

## Adaptive AI-based surrogate modelling via transfer learning for DEM simulation of multi-component segregation

Hadi, A.H.; Moradi, M.; Pang, Y.; Schott, D.L.

**DOI**

[10.1038/s41598-024-78455-7](https://doi.org/10.1038/s41598-024-78455-7)

**Publication date**

2024

**Document Version**

Final published version

**Published in**

Scientific Reports

**Citation (APA)**

Hadi, A. H., Moradi, M., Pang, Y., & Schott, D. L. (2024). Adaptive AI-based surrogate modelling via transfer learning for DEM simulation of multi-component segregation. *Scientific Reports*, 14(1), Article 27003. <https://doi.org/10.1038/s41598-024-78455-7>

**Important note**

To cite this publication, please use the final published version (if applicable).  
Please check the document version above.

**Copyright**

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

**Takedown policy**

Please contact us and provide details if you believe this document breaches copyrights.  
We will remove access to the work immediately and investigate your claim.



# OPEN Adaptive AI-based surrogate modelling via transfer learning for DEM simulation of multi-component segregation

Ahmed Hadi<sup>1✉</sup>, Morteza Moradi<sup>2</sup>, Yusong Pang<sup>1</sup> & Dingena Schott<sup>1</sup>

Segregation of granular materials is a critical challenge in many industries, often aimed at being controlled or minimised. The discrete element method (DEM) offers valuable insights into this phenomenon. However, calibrating DEM models is a crucial, albeit time-consuming, step. Recently, using machine learning (ML)-based surrogate models (SMs) in the calibration process has emerged as a promising solution. Nevertheless, developing such SMs is challenging due to the high number of DEM simulations required for training. Additionally, choosing a suitable ML model is not trivial. This study aims to develop SMs that effectively link particle-particle and particle-wall DEM interaction parameters to segregation of a multi-component mixture. We evaluate several ML models, ranging from artificial neural networks to ensemble learning, that are trained on a very cost-effective dataset, employing Bayesian optimisation with cross-validation to tune their hyperparameters. Next, we introduce a novel transfer learning (TL)-based approach that leverages knowledge from a few scenarios to handle new “unseen” ones. This method enables the construction of adaptive SMs for unseen scenarios, such as a new initial configuration (IC) of granular mixtures, without the need for a full-sized dataset. Our findings indicate that Gaussian process regression (GPR) efficiently builds accurate SMs on a very small dataset. We also demonstrate that only a few samples are required to build an accurate SM for the unseen IC, which significantly reduces the data preparation burden. By incorporating one and five samples from unseen scenarios to update the TL-GPR-based surrogate model, the SM's performance (based on  $R^2$ ) on unseen scenarios improves by 17 and 47%, respectively. The insights and methodology presented in this study will facilitate and accelerate the development of accurate SMs for DEM calibration, assisting in developing reliable DEM models in a shorter timeframe.

**Keywords** Machine learning, Transfer learning, Granular materials, Segregation, Discrete element method, DEM calibration

Granular segregation is an occurrence in which flowing particles with similar properties (such as size, density, or shape) accumulate in specific areas. Segregation is regarded unfavourable in the majority of applications because it might reduce the homogeneity of the granular mixtures and, consequently, is aimed at being controlled/minimised<sup>1</sup>. To achieve this goal, a thorough understanding of segregation and the factors influencing it is required.

Numerous experimental studies have aimed to unravel the segregation phenomenon since the 1970s<sup>2–5</sup>. While these studies provided useful insights into segregation, the experimental approaches to studying segregation generally suffer from several limitations<sup>6</sup>. These include the difficulty in collecting the required samples for segregation measurements, limitations in obtaining particle-scale data, as well as being expensive and time-consuming.

Recent advancements in computational power have led to widespread usage of the discrete element method (DEM), initially introduced by Cundall and Strack<sup>7</sup>, as a useful alternative to experiments for studying granular materials. Especially for segregation, DEM has major advantages over experiments, as it allows for the modelling of granular mixtures with any combinations of size, density, and shape while providing comprehensive particle-

<sup>1</sup>Department of Maritime and Transport Technology, Faculty of Mechanical Engineering, Delft University of Technology, Delft 2628CD, The Netherlands. <sup>2</sup>Center of Excellence in Artificial Intelligence for structures, prognostics & health management, Aerospace Engineering Faculty, Delft University of Technology, Kluyverweg 1, Delft, 2629HS, The Netherlands. ✉email: A.H.Hadi-1@tudelft.nl

level information that is difficult or impossible to obtain in physical experiments<sup>6</sup>. While DEM is widely used, achieving a balance between model accuracy and computational efficiency remains a challenge<sup>8</sup>. The accuracy of the DEM model heavily depends on the proper determination of its parameters through a process called calibration. However, the calibration process can be time-consuming, particularly for multi-component mixtures, where the number of DEM parameters significantly increases.

Trial and error is still extensively employed for calibrating DEM models<sup>9–13</sup>. However, it is not only inefficient but also depends on the user's expertise and barely results in an optimal parameter set<sup>14</sup>. To systematically calibrate the DEM model, several approaches have been proposed. Typically, these approaches use optimisation techniques to update the parameters and determine the calibrated parameter set. Examples include using advanced design of experiments (DoE) in combination with simple optimisation algorithms<sup>15</sup>, particle swarm optimisation<sup>16</sup>, and genetic algorithms<sup>17,18</sup>. However, these methods are still not computationally efficient due to the high number of simulations required<sup>14</sup>.

Richter et al.<sup>14</sup> conducted a thorough literature review on various optimisation techniques, concluding that surrogate-based optimisation is the most suitable approach for DEM calibration. A surrogate model (SM) is an approximation of a more complex and computationally expensive model (such as DEM) aimed at mapping the relationship between the model's input(s) and output(s)<sup>19</sup>. They can be built using advanced mathematics or machine learning (ML). Surrogate-based optimisation is effective at finding a global optimum, is computationally efficient, and can handle parameter limitations and multi-objective problems<sup>14</sup>. Additionally, ML-based surrogates can take advantage of the rapid progress in the field of machine learning in other fields<sup>13,20</sup>. Several studies have used surrogate-based optimisation for DEM calibration. This includes using Gaussian process regression (GPR) and Kriging<sup>14,21–24</sup>, multi-objective reinforcement learning<sup>25</sup>, Bayesian filtering<sup>26,27</sup>, multi-variate regression analysis<sup>28</sup>, neural networks<sup>29–31</sup>, and random forest (RF)<sup>13</sup>.

Despite the advancement of surrogate-based DEM calibration, several challenges remain to be addressed. Firstly, the vast array of available algorithms can make it challenging to choose the most suitable approach, often leading to subjective decision-making. Secondly, while using the SM reduces the computational cost of the DEM calibration, training the SMs themselves, especially when employing sampling techniques such as Latin Hypercube Sampling (LHS), requires a substantial number of simulations. Thirdly, most of the studies consider only a limited number of DEM parameters to construct the SM, potentially overlooking significant DEM parameters. Lastly, most studies aim at single granular materials, and to the best of the authors' knowledge, no study has yet explored surrogate modelling for multi-component granular mixtures. This study attempts to address these challenges by developing SMs that effectively link particle-particle and particle-wall DEM interaction parameters to segregation. We demonstrate this on the basis of a case study for reliable estimation of radial segregation of multi-component mixture in a heap.

The objective of this paper is twofold:



1. We evaluate several ML models to develop surrogate models for DEM simulations involving a two-component mixture (i.e., pellet-sinter) that flows from a hopper through a chute into a receiving bin. Our goal is to develop SMs that capture the relationship between all particle-particle and particle-wall DEM interaction parameters to radial segregation in the heap. To investigate the effect of the initial configuration (IC) of the mixture within the hopper on heap segregation, we vary the mixing degree, pellet-to-sinter mass ratio, and layering order within the hopper. For each individual IC, we use the definitive screening design (DSD), a cost-effective three-level DoE technique, to efficiently create our dataset. To construct effective SMs, we encode ICs, which consist of a combination of categorical and numerical variables, to prepare them as input features for the SMs.
2. Following the identification of the most effective SMs, we innovatively implement a transfer learning (TL) approach to transform the surrogate into an adaptive SM tailored for new, unseen ICs, named the 'transfer learning-based surrogate model (TL-SM)'; thereby addressing our second objective. In pursuit of this, we systematically exclude one IC from the training-validation dataset, designated as the 'unseen IC', which serves as the target domain for TL. Subsequently, we train and cross-validate the SM coupled with Bayesian optimisation (BO) using the remaining dataset as the source domain for TL. Utilising the TL methodology, we deploy the pre-trained ML model as the surrogate for the unseen target IC. We then update and retrain the SM by integrating new data points from the unseen IC while monitoring performance enhancements. The effectiveness of the proposed data-driven SM is assessed through nested cross-validation (NCV), which involves iteratively excluding each IC. Additionally, the stability of the TL-SM is evaluated using distinct random seed numbers for weight and bias initialisation.

Achieving these two objectives will pave the way for efficiently building generalised SMs for various scenarios. These SMs, in turn, will facilitate and speed up the DEM calibration process, contributing to the development of more robust and reliable DEM models in a significantly shorter time.

## Simulation method and established dataset

### Discrete element method

We used the Hertz-Mindlin (no-slip)<sup>32</sup> contact model with an elastic-plastic spring-dashpot rolling friction model (referred to as "type C" in<sup>33</sup>) in our DEM model. This contact model has been successfully employed in past studies on pellets and sinter<sup>34,35</sup>. Detailed equations and more information on the contact model are addressed in the relevant literature<sup>32–34,36</sup>. We developed the DEM model using the commercial software EDEM version 2022.3, where all of the simulations were performed on the DelftBlue high-performance cluster<sup>37</sup>.

DEM Parameter	Pellet	Sinter	Geometry
Shear modulus ( $G$ )	1e+8 Pa <sup>35</sup>	1e+8 Pa <sup>35</sup>	2e+11 Pa <sup>38</sup>
Poisson's ratio ( $\nu$ )	0.25 <sup>35</sup>	0.25 <sup>35</sup>	0.3 <sup>38</sup>
Solid density ( $\rho_s$ )	1951 (kg/m <sup>3</sup> )	1731 (kg/m <sup>3</sup> )	7800 (kg/m <sup>3</sup> ) <sup>38</sup>
Particle size distribution	10-12.5 (mm): 32.8% 12.5-16 (mm): 67.2%	5.6-8 (mm): 14% 8-10 (mm): 22.2 % 10-12.5 (mm): 25.9% 12.5-16 (mm): 21.3% 16-20 (mm): 12.2% 20-25 (mm): 4.4%	NA
Particle shape			NA
Time step	7.94e-6 s (20% of Rayleigh time step)		

**Table 1.** Intrinsic material properties used in DEM simulations.

Factor		Low level (-1)	Middle level (0)	High level (+1)
Pellet-pellet	$\mu_{s,pp}$	0.21 <sup>39</sup>	0.455	0.7 <sup>40</sup>
	$\mu_{r,pp}$	0.05 <sup>41</sup>	0.145	0.24 <sup>38</sup>
	$C_{r,pp}$	0.3 <sup>42</sup>	0.5	0.7 <sup>43</sup>
Sinter-sinter	$\mu_{s,ss}$	0.43 <sup>44</sup>	0.595	0.76 <sup>38</sup>
	$\mu_{r,ss}$	0.08 <sup>35</sup>	0.23	0.38 <sup>38</sup>
	$C_{r,ss}$	0.01 <sup>45</sup>	0.18	0.35 <sup>38</sup>
Pellet-sinter	$\mu_{s,ps}$	0.21	0.485	0.76
	$\mu_{r,ps}$	0.05	0.215	0.38
	$C_{r,ps}$	0.01	0.355	0.7
Pellet-geometry	$\mu_{s,pg}$	0.31 <sup>42</sup>	0.405	0.5 <sup>46</sup>
	$\mu_{r,pg}$	0.05 <sup>42</sup>	0.2	0.35 <sup>40</sup>
	$C_{r,pg}$	0.2 <sup>43</sup>	0.41	0.62 <sup>38</sup>
Sinter-geometry	$\mu_{s,sg}$	0.38 <sup>34</sup>	0.64	0.9 <sup>47</sup>
	$\mu_{r,sg}$	0.08 <sup>35</sup>	0.14	0.2 <sup>34</sup>
	$C_{r,sg}$	0.05 <sup>47</sup>	0.275	0.5 <sup>45</sup>

**Table 2.** Investigated DEM parameters with their low, middle and high values ( $\mu_s$  = coefficient of sliding friction,  $\mu_r$  = coefficient of rolling friction,  $C_r$  = coefficient of restitution). The underlined values for pellet-pellet and sinter-sinter parameters used for pellet-sinter interactions.

We simulated the mixture of sinter and iron ore pellets, as an example of a multi-component mixture used in blast furnace. The intrinsic material properties as fixed and varied interaction parameters were employed, which are listed in Tables 1 and 2, respectively.

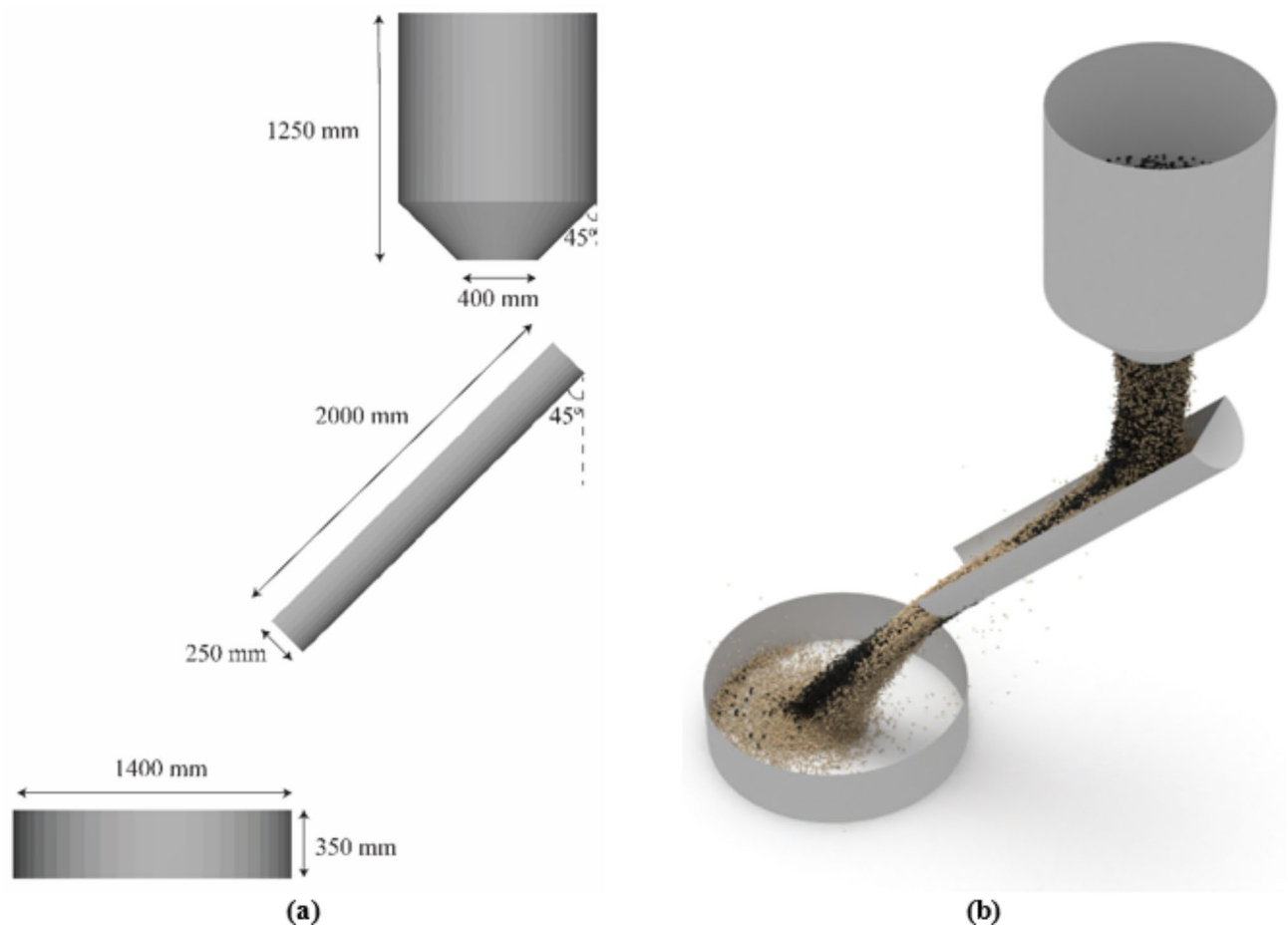
### System and geometry

A system of geometries composed of a hopper, a chute and a receiving bin was used, as shown in (Fig. 1a). The sequence of the simulations' steps is as follows: First, a mixture of pellets and sinter was generated in the hopper. Next, the outlet of the hopper was opened, allowing the materials to discharge from the hopper under the influence of gravity. Finally, the materials were accumulated in the receiving bin after a chute flow (see Fig. 1b).

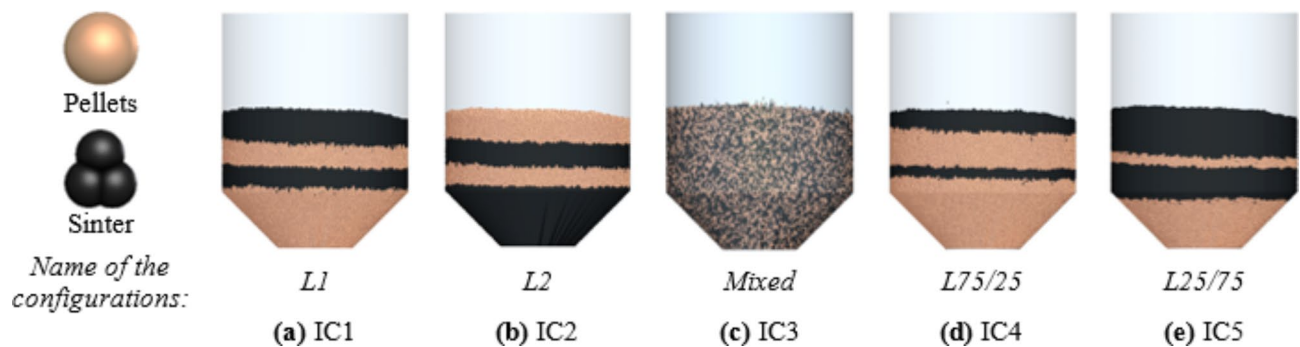
Since the IC of the mixture within the hopper significantly influences the final segregation in the heap, we used various initialisations in the hopper, as illustrated in Fig. 2.

### Quantifying segregation in heap

At the end of the simulations, a heap of the mixture of pellets and sinter was formed, as illustrated in Fig. 3a. Segregation in the heap can be measured in different directions, namely radial, vertical, and circumferential. This



**Fig. 1.** (a) The geometry employed in the simulations and their dimensions, (b) the flow of the mixture of pellets and sinter from the hopper to the receiving bin through the chute.



**Fig. 2.** Various initial configurations (ICs) of pellets and sinter in the simulations (copper and black particles represent pellets and sinter, respectively).

study specifically targets radial segregation. To measure the radial segregation, the heap was first divided into a number ( $m$ ) of radial bins (see Fig. 3b). Next, the mass ratio of one component such as pellets within each bin ( $C_{pm}$ ) was determined. Then, the segregation was quantified using the relative standard deviation (RSD):

$$RSD = \frac{\sigma}{\mu} \quad (1)$$

where  $\sigma$  and  $\mu$  are the standard deviation and the mean of  $C_{pm}$ s, respectively.



**Fig. 3.** (a) Side view of the heap formed within the receiving bin, (b) radial bins used to quantify radial segregation.

Initial configuration	Segregation index	Pellets mass ratio	Layering mode
IC1	1	0.5	1
IC2	1	0.5	2
IC3	0	0.5	3
IC4	1	0.75	1
IC5	1	0.25	1

**Table 3.** Three features used to describe the initial configurations (see Fig. 2) along with their values.

### Established dataset and feature engineering

Because there is a high number of DEM parameters (15, as listed in Table 2) to vary, we aimed to use a sampling strategy that minimises the number of DEM simulations required. To achieve this, we employed the definitive screening design (DSD), a unique three-level design that was first presented by Nachtsheim and Jones<sup>48</sup>. The power of DSD is that, in addition to the main effects, it can identify two-factor and quadratic terms. For an odd number of  $k$  variables (as in this study),  $2k + 3$  runs are needed. Furthermore, to increase the DSD design's power, Jones and Nachtsheim<sup>49</sup> suggested adding four extra runs. Therefore, only 37 simulations were required to establish a DSD design for 15 DEM interaction parameters. The DSD design is presented in Table A.1 in the Appendix.

As we conducted the DSD for five different ICs in the hopper (see Fig. 2), the generated dataset comprised a total of 185 simulation samples. To effectively distinguish between these ICs, it was essential to perform feature engineering to create additional features that describe these configurations. This feature engineering could potentially enhance the performance of ML models<sup>50</sup>. We specifically selected three features: segregation index, pellets mass ratio, and layering mode, which can distinguish between the five ICs used. Since the layering mode is a categorical variable, we employed the label encoding technique, which is known for its computational simplicity, to convert it into a numerical format<sup>51</sup>. Table 3 presents these three features along with their values for all ICs.

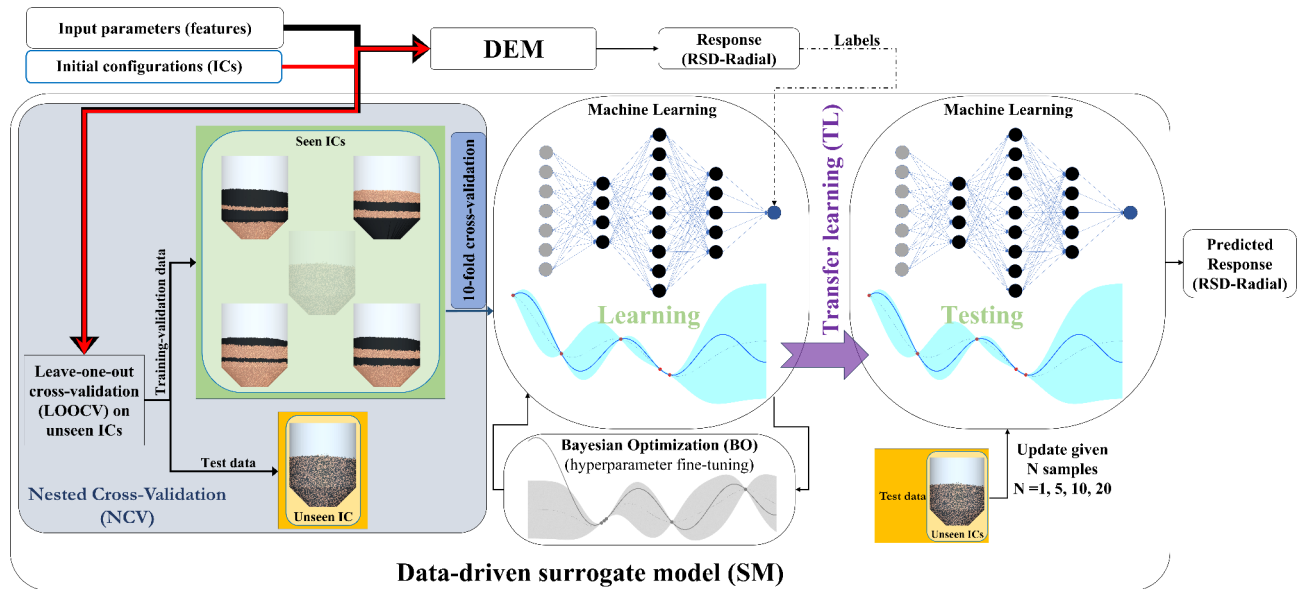
### Data-driven surrogate models

The overall proposed framework for designing data-driven SMs is illustrated in Fig. 4. The dataset underwent nested cross-validation (NCV), one of the most rigorous validation approaches, which involves two other cross-validation (CV) steps. In this section, the ML models employed in this study to build surrogates for DEM are briefly described. These models include linear regression, support vector machine (SVM), regression tree, ensemble learning, Gaussian process regression (GPR), and artificial neural network (ANN). Subsequent sections elaborate on NCV and hyperparameter optimisation for the aforementioned ML models.

### Linear regression

Linear regression is a widely used statistical tool for modelling the linear relationship between independent (inputs) and dependent (output) variables. In the case of only one independent variable, it is called a “simple





**Fig. 4.** The overall proposed framework to design data-driven SM leveraged by TL.

linear regression model”, and when there is more than one independent variable, it is referred to as a “multiple linear regression model”<sup>52</sup>, which is the case in the current work. Considering the original dataset composed of  $n$  data points as  $D = \{(x_i, y_i), i = 1, 2, \dots, n\}$ , the linear regression model is expressed as:

$$y = \beta_0 + \sum_{i=1}^n (\beta_i x_i + \epsilon_i) \quad (2)$$

where  $y$  is the vector of dependent variables (i.e., observed response),  $\beta$  is the coefficients vector,  $\beta_0$  is the intercept (or bias in machine learning), and  $\epsilon_i$  is the random error term. The objective in the linear regression is to minimise the sum of squared errors (SSE) between the predicted ( $\hat{y}_i$ ) and actual ( $y_i$ ) values, which are calculated using the following equation:

$$SSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3)$$

There are several techniques to improve the interpretation of linear regression models, including linear regression with interactions, robust linear regression, and stepwise regression. Linear regression with interactions takes the interactions between the independent variables into account, allowing for modelling of complex relationships among them. Robust linear regression helps mitigate the impacts of outliers, leading to more reliable estimates<sup>53</sup>. Stepwise regression aids in refining the model by iteratively adding or removing them based on statistical criteria, ensuring that the most significant variables are included in the model. All these techniques contribute to the development of more accurate linear regression models across various scenarios<sup>54</sup>.

### Support vector machine (SVM)

Support vector machines (SVMs) are efficient statistical learning models for classification and regression tasks<sup>55</sup>. SVMs are known for finding the optimal decision boundary, known as the maximum-margin, which can effectively separate various classes in the data. Because of this feature, SVMs are very effective at handling complicated datasets.

In training data, where  $x_i$  is the multivariate set of  $n$  observations, the goal in support vector regression is to determine the estimating function  $f(x)$ , which takes the form<sup>56</sup>:

$$f(x) = w^T G(x) + b \quad (4)$$

where  $w$  is the weight vector,  $b$  denotes the bias term and  $G(x)$  is a set of linear or non-linear kernel functions (e.g., quadratic, cubic, etc.). To determine  $w$  and  $b$ , the following objective function is to minimise<sup>57</sup>:

$$\frac{1}{2} w^T w + C \sum_{i=1}^n (\xi_i + \xi_i^*) \quad (5)$$

subject to:

$$\begin{cases} w^T G(x_i) + b - y_i \leq \epsilon + \xi_i^* \\ -(w^T G(x_i) + b - y_i) \leq \epsilon + \xi_i \\ \xi_i, \xi_i^* \geq 0, \quad i = 1, 2, \dots, n \end{cases} \quad (6)$$

where  $C$  is the box constraint,  $x_i$  and  $y_i$  are the input and output vectors, respectively, and  $\xi_i$  and  $\xi_i^*$  are positive slack variables. SVMs utilise kernel functions ( $k(x, x')$ , where  $x$  and  $x'$  are two data points) to handle non-linear relationships between the input and output vectors. Even if the original input space is not linearly separable, kernel functions enable SVMs to implicitly transfer input vectors into a higher-dimensional space where the data may be more separable. In this case, the decision function (Eq. (4)) takes the form:

$$f(x) = \sum_{i=1}^n (\alpha_i - \alpha_i^*) k(x_i, x') + b \quad (7)$$

where  $\alpha_i$  and  $\alpha_i^*$  are Lagrange multipliers. For example, the decision function for the radial basis function (RBF) kernel is as follows:

$$f(x) = \sum_{i=1}^n (\alpha_i - \alpha_i^*) \exp(-\gamma \|x_i - x'\|^2) + b \quad (8)$$

where  $\gamma$  is a parameter for the RBF kernel.

### Regression tree

Decision trees are predictive models that partition the feature space to predict the label associated with an instance by traversing from the tree's root to a leaf<sup>58</sup>. Regression trees are a specific type of decision trees designed for predicting numerical values. They recursively divide the input space ( $x_i$ ) into  $J$  number of disjoint regions ( $R_1, R_2, \dots, R_J$ ) using splitting rules. Regression tree splitting rules are derived from the minimisation of the sum of squared errors inside each division:

$$(j, s) = \operatorname{argmin} \left[ \sum (y_i - c_m)^2 + \sum (y_i - c'_m)^2 \right] \quad (9)$$

where  $j$  and  $s$  are the index and the threshold value of the feature used for splitting, respectively.  $c_m$  and  $c'_m$  are the constant predictions for regions  $R_m$  and  $R'_m$ . The main parameter in regression trees is the minimum leaf size, which represents the minimum number of samples required to create a terminal node (leaf) in the process of building the tree.

### Ensemble learning

While regression trees are easy to interpret and fast for fitting and prediction, like other weak learners, they are susceptible to overfitting and have sensitivity to training data. Integrating several weak learners makes the model more resilient and less prone to overfitting, as each learner depends on a different set of data points. Ensemble learning in ML is the process of combining several weak learners. Given regression trees, one way to overcome this issue is to construct a weighted collection of multiple regression trees to build models called ensembles of trees. Combining many regression trees generally improves the prediction capability and accuracy. Several ensemble learning methods exist, including bagging and boosting.

Bagging (bootstrap aggregation) involves training many weak (base) learners (parallelly) simultaneously and integrating them using averaging techniques<sup>59</sup>. Considering the original data set as  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ , first, a number of bootstrap samples ( $D_i, i = 1, 2, \dots, B$ ) is created by randomly choosing  $n$  samples from  $D$  with replacement. Then, a base learner  $f_i$  is trained based on  $D_i$  to minimise the error between  $y$  and  $f_i(x)$ . Finally, the aggregated prediction model  $f(x)$  is obtained by averaging the predictions:

$$f(x) = \frac{1}{B} \sum_{i=1}^B f_i(x) \quad (10)$$

By training each learner with the output of the preceding learner, the boosting approach progressively boosts the model's overall performance. One boosting technique used for building regression ensembles is least-squares boosting (LSBoost)<sup>60</sup>. This technique successively fits a set of weak learners (e.g., decision trees), with each new learner trained to reduce residual errors from the ensemble's total predictions. The approach iteratively improves the ensemble's predictions by including fresh weak learners. First, the ensemble prediction is initialised as the mean of the target values ( $y_i, i = 1, 2, \dots, n$ ):

$$f_0(x) = \frac{1}{n} \sum_{i=1}^n y_i \quad (11)$$

Then, for iteration  $m$  ( $m = 1, 2, \dots, M$ ), the residuals between the target values and the accumulated prediction ( $f_{m-1}(x_i)$ ) for each observation is calculated as:

$$r_{im} = y_i - f_{m-1}(x_i) \quad 1 \leq i \leq n \quad (12)$$



Next, a new weak learner ( $h_m$ ) is trained by fitting it to the residuals:

$$h_m = \underset{h}{\operatorname{argmin}} \left( \frac{1}{2n} \sum_{i=1}^n [r_{im} - h(x_i)]^2 \right) \quad (13)$$

Finally, the ensemble model is updated:

$$f_m(x) = f_{m-1}(x) + \eta_m h_m \quad (14)$$

where  $\eta$  is the learning rate (the shrinkage parameter), which controls the contribution of each weak learner and ranges from 0 to 1.

### Gaussian process regression (GPR)

Gaussian process regression (GPR) is a probabilistic and non-parametric kernel-based machine learning regression model<sup>61</sup> rooted in Bayesian principles. Due to its simplicity of use and flexibility in obtaining hyperparameters, GPR is well-suited to handle small-sized datasets and nonlinear problems<sup>62</sup>. A Gaussian process (GP) is a collection of random variables having Gaussian distribution and is fully defined by its mean function  $\mu(x)$  and covariance kernel function  $k(x, x')$ .

Considering  $x_i$  and  $y_i$  as the input and corresponding output vectors, respectively, the GPR model with Gaussian noise is formulated as:

$$y_i = f(x_i) + \epsilon_i \quad (15)$$

where  $\epsilon_i$  denotes a constant additive noise term assumed to follow a Gaussian distribution with a mean of 0 and a standard deviation of  $\sigma$  (i.e.,  $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ ). The objective of GPR is to infer the function  $f$  in a non-parametric and Bayesian approach, utilizing the provided training dataset  $\{(x_i, y_i); i = 1, 2, \dots, n\}$ . A prior distribution on  $f$  needs to be established in order to learn this function. Typically, this prior is utilized to encapsulate qualitative attributes of the function such as continuity, differentiability, or periodicity. In GPR, the prior distribution for  $f$  as the regression function is represented by:

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x')) \quad (16)$$

In this formulation, while the mean function  $\mu(x)$  is often set constant, the covariance kernel  $k(x, x')$  varies. When the values of the function  $f(x_i)$  have a joint Gaussian distribution defined by  $\mu(x)$  and  $k(x, x')$  for every finite set of inputs  $x_i$ , then the function  $f(x)$  is a GP, implying:

$$\begin{bmatrix} f(x_1) \\ ? \\ f(x_n) \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \mu(x_1) \\ ? \\ \mu(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ ? & \ddots & ? \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \right) \quad (17)$$

which using the notation below:

$$\boldsymbol{\mu} \triangleq \begin{bmatrix} \mu(x_1) \\ ? \\ \mu(x_n) \end{bmatrix}; \mathcal{K} \triangleq \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ ? & \ddots & ? \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}; \parallel(x) \triangleq \begin{bmatrix} k(x_1, x) \\ ? \\ k(x_n, x) \end{bmatrix} \quad (18)$$

the equations can be simplified. In the process of learning functions through GPR, the implications of expanding Eq. (17) by including a new data point  $x_*$ , separate from the training data, are being considered. The objective is to predict the value of the function at this particular location, i.e.,  $f(x_*)$ . To do so, given the already observed values  $\mathcal{Y} = [y_1 \dots y_n]^T$ , the relationship can be expressed by incorporating Eqs. (15)–(18):

$$\begin{bmatrix} \mathcal{Y} \\ f(x_*) \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \boldsymbol{\mu} \\ \mu(x_*) \end{bmatrix}, \begin{bmatrix} \mathcal{K} + \sigma^2 \mathbf{I}_n & \parallel(x_*) \\ \parallel(x_*)^T & k(x_*, x_*) \end{bmatrix} \right) \quad (19)$$

Here,  $\mathbf{I}_n$  is the  $N \times N$  identity matrix. Conditioning on the new data based on observations, the posterior probability distribution for  $f(x_*)$  can be estimated as:

$$f(x_*) | \mathcal{Y} \sim \mathcal{N}(\mu_*, \sigma_*^2) \quad (20)$$

where:

$$\mu_* = \mu(x_*) + \parallel(x_*)^T (\mathcal{K} + \sigma^2 \mathbf{I}_n)^{-1} (\mathcal{Y} - \boldsymbol{\mu}) \quad (21)$$

$$\sigma_*^2 = k(x_*, x_*) - \parallel(x_*)^T (\mathcal{K} + \sigma^2 \mathbf{I}_n)^{-1} \parallel(x_*) \quad (22)$$

The posterior probability distribution is Gaussian once more, allowing for Bayesian reasoning on the function  $f$ . One noteworthy aspect of these formulations is that the function's posterior expected value,  $\mathbb{E}(f(x_*) | \mathcal{Y})$ , could be stated using a weighted sum of kernel functions:

$$\mathbb{E}(f(x_*)|\mathcal{Y}) = \|(x_*)^T(\mathcal{K} + \sigma^2 \mathbf{I}_n)^{-1}(\mathcal{Y} - \boldsymbol{\mu}) = \sum_{i=1}^n k(x_*, x_n) \alpha_i \quad (23)$$

where:

$$\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \triangleq (\mathcal{K} + \sigma^2 \mathbf{I}_n)^{-1}(\mathcal{Y} - \boldsymbol{\mu}) \quad (24)$$

To delve into the GPR model, the weighted sum in Eq. (24) is advantageous as it facilitates computations that would otherwise be challenging.

The details on the kernel functions used in this study are presented in Table B.1 in the Appendix. It is possible to use either isotropic or non-isotropic kernel functions with GPR. In contrast to isotropic kernels, non-isotropic ones give each predictor variable a distinct correlation length scale. This results in an improved accuracy at the cost of slowing down the fitting process.

### Artificial neural network (ANN)

An artificial neural network (ANN) is an ML model inspired by neuronal organisation in animal brains. ANN is composed of interconnected nodes which are arranged into several layers. These layers are typically organised into three groups: the input layer, hidden layers, and the output layer. Each node (or neuron) conducts a basic computation, and the connections between nodes transport weighted signals from one layer to the next<sup>63,64</sup>.

The output of the NN is computed through feedforward propagation<sup>65</sup>. Considering the input vector as  $x$ , the activation of each neuron in layer  $l$  as  $a^{(l)}$ , the weight matrix linking layer  $l$  to layer  $l+1$  as  $W^{(l)}$ , and the bias term for layer  $l$  as  $b^{(l)}$ , the feedforward computation is as follows:

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)} \quad (25)$$

$$a^{(l+1)} = g(z^{(l+1)}) \quad (26)$$

where  $z^{(l+1)}$  is the input to layer  $l+1$  and  $g(z)$  is the activation function which is applied element-wise to the input. In this study, we used the rectified linear unit (ReLU), Tanh, and sigmoid activation functions, whose formulas are provided in Table B.2 in the Appendix.

### Hyperparameters optimisation and model validation

Overfitting is a common challenge in ML models, requiring the use of cross-validation (CV) methods to validate the effectiveness of the model. As illustrated in Fig. 5, we employed nested cross-validation (NCV), which consists of two CV steps:

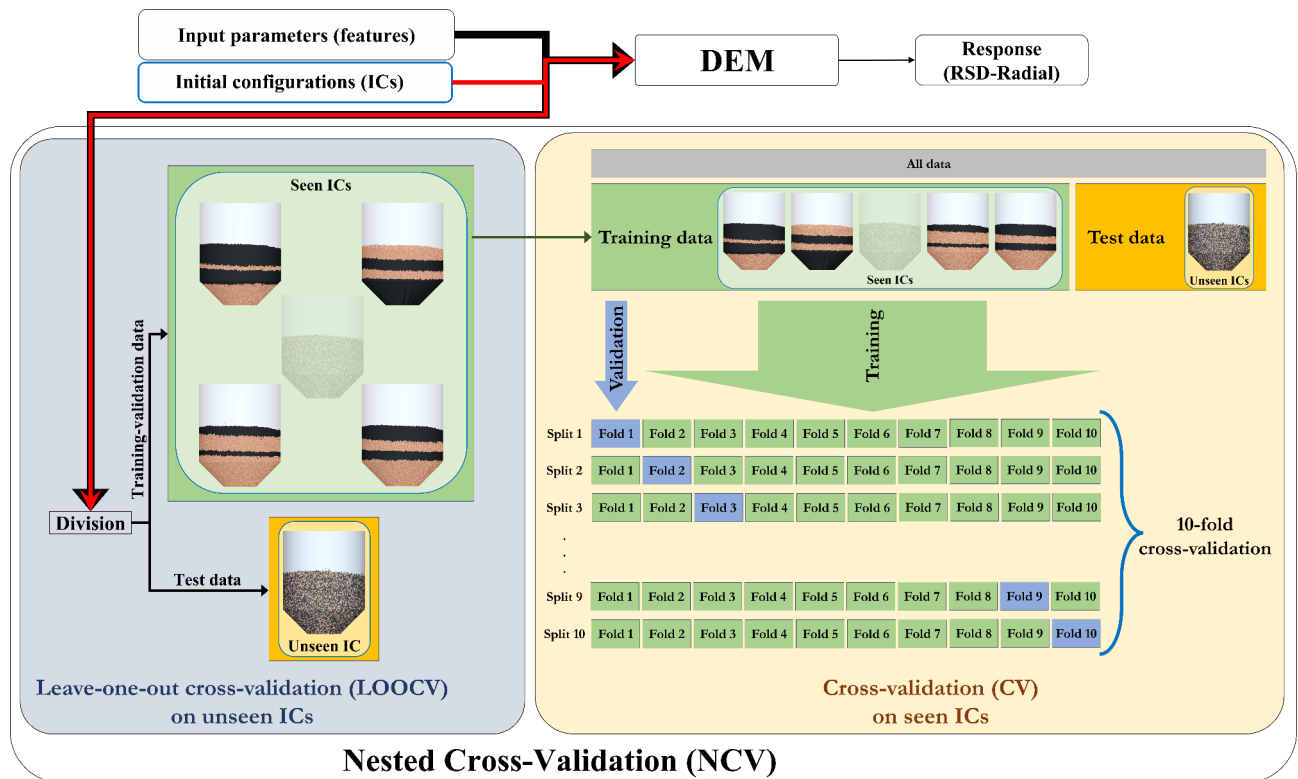
- I. An external loop conducts CV on the dataset based on the number of ICs, excluding one IC at each iteration to create a distinct test set of “unseen ICs”.
- II. An internal loop performs CV on the remaining dataset after the execution of the external loop to tune hyperparameters and mitigate overfitting.

This NCV technique properly estimates model performance by combining 5-fold outer loops based on the number of ICs with 10-fold inner loops. In the outer loop, model performance assessment occurs through the partitioning of the dataset into a training-validation set (comprising four ICs) and a distinct test set (comprising one IC). The outer loop is also referred to as leave-one-out cross-validation (LOOCV)<sup>66</sup> on ICs. The training-validation set undergoes further subdivision into diverse folds using a 10-fold CV to estimate the generalisation error and fine-tune hyperparameters. To this end, Bayesian Optimisation (BO) algorithms<sup>67</sup> were employed to adjust hyperparameters. To determine the minimum or maximum of a function, BO combines Bayesian inference with optimisation techniques. BO constructs a prior distribution over parameters, updates it with data, and selects promising parameters using an acquisition function. This iterative process efficiently explores the function space until it converges on the optimal parameters. Compared to exhaustive search techniques such as grid search or random search, this method effectively explores the hyperparameter space and frequently requires fewer trials.

### Transfer learning (TL) for unseen ICs

In this section, we explore the application of transfer learning (TL) to enhance the performance of the SM when confronting a new, previously unseen IC. TL is a powerful technique that leverages knowledge gained from related tasks or domains to improve performance on a target task<sup>68–70</sup>. TL is specifically advantageous when providing a sufficient number of training samples is costly.

In the context of granular material segregation, the IC is crucial as it can significantly influence segregation outcomes<sup>6</sup>. Consequently, if the IC changes, the DEM model must be recalibrated, which is very time-consuming. To address this challenge, we can leverage prior knowledge gained from previously encountered ICs as the source domain to pretrain the SMs. Subsequently, we can transfer the pretrained SMs as base learners for new, unseen ICs, treating them as the target domain in TL. In the next stage, the pretrained SMs are updated by incorporating a small number of samples from the target domain through model retraining while retaining the prior information from the source domain. This approach allows TL to expedite the learning process, as the SMs require fewer samples from the unseen ICs to achieve effective learning. Additionally, it eliminates the need



**Fig. 5.** The process of nested cross-validation (NCV), given LOOCV in the outer loop and 10 folds in the inner loop.

for full retraining and cross-validation, coupled with BO, using data from both the source and target domains, thereby reducing computational demands.

Considering  $x_i$  and  $y_i$  representing the input and output vectors, TL involves extracting knowledge from the source domain ( $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{N_s}$ ), and use it to pretrain the model for the target domain ( $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{N_t}$ ). Here, we investigate the effectiveness of TL in updating the model by varying the number of samples from the newly unseen IC. This includes the following steps:

- **Initial model training:** Initially, the SM model is trained using cross-validation on data from four out of the five ICs, yielding a baseline model  $f_s$ . This baseline model provides a starting point for the TL approach.

For example, the training of the source model for GPR can be expressed as:

$$\{f_s(x, \mathbf{w}_s, \boldsymbol{\theta}_0) \sim \mathcal{GP}(\mu(x), k(x, x')) | (\mathbf{w}_0, \boldsymbol{\theta}_0) : x \in \mathcal{D}_s\} \quad (27)$$

where the initial learnable parameters are denoted as  $\mathbf{w}_0$  and the trained learnable parameters are denoted as  $\mathbf{w}_s$ . If we denote initial hyperparameters of  $f(x)$  with  $\boldsymbol{\theta}_0$ , the BO algorithm – given  $x \in \mathcal{D}_s$  – is applied to tune hyperparameters of  $f_s(x)$ , after which they can be denoted by  $\boldsymbol{\theta}_s$ .

- **Testing the unseen IC:** Subsequently, the performance of the pretrained baseline model is evaluated on the data from the new unseen IC.

$$\{f_s(x, \mathbf{w}_s, \boldsymbol{\theta}_s) : x \in \mathcal{D}_t\} \quad (28)$$

However, the results indicated suboptimal performance, highlighting the need for further model refinement.

- **TL with limited samples:** To address the limitations in performance, the transferred model is updated using limited samples from the target domain  $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{N_t}$  alongside the source domain data  $\mathcal{D}_s$ . For example, the updating of the source model for GPR can be expressed as:

$$\{f_t(x, \mathbf{w}_t, \boldsymbol{\theta}_s) \sim \mathcal{GP}(\mu(x), k(x, x')) | (\mathbf{w}_s, \boldsymbol{\theta}_s) : x \in (\mathcal{D}_s \cup \mathcal{D}_t)\} \quad (29)$$

Specifically, we experimented with  $N_t$  values of 1, 5, 10, and 20 samples available from the new unseen IC to investigate the impact of varying sample sizes of the target domain on model improvement.

## Evaluation metrics

We used several metrics to compare the performance of the trained ML models. These metrics can be categorised into two groups: metrics that evaluate the accuracy of the models and those that assess the speed of the training and prediction processes. Regarding the first group, we used root-mean-square error (RMSE), coefficient of determination (R-squared or  $R^2$ ), and mean-absolute-error (MAE), with the following equations:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}} \quad (30)$$

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (31)$$

$$MAE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)}{n} \quad (32)$$

where  $n$  is the number of data points,  $y$  is the actual value vector,  $\hat{y}$  is the predicted value vector, and  $\bar{y}$  is the mean of actual values.

In addition to the metrics used to assess accuracy, we also employed two additional metrics: training time and prediction speed. The former indicates the time required for the model to be trained (in seconds), while the latter represents the number of predictions the model can make per second. Therefore, models with low training times and high prediction speeds are preferred.

## Results and discussion

In this section, the performance of various ML models (mentioned in Section 3), considered as the SM, is first compared in the subsection “surrogate model selection”. The influence of including or excluding initial configurations (ICs) as complementary input features through label encoding is investigated to select the best models and determine the optimal approach for shaping the feature input. This step is crucial for further evaluations and subsequent steps toward nested cross-validation (NCV) on unseen ICs and updating the SM using TL, as will be discussed in the subsection “Transfer Learning (TL) for Unseen ICs”. We used MATLAB 2022a on a laptop with an Intel Core i7-8665U CPU and 16 GB of RAM to train and evaluate ML models.

### Surrogate model selection

We trained various ML models to compare and select those showing the best performance for the next steps. The models were trained under two distinct scenarios with respect to ML inputs: (1) using only DEM interaction parameters, and (2) considering extra inputs to characterise the ICs of the mixture within the hopper (see Fig. 2). Figure 6 illustrates an example of the regression tree model's performance for both scenarios. Obviously, the model's performance is significantly enhanced in the case of the extra inputs related to ICs. This improvement is anticipated as the segregation (or degree of mixing) of multiple materials heavily depends on their ICs<sup>6</sup>. The results for the first scenario (i.e., excluding ICs as ML inputs) are given in Table B.3 of the Appendix, where it is evident that the model's performance is unsatisfactory even with optimised hyperparameters. As a result, we proceed with the second scenario to train the models.

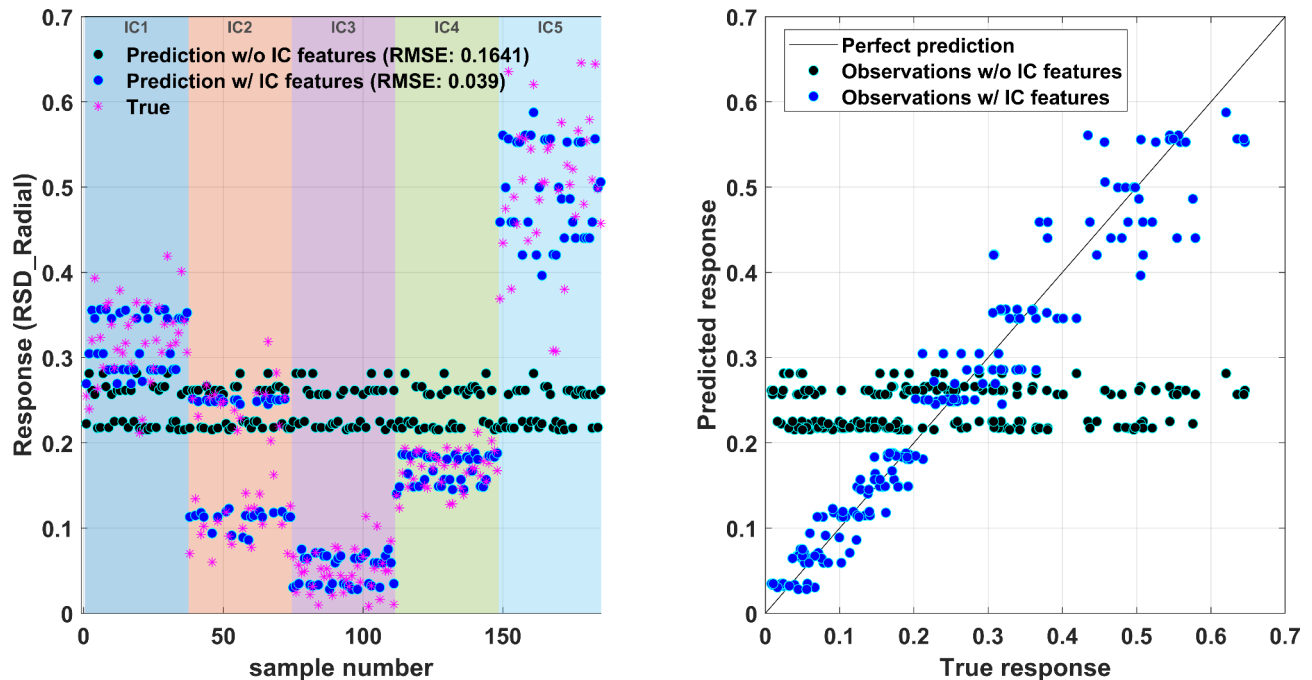
The training results of various ML models are presented in Table 4, where ICs were included as complementary features. The training was performed with 5-fold cross-validation on 185 samples and 18 features. Optimisable models were fine-tuned using the Bayesian optimisation (BO) algorithm with 50 iterations. Notably, the performance of the models significantly improved when the hyperparameters were optimised using BO, underscoring the importance of fine-tuning hyperparameters in the ML training process. It is also worth mentioning that despite increasing the number of hidden layers and neurons, the performance of ANN did not improve, possibly due to overfitting or vanishing gradient problems<sup>71</sup>.

Based on the evaluation metrics employed in this study, the optimal model is characterised by minimal error (i.e., RMSE and MAE), maximal  $R^2$ , low training time, and high prediction speed. According to Table 4, we identify two models that fulfil the majority of these criteria simultaneously: Gaussian process regression (GPR) and ensemble of trees, with GPR being superior across all metrics.

### Transfer learning (TL) for unseen ICs

In this section, we present the outcomes of the TL method and updating process utilised in this study, as explained in Section 5. Results for TL-based ensemble learning (TL-Ensemble) and GPR (TL-GPR) are given in Tables 5 and 6, respectively. To ensure stability and repeatability, we used five different random seed numbers to initialise the ML model's parameters during training. This strategy ensures a reliable assessment of the model's performance. The results in Tables 5 and 6 are reported as (mean  $\pm$  standard deviation) resulting from these five repetitions. All training procedures were performed with 10-fold cross-validation and 18 features. Additionally, hyperparameters were fine-tuned using the BO algorithm. The hyperparameters of these models together with their search space are provided in Table B.4 and Table B.5 in the Appendix.

Initially, we cross-validated the models (i.e., Ensemble learning and GPR) using all 185 available samples given ten folds to establish a benchmark for comparison. The results are displayed in the first row of Tables 5 and 6, labelled as “All ICs were seen”. Then, we iteratively applied the transfer learning approach across all ICs. For each iteration, one IC was excluded, and the model was pretrained on the remaining four ICs (i.e., on



**Fig. 6.** Comparison of the impact of input features with (w/) and without (w/o) initial configurations (ICs) on fine-tuned regression tree outcomes using Bayesian optimisation (BO), assessed via 5-fold cross-validation.

$4 \times 37 = 148$  samples) via a 10-fold cross-validation coupled with BO fine-tuning of hyperparameters. The results of the validation phase are presented under “Validation”, excluding the training phase’s results. Next, we tested the model’s performance on the “unseen IC” as the target domain of TL that was previously excluded. We ran tests where the pretrained model was retrained with 0, 1, 5, 10, and 20 samples from the unseen IC, monitoring its performance each time. These test results are shown under “Unseen IC”. For an overall assessment of the model’s performance across all unseen ICs, we calculated the average of the TL-based outcomes across all unseen ICs for retraining with different available samples of 0, 1, 5, 10, and 20 from the target domain. These averages are displayed at the bottom of Tables 5 and 6, indicated as “Mean”.

To visually illustrate the impact of the updating process on the pretrained TL-SMs, Fig. 7 shows predicted versus true responses for two different unseen initial configurations, IC1 and IC3, where they were updated with 0 (no update), 1, and 5 samples from the target domain. The models’ predictions were initialised using the first random seed number in this figure. As illustrated, providing even a few samples from the unseen IC—the target domain—for updating leads to a significant improvement. Comprehensive results for various unseen ICs, random seed numbers, and the model updated with different numbers of samples (0, 1, 5, 10, 20) from the unseen IC under test are provided in two videos in the supplementary material for both TL-Ensemble and TL-GPR. These videos allow easy tracking of the updating process and the effect on targeting different unseen ICs.

Figure 8 illustrates a comparison between the “mean” performance of TL-Ensemble and TL-GPR across different numbers of available samples from the unseen IC. As shown, TL-GPR consistently outperforms Ensemble-TL across all available sample sizes, highlighting the superiority of TL-GPR over TL-Ensemble. Consequently, we focus our attention on TL-GPR for further analysis of the results.

Figure 9 illustrates the reduction in RMSE in percentage for different sample sizes used to update the TL-GPR model. The reduction was calculated as  $(|RMSE_{IC_i} - RMSE_{IC_0}| / RMSE_{IC_0}) \times 100$ , where  $(i = 1, 5, 10, 20)$  corresponds to different numbers of samples to retrain SM, and  $IC_0$  denotes the case where no update happens to the pretrained model. The bar graph demonstrates that updating the TL-GPR model with just one sample from the unseen IC results in a significant reduction ( $\sim 50\%$ ) in the RMSE. Figure 8a also shows that by updating the model with only a few new samples (e.g. 5 samples) from the unseen IC, its accuracy for the new IC can approach that of the validation set. This finding underscores the efficiency of updating the SM with a minimal number of new DEM simulations to achieve improved accuracy of the SM for unseen IC.

It is important to note that the results and analyses discussed above are based on the average performance of the TL-based model across all ICs. However, according to Table 6, the model’s performance varies depending on which IC is considered as unseen. To facilitate a clearer comparison, Fig. 10 presents the RMSE and  $R^2$  of TL-GPR across all ICs and for different numbers of samples used to update the pretrained model. It reveals that, when the model is not updated (i.e., 0 sample), the TL-GPR model exhibits the poorest performance for IC3 and IC2, characterised by the highest RMSE and lowest  $R^2$ .

The observed performance for IC3, where materials within the hopper are fully-mixed (see Fig. 2), was anticipated. This is because there is no comparable data in the training dataset; the other four initial configurations (IC1, IC2, IC4, and IC5) have fully segregated initial configurations, leading to a far different data distribution for IC3. Similarly, the relatively less optimal performance of the model for IC2 can be attributed to its unique

Model	Adjustments	Metrics based on validation			Training time (sec)	Prediction speed* (≈ observation/sec)
		RMSE	R <sup>2</sup>	MAE		
Linear Regression	–	0.0506	0.91	0.0406	9.98	1600
	Interactions	0.0244	0.98	0.0200	33.32	620
	Robust	0.0509	0.91	0.0404	31.00	910
	Stepwise	0.0237	0.98	0.0188	539.56	3000
Regression Tree	Fine	0.0423	0.93	0.0301	28.25	3400
	Medium	0.0420	0.94	0.0293	27.12	3700
	Coarse	0.1288	0.39	0.1048	25.35	3000
	Optimisable (BO)	0.0390	0.94	0.0277	34.97	5600
SVM	Kernel: linear	0.0505	0.91	0.0388	19.18	3500
	Kernel: quadratic	0.0310	0.96	0.0245	17.22	2800
	Kernel: cubic	0.0384	0.95	0.0300	14.18	3400
	Kernel: Fine Gaussian	0.1616	0.04	0.1325	43.12	4700
	Kernel: Medium Gaussian	0.0642	0.85	0.0478	42.76	4500
	Kernel: Coarse Gaussian	0.0755	0.79	0.0585	41.94	3000
	Optimisable (BO)	0.0251	0.98	0.0201	461.24	5900
Ensemble of Trees	Boosted	0.0362	0.95	0.0265	41.19	790
	Bagged	0.0703	0.82	0.0590	40.11	650
	<b>Optimisable (BO)</b>	<b>0.0231</b>	<b>0.98</b>	<b>0.0173</b>	<b>472.77</b>	<b>260</b>
GPR	Kernel = Exponential	0.0485	0.91	0.0370	36.89	2100
	Kernel = Squared Exponential	0.0239	0.98	0.0184	38.62	1100
	Kernel = Rational Quadratic	0.0239	0.98	0.0184	36.13	2200
	Kernel = Matern 5/2	0.0255	0.98	0.0197	37.68	3400
	<b>Optimisable (BO)</b>	<b>0.0162</b>	<b>0.99</b>	<b>0.0119</b>	<b>348.61</b>	<b>4600</b>
ANN	One layer with [10] neurons	0.0378	0.95	0.0282	34.86	1000
	One layer with [25] neurons	0.0452	0.93	0.0349	34.10	2300
	One layer with [100] neurons	0.0703	0.82	0.0560	33.43	1200
	Two layers with [10,10] neurons	0.0719	0.81	0.0541	32.68	3300
	Three layers with [10,10,10] neurons	0.0921	0.69	0.0637	31.86	2800
	Optimisable (BO)	0.0245	0.98	0.0201	602.92	9400

**Table 4.** Performance comparison of regression models, including 3 features representing initial configurations (ICs). Optimisable models’ hyperparameters were fine-tuned using the BO algorithm. \*For reference, each DEM simulation takes approximately 3 h (10,800 s), highlighting the significant speed advantage of the surrogate models.

feature, i.e., having a reversed layering order, opposite to IC1, IC4, and IC5. Nevertheless, in both IC2 and IC3 cases, after updating the model with only a few data points, a significant improvement in performance is observed. For instance, in IC3, updating the model with only 1 and 5 samples results in a remarkable reduction in RMSE by 68% and 85%, respectively.

Conclusion

In this study, we successfully demonstrated a framework for developing surrogate models (SMs) that effectively link particle-particle and particle-wall DEM interaction parameters to the segregation of a multi-component mixture. We first examined various ML models to develop SMs capable of estimating radial segregation in the heap based on DEM parameters and the initial configuration (IC) of the mixture. We found that developing accurate SMs requires consideration of features describing IC through feature engineering. Moreover, we emphasise that fine-tuning hyperparameters is crucial to obtaining the optimal performance of ML-based SMs. Among the six ML models tested, Ensemble learning and Gaussian process regression (GPR) demonstrated the best performance.

Next, we developed an adaptive SM leveraging a transfer learning (TL)-based approach using Ensemble learning and GPR. Cross-validation, coupled with Bayesian optimisation for fine-tuning the SM’s hyperparameters, was conducted using four ICs as the pretraining phase of TL, while predictions and a retraining phase made for a fifth “unseen IC”. Model performance was monitored after updating and retraining the TL-SMs with access to different numbers of samples from the unseen IC. We observed that TL-GPR consistently outperformed TL-Ensemble.

Our findings indicate that the performance of TL-SMs varied depending on the specific unseen IC. When testing the pretrained TL-SMs with new ICs possessing specifications not included in the source dataset, their performance appears to be relatively lower. For instance, IC3 is the only configuration out of five in which the

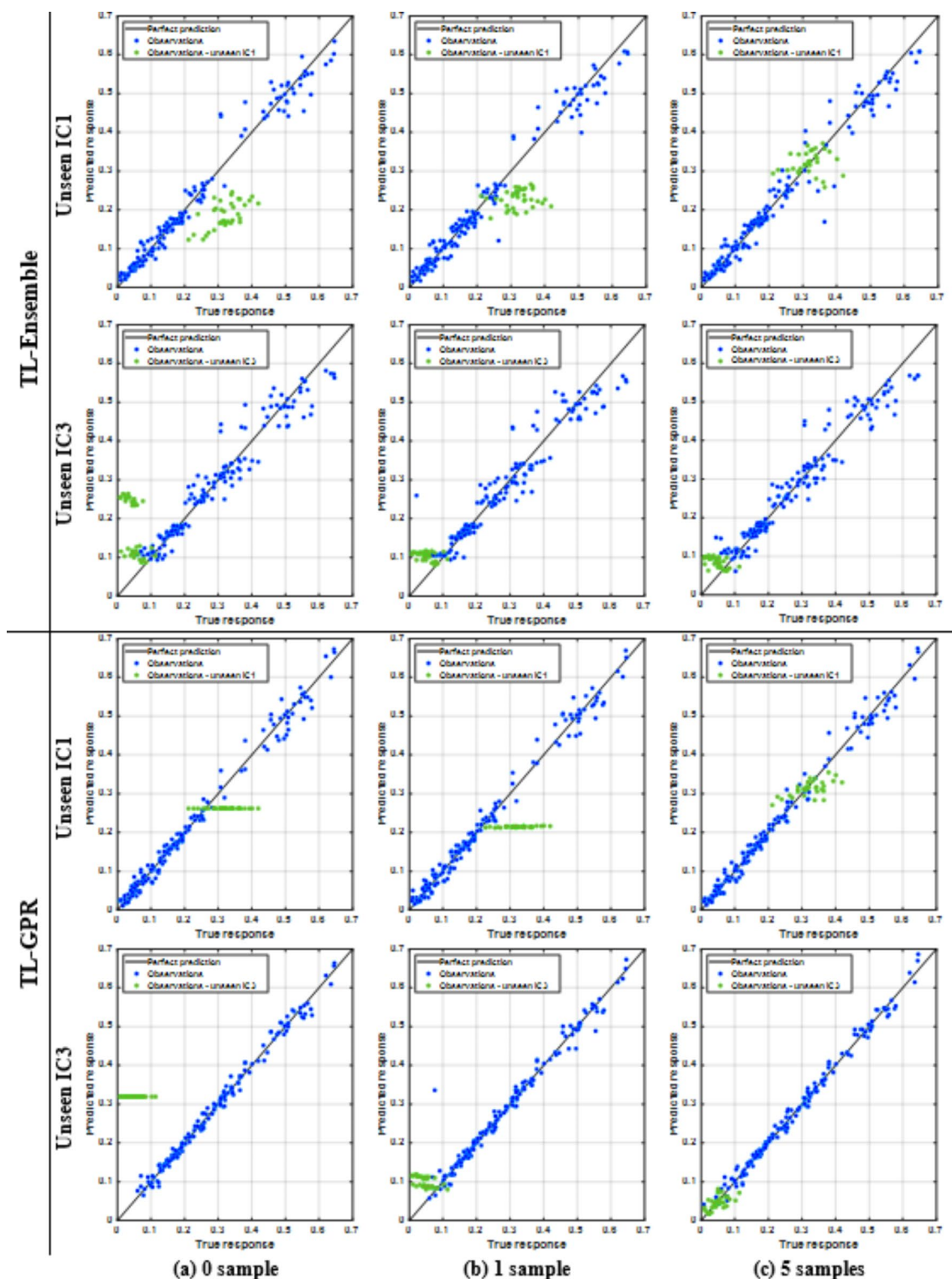


Unseen ICs	Num. of available samples to update	Metrics based on validation											
		RMSE				R <sup>2</sup>				MAE			
		Validation		Unseen IC		Validation		Unseen IC		Validation		Unseen IC	
All ICs were seen		0.0217	±	0.0012	NA	±	0.0012	±	0.0012	±	0.0012	±	0.0012
1	0	0.0286	±	0.0019	0.1426	±	0.0116	±	0.00	0.00	0.0211	±	0.0123
1	1	0.0310	±	0.0017	0.0987	±	0.0269	±	0.00	0.00	0.0218	±	0.0276
1	5	0.0298	±	0.0026	0.0524	±	0.0104	±	0.01	0.04	0.0213	±	0.0092
1	10	0.0296	±	0.0031	0.0350	±	0.0061	±	0.01	0.40	0.0216	±	0.0047
1	20	0.0270	±	0.0011	0.0259	±	0.0035	±	0.00	0.63	0.0200	±	0.0031
2	0	0.0216	±	0.0022	0.1023	±	0.0231	±	0.00	0.00	0.0158	±	0.0177
2	1	0.0234	±	0.0042	0.1016	±	0.0162	±	0.01	0.00	0.0164	±	0.0127
2	5	0.0287	±	0.0060	0.0892	±	0.0154	±	0.01	0.04	0.0196	±	0.0121
2	10	0.0313	±	0.0027	0.0569	±	0.0147	±	0.01	0.41	0.0214	±	0.0135
2	20	0.0263	±	0.0031	0.0363	±	0.0117	±	0.01	0.73	0.0193	±	0.0105
3	0	0.0302	±	0.0041	0.1516	±	0.0008	±	0.01	0.00	0.0220	±	0.0007
3	1	0.0330	±	0.0057	0.1262	±	0.0324	±	0.02	0.00	0.0229	±	0.0243
3	5	0.0348	±	0.0048	0.0789	±	0.0311	±	0.01	0.00	0.0242	±	0.0233
3	10	0.0323	±	0.0040	0.0405	±	0.0188	±	0.01	0.03	0.0228	±	0.0167
3	20	0.0301	±	0.0040	0.0246	±	0.0061	±	0.01	0.15	0.0210	±	0.0050
4	0	0.0262	±	0.0020	0.1539	±	0.0002	±	0.00	0.00	0.0195	±	0.0001
4	1	0.0290	±	0.0026	0.1009	±	0.0493	±	0.00	0.00	0.0206	±	0.0507
4	5	0.0269	±	0.0016	0.0210	±	0.0078	±	0.00	0.26	0.0203	±	0.0061
4	10	0.0257	±	0.0019	0.0167	±	0.0025	±	0.00	0.37	0.0196	±	0.0024
4	20	0.0250	±	0.0027	0.0126	±	0.0027	±	0.00	0.52	0.0188	±	0.0021
5	0	0.0194	±	0.0012	0.1858	±	0.0007	±	0.00	0.00	0.0150	±	0.0002
5	1	0.0287	±	0.0055	0.1378	±	0.0574	±	0.03	0.13	0.0183	±	0.0600
5	5	0.0350	±	0.0107	0.1134	±	0.0596	±	0.05	0.20	0.0228	±	0.0616
5	10	0.0408	±	0.0205	0.1068	±	0.0564	±	0.10	0.31	0.0275	±	0.0590
5	20	0.0284	±	0.0031	0.0480	±	0.0081	±	0.01	0.62	0.0203	±	0.0083
Mean	0	0.0252	±	0.0023	0.1472	±	0.0073	±	0.01	0.00	0.0187	±	0.0062
	1	0.0290	±	0.0040	0.1130	±	0.0364	±	0.01	0.03	0.0200	±	0.0350
	5	0.0310	±	0.0051	0.0710	±	0.0248	±	0.02	0.11	0.0216	±	0.0224
	10	0.0320	±	0.0064	0.0512	±	0.0197	±	0.03	0.30	0.0226	±	0.0193
	20	0.0274	±	0.0028	0.0295	±	0.0064	±	0.01	0.53	0.0199	±	0.0058

Table 5. Performance of TL-Ensemble averaged over 5 different random seed numbers for initialisation.

Unseen ICs	Num. of available samples to update	Metrics based on validation											
		RMSE				R <sup>2</sup>				MAE			
		Validation		Unseen IC		Validation		Unseen IC		Validation		Unseen IC	
All ICs were seen		0.0150	± 0.0011	0.1024	± 0.0553	0.99	± 0.00	0.05	± 0.10	0.0112	± 0.0006	0.0914	± 0.0483
1	0	0.0194	± 0.0009	0.1024	± 0.0553	0.99	± 0.00	0.05	± 0.10	0.0144	± 0.0006	0.0914	± 0.0483
1	1	0.0234	± 0.0067	0.0515	± 0.0351	0.98	± 0.01	0.38	± 0.33	0.0152	± 0.0012	0.0452	± 0.0335
1	5	0.0197	± 0.0011	0.0205	± 0.0097	0.99	± 0.00	0.77	± 0.23	0.0145	± 0.0008	0.0166	± 0.0075
1	10	0.0188	± 0.0019	0.0210	± 0.0139	0.99	± 0.00	0.74	± 0.22	0.0140	± 0.0015	0.0167	± 0.0108
1	20	0.0177	± 0.0012	0.0147	± 0.0055	0.99	± 0.00	0.90	± 0.06	0.0130	± 0.0010	0.0117	± 0.0033
2	0	0.0140	± 0.0011	0.1719	± 0.0070	0.99	± 0.00	0.00	± 0.00	0.0104	± 0.0007	0.1526	± 0.0074
2	1	0.0186	± 0.0034	0.1015	± 0.0082	0.99	± 0.00	0.00	± 0.00	0.0117	± 0.0007	0.0839	± 0.0063
2	5	0.0176	± 0.0023	0.0571	± 0.0214	0.99	± 0.00	0.40	± 0.36	0.0118	± 0.0007	0.0480	± 0.0189
2	10	0.0155	± 0.0008	0.0320	± 0.0034	0.99	± 0.00	0.82	± 0.03	0.0112	± 0.0005	0.0266	± 0.0032
2	20	0.0159	± 0.0011	0.0224	± 0.0048	0.99	± 0.00	0.90	± 0.04	0.0117	± 0.0006	0.0187	± 0.0038
3	0	0.0158	± 0.0011	0.1803	± 0.1031	0.99	± 0.00	0.00	± 0.00	0.0118	± 0.0007	0.1758	± 0.1044
3	1	0.0220	± 0.0056	0.0575	± 0.0269	0.98	± 0.01	0.01	± 0.03	0.0126	± 0.0014	0.0502	± 0.0254
3	5	0.0166	± 0.0038	0.0278	± 0.0068	0.99	± 0.01	0.10	± 0.11	0.0123	± 0.0022	0.0227	± 0.0060
3	10	0.0156	± 0.0013	0.0247	± 0.0069	0.99	± 0.00	0.21	± 0.18	0.0115	± 0.0010	0.0207	± 0.0067
3	20	0.0168	± 0.0011	0.0191	± 0.0023	0.99	± 0.00	0.43	± 0.10	0.0123	± 0.0009	0.0156	± 0.0023
4	0	0.0174	± 0.0012	0.0209	± 0.0056	0.99	± 0.00	0.23	± 0.28	0.0136	± 0.0010	0.0186	± 0.0059
4	1	0.0184	± 0.0019	0.0158	± 0.0044	0.99	± 0.00	0.46	± 0.27	0.0142	± 0.0014	0.0134	± 0.0039
4	5	0.0177	± 0.0016	0.0100	± 0.0040	0.99	± 0.00	0.74	± 0.24	0.0138	± 0.0011	0.0080	± 0.0033
4	10	0.0171	± 0.0010	0.0110	± 0.0030	0.99	± 0.00	0.71	± 0.15	0.0133	± 0.0007	0.0091	± 0.0025
4	20	0.0161	± 0.0010	0.0070	± 0.0010	0.99	± 0.00	0.90	± 0.04	0.0120	± 0.0008	0.0055	± 0.0005
5	0	0.0163	± 0.0007	0.0957	± 0.0256	0.98	± 0.00	0.13	± 0.26	0.0124	± 0.0005	0.0864	± 0.0253
5	1	0.0295	± 0.0139	0.0641	± 0.0244	0.91	± 0.07	0.40	± 0.35	0.0140	± 0.0016	0.0503	± 0.0190
5	5	0.0181	± 0.0020	0.0415	± 0.0147	0.98	± 0.00	0.72	± 0.21	0.0132	± 0.0012	0.0328	± 0.0108
5	10	0.0180	± 0.0019	0.0263	± 0.0024	0.98	± 0.00	0.90	± 0.02	0.0133	± 0.0012	0.0214	± 0.0022
5	20	0.0168	± 0.0012	0.0235	± 0.0045	0.99	± 0.00	0.93	± 0.02	0.0125	± 0.0008	0.0194	± 0.0033
Mean	0	0.0166	± 0.0010	0.1142	± 0.0393	0.99	± 0.00	0.08	± 0.13	0.0125	± 0.0007	0.1050	± 0.0383
	1	0.0224	± 0.0063	0.0581	± 0.0198	0.97	± 0.02	0.25	± 0.20	0.0135	± 0.0013	0.0486	± 0.0177
	5	0.0180	± 0.0021	0.0314	± 0.0113	0.99	± 0.00	0.55	± 0.23	0.0131	± 0.0012	0.0256	± 0.0093
	10	0.0170	± 0.0014	0.0230	± 0.0059	0.99	± 0.00	0.67	± 0.14	0.0127	± 0.0010	0.0189	± 0.0051
	20	0.0167	± 0.0011	0.0174	± 0.0036	0.99	± 0.00	0.81	± 0.05	0.0123	± 0.0008	0.0142	± 0.0027

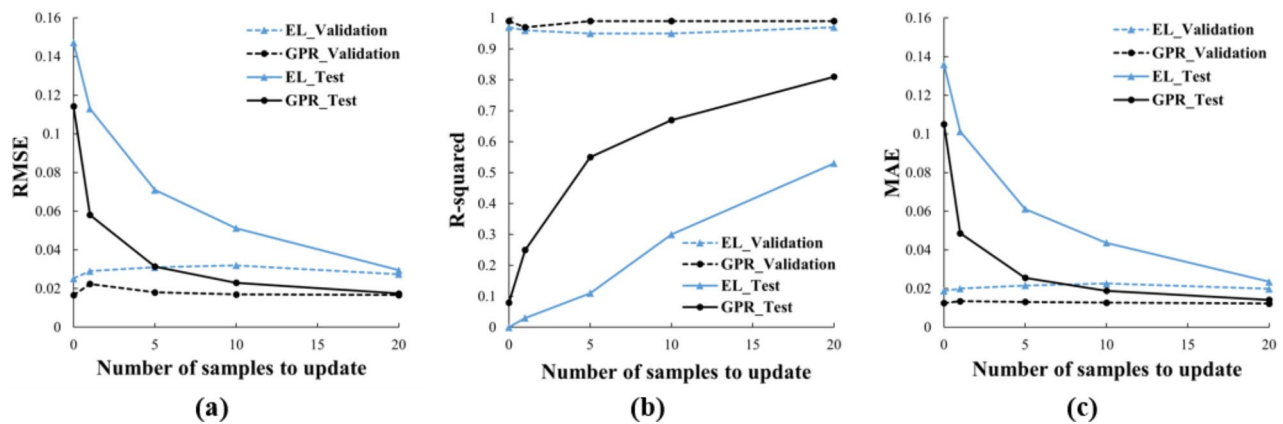
Table 6. Performance of TL-GPR averaged over 5 different random seed numbers for initialisation.



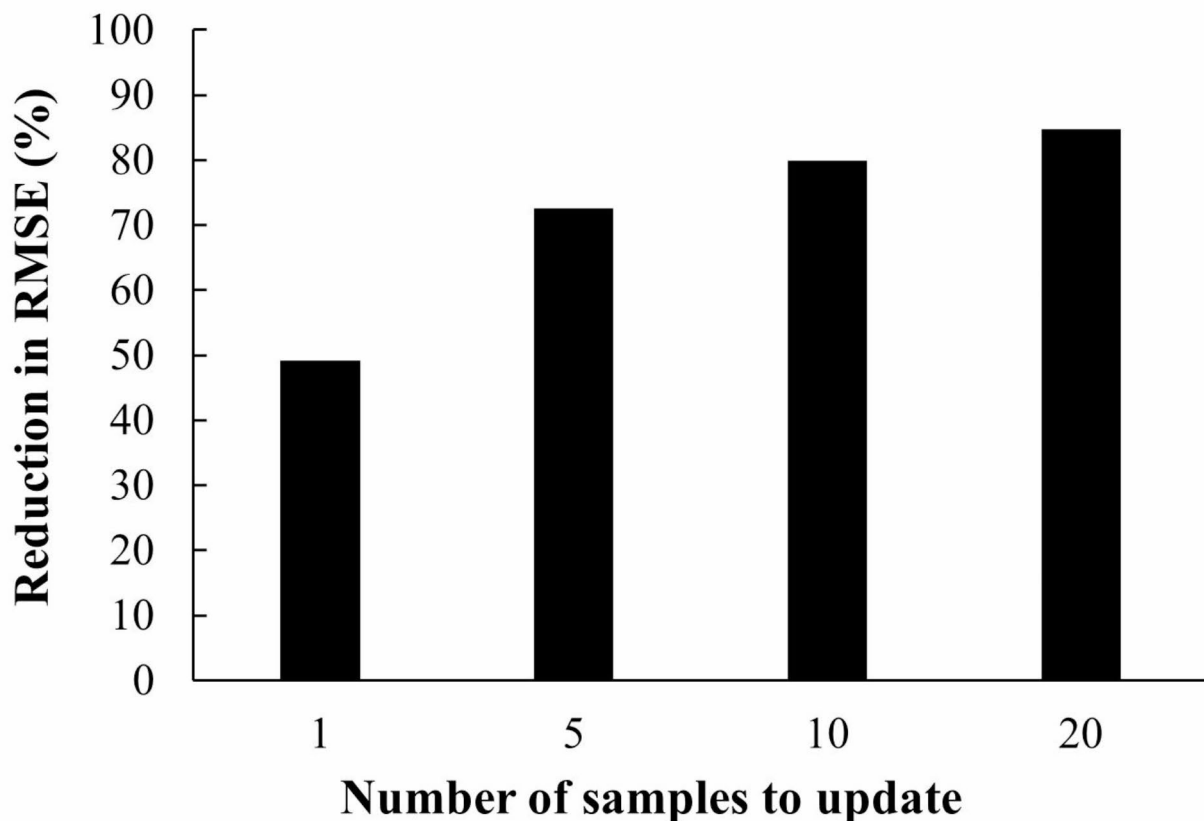
**Fig. 7.** Predicted and true responses for unseen initial configurations IC1 and IC3 with TL-Ensemble and TL-GPR, updated with varying numbers of samples: (a) 0 (no update), (b) 1, and (c) 5 from the unseen IC.

materials are fully-mixed, and the TL-SMs' performance for IC3 was inferior compared to other ICs. However, the performance significantly improved after updating the model with a few samples from the “unseen IC”. For instance, the RMSE of TL-GPR was reduced by an average of 50% by retraining the model with just one additional sample, highlighting the adaptability and effectiveness of our proposed adaptive TL approach.

To overcome the challenge encountered by the TL model for IC3 (i.e., fully-mixed configuration), additional intermediate initial configurations between fully-segregated and fully-mixed ones can be incorporated.

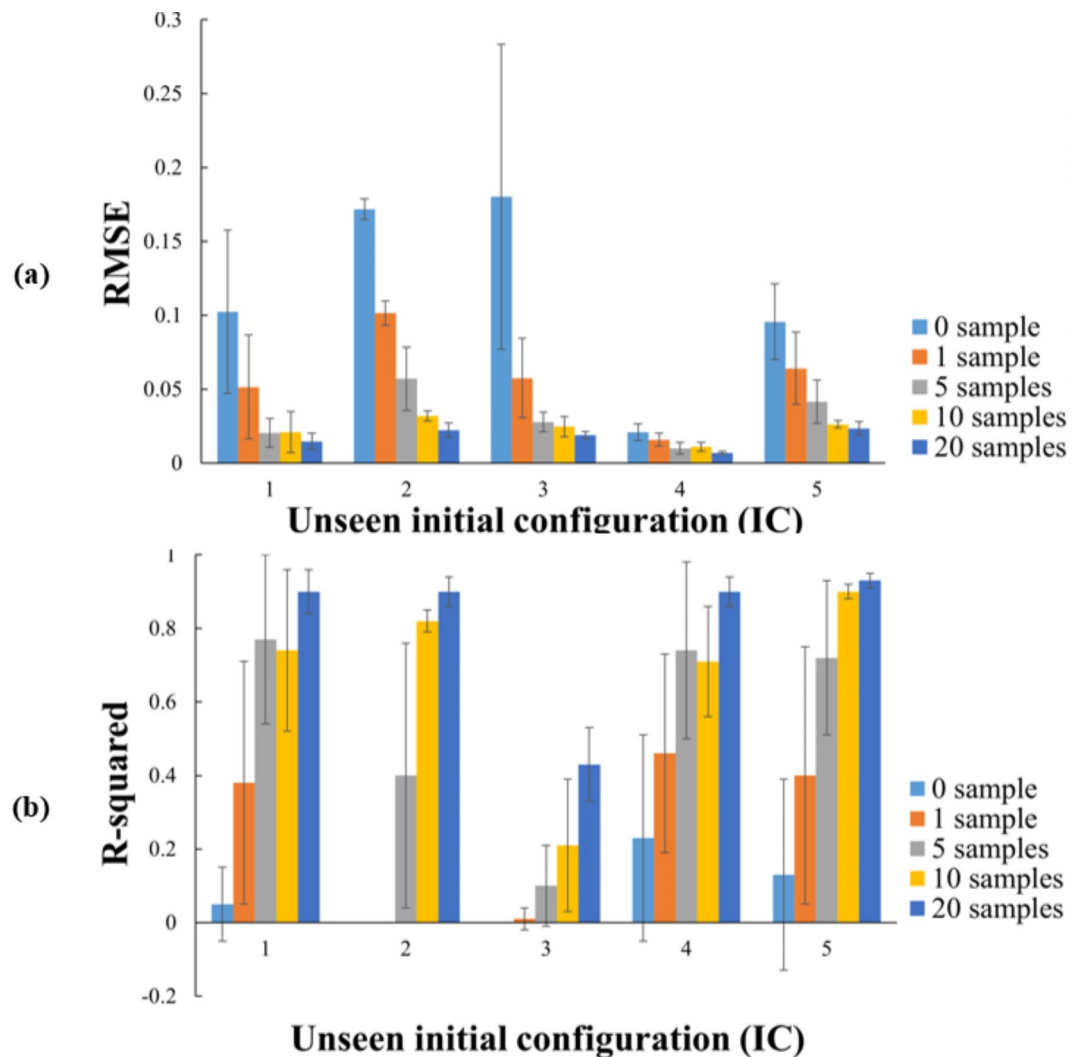


**Fig. 8.** Comparison between the mean performances of TL-Ensemble and TL-GPR during validation and test phases for different numbers of available samples from unseen IC to update in terms of (a) RMSE, (b)  $R^2$ , and (c) MAE.



**Fig. 9.** Reduction in RMSE of TL-GPR model following updating with different numbers of samples from unseen ICs.

Additionally, our surrogate model is currently constrained to predicting segregation for pellets and sinter with specified material properties and certain geometrical properties of the system. To improve the generalisability of the model, it is essential to vary and include these material- and geometry-related properties in the training phase. Furthermore, our surrogate model is developed for only one response variable. However, in DEM model calibration problems, multiple responses are typically considered simultaneously. Future research endeavours could focus on addressing these challenges, including incorporating multiple response variables and exploring a broader range of material and geometric properties to improve model performance and applicability.



**Fig. 10.** Accuracy of TL-GPR across various unseen ICs and for different numbers of samples to update the model in terms of (a) RMSE and (b) R-squared.

The authors would like to acknowledge dr.ir. Jan van der Stel and ir. Allert Adema from Tata Steel IJmuiden for the insightful discussions in the context of blast furnace.

### Data availability

The data presented in this study is available upon request from the corresponding author.

Received: 23 August 2024; Accepted: 30 October 2024

Published online: 06 November 2024

### References

1. Rosato, A. D., Blackmore, D. L., Zhang, N. & Lan, Y. A perspective on vibration-induced size segregation of granular materials. *Chem. Eng. Sci.* **57**, 265–275 (2002).
2. Gray, J. M. N. T. Particle segregation in dense granular flows. **50**, 407–433. (2018).
3. Shinohara, K. & Miyata, S. I. Mechanism of density segregation of particles in filling vessels. *Industrial Eng. Chem. Process. Des. Dev.* **23**, 423–428 (1984).
4. Jain, N., Ottino, J. M. & Lueptow, R. M. Regimes of segregation and mixing in combined size and density granular systems: an experimental study. *Granul. Matter.* **7**, 69–81 (2005).
5. Duffy, S. P. & Puri, V. M. Primary segregation shear cell for size-segregation analysis of binary mixtures. *Kona Powder Part. J.* **20**, 196–207 (2002).
6. Hadi, A., Roepal, R., Pang, Y. & Schott, D. L. DEM modelling of segregation in Granular materials: a review. *Kona Powder Part. J.* <https://doi.org/10.14356/kona.2024017> (2023).
7. Cundall, P. A. & Strack, O. D. L. A discrete numerical model for granular assemblies. *Geotechnique* **29**, 47–65 (1979).
8. Roepal, R., Pang, Y., Adema, A., van der Stel, J. & Schott, D. Modelling of phenomena affecting blast furnace burden permeability using the discrete element method (DEM)—A review. *Powder Technol.* **415** <https://doi.org/10.1016/j.powtec.2022.118161> (2023).

9. Lee, S. J., Hashash, Y. M. A. & Nezami, E. G. Simulation of triaxial compression tests with polyhedral discrete elements. *Comput. Geotech.* **43**, 92–100 (2012).
10. Huang, H. & Tutumluer, E. Discrete element modeling for fouled railroad ballast. *Constr. Build. Mater.* **25**, 3306–3312 (2011).
11. Obermayr, M., Dressler, K., Vrettos, C. & Eberhard, P. Prediction of draft forces in cohesionless soil with the discrete element Method. *J. Terramech.* **48**, 347–358 (2011).
12. Roessler, T., Richter, C., Katterfeld, A. & Will, F. Development of a standard calibration procedure for the DEM parameters of cohesionless bulk materials—part I: solving the problem of ambiguous parameter combinations. *Powder Technol.* **343**, 803–812 (2019).
13. Irazábal, J., Salazar, F. & Vicente, D. J. A methodology for calibrating parameters in discrete element models based on machine learning surrogates. *Comput. Part. Mech.* **10**, 1031–1047 (2023).
14. Richter, C., Rößler, T., Kunze, G., Katterfeld, A. & Will, F. Development of a standard calibration procedure for the DEM parameters of cohesionless bulk materials – part II: efficient optimization-based calibration. *Powder Technol.* **360**, 967–976 (2020).
15. Yoon, J. Application of experimental design and optimization to PFC model calibration in uniaxial compression simulation. *Int. J. Rock Mech. Min. Sci.* **44**, 871–889 (2007).
16. Heß, G., Richter, C. & Katterfeld, A. Simulation of the dynamic interaction between bulk material and heavy equipment: Calibration and validation. In *ICBMH – 12th International Conference on Bulk Materials Storage, Handling and Transportation, Proceedings 427–436*. (2016).
17. Mohajeri, M. J., Do, H. Q. & Schott, D. L. DEM calibration of cohesive material in the ring shear test by applying a genetic algorithm framework. *Adv. Powder Technol.* **31**, 1838–1850 (2020).
18. Do, H. Q., Aragón, A. M. & Schott, D. L. A calibration framework for discrete element model parameters using genetic algorithms. *Adv. Powder Technol.* **29**, 1393–1403 (2018).
19. Fransen, M. P., Langelaar, M. & Schott, D. L. Application of DEM-based metamodels in bulk handling equipment design: Methodology and DEM case study. *Powder Technol.* **393**, 205–218 (2021).
20. Chinesta, F., Cueto, E. & Klusemann, B. Empowering materials processing and performance from data and AI. *Materials* **14** <https://doi.org/10.3390/ma14164409> (2021).
21. Rackl, M. & Hanley, K. J. A methodical calibration procedure for discrete element models. *Powder Technol.* **307**, 73–83 (2017).
22. Grobbel, J., Brendelberger, S., Henninger, M., Sattler, C. & Pitz-Paal, R. Calibration of parameters for DEM simulations of solar particle receivers by bulk experiments and surrogate functions. *Powder Technol.* **364**, 831–844 (2020).
23. De Pue, J., Di Emidio, G., Flores, V., Bezuijen, R. D. & Cornelis, W. M. A. Calibration of DEM material parameters to simulate stress-strain behaviour of unsaturated soils during uniaxial compression. *Soil Tillage. Res.* **194**, (2019).
24. Fransen, M. P., Langelaar, M. & Schott, D. L. Including stochastics in metamodel-based DEM model calibration. *Powder Technol.* **406**, (2022).
25. Westbrink, F., Elbel, A., Schwung, A. & Ding, S. X. Optimization of DEM parameters using multi-objective reinforcement learning. *Powder Technol.* **379**, 602–616 (2021).
26. Cheng, H. et al. An iterative bayesian filtering framework for fast and automated calibration of DEM models. *Comput. Methods Appl. Mech. Eng.* **350**, 268–294 (2019).
27. Hartmann, P., Cheng, H. & Thoeni, K. Performance study of iterative bayesian filtering to develop an efficient calibration framework for DEM. *Comput. Geotech.* **141**, (2022).
28. El-Kassem, B., Salloum, N., Brinz, T., Heider, Y. & Markert, B. A multivariate regression parametric study on DEM input parameters of free-flowing and cohesive powders with experimental data-based validation. *Comput. Part. Mech.* **8**, 87–111 (2021).
29. Benvenuti, L., Kloss, C. & Pirker, S. Identification of DEM simulation parameters by Artificial neural networks and bulk experiments. *Powder Technol.* **291**, 456–465 (2016).
30. Zhou, H., Hu, Z., Chen, J., Lv, X. & Xie, N. Calibration of DEM models for irregular particles based on experimental design method and bulk experiments. *Powder Technol.* **332**, 210–223 (2018).
31. Ye, F. et al. Calibration and verification of DEM parameters for dynamic particle flow conditions using a backpropagation neural network. *Adv. Powder Technol.* **30**, 292–301 (2019).
32. Zhu, H. P., Zhou, Z. Y., Yang, R. Y. & Yu, A. B. Discrete particle simulation of particulate systems: theoretical developments. *Chem. Eng. Sci.* **62**, 3378–3396 (2007).
33. Ai, J., Chen, J. F., Rotter, J. M. & Ooi, J. Y. Assessment of rolling resistance models in discrete element simulations. *Powder Technol.* **206**, 269–282 (2011).
34. Tripathi, A. et al. Quantitative DEM simulation of pellet and sinter particles using rolling friction estimated from image analysis. *Powder Technol.* **380**, 288–302 (2021).
35. Chakrabarty, A., Biswas, R., Basu, S. & Nag, S. Characterisation of binary mixtures of pellets and sinter for DEM simulations. *Adv. Powder Technol.* **33**, (2022).
36. Wensrich, C. M. & Katterfeld, A. Rolling friction as a technique for modelling particle shape in DEM. *Powder Technol.* **217**, 409–417 (2012).
37. Viera Valencia, L. F. & Garcia Giraldo, D. *Angewandte Chemie International Edition*, 6(11), 951–952. **2** <https://www.tudelft.nl/dhp/c/ark/44463/DelftBluePhase1> (2019).
38. Wei, H. et al. Measurement and simulation validation of DEM parameters of pellet, sinter and coke particles. *Powder Technol.* **364**, 593–603 (2020).
39. Yu, Y. & Saxén, H. Particle flow and behavior at bell-less charging of the blast furnace. *Steel Res. Int.* **84**, 1018–1033 (2013).
40. Mitra, T. Modeling of burden distribution in the blast furnace. (2016).
41. Yu, Y. & Saxén, H. Flow of pellet and coke particles in and from a fixed chute. *Ind. Eng. Chem. Res.* **51**, 7383–7397 (2012).
42. Lu, Y., Jiang, Z., Zhang, X., Wang, J. & Zhang, X. Vertical section observation of the solid flow in a blast furnace with a cutting method. *Metals* **9**, (2019).
43. Yu, Y. & Saxén, H. Experimental and DEM study of segregation of ternary size particles in a blast furnace top bunker model. *Chem. Eng. Sci.* **65**, 5237–5250 (2010).
44. Mio, H. et al. Validation of particle size segregation of sintered ore during flowing through laboratory-scale chute by discrete element method. *ISIJ Int.* **48**, 1696–1703 (2008).
45. Basu, S. et al. Modeling and simulation of mechanical degradation of iron ore sinter in a complex transfer chute system using the discrete element model and a particle breakage model. *Powder Technol.* **417**, (2023).
46. Barrios, G. K. P., de Carvalho, R. M., Kwade, A. & Tavares, L. M. Contact parameter estimation for DEM simulation of iron ore pellet handling. *Powder Technol.* **248**, 84–93 (2013).
47. Izard, E., Moreau, M. & Ravier, P. Discrete element method simulation of segregation pattern in a sinter cooler charging chute system. *Particuology* **59**, 34–42 (2021).
48. Jones, B. & Nachtsheim, C. J. A class of three-level designs for definitive screening in the presence of second-order effects. *J. Qual. Technol.* **43**, 1–15 (2011).
49. Jones, B. & Nachtsheim, C. J. Effective design-based model selection for definitive screening designs. *Technometrics* **59**, 319–329 (2017).
50. Zheng, A. & Casari, A. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists* (O'Reilly Media, Inc.; 2018).
51. Hancock, J. T. & Khoshgoftaar, T. M. Survey on categorical data for neural networks. *J. Big Data* **7**, (2020).



52. Freedman, D. A. Statistical models: theory and practice. *Stat. Models Theory Pract.* <https://doi.org/10.1017/CBO9780511815867> (2009).
53. Welsch, R. E. Robust regression using iteratively reweighted least-squares. *Commun. Stat. Theory Methods* **6**, 813–827 (1977).
54. Neter, J., Kutner, M. H., Nachtsheim, C. J. & Wasserman, W. Applied linear statistical models. (1996).
55. Hearst, M. A., Scholkopf, B., Dumais, S., Osuna, E. & Platt, J. Support vector machines. *IEEE Intell. Syst. Their Appl.* **13**, 18–28 (1998).
56. Vapnik, V. N. The Nature of Statistical Learning Theory. *Nat. Stat. Learn. Theory.* <https://doi.org/10.1007/978-1-4757-2440-0> (1995).
57. Smola, A. J. & Schölkopf, B. A tutorial on support vector regression. *Stat. Comput.* **14** 199–222. <https://doi.org/10.1023/B:STCO.0000035301.49549.88> (2004).
58. Witten, I. H., Frank, E. & Geller, J. Data mining: Practical machine learning tools and techniques with Java implementations. *SIGMOD Record.* **31**, 76–77 (2002).
59. Breiman, L. Bagging predictors. *Mach. Learn.* **24**, 123–140 (1996).
60. Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32 (2001).
61. Rasmussen, C. E. Gaussian processes in machine learning. In *Summer school on machine learning* 63–71 (Springer, 2003).
62. Wu, Z. et al. Machine learning approach to predicting the macro-mechanical properties of rock from the meso-mechanical parameters. *Comput. Geotech.* **166**, 105933. (2024).
63. Shalev-Shwartz, S. & Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. *Understanding Machine Learning: From Theory to Algorithms* 9781107057 (2013).
64. Bishop, C. M. Pattern recognition and machine learning. *Springer Google Schola* **2**, 5–43 (2006).
65. Haykin, S. *Neural Networks and Learning Machines*, 3/E. (Pearson Education India, 2009).
66. Arlot, S. & Celisse, A. A survey of cross-validation procedures for model selection. *Stat. Surv.* **4**, 40–79 (2010).
67. Snoek, J., Larochelle, H. & Adams, R. P. Practical bayesian optimization of machine learning algorithms. *Adv. Neural. Inf. Process. Syst.* **4**, 2951–2959 (2012).
68. Weiss, K., Khoshgoftaar, T. M. & Wang, D. D. A survey of transfer learning. *J. Big Data* **3**, (2016).
69. Pan, S. J. & Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* **22** 1345–1359. <https://doi.org/10.1109/TKDE.2009.191> (2010).
70. Iman, M., Arabnia, H. R. & Rasheed, K. A Review of deep transfer learning and recent advancements. *Technologies* **11** <https://doi.org/10.3390/technologies11020040> (2023).
71. Sheela, K. G. & Deepa, S. N. Review on methods to fix number of hidden neurons in neural networks. *Math. Problems Eng.* (2013).

## Acknowledgements

This research was carried out under project number T18019 in the framework of the Research Program of the Materials innovation institute (M2i) ([www.m2i.nl](http://www.m2i.nl)) supported by the Dutch government. The authors would like to acknowledge ir. Jan van der Stel and dr.ir. Allert Adema from Tata Steel IJmuiden for the insightful discussions in the context of blast furnace.

## Author contributions

Conceptualisation, A.H., and M.M.; methodology, A.H., and M.M.; software, A.H., and M.M.; validation, A.H., M.M., Y.P., and D.S.; formal analysis, A.H., M.M.; investigation, A.H., M.M.; writing—original draft preparation, A.H., M.M.; writing—review and editing, A.H., M.M., Y.P., and D.S.; visualization, A.H., and M.M.; supervision, Y.P., and D.S.; All authors have read and agreed to the published version of the manuscript.

## Declarations

## Competing interests

The authors declare no competing interests.

## Additional information

**Supplementary Information** The online version contains supplementary material available at <https://doi.org/10.1038/s41598-024-78455-7>.

**Correspondence** and requests for materials should be addressed to A.H.

**Reprints and permissions information** is available at [www.nature.com/reprints](http://www.nature.com/reprints).

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2024