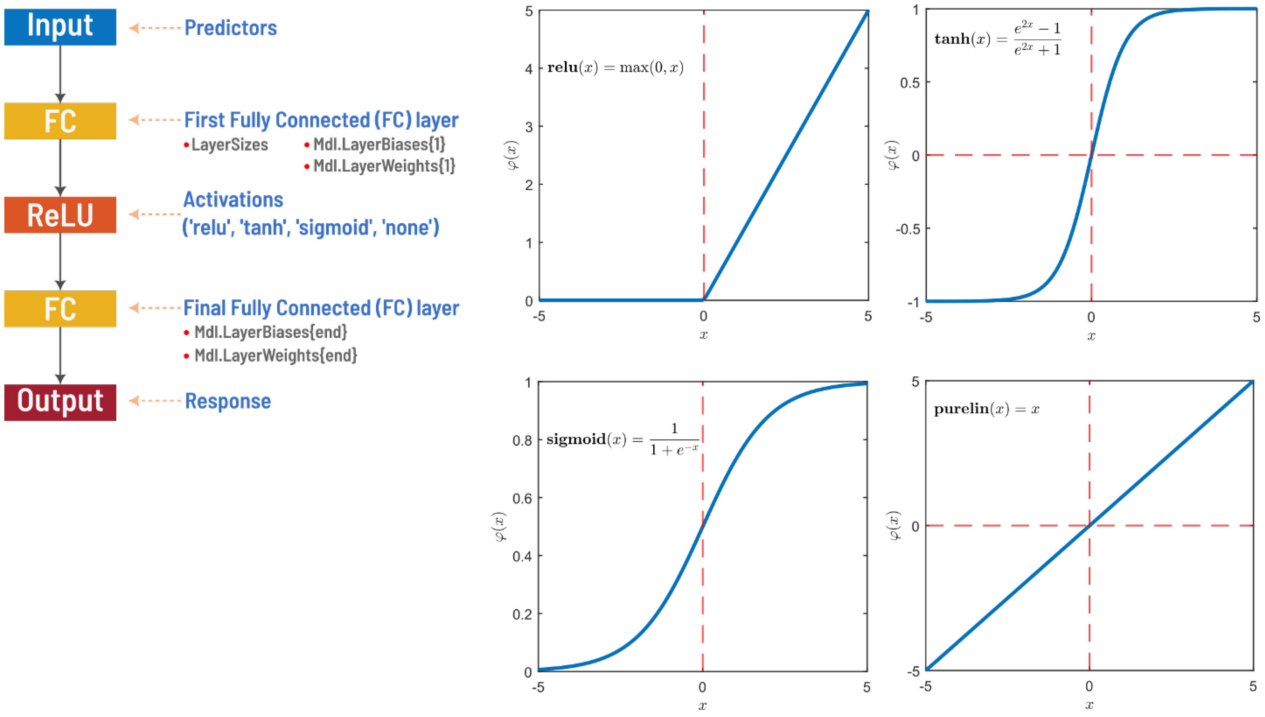

Figure 1. Typical feed-forward, fully connected neural networks for regression [29] (**left**) and activation functions used in the modeling task (**right**).

Bayesian optimization is an iterative algorithm with a probabilistic surrogate model and an acquisition function to select the next evaluation point. Each iteration involves

fitting the surrogate model to all past observations of the objective function. The acquisition function uses the predictive distribution of the probabilistic model to guide the selection of candidate points and balance exploration and exploitation. This approach, which is less costly than directly evaluating the expensive black-box function, allows for comprehensive optimization of the acquisition function. Although various acquisition functions are available, the expected improvement (EI) is often preferred because it can be calculated analytically if the model prediction y for the configuration λ follows a normal distribution, as in the following Equations (1) and (2) [40–42]:

$$\mathbb{E}[\mathbb{I}(\lambda)] = \mathbb{E}\lfloor max(f_{min} - y, 0)\rfloor \tag{1}$$

$$\mathbb{E}[\mathbb{I}(\lambda)] = (f_{min} - \mu(\lambda))\Phi\left(\frac{f_{min} - \mu(\lambda)}{\sigma}\right) + \sigma\varnothing\left(\frac{f_{min} - \mu(\lambda)}{\sigma}\right) \tag{2}$$

where $\varnothing(\cdot)$ and $\Phi(\cdot)$ denote the standard normal density and standard normal distribution function, respectively, and f_{min} represents the best observed value thus far.

When searching for optimal values for the regression hyperparameters, MATLAB offers various optimization techniques, including *Bayesopt*, *grid search*, and *random search*, with Bayesian optimization (bayesopt) being the default setting. When selecting the hyperparameters for the subsequent experiment, any of the acquisition functions can be used, such as *expected improvement per second plus*, *expected improvement*, *expected improvement plus*, *expected improvement per second*, *lower confidence bound*, or *probability of improvement*, as the optimization option.

In supervised learning, a fundamental concern revolves around the accuracy of the resulting model. A critical challenge in this context is the phenomenon of overfitting [43], where a model fits perfectly on the dataset on which it was trained, but has difficulty generalizing effectively to new, unseen data. Conversely, simpler models, such as a least-squares line, are less susceptible to being influenced by inherent noise but may inadequately capture the relationship between variables, leading to underfitting. It is unlikely that both overfitting and underfitting models will generalize well, demonstrating the importance of finding a balance between the two models [44].

Assessing the generalization ability of a model raises an important question: How can we assess its performance on new data? The ideal approach is to evaluate the model with new data from the same population as the training dataset [45]. However, practical constraints often limit the ability to conduct independent validation studies. Therefore, it is advisable to first estimate the predictive performance before investing resources in external validation studies. Usually, this estimation is performed using resampling techniques, such as cross-validation [44].

There are many alternative CV schemes, and the k-fold CV is one of the most popular. The k-fold CV is a widely used resampling method in the practical application of statistical learning methods. The set of observations are randomly divided into k groups (folds) of approximately equal size. In each iteration, the first fold is designated as the validation set, and the model is trained on the remaining k − 1 folds. The mean square error (MSE) is then calculated for the observations in the remaining group. This process is repeated k times, resulting in k estimates of the test error (MSE_1, MSE_2,. . ., MSE_k). The final k-fold CV estimate is obtained by averaging these values, as in the following Equation (3) [33]:

$$CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{3}$$

Finally, the observations of the test set are used as an independent dataset to evaluate the model's performance and to select the model. The model with the smallest test RMSE and the smallest mean absolute error (MAE) is selected as the final model, as these metrics ensure that the selected model generalizes well to a new dataset. In addition to RMSE, mean square error (MSE), mean absolute error (MAE), and coefficient of determination (R^2)

are used to evaluate the model during the test performance analysis, as in the following Equations (4)–(7):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (4)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (5)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \quad (6)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (7)$$

where n is the total number of experimental observations, y_i is the experimental values for each of the runs, $\hat{y}_i$ is the predicted value for each of the observed values, and $\overline{y}$ is the mean of the experimental values of the runs.

We assessed the cross-validation loss of neural network regression models by examining various hyperparameter settings, including regularization strengths (lambda), activation functions ('relu', 'tanh', 'sigmoid', and 'none'), hidden layer sizes, and hidden node sizes. We then fine-tuned the hyperparameters using Bayesian optimization to find the optimal configuration that minimizes model error and improves performance.

2.1. Analysis of the Dataset

In this study, the prediction of the hardness of heat-treated steel based on chemical composition and Jominy distance is investigated. The relationship between the Jominy distance and hardness, often referred to as the hardenability curve, serves as the central aspect of the analysis. It is assumed that the experiments conform to ASTM A-255 standards [46] to ensure consistency and reliability in data collection. In particular, it is assumed that the heat treatment procedures for the steel specimens meet the optimum conditions specified in the ASTM guidelines [46] to ensure the consistency and reliability of the test setup. In addition, the grain size of the steel samples is standardized to a value of seven on the ASTM scale, which contributes to the uniformity and reproducibility of the heat treatment process [2].

The dataset contains data on measurements on the Rockwell hardness scale C at distances of 1.5, 3, 5, 7, 9, 11, 13, 15, 20, 25, 30, 40, and 50 mm. It also contains information on the percentage of mass fraction of concentration of the seven basic alloying elements found in the group of alloyed structural alloy steel, namely C, Mn, Si, Cr, Ni, Mo, and Cu. Table 1 outlines the ranges of predictor parameters used in modeling the hardenability of steels.

Table 1. Ranges of predictor parameters used to model hardenability of steels.

Range (% Mass)	C	Mn	Si	Cr	Ni	Mo	Cu	Jominy Distance (mm)
Minimum	0.12	0.36	0.12	0.09	0.04	0.0098	0.07	1.50
Maximum	0.70	1.40	0.41	1.92	2.739	0.43	0.34	50

In MATLAB, the *cvpartition* function is a versatile tool for creating data partitions that is often used in machine learning and statistical analysis tasks. A typical application of this function is holdout validation, where the dataset is partitioned into a training set and a test set (or holdout set). The syntax *cvpartition(n, 'HoldOut', p)* generates a random, non-stratified partition for holdout validation on a dataset with *n* observations. Here, *p* represents the proportion of observations assigned to the test set, while the remaining data form the training set. This method enables straightforward and efficient data partitioning for training and evaluation purposes in machine learning workflows [29].

Table 2 presents the descriptive statistics of the training and test datasets. It provides a comprehensive insight into the characteristics of the datasets and illustrates their distribution patterns and variability. In it, we examine critical statistical parameters, namely the maximum, minimum, mean, and standard deviation of the percentage mass fractions of seven elemental constituents and the Jominy distance, an indicator of hardenability. In this way, we examine the spectrum, central tendency, and scattering tendencies of the alloy compositions in the training and test datasets.

Table 2. Descriptive statistics of the training and test datasets used in modeling the hardenability of steels.

Dataset Statistics	Datasets	Predictors							
		C	Mn	Si	Cr	Ni	Mo	Cu	Jominy Distance
Maximum	Training	0.7	1.4	0.41	1.92	2.739	0.43	0.34	50
	Test	0.7	1.4	0.41	1.92	2.739	0.43	0.34	50
Minimum	Training	0.12	0.36	0.12	0.09	0.04	0.0098	0.07	1.5
	Test	0.12	0.36	0.12	0.09	0.04	0.0098	0.07	1.5
Average	Training	0.28	0.83	0.26	1.00	0.47	0.12	0.17	17.54
	Test	0.28	0.82	0.26	1.01	0.50	0.12	0.16	18.28
Standard Deviation	Training	0.12	0.26	0.04	0.33	0.58	0.09	0.05	14.18
	Test	0.13	0.26	0.04	0.34	0.60	0.09	0.05	14.89

The goal of predictive hardenability modeling is to create a model based on the training dataset and then to apply this model to the test dataset. However, to achieve the best results, the training dataset must be a representative sample of the data we want to apply it to (i.e., the test dataset). Otherwise, our model will be inadequate at best or utterly useless at worst. The Kolmogorov–Smirnov test checks whether the distributions of the two samples, namely the training and test datasets, differ significantly. The null hypothesis is that the samples come from the same distribution [47]. Table 3 presents the results of the Kolmogorov–Smirnov test, which is used to assess the goodness of fit of the training and test datasets.

Table 3. Kolmogorov–Smirnov test results of goodness of fit of the training and test datasets.

Features	C	Mn	Si	Cr	Ni	Mo	Cu	Jominy Distance	Hardness
KS test value	0.026	0.025	0.027	0.026	0.022	0.027	0.020	0.033	0.023
p-value	0.65	0.70	0.62	0.67	0.82	0.60	0.91	0.37	0.81

To assess the similarity of the feature distributions between the training and test datasets, we performed Kolmogorov–Smirnov (KS) tests at a 5% significance level for each feature. The KS test values, representing the maximum absolute differences between the cumulative distribution functions, were calculated for the features C, Mn, Si, Cr, Ni, Mo, Cu, Jominy distance, and hardness (HRC). The KS test values obtained are between 0.020 and 0.033, indicating relatively small differences between the distributions. In addition, *p*-values were calculated for each test, ranging from 0.37 to 0.91. Significantly, all calculated *p*-values were relatively high, indicating no substantial evidence to reject the null hypothesis of similarity between the distributions. These results provide confidence in the similarity of the feature distributions between the training and test datasets and confirm the robustness of our partitioning of the dataset for training and evaluating the models, i.e., the model is likely to generalize well to the unseen test dataset, see Figure 2.

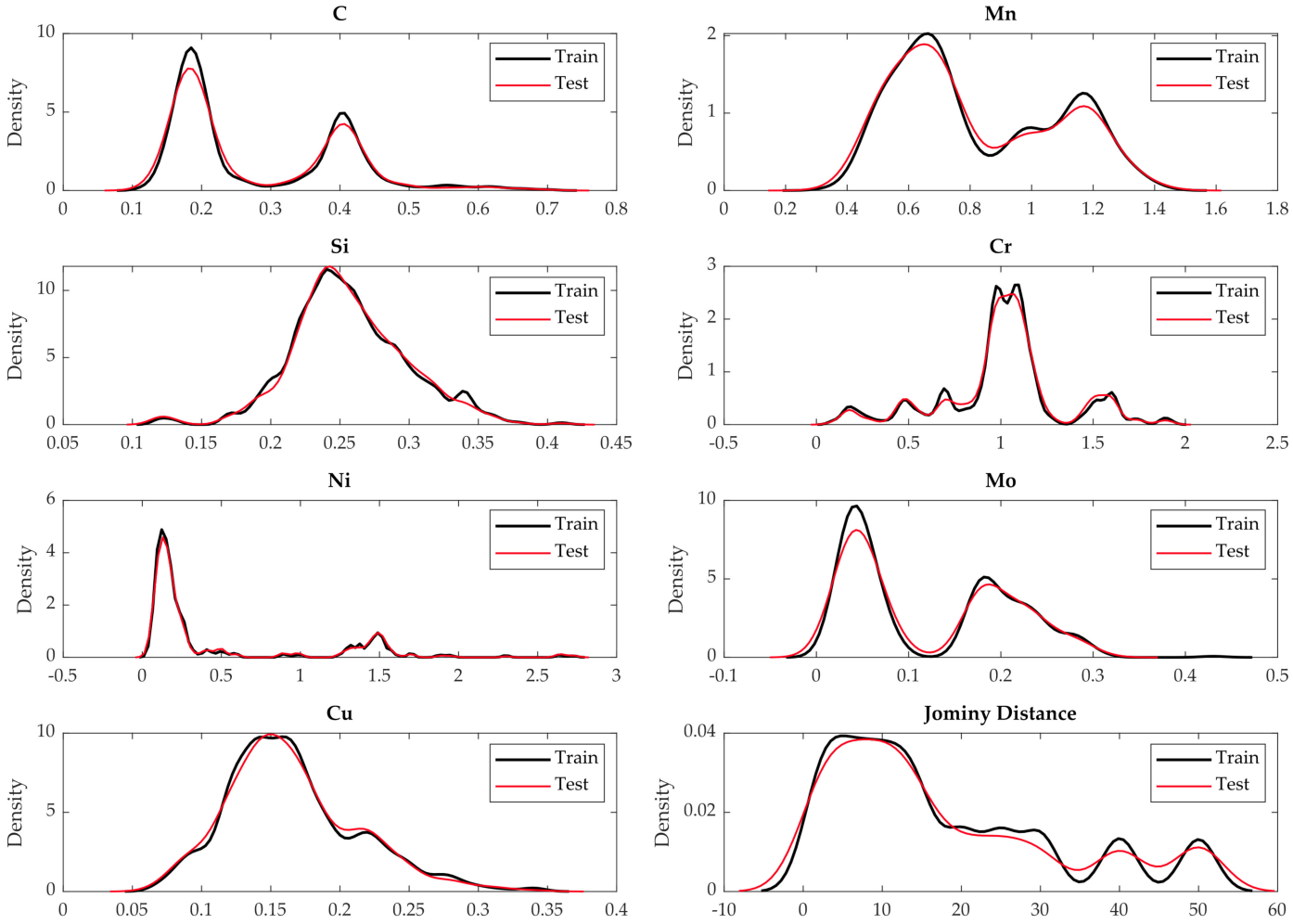

Figure 2. Training and test datasets' split distribution.

2.2. Experimental Setup

The experiments were performed with MATLAB R2023b using the Statistics and Machine Learning Toolbox (MATLAB 23.2, R2023b) for custom training experiments for machine learning and Bayesian optimization. In addition, the Parallel Computing Toolbox (MATLAB 23.2, R2023b) facilitated the simultaneous execution of multiple trials. The MATLAB Experiment Manager was crucial for planning and running experiments to train and compare the neural network models. In the Experiment Manager, a set of hyperparameters were explored using Bayesian optimization, with each experiment generating a set of results for each run. These results comprised 1500 trials, each representing a distinct combination of hyperparameters. The pseudocode in Appendix A outlines the process for training a regression neural network model, tuning the hyperparameters, and computing associated metrics in the experimental settings.

The parameters optimized during the modeling process are 'Activations', 'Lambda', 'LayerSizes', and 'Standardize'. These parameters are options for the optimization of hyperparameters in connection with the fitting of a neural network regression model (*fitrnet*). The specific functions and effects of each parameter are explained below:

1. *Activations*: This parameter refers to the choice of activation functions used in the neural network layers. Activation functions introduce non-linearity into the neural network and allow it to learn complex relationships in the data.
2. *Lambda*: This parameter represents the strength of regularization, often referred to as λ (lambda), which penalizes significant coefficients in the neural network model. Regularization helps prevent overfitting by discouraging overly complex models.

3. *LayerSizes*: This parameter refers to the size (number of neurons) of the hidden layers in the neural network. The selection of appropriate layer sizes is crucial for the balance between model complexity and performance.
4. *Standardize*: This parameter specifies whether or not each numeric predictor variable should be standardized before fitting the neural network. When standardizing, the predictor variables are scaled to have a mean of zero and a unit standard deviation. This can improve convergence and performance, especially if the predictors have different scales.

There are numerous normalization methods, including statistical normalization, standardization, sigmoidal normalization, and others. In this study, standardization is used because it results in consistent variances in the features and normalizes the input features to the same range [48].

In hyperparameter optimization, the goal is to find the combination of these parameters that minimizes the cross-validation loss and, thus, optimizes the performance of the neural network model for the task at hand [29].

Five distinct experiments with different hyperparameter settings were conducted for the hyperparameter optimization process. Using the BO methods of hyperparameter optimization and cross-validation schemes, different combinations of hyperparameters were investigated in the experiments, including the number of layers, activation functions, standardization techniques, regularization parameters and layer sizes. Systematically varying these parameters and performing 5-fold and 10-fold cross-validation experiments, the study aimed to identify optimal model configurations that balance model complexity and generalizability, thereby improving the accuracy of predicting hardenability. The experiments were run for 1500 trials, and validation RMSE was recorded as the performance metric.

1. Experiment 1 (10-fold CV): NumLayers [1, 2], Activations ['relu', 'tanh', 'sigmoid', 'none'], Standardize ['true', 'false'], lambda [1.9172×10^{-9}, 0.1], Layer 1 [1, 300] and Layer 2 [1, 300].
2. Experiment 2 (5-fold CV): NumLayers [1, 2], Activations ['relu', 'tanh', 'sigmoid', 'none'], Standardize ['true', 'false'], lambda [1.9172×10^{-9}, 0.1], Layer 1 [1, 300] and Layer 2 [1, 300].
3. Experiment 3 (10-fold CV): Layer 1 [1, 300], Activations ['relu', 'tanh', 'sigmoid', 'none'], Standardize ['true', 'false'] and lambda [1.9172×10^{-9}, 0.1].
4. Experiment 4 (10-fold CV): Layer 1 [1, 100], Activations ['relu', 'tanh', 'sigmoid'], Standardize ['true', 'false'] and lambda [1.9172×10^{-9}, 0.1].
5. Experiment 5 (10-fold CV): Layer 1 [1, 30], Activations ['relu', 'tanh', 'sigmoid'], Standardize ['true', 'false'] and lambda [1.9172×10^{-9}, 0.1].

For the tuning of the hyperparameter process, 5-fold and 10-fold CV were employed to protect the model against overfitting. Experiment 1 and Experiment 2 have the same hyperparameter settings with different cross-validation folds.

In this study, the main dataset of 6136 observations is randomly split into two sets: A training dataset contains 85% of the observations and is used to determine the model parameters and hyperparameters, and a separate model test dataset contains the remaining 15% of the observations. The training dataset is then used for k-fold cross-validation during the training and validation phase of model development. The training dataset containing the observations is randomly split into k mutually exclusive folds of approximately equal size. In each iteration, the first fold is determined as the validation set, and the model is trained on the remaining $k - 1$ folds. The root mean square error (RMSE) is then calculated for the observations in the remaining fold. This process is repeated k times, resulting in k estimates of the validation error. The average of these results is the final estimate of the k-fold CV and is reported as the validation RMSE of the model.

3. Results

This section provides a comprehensive overview of model training and tuning, performance analysis, and model evaluation and selection to obtain the best-performing model for prediction and deployment. The performance of the various neural network configurations was evaluated using the root mean square error (RMSE) and the configuration with the lowest RMSE was selected for subsequent model testing using a separate test dataset. The tuned models were thoroughly evaluated to determine the most appropriate configuration for accurate hardness prediction. All models used standardized datasets and the sigmoid activation function that appeared to be effective for the modeling task.

3.1. Experimental Results Analysis

Table 4 and Figure 3 presents five optimal models determined by Bayesian optimization of the hyperparameters in five different experiments, and their performance evaluations. These results indicate the performance of each model. The RMSE and R^2 values for both the training and test datasets show how well the models generalize in relation to unseen dataset.

Table 4. Optimal models were determined through Bayesian optimization of the hyperparameters in the five experiments.

Model	Architecture	Lambda	RMSE		R^2		MSE	MAE
			Train	Test	Train	Test	Test	Test
I (10-fold CV)	8-110-104-1	0.0001	1.0324	1.1201	0.9914	0.9892	1.2546	0.7214
II (5-fold CV)	8-93-120-1	0.0002	1.0690	1.0976	0.9907	0.9896	1.2046	0.7101
III (10-fold CV)	8-298-1	1.0816×10^{-7}	1.1544	1.3222	0.9892	0.9849	1.7483	0.8560
IV (10-fold CV)	8-100-1	0.0001	1.2083	1.3529	0.9882	0.9842	1.8305	0.8897
V (10-fold CV)	8-30-1	2.367×10^{-5}	1.5505	1.6299	0.9805	0.9771	2.6565	1.1553

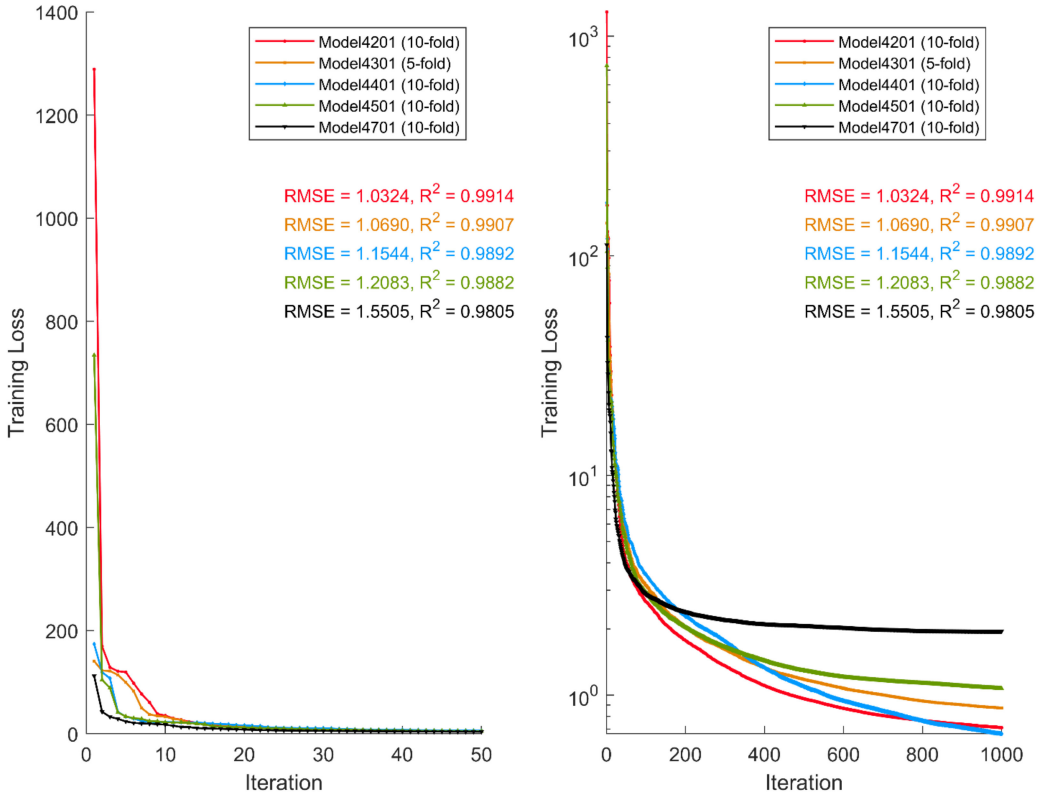

Figure 3. Comparison of the training loss of the best five models from the five experimental runs.

All models perform well, as shown by the relatively low RMSE values for both the training and test datasets. The R^2 values, which represent the proportion of variance explained, are consistently high, indicating that the models capture a significant portion of the variability in the data.

Model II is characterized by the lowest RMSE in the test dataset (1.0976), indicating a higher predictive accuracy compared to the other models. The performance of the model is shown in Figure 4 for more clarification. Model V has shown the highest RMSE on the test dataset (1.6299), indicating that it is less effective when generalizing to new data.

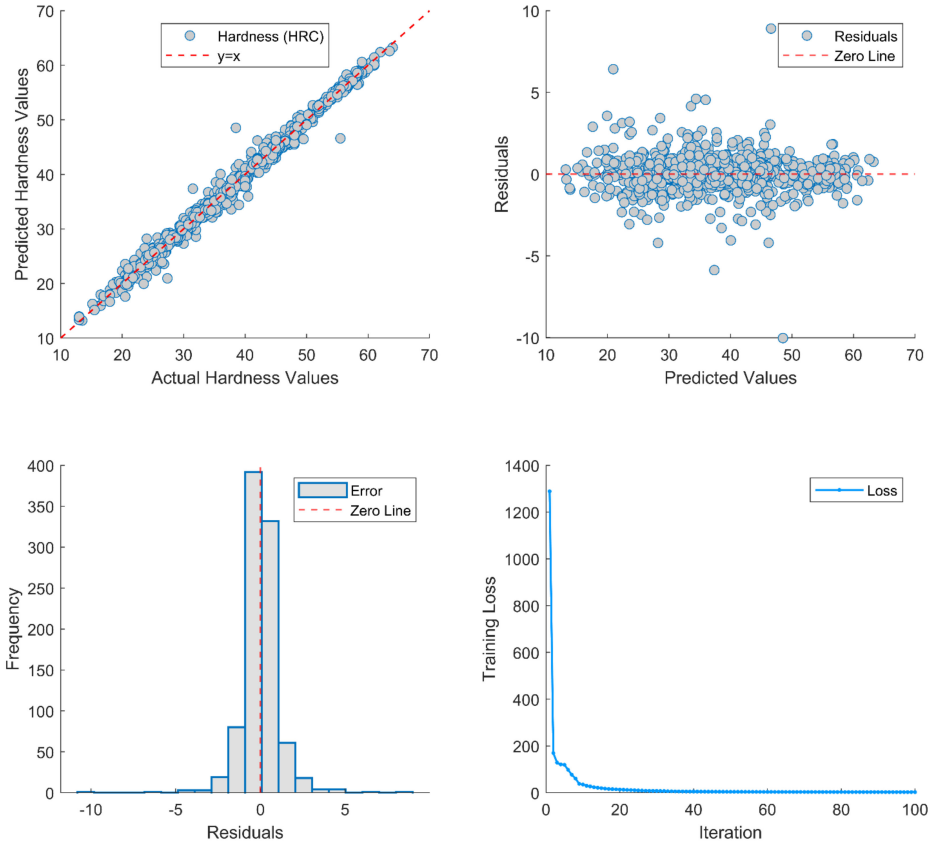

Figure 4. Performance evaluation of Model II (model4301) on an unseen new test dataset.

The differences in architecture (number of hidden layers and nodes) between the models do not consistently correlate with performance. Models I, III, IV, and V show similar performance values despite their different architectures. The lambda values for regularization vary, indicating the influence of regularization strength on model performance.

Of the top models from each experiment, Model II proves to be the most promising, as it has the lowest RMSE for the test set and high R^2 values. It appears to be the best-performing model among the options offered and shows superior predictive accuracy for the new unseen test set.

3.2. Experiment 1 and Experiment 2 Comparative Analysis

In Experiment 1, a 10-fold cross-validation approach was used to evaluate the performance of the model under different hyperparameter configurations. The parameters considered include the number of layers (*NumLayers*) from 1 to 2, activation functions (activations) including 'relu', 'tanh', 'sigmoid' and 'none', standardization (standardize) options of 'true' or 'false', regularization parameter (lambda) with values between 1.9172×10^{-9} and 0.1, and the sizes of Layer 1 and Layer 2 ranging from 1 to 300. In Experiment 2, a 5-fold cross-validation was performed to evaluate the robustness of the model under the same hyperparameter variations. These experiments aimed to systematically investigate

the effects of different configurations on the predictive performance of the model and to gain insights into the optimal hyperparameter settings for each regression task.

The process of identifying the optimal model for predicting hardness in the regression task involved three important steps. First, the fifteen best-performing models from each experiment were selected, highlighting their performance in the test datasets. Then, the selection was refined to the five best models. The results are shown in Tables 5 and 6. In the final stage, a comparative analysis was performed between the best model from each experiment and the best models from the training datasets. Ultimately, the most effective test model was identified for subsequent prediction and deployment.

Table 5. The five best models from Experiment 1 (10-fold CV).

Models	Activation	Lambda	L1	L2	RMSE		R^2		MSE	MAE
					Train	Test	Train	Test	Test	Test
Model 4204	'sigmoid'	0.000214222	125	148	1.0388	1.1125	0.9913	0.9893	1.2376	0.7407
Model 4208	'sigmoid'	6.93×10^{-5}	85	141	1.0426	1.0790	0.9912	0.9900	1.1643	0.6938
Model 4210	'tanh'	0.001612633	82	154	1.0451	1.1023	0.9912	0.9895	1.2150	0.6988
Model 4211	'tanh'	0.00199953	74	175	1.0460	1.1083	0.9911	0.9894	1.2284	0.7068
Model 4212	'sigmoid'	7.22×10^{-5}	73	112	1.0463	1.0963	0.9911	0.9896	1.2019	0.7184

Table 6. The five best models from Experiment 1 (5-fold CV).

Models	Activation	Lambda	L1	L2	RMSE		R^2		MSE	MAE
					Train	Test	Train	Test	Test	Test
Model 4301	'sigmoid'	0.000185376	93	120	1.0690	1.0976	0.9907	0.9896	1.2046	0.7101
Model 4303	'sigmoid'	0.000136655	116	112	1.0768	1.0826	0.9906	0.9899	1.1720	0.6885
Model 4307	'sigmoid'	0.000224495	82	123	1.0968	1.1109	0.9903	0.9894	1.2340	0.7269
Model 4308	'sigmoid'	8.70×10^{-5}	91	153	1.0971	1.0830	0.9903	0.9899	1.1729	0.6937
Model 4314	'sigmoid'	0.000146841	87	167	1.1033	1.0982	0.9901	0.9896	1.2059	0.7040

These results provide a comprehensive insight into the performance of the models under different cross-validation schemes and help to select the most effective models for predicting hardness in the regression task.

Based on the metrics provided in Table 7 and Figures 5 and 6, Model 4208 appears to outperform the other models, except for in terms of the mean absolute error (MAE). It achieves a relatively low MSE of 1.1643, indicating its ability to minimize the squared differences between predicted and actual values. The root mean squared error (RMSE) is also comparatively low at 1.0790, indicating accurate predictions with less variability. Although the MAE is slightly higher compared to some other models, it is still reasonable at 0.6938, indicating a relatively small average absolute error.

Table 7. Comparing the results of the best model from each experiment and the best models from the training datasets.

Model	MSE	RMSE	MAE	R	R^2	STD	MaxAE
Model 4201	1.2546	1.1201	0.7214	0.9946	0.9892	1.1204	11.0010
Model 4208	1.1643	1.0790	0.6938	0.9951	0.9900	1.0790	9.8748
Model 4301	1.2046	1.0976	0.7101	0.9948	0.9896	1.0979	10.0270
Model 4303	1.1720	1.0826	0.6885	0.9950	0.9899	1.0827	11.1998

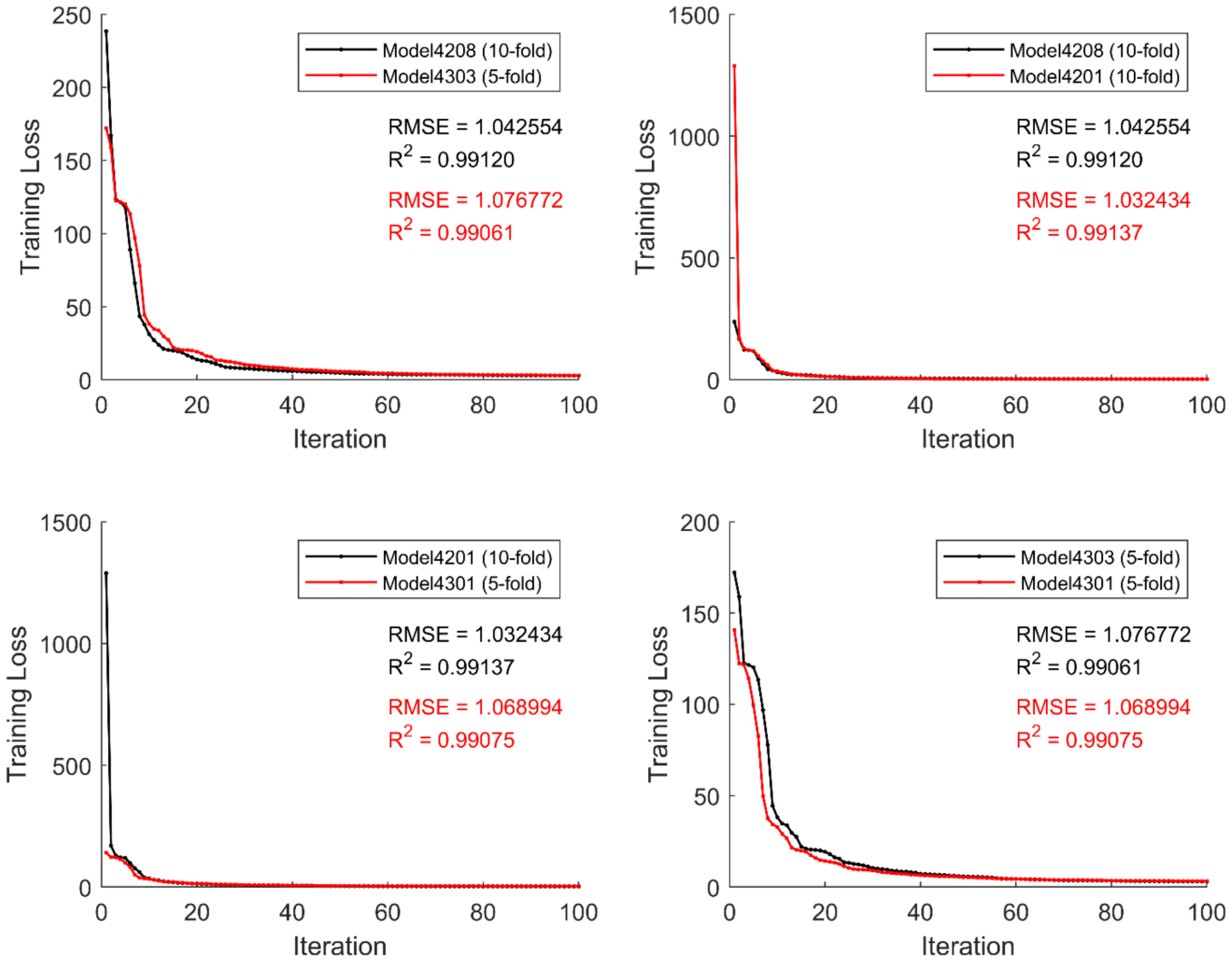

Figure 5. Comparison of training losses for the two best models from the training and test datasets.

In addition, Model 4208 has a high correlation coefficient (R) of 0.9950, indicating a strong linear relationship between the predicted and actual values. The coefficient of determination (R^2) is 0.98996, which means that the model captures a substantial portion of the variance in the data. The standard deviation (STD) is 1.07898, which indicates a relatively low dispersion of the residuals.

The table displays the performance metrics of different models in predicting hardness. Each model is assessed based on various metrics, including mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), correlation coefficient (R), coefficient of determination (R^2), standard deviation (STD), and maximum absolute error (MaxAE).

To summarize, Model 4208 is a promising candidate for hardness prediction. It shows a balanced performance on various metrics, with a particularly notable strength in minimizing the squared error and a high correlation with the actual data.

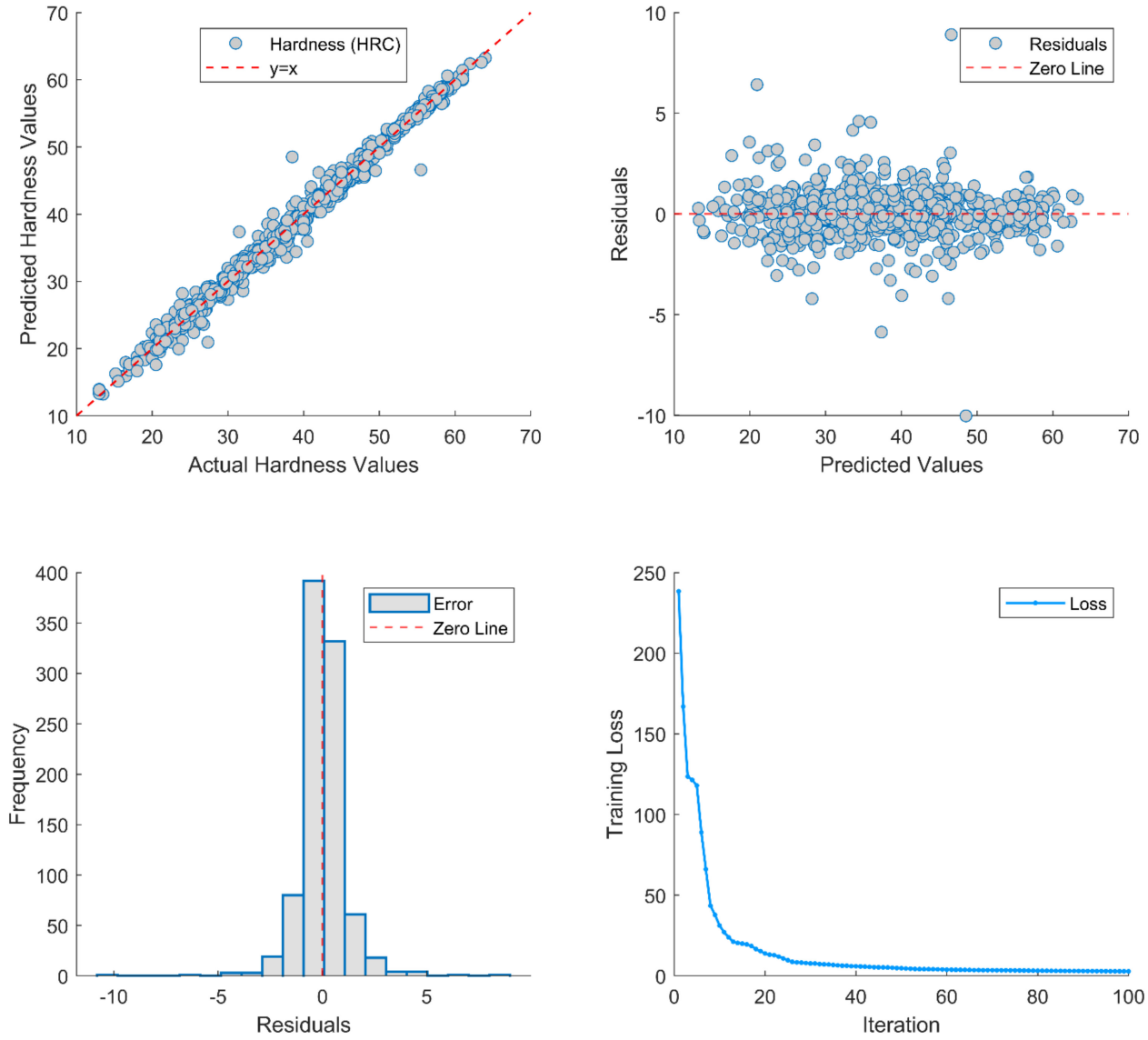

Figure 6. Performance evaluation of Model 4208 on an unseen new test dataset.

3.3. Illustrative Example of Model 4208 to Determine Hardenability Curves

To evaluate the performance of the model, the hardenability curves of the 41Cr4 and 42CrMo4 steel (according standard [49]) grades are compared by examining the experimental and predicted results.

In this particular example, the performance of the model is evaluated by comparing the predicted hardness values with the observed (experimental) values for the 41Cr4 and 42CrMo4 steel grades. As evident from Table 8 and Figure 7, the results show close agreement between the experimental and predicted values. The close agreement between the experimental and predicted hardness values indicates a high degree of accuracy and reliability in the model's ability to predict the hardenability of the steels. This robust predictive capability indicates the model's potential for optimizing heat treatment processes, assisting in material selection, and improving overall efficiency in the design and manufacture of steel components.

Table 8. Experimental and predicted hardness for 41Cr4 and 42CrMo4 steel grades.

Steel Grade		Distance in mm from Quenched End												
		Hardness in HRC												
		1.5	**3**	**5**	**7**	**9**	**11**	**13**	**15**	**20**	**25**	**30**	**40**	**50**
41Cr4	Exp.	58.50	57.50	56.75	55.25	54.00	51.75	49.25	45.50	40.50	38.00	37.75	29.00	24.00
	Pred.	58.38	57.71	56.57	55.49	54.05	51.69	48.70	45.61	40.64	38.34	36.39	29.39	23.90
42CrMo4	Exp.	56.33	55.33	54.17	53.50	52.00	49.00	45.17	42.00	37.67	35.17	34.00	31.83	28.67
	Pred.	56.17	55.55	54.55	53.57	51.94	48.91	45.41	42.33	37.57	35.42	33.96	31.24	29.02

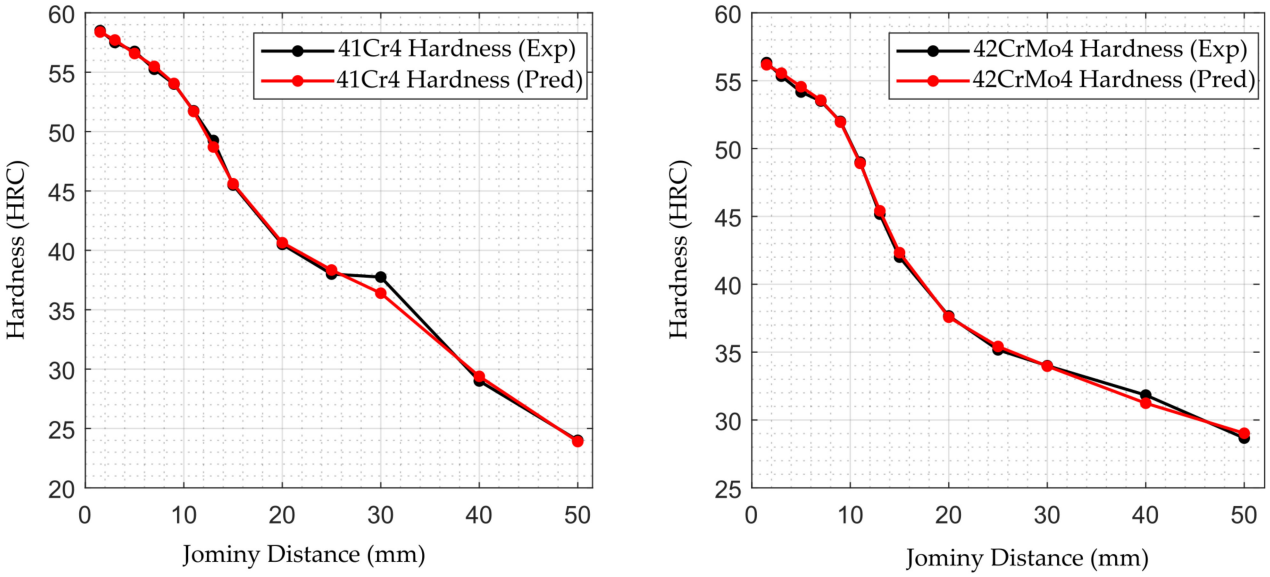

Figure 7. The target and predicted hardenability curves of 41Cr4 and 42CrMo4 steel grades.

3.4. Illustrative Example of Model 4208 to Determine Restricted Hardenability Band of 42CrMo4 Steel Grade

In this section, the use case of the model in determining the hardenability band is discussed. Determining the H-band offers numerous advantages, including optimized material selection, enhanced component performance, cost-effective manufacturing, prevention of defects and failures, improved consistency and quality control, tailored material properties, and customization to specific customer requirements. Moreover, it aids in restricting the chemical composition ranges for applications and ensures the desired performance characteristics.

Due to variations in chemical composition among different heats of the same steel grade, hardenability bands are defined using the Jominy end-quench test. The upper curve indicates maximum hardness values corresponding to the upper composition limits, while the lower curve represents minimum hardness values for the lower composition limits. Hardenability bands are valuable for both suppliers and customers, with most steels now purchased based on these bands. Suppliers ensure that a significant percentage, typically 93 or 95%, of mill heats meeting chemical specifications fall within the specified hardenability band [50].

The model can be used either manually or can be combined with Bayesian optimization or a genetic algorithm [14] to determine the limiting chemical compositions values within the required hardness stated by the customer. The integration of artificial neural networks with various modeling techniques, such as mathematical modeling, computational intelligence, and artificial intelligence is a common practice. The amalgamation of different methods within a single model enables the exploration of a larger problem space and leads

to a synergistic effect that utilizes the strengths of the individual methods [23,51]. Table 9 shows the chemical compositions determined when comparing the 42CrMo4 steel grade hardness with the DIN EN 10083-3 technical standard [49].

Table 9. 42CrMo4 steel grade and its chemical composition for calculating H-band.

	Chemical Composition (% Mass)		C	Mn	Si	Cr	Ni	Mo	Cu
42CrMo4	+HH	Max	0.45	0.81	0.28	1.08	0.12	0.23	0.22
		Min	0.4	0.68	0.22	0.91	0.13	0.16	0.11
	+HL	Max	0.45	0.82	0.28	1.12	0.12	0.25	0.25
		Min	0.4	0.61	0.18	0.95	0.04	0.18	0.07

Table 10 shows the predicted maximum (Max) and minimum (Min) hardness values for the restricted hardenability towards the top of the scale (+HH) and restricted hardenability towards the bottom of the scale (+HL) bands of 42CrMo4 steel grade. The predicted values were determined using the developed model. In addition, the DIN EN 10083-3 provides reference values for the 42CrMo4 (+H) maximum (Max) and minimum (Min) hardness for comparison. This is visually depicted in Figure 8 for enhanced clarity.

Table 10. 42CrMo4 steel grades hardness with restricted hardenability scatter bands (+HH and +HL grades).

42CrMo4		Distance in mm from Quenched End												
		Hardness in HRC												
		1.5	3	5	7	9	11	13	15	20	25	30	40	50
+HH	Max	59.37	58.97	58.19	57.48	56.99	56.70	56.35	55.84	54.23	51.95	49.67	44.65	40.49
	Min	54.84	54.20	53.20	52.24	50.45	47.29	43.66	40.66	37.15	35.87	34.68	32.41	30.58
+HL	Max	57.67	57.14	56.23	55.45	54.92	54.58	54.15	53.53	51.61	49.16	46.78	42.31	38.69
	Min	52.88	52.32	51.34	50.00	47.41	43.67	40.11	37.24	33.35	32.34	31.82	30.36	28.31

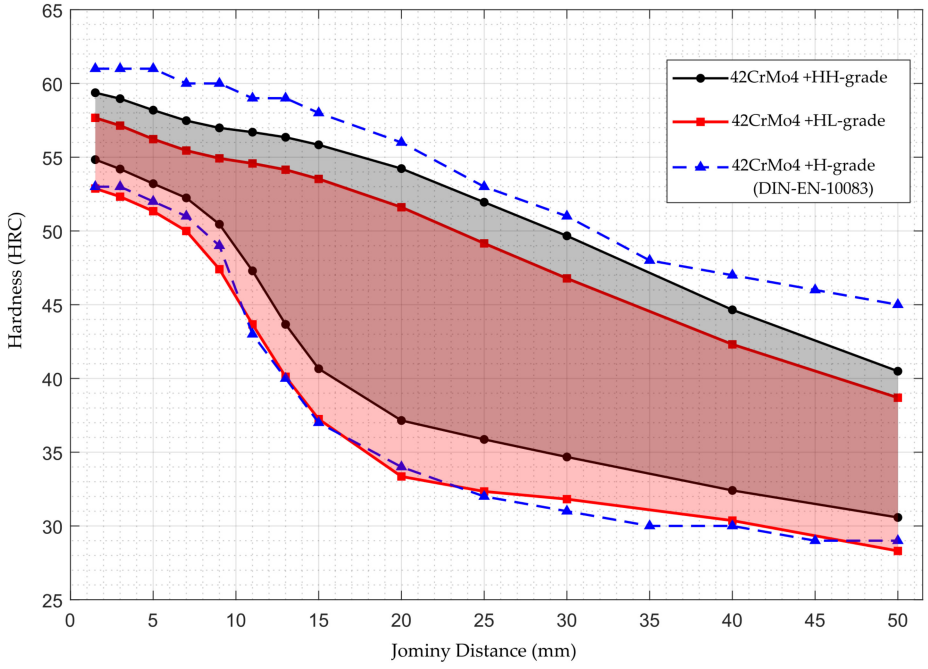

Figure 8. Restricted hardenability scatter band (RH-Band) for the 42CrMo4 steel grade.

Comparing the predicted values with the standard reference values shows that the predictions of the model are generally within the specified hardness ranges, both for the hardenability and hardenability-limited bands. The close agreement between the predicted values and standard values indicates that the model performs well in estimating the hardenability properties of the 42CrMo4 steel grade. This information is critical to ensure that the steel meets the specified hardness requirements for various applications and contributes to effective material selection and component design.

Since the model is data-based and has been trained on datasets comprising nearly 40 steel grades with chemical composition and Jominy distance information, its application extends to the prediction of hardenability curves for these specific steel grades. However, it is important to note one limitation of the model: it cannot predict results for features beyond the range covered by the training datasets.

4. Discussion

The results of the experiments not only provided valuable insights into the performance of the neural network regression model in predicting the hardenability of steels, but also shed light on the effects of the chosen hyperparameters and cross-validation strategies on the robustness of the model and the prediction accuracy through a comparative analysis.

The results obtained from the neural network regression (*fitrnet*) models, particularly Model 4208 (8-85-141-1), demonstrate their remarkable effectiveness in predicting the hardness of steels. The rigorous hyperparameter tuning, facilitated by Bayesian optimization, contributes significantly to the performance of the models. The 5-fold and 10-fold cross-validation schemes ensure the reliability and generalizability of the models and provide robust insights into their predictive capabilities. The 10-fold cross-validation proved to be the optimal choice in this study, which is in line with recommendations from the previous literature [34]. When comparing various evaluation metrics, such as mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE), and coefficient of determination (R^2), the 4208 model comes out on top. With an impressive RMSE of 1.0790 and an R^2 of 0.9900, the model demonstrates excellent accuracy and reliability across the assessment measures.

The illustrative example in the research paper shows the practical application of Model 4208 in determining the hardenability band for a particular steel grade. This application highlights the effectiveness of the model in predicting and optimizing heat treatment results and underscores its practical utility in materials engineering.

In summary, the experiments highlight the predictive capabilities of *fitrnet* models in determining the hardenability of steel, with Model 4208 demonstrating exceptional accuracy and robustness. The inclusion of Bayesian optimization improves the efficiency of hyperparameter tuning and highlights its importance in improving model performance.

5. Conclusions

The experiments conducted shed light on the effectiveness of the regression neural network (*fitrnet*) models in predicting steel hardness. The models, which were rigorously tuned using Bayesian optimization, have impressive predictive capabilities for steel hardness. Among the models, Model 4208 (8-85-141-1) stands out, demonstrating excellent accuracy and robustness across a range of assessment measures. The comprehensive cross-validation schemes (5-fold and 10-fold) ensure the reliability of the models and their generalization to unseen data. The inclusion of Bayesian optimization not only improves the efficiency of hyperparameter tuning but also highlights its importance in refining model performance. The performance of the model is also demonstrated through examples which show its effectiveness in tackling practical problems. This research provides valuable insights into the field of modeling material science and engineering properties. It highlights the potential of regression neural networks in improving hardenability prediction and lays the foundation for future advances.

Author Contributions: Conceptualization, W.F.G. and W.S.; methodology, W.F.G.; software, W.F.G.; validation, W.S. and G.F.B.; formal analysis, W.F.G.; investigation, W.S.; resources, W.S.; data curation, W.F.G. and W.S.; writing—original draft preparation, W.F.G.; writing—review and editing, G.F.B.; visualization, W.F.G.; supervision, W.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

This pseudocode outlines the process for training a regression neural network model, tuning hyperparameters, and computing associated metrics within the experiment framework. It includes functions for conducting the experiment, tuning the hyperparameters, creating the regression model, and performing cross-validation.

```
Procedure Regression_Model_Optimization_Experiment(params, monitor):
    # Load training data
    fileData = load("trainingData.mat")
    trainingData = fileData.predictorMatrix
    responseData = fileData.responseData

    # Specify metrics
    monitor.Metrics = ["ValidationRMSE", "ValidationRSquared"]

    # Train regression model and compute metric values
    trainedModel, validationRMSE, validationRSquared =
trainRegressionModel(trainingData, responseData, params)

    # Record metric values
    monitor.recordMetrics(1, ValidationRMSE = validationRMSE, ValidationRSquared =
validationRSquared)
    return trainedModel

Procedure Train_Regression_Model(trainingData, responseData, params):
    # Train a regression model
    regressionNeuralNetwork = fitrnet(trainingData, responseData, paramNameValuePairs
= namedargs2cell(params))

    # Create the result struct with predict function
    predictorExtractionFcn = lambda x: array2table(x, 'VariableNames',
predictorNames)
    neuralNetworkPredictFcn = lambda x: predict(regressionNeuralNetwork, x)
    trainedModel.predictFcn = lambda x:
neuralNetworkPredictFcn(predictorExtractionFcn(x))

    # Add additional fields to the result struct
    trainedModel.RegressionNeuralNetwork = regressionNeuralNetwork
    trainedModel.About = 'This struct is a trained regression model to predict
hardness of various steel grades.'
    trainedModel.HowToPredict = 'To make predictions on a new predictor column
matrix, X, use: \n yfit = trainedModel.predictFcn(X).'
```

Figure A1. *Cont.*

```
# Perform cross-validation
    partitionedModel = crossval(trainedModel.RegressionNeuralNetwork, 'KFold', 10)

    # Compute validation predictions
    validationPredictions = kfoldPredict(partitionedModel)

    # Compute validation RMSE
    validationRMSE = sqrt(kfoldLoss(partitionedModel, 'LossFun', 'mse'))

    # Compute validation R-squared
    isNotMissing = ~isnan(validationPredictions) & ~isnan(responseData)
    sumSquaredResiduals = sum((responseData(isNotMissing) -
validationPredictions(isNotMissing)).^2)
    sumSquaredTotal = sum((responseData(isNotMissing) -
mean(responseData(isNotMissing))).^2)
    validationRSquared = 1 - sumSquaredResiduals/sumSquaredTotal

    # Create validation Predicted vs. Actual plot
    fig = figure('Name', 'Predicted vs. Actual (Validation)')
    ax = axes(fig)
    hold(ax, 'on')
    axis(ax, 'square')
    line(ax, [min(responseData), max(responseData)], [min(responseData),
max(responseData)], 'Color', 'black', 'LineWidth', 2)
    plot(ax, responseData, validationPredictions, 'ko', 'MarkerFaceColor', '#0072BD')
    xlabel(ax, 'True response')
    ylabel(ax, 'Predicted response')
    xlim(ax, 'padded')
    ylim(ax, 'padded')
    title(ax, 'Predictions: NeuralNetwork')
    return trainedModel, validationRMSE, validationRSquared
```

Figure A1. Regression model optimization pseudocode.

References

1. Sitek, W.; Trzaska, J. Practical Aspects of the Design and Use of the Artificial Neural Networks in Materials Engineering. *Metals* **2021**, *11*, 1832. [CrossRef]
2. Sitek, W.; Trzaska, J.; Gemechu, W.F. Modelling and Analysis of the Synergistic Alloying Elements Effect on Hardenability of Steel. *Arch. Foundry Eng.* **2022**, *22*, 102–108. [CrossRef]
3. Rajan, K. Materials informatics. *Mater. Today* **2005**, *8*, 38–45. [CrossRef]
4. Sha, W.X.; Guo, Y.Q.; Yuan, Q.; Tang, S.; Zhang, X.F.; Lu, S.F.; Guo, X.; Cao, Y.C.; Cheng, S.J. Artificial Intelligence to Power the Future of Materials Science and Engineering. *Adv. Intell. Syst.* **2020**, *2*, 1900143. [CrossRef]
5. Smoljan, B. Computer simulation of mechanical properties, stresses and strains of quenched steel specimen. *J. Achiev. Mater. Manuf. Eng.* **2006**, *19*, 81–85.
6. Homberg, D. A numerical simulation of the Jominy end-quench test. *Acta Mater.* **1996**, *44*, 4375–4385. [CrossRef]
7. Li, M.V.; Niebuhr, D.V.; Meekisho, L.L.; Atteridge, D.G. A computational model for the prediction of steel hardenability. *Metall. Mater. Trans. B-Process Metall. Mater. Process. Sci.* **1998**, *29*, 661–672. [CrossRef]
8. Vermeulen, W.G.; van der Wolk, P.J.; de Weijer, A.P.; van der Zwaag, S. Prediction of Jominy hardness profiles of steels using artificial neural networks. *J. Mater. Eng. Perform.* **1996**, *5*, 57–63. [CrossRef]
9. Dobrzanski, L.A.; Sitek, W. Comparison of hardenability calculation methods of the heat-treatable constructional steels. *J. Mater. Process. Technol.* **1997**, *64*, 117–126. [CrossRef]
10. Geng, X.X.; Wang, S.Z.; Ullah, A.; Wu, G.L.; Wang, H. Prediction of Hardenability Curves for Non-Boron Steels via a Combined Machine Learning Model. *Materials* **2022**, *15*, 3127. [CrossRef]
11. Kovacic, M. Genetic Programming and Jominy Test Modeling. *Mater. Manuf. Process.* **2009**, *24*, 806–808. [CrossRef]
12. Sponzilli, J.T.; Walter, G.H.; Doane, D.V. Development of Restricted Hardenability Band Steels. *SAE Trans.* **1987**, *96*, 1183–1190.
13. Canale, L.C.F.; Albano, L.; Totten, G.E.; Meekisho, L. 12.03—Hardenability of Steel. In *Comprehensive Materials Processing*; Hashmi, S., Batalha, G.F., Van Tyne, C.J., Yilbas, B., Eds.; Elsevier: Amsterdam, The Netherlands, 2014; pp. 39–97.
14. Sitek, W.; Dobrzanski, L.A. Application of genetic methods in materials' design. *J. Mater. Process. Technol.* **2005**, *164*, 1607–1611. [CrossRef]
15. Mukherjee, M.; Singh, S.B. Artificial Neural Network: Some Applications in Physical Metallurgy of Steels. *Mater. Manuf. Process.* **2009**, *24*, 198–208. [CrossRef]
16. Liu, Y.; Zhao, T.L.; Ju, W.W.; Shi, S.Q. Materials discovery and design using machine learning. *J. Mater.* **2017**, *3*, 159–177. [CrossRef]
17. Mueller, T.; Kusne, A.G.; Ramprasad, R. Machine learning in materials science: Recent progress and emerging applications. *Rev. Comput. Chem.* **2016**, *29*, 186–273.
18. Wei, J.; Chu, X.; Sun, X.Y.; Xu, K.; Deng, H.X.; Chen, J.G.; Wei, Z.M.; Lei, M. Machine learning in materials science. *Infomat* **2019**, *1*, 338–358. [CrossRef]

19. Chan, C.H.; Sun, M.Z.; Huang, B.L. Application of machine learning for advanced material prediction and design. *Ecomat* **2022**, *4*, e12194. [CrossRef]
20. Sha, W.; Edwards, K.L. The use of artificial neural networks in materials science based research. *Mater. Des.* **2007**, *28*, 1747–1752. [CrossRef]
21. Bhadeshia, H. Neural networks in materials science. *ISIJ Int.* **1999**, *39*, 966–979. [CrossRef]
22. Bhadeshia, H.; Dimitriu, R.C.; Forsik, S.; Pak, J.H.; Ryu, J.H. Performance of neural networks in materials science. *Mater. Sci. Technol.* **2009**, *25*, 504–510. [CrossRef]
23. Trzaska, J.; Sitek, W. A Hybrid Method for Calculating the Chemical Composition of Steel with the Required Hardness after Cooling from the Austenitizing Temperature. *Materials* **2024**, *17*, 97. [CrossRef] [PubMed]
24. Bishop, C.M.; Bishop, H. The Deep Learning Revolution. In *Deep Learning: Foundations and Concepts*; Springer International Publishing: Berlin/Heidelberg, Germany, 2024; pp. 1–22.
25. Hanza, S.S.; Marohnic, T.; Iljkic, D.; Basan, R. Artificial Neural Networks-Based Prediction of Hardness of Low-Alloy Steels Using Specific Jominy Distance. *Metals* **2021**, *11*, 714. [CrossRef]
26. Reddy, N.S.; Krishnaiah, J.; Hong, S.G.; Lee, J.S. Modeling medium carbon steels by using artificial neural networks. *Mater. Sci. Eng. A-Struct. Mater. Prop. Microstruct. Process.* **2009**, *508*, 93–105. [CrossRef]
27. Dobrzanski, L.A.; Sitek, W. Application of a neural network in modelling of hardenability of constructional steels. *J. Mater. Process. Technol.* **1998**, *78*, 59–66. [CrossRef]
28. Santhosh, A.J.; Tura, A.D.; Jiregna, I.T.; Gemechu, W.F.; Ashok, N.; Ponnusamy, M. Optimization of CNC turning parameters using face centred CCD approach in RSM and ANN-genetic algorithm for AISI 4340 alloy steel. *Results Eng.* **2021**, *11*, 100251. [CrossRef]
29. MathWorks. Statistics and Machine Learning Toolbox: Analyze and Model Data Using Statistics and Machine Learning. 2023. Available online: https://www.mathworks.com/help/pdf_doc/stats/stats.pdf (accessed on 14 March 2024).
30. Kuhn, M.; Johnson, K. Data Pre-processing. In *Applied Predictive Modeling*; Springer: New York, NY, USA, 2013; pp. 27–59.
31. de Zarza, I.; de Curtò, J.; Calafate, C.T. Optimizing Neural Networks for Imbalanced Data. *Electronics* **2023**, *12*, 2674. [CrossRef]
32. Fontanari, T.; Fróes, T.C.; Recamonde-Mendoza, M. Cross-validation Strategies for Balanced and Imbalanced Datasets. In *Brazilian Conference on Intelligent Systems*; Springer International Publishing: Cham, Switzerland, 2022; pp. 626–640. [CrossRef]
33. James, G.; Witten, D.; Hastie, T.; Tibshirani, R.; Taylor, J. Resampling Methods. In *An Introduction to Statistical Learning: With Applications in Python*; Springer Texts in Statistics (STS); Springer International Publishing: Berlin/Heidelberg, Germany, 2023; pp. 201–228.
34. Breiman, L.; Spector, P. Submodel selection and evaluation in regression—The X-random case. *Int. Stat. Rev.* **1992**, *60*, 291–319. [CrossRef]
35. Bischl, B.; Binder, M.; Lang, M.; Pielok, T.; Richter, J.; Coors, S.; Thomas, J.; Ullmann, T.; Becker, M.; Boulesteix, A.L.; et al. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2023**, *13*, 1484. [CrossRef]
36. Snoek, J.; Larochelle, H.; Adams, R.P. Practical Bayesian optimization of machine learning algorithms. In Proceedings of the Advances in Neural Information Processing Systems 25 (NIPS 2012), Lake Tahoe, NV, USA, 3–6 December 2012.
37. Bhandari, U.; Chen, Y.H.; Ding, H.; Zeng, C.Y.; Emanet, S.; Gradl, P.R.; Guo, S.M. Machine-Learning-Based Thermal Conductivity Prediction for Additively Manufactured Alloys. *J. Manuf. Mater. Process.* **2023**, *7*, 160. [CrossRef]
38. Lee, S.Y.; Byeon, S.; Kim, H.S.; Jin, H.; Lee, S. Deep learning-based phase prediction of high-entropy alloys: Optimization, generation, and explanation. *Mater. Des.* **2021**, *197*, 109260. [CrossRef]
39. Nocedal, J.; Wright, S.J. Quasi-Newton Methods. In *Numerical Optimization*; Springer Series in Operations Research and Financial Engineering; Springer: New York, NY, USA, 2006; pp. 135–163.
40. Feurer, M.; Hutter, F. Hyperparameter Optimization. In *Automated Machine Learning: Methods, Systems, Challenges*; Hutter, F., Kotthoff, L., Vanschoren, J., Eds.; The Springer Series on Challenges in Machine Learning; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 3–33.
41. Jones, D.R.; Schonlau, M.; Welch, W.J. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* **1998**, *13*, 455–492. [CrossRef]
42. Kim, Y.; Chung, M. An Approach to Hyperparameter Optimization for the Objective Function in Machine Learning. *Electronics* **2019**, *8*, 1267. [CrossRef]
43. Berrar, D.; Dubitzky, W. Overfitting. In *Encyclopedia of Systems Biology*; Dubitzky, W., Wolkenhauer, O., Cho, K.-H., Yokota, H., Eds.; Springer: New York, NY, USA, 2013; pp. 1617–1619.
44. Berrar, D. Cross-Validation. In *Encyclopedia of Bioinformatics and Computational Biology*; Ranganathan, S., Gribskov, M., Nakai, K., Schönbach, C., Eds.; Academic Press: Oxford, UK, 2019; pp. 542–545.
45. Simon, R. Supervised analysis when the number of candidate features (p) greatly exceeds the number of cases (n). *ACM SIGKDD Explor. Newsl.* **2003**, *5*, 31–36. [CrossRef]
46. ASTM. *A225 Standard Method for End-Quench Test for Hardenability of Steel*; ASTM: West Conshohocken, PA, USA, 2010.
47. Massey, F.J. The Kolmogorov-Smirnov Test for Goodness of Fit. *J. Am. Stat. Assoc.* **1951**, *46*, 68–78. [CrossRef]
48. Bataineh, M.; Marler, T. Neural network for regression problems with reduced training sets. *Neural Netw.* **2017**, *95*, 1–9. [CrossRef]

49. *DIN EN 10083-3*; Steels for Quenching and Tempering—Part 3: Technical Delivery Conditions for Alloy Steels. DIN: Berlin, Germany, 2007.
50. Liscic, B. Hardenability. In *Steel Heat Treatment Handbook—2 Volume Set*, 2nd ed.; Totten, G.E., Ed.; CRC Press: Boca Raton, FL, USA, 2007; pp. 213–276.
51. Datta, S.; Chattopadhyay, P.P. Soft computing techniques in advancement of structural metals. *Int. Mater. Rev.* **2013**, *58*, 475–504. [CrossRef]