Applied Physics A (2024) 130:588
https://doi.org/10.1007/s00339-024-07728-9

Applied Physics A
Materials Science & Processing

Exploring active learning strategies for predictive models in mechanics of materials

Yingbin Chen[1] · Phillip Deierling[1] · Shaoping Xiao[1]

Received: 12 May 2024 / Accepted: 5 July 2024

Abstract
Machine learning (ML) has found widespread applications in predicting material properties and mechanical behaviors across various scales in computational materials science. This data-driven approach typically relies on large datasets to train predictive models. However, labeling data samples through numerical simulations in materials science can be computationally intensive. In response to this challenge, this research delves into the utilization of active learning (AL) strategies to selectively label the most informative data samples for regression and classification models. Additionally, Several novel AL strategies were developed to enhance the development of probabilistic ML models. Through illustrative examples, this study demonstrated that AL could significantly boost ML training efficiency by labeling only a small subset of data samples while achieving exceptional model performance.

Keywords Active learning · Molecular dynamics · Peridynamics · Machine learning

1 Introduction

The emergence of Artificial Intelligence (AI), particularly Machine Learning (ML) and Deep Learning (DL), has revolutionized computational approaches in materials science [1]. Early studies have demonstrated the versatility of Artificial Neural Networks (ANNs) in predicting material properties [2], evaluating fatigue life [3], analyzing crack and damage [4], and approximating dynamic mechanical behaviors [5]. Moreover, ANNs have been pivotal in optimizing manufacturing processes [6] and facilitating design improvements [7]. They have been seamlessly integrated into multiscale modeling and simulations, enabling the transfer of information across different spatial scales to enhance our understanding of material behavior [8–10]. ML and DL techniques have further accelerated the discovery of novel materials and the analysis of chemical sensing compounds [11]. Recent reviews such as [12, 13] offered comprehensive insights into these advancements. However, employing data-driven approaches in computational materials science often entails extensive data generation through numerical modeling and simulations [14–16], which can be time-consuming. Therefore, optimizing data generation processes to minimize redundant computations is crucial.

Active Learning (AL) [17] can effectively address the above-mentioned issue. It is a subfield of ML designed to identify and label the most informative samples from a large pool of unlabeled data for labeling, thereby achieving maximum model performance with minimal data usage. AL operates under different scenarios: pool-based [18], stream-based [19], and membership query synthesis-based [20]. In pool-based AL, the algorithm selects informative samples from a pool containing all available unlabeled samples. In contrast, stream-based AL processes and selects valuable samples sequentially from a continuous stream of data. Membership query synthesis-based AL, on the other hand, can generate unlabeled samples within the input space without a predefined pool. This paper focuses on pool-based AL, which is particularly well-suited for datasets in materials

Phillip Deierling and Shaoping Xiao have contributed equally to this work.

✉ Yingbin Chen
yingbin-chen@uiowa.edu

Phillip Deierling
phillip-deierling@uiowa.edu

Shaoping Xiao
shaoping-xiao@uiowa.edu

[1] Department of Mechanical Engineering, Iowa Technology Institute, University of Iowa, 3131 Seamans Center, Iowa City, IA 52242, USA

Published online: 24 July 2024

science. Notably, the effectiveness of pool-based AL largely depends on the query strategies employed to identify the most informative samples. These strategies can be broadly categorized into approaches suitable for classification tasks [21–24] and those tailored for regression tasks [25–28].

AL has broad applications in different fields, including visual data processing, Natural Language Processing (NLP), gene expression, and more [29]. In visual data processing, for example, Yuan et al. [30] demonstrated that AL could be employed to selectively annotate video sequences, significantly enhancing the performance of Convolutional Neural Network (CNN) models for visual tracking while reducing labeling costs. AL has also made significant inroads into NLP. A recent example by Schröder [31] investigated the performance of uncertainty-based query strategies in transformer-based language models for text classification. The study demonstrated that traditional strategies, previously considered less effective, achieved notable success with transformers. For gene expression, Begum et al. [32] explored the use of an AL model integrated with a support vector machine (SVM) and a feature-selection algorithm called Symmetrical Uncertainty (SU). This approach was applied to identify biomarkers in cancer gene expression data, enhancing the accuracy of cancer predictions while also reducing the labeling cost and effort involved in the analysis. Furthermore, Xiang et al. [33] proposed an AL approach that selected experimental points close to the limit state surface (LSS) from the Monte Carlo population. The authors innovatively integrated a weighted sampling technique to ensure that these selected points are uniformly distributed. This method significantly advanced the application of AL in structural reliability analysis.

AL has also shown significant promise in materials science, with various applications tailored to specific goals. Lookman et al.'s review paper [34] highlighted the effectiveness of AL coupled with adaptive sampling to streamline high-throughput density functional theory calculations for data labeling. Notable achievements included the discovery of piezoelectrics with substantial electrostrains [35] and the development of surrogate models to optimize optoelectronic devices [36]. In a recent study by Farache et al. [37], AL showcased its power in identifying multiple principal component alloys (MPCAs) with high melting temperatures through molecular dynamics (MD) simulations. Their work introduced a fully autonomous workflow that utilized Random Forests within the AL framework to manage the inherent uncertainties in MD simulations. This approach enabled efficient exploration of the high-dimensional compositional space of MPCAs. Additionally, Allotey et al. [38] proposed integrating Graph Neural Networks (GNNs) with Gaussian Processes (GPs) in an AL framework to predict the properties of solid-state materials. Employing an entropy-based sampling approach, they prioritized labeling unlabeled samples with the highest uncertainty predicted by GP. This AL strategy, grounded in information theory, maximizes entropy reduction across the dataset, thereby optimizing the learning process. Remarkably, this method has demonstrated a twofold increase in the rate of model performance improvement on test datasets compared to random sampling.

This paper explored various query strategies developed by Wu et al. [39] and Lewis [40], and assessed their impacts on the performance of ML models in studying the mechanics of materials across different scales, comparing them with random sampling. The study examined one-dimensional molecule chains and three-dimensional metal-ceramics composite materials. Both regression and classification tasks were investigated, such as predicting material strength and forecasting material failure. Data labeling was conducted using MD or peridynamics simulations. The ML methods utilized in this study included SVMs, ANNs, and CNNs. Additionally, drawing inspiration from the previous work [39], several novel strategies for probabilistic ML regression models based on the Maximum Likelihood Estimation (MLE) were designed. Uncertainties were introduced in data collection during MD simulations, and the impact of these AL strategies on improving model performance was evaluated.

The research makes two primary contributions. Firstly, it comprehensively examines the applications of greedy sampling and its variations [39] for regression tasks, as well as uncertainty sampling for classification tasks, providing insights into their applicability and effectiveness in predicting material properties and behaviors. This offers an alternative and more efficient approach to training ML models in a data-driven manner for multiscale modeling and simulations [1]. Secondly, the study proposes several novel AL strategies for probabilistic ML models that output probability distributions rather than single values in regression tasks. Existing studies, such as those by Khatamsaz et al. [41] and Allotey et al. [38], have utilized GP for uncertainty (i.e., entropy) estimation to select the most informative samples for labeling. However, their AL frameworks were only applied to deterministic ML models that did not provide predictions with probabilities or confidence intervals. In contrast, our developed AL strategies, aligned with the pool-based greedy sampling framework, are specifically designed for probabilistic ML models in regression tasks. These user-friendly and

versatile strategies allow straightforward adaptation across various contexts.

This paper is structured as follows. After the introduction, the Methods section provides a comprehensive overview of the simulation techniques, including MD, peridynamics, and various ML methods. Subsequently, the AL section introduces several specific pool-based AL strategies with algorithms outlined for classification and regression tasks. Additionally, novel AL strategies are developed for probabilistic ML models. Following this, the Examples and Discussions section presents case studies on one-dimensional molecule chains and three-dimensional metal-ceramic composites, illustrating the practical applications and benefits of these AL approaches in predicting the material properties and mechanical behavior. Finally, the Conclusion summarizes our findings and outlines future research directions.

2 Methods

2.1 Molecular dynamics

MD stands as a premier simulation technique extensively used to explore physical phenomena and mechanical behaviors at the nanoscale [42, 43]. Fundamentally rooted in classical Newtonian motion, MD simulations offer detailed insights into the molecular structure and dynamics of materials, providing valuable information on atomistic displacements and velocities. By employing Newton's equations of motion, depicted below, the trajectories of atoms and molecules can be determined, allowing researchers to explore and understand the intricate behaviors occurring at the molecular level.

$$m_i \boldsymbol{a}_i = \boldsymbol{f}_i = -\frac{\partial U}{\partial \boldsymbol{r}_i} \tag{1}$$

where m_i is the mass of atom i, $\boldsymbol{a}_i$ is its acceleration, $\boldsymbol{f}_i$ is the atomistic force applied on atom i, $\boldsymbol{r}_i$ is atom i's positive vector, and U is the total potential energy of the simulated system.

After the accelerations are computed from Eq. (1), the atomistic velocities ($\boldsymbol{v}_i$) and displacements ($\boldsymbol{u}_i$) can be updated via time integration using the following velocity Verlet method.

$$\boldsymbol{u}_i(t+\Delta t) = \boldsymbol{u}_i(t) + \boldsymbol{v}_i(t)\Delta t + \frac{1}{2}\boldsymbol{a}_i(t)\Delta t^2 \tag{2}$$

$$\boldsymbol{a}_i(t+\Delta t) = \frac{\boldsymbol{f}_i(\boldsymbol{r}_i(t+\Delta t))}{m_i} \tag{3}$$

$$\boldsymbol{v}_i(t+\Delta t) = \boldsymbol{v}_i(t) + \frac{\Delta t}{2}[\boldsymbol{a}_i(t) + \boldsymbol{a}_i(t+\Delta t)] \tag{4}$$

where Δt is the time step.

The initial configuration of the simulated molecular system can be achieved through relaxation, wherein the volume and temperature of the simulated system are maintained over a specified number of time steps. Once the system attains thermodynamic equilibrium, the atomic velocities conform to the Maxwell-Boltzmann distribution, as formulated below. This distribution describes how the velocities (or energies) of a mixture of atoms vary at a particular temperature T.

$$P(v) = \sqrt{\frac{2}{\pi}}\left(\frac{m}{k_B T}\right)^{\frac{3}{2}} v^2 e^{-\frac{mv^2}{2k_B T}} \tag{5}$$

where $k_B = 1.38 \times 10^{-23} JK^{-1}$ is the Boltzmann constant.

One technique for regulating the simulated system, consisting of N atoms, to achieve a desired temperature T_0 is velocity scaling. Following the update of atomistic velocities by solving the equations of motion, i.e., Eq. (1), the system's absolute temperature can be determined by calculating the average kinetic energy from the following equation.

$$T = \frac{2}{nk_B}\frac{1}{N}\sum_i \frac{m_i v_i^2}{2} \tag{6}$$

where n is the number of degrees of freedom per atom. Subsequently, the velocities are rescaled by multiplying a scalar of $\sqrt{T_0/T}$. Other commonly used temperature regulation techniques include Berendsen, Andersen, and Nose-Hoover thermostats [44].

Moreover, the subsequent equation can be employed to evaluate the atomic-level Cauchy stress tensor, denoted as $\boldsymbol{\sigma}$, using the interatomic distances $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$ and forces $\boldsymbol{f}_{ij}$ obtained after MD simulations.

$$\boldsymbol{\sigma} = \frac{1}{2V}\sum_{i,j(i\neq j)} \boldsymbol{r}_{ij} \otimes \boldsymbol{f}_{ij} \tag{7}$$

where V is the total volume of the simulated system, and $\otimes$ represents the tensor product of two vectors.

2.2 Peridynamics

Peridynamics [45] serves as a valuable approach for micro- or macro-scale simulations grounded in continuum mechanics principles. Diverging from traditional numerical methods like finite element methods [46] or meshfree particle methods [47], peridynamics models reformulate the governing equation by substituting

derivative terms with integral terms. In peridynamics, the mechanical behavior of discrete points is determined by interaction forces exerted by adjacent points within a specified range, referred to as the horizon. Thus, peridynamics is classified as a type of nonlocal method, which also encompasses the Nonlocal Operator Method (NOM) [48, 49] and dual-horizon peridynamics—an advanced version of the traditional peridynamic approach [50]. These nonlocal methods effectively address challenges that the local continuum mechanics approach faces, particularly in modeling complex phenomena such as material fractures and the interfaces of composites. Peridynamics can be categorized into bond-based and state-based. In bond-based peridynamics, the simulation domain is discretized into equally spaced material points connected by virtual bonds. The equations of motion for a material point $\mathbf{x}$, considering other material points $\mathbf{x}'$ within its neighborhood $H(\mathbf{x})$, can be expressed as follows.

$$\rho(\mathbf{x})\ddot{\boldsymbol{u}}(\mathbf{x},t) = \int_{H_{\mathbf{x}}} \boldsymbol{f}(\boldsymbol{\eta},\boldsymbol{\xi},t)dV_{\mathbf{x}'} + \boldsymbol{b}(\mathbf{x},t) \tag{8}$$

where ρ represents the density, $\boldsymbol{u}$ denotes the displacement, $\boldsymbol{f}$ denotes the pairwise bond force vector, and $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$ are the relative position and displacement vectors between $\mathbf{x}$ and $\mathbf{x}'$, respectively. Additionally, $\boldsymbol{b}$ stands for the body force vector.

The bond force can be calculated based on bond stretch, and the bond micromodulus can be determined from material properties. It is noteworthy that classical bond-based peridynamics is designed exclusively for materials with a Poisson's ratio of 0.25. However, to account for the effects of bond rotation in more general materials, Zhu and Ni [51] introduced updates to the calculation of bond force.

In this study, we employ state-based peridynamics [52], in which the governing equation is established as:

$$\rho(\mathbf{x})\ddot{\boldsymbol{u}}(\mathbf{x},t) = \int_{H_{\mathbf{x}}} \left\{\underline{\mathbf{T}}[\mathbf{x},t]\langle\mathbf{x}'-\mathbf{x}\rangle - \underline{\mathbf{T}}[\mathbf{x}',t]\langle\mathbf{x}-\mathbf{x}'\rangle\right\}dV_{\mathbf{x}'} + b(\mathbf{x},t) \tag{9}$$

Here, the bond force vector $\boldsymbol{f}$, as specified in Eq. (8), is redefined as $\underline{\mathbf{T}}[\mathbf{x},t]\langle\mathbf{x}'-\mathbf{x}\rangle - \underline{\mathbf{T}}[\mathbf{x}',t]\langle\mathbf{x}-\mathbf{x}'\rangle$. The terms $\underline{\mathbf{T}}[\mathbf{x},t]\langle\mathbf{x}'-\mathbf{x}\rangle$ and $\underline{\mathbf{T}}[\mathbf{x}',t]\langle\mathbf{x}-\mathbf{x}'\rangle$ refer to the force vector state. The vector state operator $\underline{\mathbf{T}}$ depends on the positions $\mathbf{x}$ or $\mathbf{x}'$ and time t. This operator maps the relative position vector between any two points $\mathbf{x}$ and $\mathbf{x}'$ onto the force vector state field. Specifically, $\underline{\mathbf{T}}$ is characterized by the relative position $\boldsymbol{\eta}$, the relative displacement $\boldsymbol{\xi}$, and the material constants including bulk modulus K and shear modulus G. Please consult Silling et al. [52] for a detailed derivation and discussion of the methodology.

In state-based peridynamics, when the relative position vector between two material points, represented by a bond, is stretched beyond the critical stretch s_C, the bond will break and never be rebuilt. The critical stretch in a three-dimensional scenario is expressed as follows:

$$s_C = \sqrt{\frac{G_{0C}}{3G + \left(\frac{3}{4}\right)^4 (K - \frac{5G}{3})\delta}} \tag{10}$$

where G_{0C} represents the critical energy release rate, and δ is the horizon size.

2.3 Machine learning

This study utilizes a range of ML methods, encompassing SVMs, ANNs or fully connected neural networks, CNNs, and a probabilistic ML model grounded in MLE.

2.3.1 Support vector machines

In shallow ML, SVMs represent a class of algorithms applicable to regression [53], classification [54], and outlier detection tasks. In SVM nonlinear regression, the predicted output prediction can be written as follows:

$$\hat{y}(\mathbf{x}) = \sum_{J=1}^{N} (\alpha_J - \alpha_J^*) K(\mathbf{x_J}, \mathbf{x}) + b \tag{11}$$

where N denotes the total number of data samples in the training set, α represents the Lagrange multipliers used to reformulate the SVM optimization problem into its dual form, and b stands for the bias term.

A fundamental aspect of SVMs is their utilization of a kernel function, denoted as $K(\mathbf{x_J}, \mathbf{x})$ in Eq. (11), to measure the similarity or proximity between two data samples in a transformed feature space. One widely adopted kernel function is the radial basis function (RBF) [55] as below, which offers advantages similar to those of the K-nearest neighbor algorithm in overcoming space complexity issues.

$$K(\mathbf{x_J}, \mathbf{x}) = e^{\gamma ||\mathbf{x_J} - \mathbf{x}||^2} \tag{12}$$

where γ represents a hyperparameter inversely proportional to the radius of influence of samples selected by the model. It is crucial for controlling the behavior of the kernel function. Another essential hyperparameter for SVMs is denoted as C, which controls the regularization strength to mitigate model

overfitting. Proper tuning of both γ and C is important to strike the optimal bias-variance balance for SVMs.

In classification, the SVM algorithm seeks to identify a hyperplane that effectively separates the data points into different classes. It defines a margin to describe the distance between the hyperplane and the data points, with those falling within the margin boundaries termed as support vectors. The objective of the SVM classification algorithm is to maximize the margin. Additionally, hyperparameter C plays a pivotal role in determining the trade-off between maximizing the margin and minimizing classification errors.

2.3.2 Neural networks

In DL, a typical ANN [56] consists of several layers with numerous neurons, including the input layer, one or more hidden layers, and the output layer. The number of neurons in the input layers depends on the number of input features, while the output layer possesses the same number of neurons as the dimensions of the output variable. Every neuron in a hidden layer connects to all the neurons on the preceding and succeeding layers, rendering the ANN a fully connected neural network.

Let's consider a training set comprising of N samples with p input features and q output variables, denoted as $\mathbf{x}_I \in R^p$ and $\mathbf{y}_I \in R^q$ where $I = 1...N$. A single training step encompasses two key processes: a feedforward pass for prediction and a backpropagation process for updating network weights. For instance, if we consider an ANN with a single hidden layer containing L neurons, the feedforward process involves projecting the original input data through each neuron into a different input feature space as

$$h_j(\mathbf{x}_I) = \phi_j(\mathbf{w}_j^T \mathbf{x}_I) + b_j \quad j = 1...L \quad I = 1...N \tag{13}$$

where $\mathbf{w}$ represents network weights, b denotes the bias, and ϕ_j is the transformation function or activation function applied to the weighted summation of input features at each neuron. Commonly used activation functions include the hyperbolic tangent, rectified linear unit (ReLU) [57], and RBF [55], among others.

Following the feedforward process, the output can be predicted as $\hat{y}_I = \mathbf{v}^T \mathbf{h}(\mathbf{x}_I)$ where $\mathbf{v}$ are network weights associates with the neurons in the output layer. For regression tasks, activation functions on the neurons in the output layer are unnecessary. However, sigmoid and softmax functions are commonly used activation functions in the output layer for binary and multiclass classification tasks, respectively. Once predictions are obtained, the loss function can be calculated, and subsequently, network weights are updated via the gradient descent method or its variations during the backpropagation process. This training step is repeated iteratively until the loss function is minimized.

CNNs [58] are a class of DL algorithms primarily designed for processing images. In contrast to conventional ANNs, which treat every pixel of images as an independent input feature, CNNs leverage convolution computations to extract spatial hierarchies and local patterns from images, aiding the network in feature recognition.

Figure 1 illustrates the typical network structure of a CNN, which consists of input layers, convolutional layers, pooling layers, and fully-connected layers. Generally, a CNN takes images as two-dimensional or three-dimensional tensors and executes four steps during the feedforward prediction process: convolution, nonlinearity, max pooling, and prediction. The convolution operation uses a sliding window approach to break down the image into many overlapped small image tiles. Each tile then undergoes processing by a small neural network, applying nonlinear active functions to generate an output feature map. Subsequently, the max pooling or downsampling operation selects windows from the feature map and outputs the maximum value of each filter. These operations of convolution, nonlinearity, and max pooling can be repeated several times before reshaping the output feature map into a one-dimensional array. This array serves as the input to the fully connected layers for final prediction, as depicted in Fig. 1.

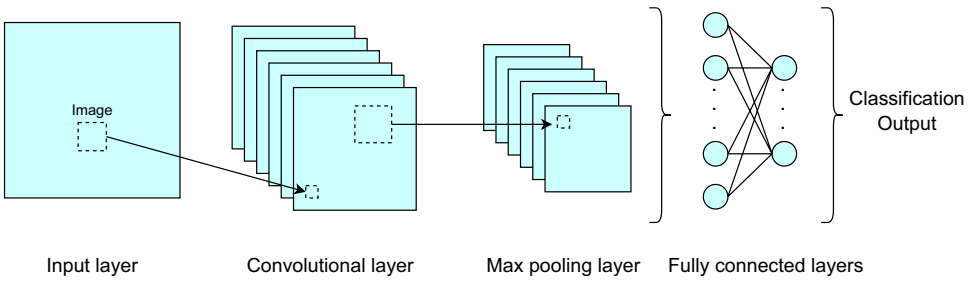

Fig. 1 Architecture of a CNN for classification

2.3.3 Probabilistic machine learning

Part of this work developed and tested several new AL strategies specifically for probabilistic ML models designed for regression tasks. We employed the maximum likelihood (MaxLike) method [59] to train a probabilistic model. Although we used ANN as the model architecture, it's worth noting that the output layer did not provide single-point prediction; instead, it offered a probability distribution encompassing all possible output targets for a given data sample.

In a non-Bayesian-based probabilistic ML approach, the MaxLike estimation serves as a common method to evaluate whether the predicted probability distribution effectively explains the training data without any prior knowledge. Consequently, the model training transforms into a probability density estimation problem aimed at identifying the proper model parameters, denoted as $\boldsymbol{\theta}$, for optimal model performance. Given a training set (x_i, y_i) where $i = 1...n$, the objective of training a probabilistic ML model is to maximize the joint probability of all observations, i.e., data samples, within the training set. Assuming the data samples are independent and identically distributed, this joint probability can be formulated as a likelihood function, represented as a product of conditional probabilities in Eqn (14).

$$L(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = P(y_1, y_2, ..., y_n|x_1, x_2, ..., x_n;\boldsymbol{\theta}) = \prod_{i=1}^{n} P(y_i|x_i;\boldsymbol{\theta}) \tag{14}$$

This study assumes the Gaussian distribution, also known as the normal distribution, denoted as $y \sim \mathcal{N}(\mu, \sigma^2)$. The probability density function of the Gaussian distribution can be expressed as

$$P(y_i|x_i;\mu_{x_i}, \sigma_{x_i}) = \frac{1}{\sigma_{x_i}\sqrt{2\pi}} exp(-\frac{(y_i - \mu_{x_i})^2}{2\sigma_{x_i}^2}) \tag{15}$$

where μ represents the mean (or expectation) of the distribution, and σ denotes the standard deviation. It is important to note that in the scenario of a single output variable, our probability ANN comprises two neurons in the output layer: one neuron represents the mean, while the other represents the standard deviation. As a result, the model predicts a Gaussian distribution that the output target follows for a given input data.

Since the conditional probability of each data sample is often small, the product of numerous small probabilities may lead to numerical instability. Hence, it is a common practice to work with the logarithm of the likelihood function defined in Eq. (14). Consequently, the task of training a probabilistic ML model can be reformulated by minimizing the Negative Log-Likelihood (NLL), depicted in Eq. (16), which serves as the cost function.

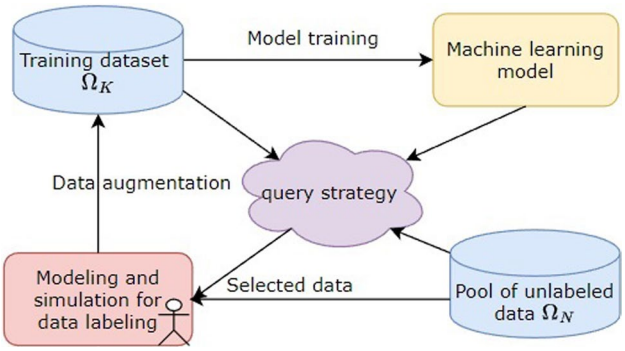

Fig. 2 Pool-based active learning process

$$NLL = -\sum_{i=1}^{n} logP(y_i|x_i;\mu_{x_i}, \sigma_{x_i}) \tag{16}$$

3 Active learning

In the realm of supervised ML, the quantity and quality of labeled data samples play a crucial role in enhancing model performance. However, labeling can be a prohibitively expensive and time-consuming task in materials science, especially when intensive computations are required to obtain a single data sample. AL, as one of the ML techniques, proves invaluable in strategically selecting training data samples for labeling. This approach enables the ML model to achieve optimal performance without redundant data samples, making the time required for data labeling and the overall training feasible and affordable.

As outlined in the introduction, AL strategies are commonly classified into pool-based, stream-based, and membership query synthesis-based methodologies. This paper specifically focuses on pool-based methodologies, where training samples are strategically chosen for labeling from a predefined pool of unlabeled data. In a typical approach [60], a pool Ω_N of N samples is initially generated without labels (outputs). A small subset Ω_K of K samples can be chosen either randomly or through a well-defined strategy. After labeling these K samples through simulations, this subset becomes the initial training set for a base ML model. Subsequently, the base model predicts the outputs for the remaining $N - K$ samples. Taking into account both the input features and predicted outputs, the most informative E samples are then selected and labeled. These newly labeled samples are incorporated into the original training set, and the model undergoes retraining. This process, as illustrated in Fig. 2, is iterated until the model meets the established performance criterion.

3.1 Active learning in classification

The essence of AL lies in its query strategy, a decisive factor in determining which data samples shall be labeled. Various query strategies for classification tasks within pool-based AL have been developed including uncertainty sampling, density-based approaches, and diversity-based approaches, among others [17]. In this study, we opt for uncertainty sampling [40] which is one of the most commonly used strategies due to its simplicity and effectiveness. It is fast and relatively easy to implement, making it a popular choice for many practical applications. Uncertainty sampling prioritizes training the model with data samples near decision boundaries. Samples located far from decision boundaries are considered to contribute less to model training. Specifically, in the context of a binary classification task, we refrain from relying solely on definitive predictions. Instead, we leverage the probability of the predictive outcome as an indicator of the uncertainty or confidence in prediction. During each iteration, the model from the previous step predicts the probabilities of all unlabeled samples. A subset of data samples with prediction probabilities around 0.5, denoting maximum uncertainty, is then carefully chosen. These samples are labeled and subsequently incorporated into the training set for the next iteration. Such a process is summarized in Algorithm 1.

3.2 Active learning in regression

In regression problems, AL also offers various query strategies, including Query By Committee (QBC) [25], Expected Model Change Maximization (EMCM) [26], and others. In our study, we adopt the Greedy Sampling (GS) approach and its variations. In reference [39], Wu et al. evaluated the performance of the GS approach and its variations against other popular pool-based AL strategies for regression QBC and EMCM. The findings indicated that GS and its variations outperformed QBC and EMCM. Notably, Yu and Kim [28] categorized the GS approach as "passive learning," distinct from typical AL methodologies. This technique selects the data samples based solely on the distribution of locations in the input feature space without taking into account the prediction results by the trained model. To be more specific, the distance between an unlabeled data sample $\boldsymbol{x}_n$ in the remaining pool (Ω_{N-K}) and a data sample $\boldsymbol{x}_m$ in the previous training set Ω_K with K labeled data samples is calculated using the following equation.

$$d_{nm} = \|\mathbf{x}_n - \mathbf{x}_m\|, \quad \mathbf{x}_n \in \Omega_{N-K}, \quad \mathbf{x}_m \in \Omega_K \tag{17}$$

The minimum distance from $\boldsymbol{x}_n$ to the data samples in Ω_K, expressed as $d_n = \min_{\boldsymbol{x}_m} d_{nm}(\boldsymbol{x}_n, \boldsymbol{x}_m)$, will be taken as the priority of this unlabeled data sample to be chosen.

Algorithm 1 Active learning for binary classification using uncertainty sampling

Require: Unlabeled data pool $\Omega_N = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, Initial labeled set size K, Expansion size E per iteration, Total iterations T
Ensure: Trained classification model $f(\cdot)$
Initialize:
  Select K samples from Ω_N to create the initial labeled training set Ω_K, either randomly or through a well-defined strategy.
  Remove Ω_K from Ω_N.
for $t = 1$ **to** T **do**
  Train the classification model $f(\cdot)$ using the current Ω_K.
  for each $\mathbf{x}_n$ in Ω_{N-K} **do**
    Predict the probability of the true class $p(\mathbf{x}_n)$ using $f(\cdot)$.
  end for
  Identify E samples with $p(\mathbf{x}_n)$ closest to 0.5, indicating maximum uncertainty.
  Label these E samples and add them to Ω_K.
  Remove these E labeled samples from Ω_{N-K}.
end for
return $f(\cdot)$

Algorithm 2 Unified active learning for regression using GS, GSO, and IGS

Require: Unlabeled data pool $\Omega_N = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, Initial labeled set size K, Expansion size E per iteration, Total iterations T, Active learning strategy $\mathcal{S} \in \{\text{GS, GSO, IGS}\}$
Ensure: Trained model $f(\cdot)$
Initialize:
  Select K samples from Ω_N to create the initial labeled training set Ω_K, either randomly or through a well-defined strategy.
  Remove Ω_K from Ω_N.
for $t = 1$ **to** T **do**
  Train the model $f(\cdot)$ using the current training set Ω_K.
  for each $\mathbf{x}_n$ in Ω_{N-K} **do**
    if $\mathcal{S} =$ GS **then**
      Calculate d_{nm} in Eqn (17) for all $\mathbf{x}_m$ in Ω_K.
    else if $\mathcal{S} =$ GSO **then**
      Predict output for $\mathbf{x}_n$ using $f(\cdot)$ to get $f(\mathbf{x}_n)$.
      Calculate d_{nm} in Eqn (18) for all $\mathbf{x}_m$ in Ω_K.
    else if $\mathcal{S} =$ IGS **then**
      Predict output for $\mathbf{x}_n$ using $f(\cdot)$ to get $f(\mathbf{x}_n)$.
      Calculate d_{nm} in Eqn (19) for all $\mathbf{x}_m$ in Ω_K.
    end if
    Compute $d_n = \min_{\mathbf{x}_m} d_{nm}$.
  end for
  Select E samples from Ω_{N-K} with the highest d_n values.
  Label these E samples and add them to Ω_K.
  Remove these E labeled samples from Ω_{N-K}.
end for
return $f(\cdot)$

Building upon the GS approach, Wu et al. [39] introduced two novel AL approaches: Greedy Sampling on the Output (GSO) and Improved Greedy Sampling (IGS) on both input and output. In the GSO approach, the ML model $f(\boldsymbol{x})$, trained with the current set of labeled data, predicts the outcomes of unlabeled data samples in the remaining pool. Unlike the distance calculation in the GS approach, represented by Eqn (17), the GSO approach bases its distance calculation on the output rather than the input features, as follows.

$$d_{nm} = |f(\mathbf{x}_n) - y_m| \tag{18}$$

where y_m is the actual output of the labeled data sample $\boldsymbol{x}_m$ in the current training set, $\boldsymbol{\Omega}_K$.

Moreover, the IGS approach incorporates both the input features and the output prediction in its distance calculation, aiming to leverage the strengths of both the GS and GSO approaches. The distance calculation is expressed below.

$$d_{nm} = \|\mathbf{x}_n - \mathbf{x}_m\| \cdot |f(\mathbf{x}_n) - y_m| \tag{19}$$

The detailed algorithm is outlined in Algorithm 2, including three different query strategies in AL for regression.

3.3 Active learning in probabilistic machine learning

In ML, data is seldom perfect, often tainted by noise stemming from natural variations or errors. Therefore, probabilistic ML techniques are crucial for effectively managing these uncertainties. Unlike conventional regression models, which yield a single deterministic value, probabilistic models offer outputs in the form of probability distributions.

In the previous discussion, three AL strategies - GS, GSO, and IGS - were described for ML regression models with deterministic predictions. While the GS strategy relies solely on the input features to select unlabeled data samples and can thus be directly applied to probabilistic ML models, the GSO and IGS strategies require revision due to the probabilistic nature of model predictions. To address this, we propose integrating the concept of Kullback-Leibler (KL) divergence [61] into these two AL strategies.

Originating from information theory, the KL divergence, also known as relative entropy, serves as a statistical measure to quantify the difference between a probability distribution and a reference probability distribution. Denoting the reference probability distribution of a random variable x as Q, with the density function $q(x)$, and considering another probability distribution P with its corresponding density function $p(x)$, the following equation calculates the KL divergence, elucidating the variance in the information conveyed by these distributions.

$$KL(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx \tag{20}$$

While the KL divergence is inherently asymmetric, its integration into AL for probabilistic models is intuitive. After training the ML model, $f(\mathbf{x})$, based on the current training set, the predicted probability distribution $f(\mathbf{x}_m)$ for a labeled data sample $\mathbf{x}_m$ serves as the reference distribution. Consequently, the distance between any unlabeled sample $\mathbf{x}_n$ and the labeled sample on output can be evaluated using the KL divergence. The adaptation leads to the revision of Eqn (18) as presented below, and the resulting strategy is termed GSO-KL.

$$d_{nm} = KL\big(f(\mathbf{x}_n) \parallel f(\mathbf{x}_m)\big) \tag{21}$$

Likewise, Eq. (19) can be adjusted as follows, and the resulting strategy is denoted as IGS-KL.

$$d_{nm} = \|\mathbf{x}_n - \mathbf{x}_m\| \cdot KL\big(f(\mathbf{x}_n) \parallel f(\mathbf{x}_m)\big) \tag{22}$$

Apart from these modifications, the remainder of the methodology closely follows the AL strategies previously employed in regression. Additional insights are provided in Algorithm 3.

Algorithm 3 Unified active learning for probabilistic ML using GS, GSO-KL, and IGS-KL

Require: Unlabeled data pool $\Omega_N = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, Initial labeled set size K, Expansion size E per iteration, Total iterations T, Active learning strategy $\mathcal{S} \in \{\text{GS, GSO-KL, IGS-KL}\}$
Ensure: Trained model $f(\cdot)$

Initialize:
  Select K samples from Ω_N to create the initial labeled training set Ω_K, either randomly or through a well-defined strategy.
  Remove Ω_K from Ω_N.
for $t = 1$ **to** T **do**
  Train the model $f(\cdot)$ using the current training set Ω_K.
  for each $\mathbf{x}_n$ in Ω_{N-K} **do**
    if $\mathcal{S} =$ GS **then**
      Calculate d_{nm} in Eqn (17) for all $\mathbf{x}_m$ in Ω_K.
    else if $\mathcal{S} =$ GSO-KL **then**
      Predict the distribution for $\mathbf{x}_n$ using $f(\cdot)$ to get $f(\mathbf{x}_n)$.
      Predict the distribution for $\mathbf{x}_m$ using $f(\cdot)$ to get $f(\mathbf{x}_m)$.
      Calculate d_{nm} in Eqn (21) for all $\mathbf{x}_m$ in Ω_K.
    else if $\mathcal{S} =$ IGS-KL **then**
      Predict the distribution for $\mathbf{x}_n$ using $f(\cdot)$ to get $f(\mathbf{x}_n)$.
      Predict the distribution for $\mathbf{x}_m$ using $f(\cdot)$ to get $f(\mathbf{x}_m)$.
      Calculate d_{nm} in Eqn (22) for all $\mathbf{x}_m$ in Ω_K.
    end if
    Compute $d_n = \min_{\mathbf{x}_m} d_{nm}$.
  end for
  Select E samples from Ω_{N-K} with the highest d_n values.
  Label these E samples and add them to Ω_K.
  Remove these E labeled samples from Ω_{N-K}.
end for
return $f(\cdot)$

4 Examples and discussions

This study explored two examples of computational modeling applied to materials science. In the first example, we employed MD simulations to analyze the mechanical behavior of a one-dimensional molecular chain. These simulations enabled us to investigate how the chain responds to different temperatures and deformation gradients. Specifically, we utilized SVMs to forecast material failure and stresses under varying conditions. Furthermore, we implemented a probability ML model to predict the probability distribution of strength. In the second example, we conducted peridynamics simulations to study metal-ceramic composites. This involved simulating the behavior of composites with varying ceramic volume fractions. To predict the tensile strength and failure of these composites, we employed CNNs.

4.1 One-dimensional molecule chain

The first example investigated the mechanical behaviors of a one-dimensional molecule chain, we referred to the configuration used in our previous study [8]. The chain consisted of 1000 atoms, each with a mass of 1.993×10^{-26} kg. A periodic boundary condition was applied. Assuming the interatomic interaction (i.e., bond) existed only between two nearest atoms, we used the classical Lennard–Jones (LJ) potential to approximate bond stretch and compression between atoms r_i and r_j as follows.

$$U(r_{ij}) = 4\varepsilon \left[\frac{1}{4} \left(\frac{r_0}{r_{ij}} \right)^{12} - \frac{1}{2} \left(\frac{r_0}{r_{ij}} \right)^{6} \right] \tag{23}$$

where $r_{ij} = |r_i - r_j|$ is the deformed bond length, $\varepsilon = 1.65 \times 10^{-18}$ J describes the potential energy well's depth, and $r_0 = 1$ nm represents the undeformed bond length.

In this example, each MD simulation was conducted for 10,000 steps, with a time step of 1 fs, at a given temperature and gradient of deformation until the simulated system reached thermodynamic equilibrium. We employed the canonical ensemble, in which the number of molecules, the volume, and the temperature were maintained constants. The overall stress was calculated and averaged every 100 time steps using Eq. (7), where $f_{ij} = \frac{\partial U(r_{ij})}{\partial r_{ij}}$. Each simulation generated one data sample with outputs of material failure status and stress (if no failure occurred) as labels for classification and regression tasks, respectively.

4.1.1 Material failure classification

A classification model has been developed to predict the failure status of this one-dimensional molecule chain. The model forecasted a binary outcome, with 0 representing non-failure and 1 indicating failure. The input features included a temperature ranging from 50K to 3000K and a deformation gradient spanning from 1.0 to 1.1. A dataset comprising 3000 data samples, evenly distributed across the input feature space, was generated. A test set was created to assess the model's performance by randomly selecting 30 data samples and labeling them. Consequently, the initial pool for AL contained 2970 unlabeled data samples.

We utilized a nonlinear SVM with RBF kernels to address the classification problem of predicting material failure. The training phase involved fine-tuning two crucial hyperparameters: C, which balances the trade-off between the model complexity and the tolerance for deviations from the margin, and γ, which defines the influence of a single training example. This tuning was accomplished through cross-validation. Specifically, we employed a grid search approach, systematically exploring a range of values for both C and γ to find the combination that resulted in the best performance on our validation dataset. The evaluation metric for assessing the model performance was accuracy, representing the proportion of correctly predicted labels.

In this example, we employed uncertainty sampling as the AL approach, starting with randomly selected five data samples (i.e., $K = 5$) from the pool. These chosen data samples were removed from the pool and labeled through MD simulations, forming the initial training set denoted as Ω_K. Following training, the model could provide probability estimates indicating the likelihood that a new data sample belonged to each class. This information served as a measure of the model's uncertainty in its predictions, a valuable aspect for implementing uncertainty sampling strategies in AL for classification tasks.

Specifically, in this binary classification problem, once the initial model was obtained, we utilized it to predict the class probabilities for all unlabeled samples in the remaining pool. Next, we selected a data sample (i.e., $E = 1$) from the pool whose prediction probability was closest to 0.5. This sample was labeled and added to the training set. The updated training set was used to retrain the classification model for subsequent predictions. This iterative process continued until the model accuracy converged. Remarkably, we observed that a total of only 37 data samples were needed to achieve a classification model with 100% accuracy. It is noteworthy that our previous study [8] utilized 861 data samples.

Figure 3 illustrates how the decision boundary evolved with the size of the training set. Notably, data samples selected using uncertainty sampling tended to cluster near the true decision boundary, resulting in more efficient training. Additionally, we compared the performances of uncertainty sampling and random sampling, which served as the baseline, in Table 1. The random sampling method followed

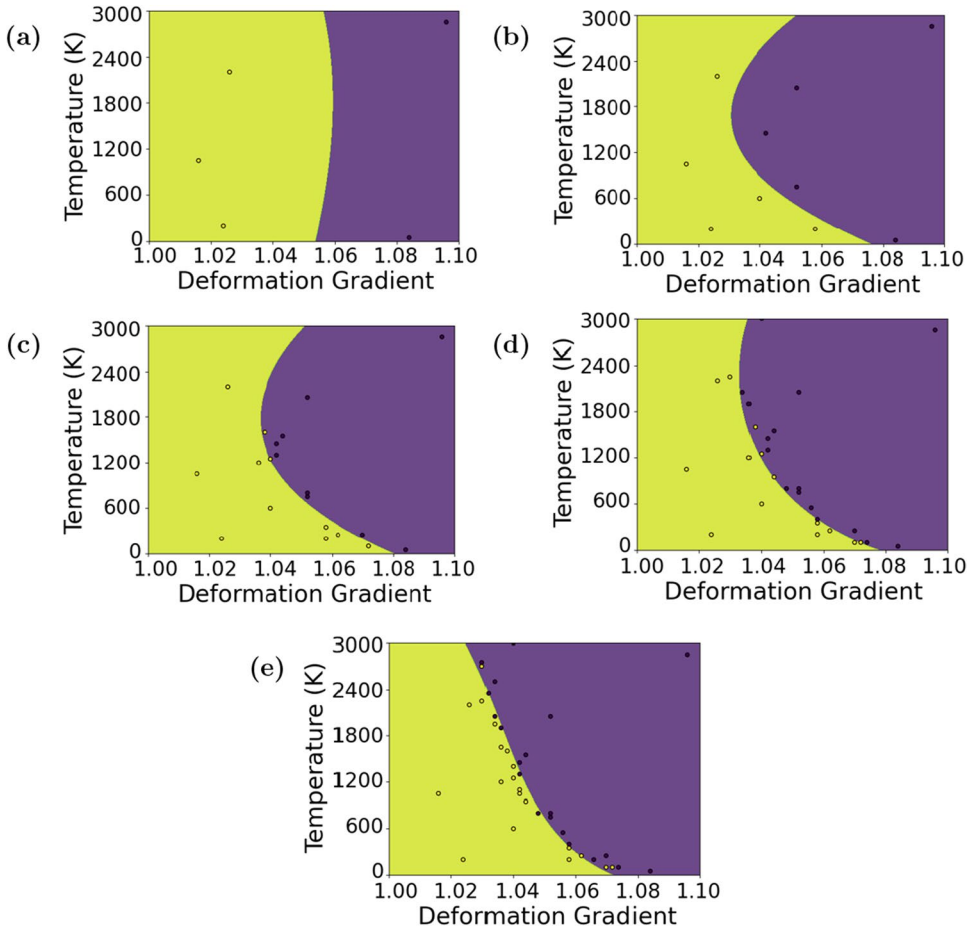

Fig. 3 The evolution of decision boundary through active learning with various numbers of training samples: **a** 5, **b** 10, **c** 20, **d** 30, and **e** 37

Table 1 The numbers of training samples needed to reach certain accuracies

Accuracy	70%	80%	90%	95%	100%
AL with uncertainty sampling	5	7	15	24	37
ML with random sampling	5	12	15	80	192

a similar iterative process to uncertainty sampling but selected data samples for labeling randomly at each iteration. It's evident from the comparison that uncertainty sampling outperformed random sampling. Specifically, uncertainty sampling achieved 100% accuracy with much fewer data samples compared to random sampling, which requires five times as many samples to reach the same level of accuracy.

4.1.2 Stress prediction

To address the stress prediction problem associated with the one-dimensional molecular chain, we developed a regression ML model. This model utilized the deformation gradient and temperature as inputs, with the output being the atomic-level stress obtained from MD simulations using Eq. (7).

In the development of our model, we leveraged the unlabeled data pool initially defined in Sect. 4.1.1 to form a new set. Notably, we employed the classification model developed in the preceding section to partition the data into subsets of "failure" and "no failure" samples. We designated the subset consisting of "no failure" samples as the new unlabeled pool Ω_N for this specific task, comprising 1296 samples for subsequent utilization in AL. Additionally, we randomly selected 30 samples for labeling to form the testing set, facilitating the evaluation of the model's performance during the training process.

For stress prediction, we utilized an SVM regression model with the RBF kernel. We employed the same grid search approach in Sect. 4.1.1 with cross-validation to fine-tune the hyperparameters C and γ. And we used the R-squared score (or the coefficient of determination, R2) as the metric on the testing set. To explore the potential enhancement of AL in this regression task, we incorporated three distinct approaches: GS, GSO, and IGS, as detailed in Sect. 3.2.

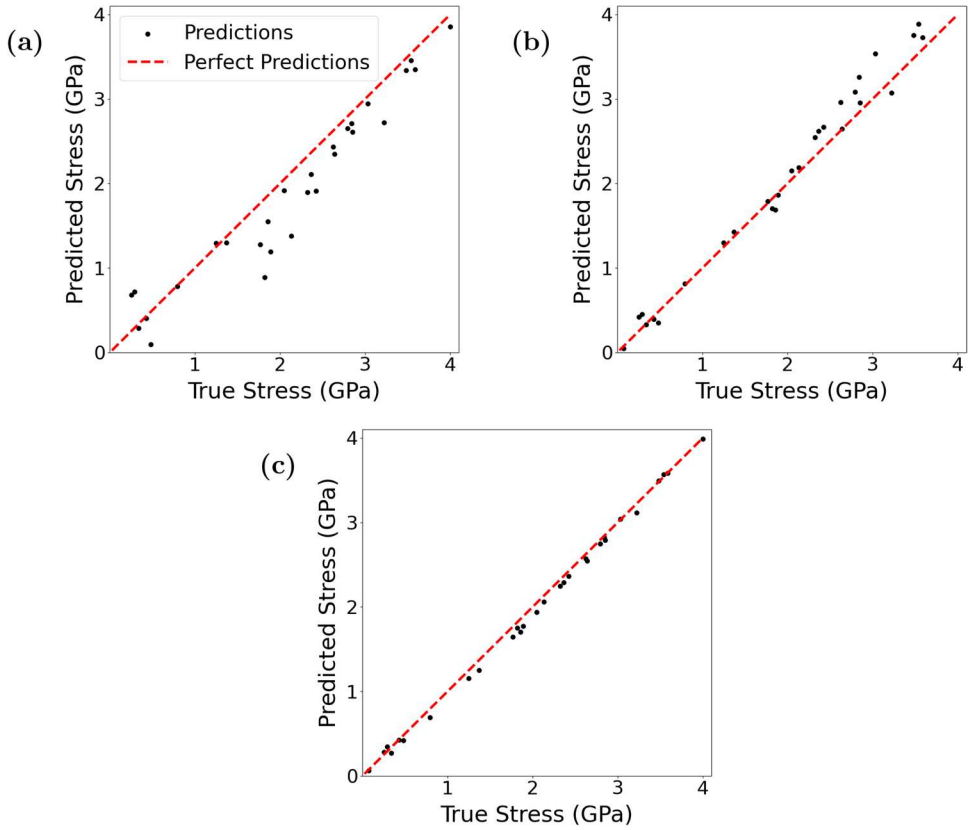

Fig. 4 The evolution of model performance on the testing set using AL with the IGS query strategy. The numbers of data samples in the training set are **a** 5, **b** 6, and **c** 10

We began with the GS method, initially identifying five data samples ($K = 5$) closest to the centroid of the pool Ω_N. These samples were labeled based on MD simulation outcomes, forming an initial training set Ω_K. Subsequently, following the procedure outlined in Algorithm 2, we iteratively selected one data sample ($E = 1$) with the largest d_n from the remaining pool of unlabeled data samples Ω_{N-K}, labeled it, and incorporated it into the training set Ω_K to update the ML model at each iteration.

Moreover, we implemented the GSO and IGS approaches. While GS served as the initial step for sample selection, GSO and IGS diverged in their selection strategies after forming the initial training set with five labeled data samples. GSO relied solely on the model's output, whereas IGS considered both input features and model predictions. Based on the steps detailed in the Algorithm 2, we continued to select, label, and integrate additional samples into Ω_K, expanding the training dataset to update the ML model until achieving the desired performance.

Using the IGS approach as an example, Fig. 4 demonstrates how the model's performance on the testing set improved with an increasing number of training data samples. The diagonal line in the figure represents perfect predictions. Notably, the regression ML model achieved an impressive R2 score of up to 0.995 with only ten training samples, showcasing remarkable efficiency. This performance stands out when compared to our previous study [8], where a similar model achieved comparable performance using 436 labeled data samples. Table 2 illustrates the

Table 2 The numbers of training samples needed to achieve specific R2 scores

R2 score	0.900	0.950	0.970	0.990	0.995
AL with GS	5	6	7	9	14
AL with GSO	5	8	9	13	14
AL with IGS	5	6	7	9	10
ML with random sampling	7	10	12	21	31
AL with IGS (training data with 1% noise)	5	6	7	10	11
AL with IGS (training data with 2% noise)	6	7	9	11	12
AL with IGS (training data with 5% noise)	5	6	9	16	17

number of training samples required for different AL approaches to achieve specific R2 scores. Importantly, all of these approaches surpassed the baseline approach, which randomly selected data samples from the pool of unlabeled data. Furthermore, the IGS approach outperformed the other two AL approaches because it considered the influences of both input features and output targets on sample selections.

Furthermore, to assess the impact of noise on our AL strategies, we conducted an experiment using the IGS approach with an additional variation in the labeling process. We maintained the same procedures as the standard IGS approach but introduced artificial noise at varying levels (1%, 2%, and 5%) to the stress values obtained from MD simulations during the labeling step for the training dataset and the test dataset remained unchanged to ensure consistent evaluation. Table 2 presents the comparative results, showing that for the IGS AL strategy, the 1% and 2% noise has little impact on performance. However, at the 5% noise level, there is a significant increase in the number of samples required to achieve similar R2 scores, especially at the high R2 score stages. These findings suggest that while IGS demonstrates a degree of resistance to noise, higher noise levels moderately diminish its effectiveness. Notably, even with noise introduced, the IGS method consistently outperforms the noise-free random sampling approach in achieving high R2 scores with fewer samples.

4.1.3 Strength distribution prediction

To incorporate uncertainty, we introduced randomness into the LJ potential (Eq. (23)). We assumed Gaussian distributions for the potential energy well's depth ϵ and the initial (i.e., undeformed) bond length r_0, with mean values of 1.65×10^{-18} J and 1 nm, respectively. The standard deviations $\mathrm{STD}_{\mathrm{scaled}}$ varied with temperature T according to the following relationship:

$$\mathrm{STD}_{\mathrm{scaled}} = \mathrm{STD}_{\mathrm{base}} \times \left(\sin(4\pi \times T_n) + \mathrm{B}\right) \tag{24}$$

where $\mathrm{STD}_{\mathrm{base}}$ represented the base values of the standard deviations for ϵ and r_0, which were 0.01×10^{-18} J and 0.02 nm, respectively. We set $B = 1.5$ to ensure that $\mathrm{STD}_{\mathrm{scaled}}$ remained positive. The normalized temperature T_n was calculated as $T_n = (T - T_0)/(T_1 - T_0)$, where $T_0 = 50K$ and $T_1 = 3000K$ represented the range of temperatures considered in this study.

We utilized the MaxLike method with an ANN framework, as elaborated in Sect. 2.3.3, to construct a probabilistic ML model for forecasting the distributions of the Ultimate Tensile Strength (UTS) across various temperatures. The ANN architecture comprised three hidden layers and one output layer. The first hidden layer contained 20 neurons, followed by a layer with 50 neurons, and finally, a layer with 20 neurons. Each neuron in these layers employed a sigmoid activation function. The output layer of the model consisted of two neurons responsible for predicting the mean and variance of the strength distribution, respectively. The loss function of the ANN, derived from Eqs. (15) and 16), is expressed as follows. It should be noted that this neural network lacks a dedicated validation set, therefore, hyperparameters were tuned based on the loss observed on the training set.

$$LOSS = \sum_{i=1}^{n} -\log \frac{1}{\sqrt{2\pi\sigma_{x_i}^2}} + \frac{1}{2\sigma_{x_i}^2}(y_i - \mu_{x_i})^2 \tag{25}$$

In our investigation of the efficacy of AL strategies within the probabilistic ML model, we employed three distinct approaches: GS, GSO-KL, and IGS-KL. The latter two were specifically designed for the probabilistic ML model, as detailed in Sect. 3.3. Initially, our dataset comprised 2951 unlabeled data samples spanning temperatures ranging from $50K$ to $3000K$. To label a single data sample at a specific temperature, we sampled potential parameters ϵ and r_0 from the previously described probability distributions. Subsequently, we conducted an MD simulation to determine the material strength, thereby assigning the label to this particular data sample. Following the guidelines outlined in Algorithm 3, we set $K = 5$ and $E = 1$, facilitating iterative model training, sample selection, labeling, and incorporation of the labeled sample into the training set. Additionally, we included random sampling as a baseline for comparative analysis.

To evaluate the performance of our probabilistic ML model, we introduced a metric termed the "KL score." This score quantifies the disparity between the predicted probability distributions generated by the ML model and the reference distributions obtained from MD simulations. We selected five distinct temperatures—500 K, 1000 K, 1500 K, 2000 K, and 2500 K—for evaluation. At each temperature, we conducted 50 MD simulations, each utilizing a sampled potential function, to determine the strengths. Subsequently, we derived the corresponding normal distributions, denoted as the reference distributions $Q_i \sim \mathcal{N}(\mu_{Q_i}, \sigma_{Q_i}^2)$ where $i = 1...5$. Conversely, the probabilistic ML model predicted

Table 3 The numbers of training samples needed to achieve specific KL scores

KL score	0.20	0.15	0.10	0.05
AL with GS	135	149	215	>400
AL with GSO-KL	48	62	124	147
AL with IGS-KL	38	47	86	169
ML with random sampling	161	175	324	>400

distributions, denoted as $P_i \sim \mathcal{N}(\mu_{P_i}, \sigma^2_{P_i})$, for these five temperatures. Finally, the KL score, calculated as the average of KL divergences in Eq. (26) across the five specified temperatures, provided a quantitative measure of the discrepancy between the predicted probability distributions and the reference distributions.

$$KL_{score} = \frac{1}{5}\sum_{i=1}^{5}\left\{\log\left(\frac{\sigma_{Q_i}}{\sigma_{P_i}}\right) + \frac{\sigma^2_{P_i} + (\mu_{P_i} - \mu_{Q_i})^2}{2\sigma^2_{Q_i}} - \frac{1}{2}\right\} \tag{26}$$

Table 3 illustrates the number of training samples required by the three AL approaches to achieve specific KL scores in comparison to the random sampling strategy. This comparison provides valuable insights into the effectiveness of AL strategies in optimizing the probabilistic ML model. It's evident that the GS approach is not comparable to the other AL approaches, as it doesn't consider outputs when selecting data samples to label. Instead, due to the randomness of the samples, it performs similarly to random sampling, the baseline approach. Given the variability in predicted probability distributions across temperatures, exploration in the output space to label data samples for ML model improvement is crucial. This conclusion is supported by the similar performance improvements between the GSO-KL and IGS-KL approaches. Furthermore, these two new approaches developed in this paper effectively quantify the differences in probability distributions between unlabeled data samples and labeled training samples. Additionally, we leveraged the IGS-KL approach to demonstrate the evolution of ML performance across various training set sizes, as depicted in Fig. 5.

4.2 Three-dimensional metal-ceramic composites

With its unique combination of high hardness and exceptional wear and corrosion resistance, TiB ceramic emerges as one of the most promising candidates for reinforcement within the Ti matrix in Ti-TiB composites [62]. The second example in this study addressed two distinct tasks: predicting the UTS of Ti-TiB composites, categorized as a regression task, and evaluating composite failure under specific deformations, which presents a classification challenge.

4.2.1 Computational model

TiB is renowned for its extraordinary material properties, attributed to its unique single crystal structures [63]. Composites comprising Ti and TiB can be synthesized through

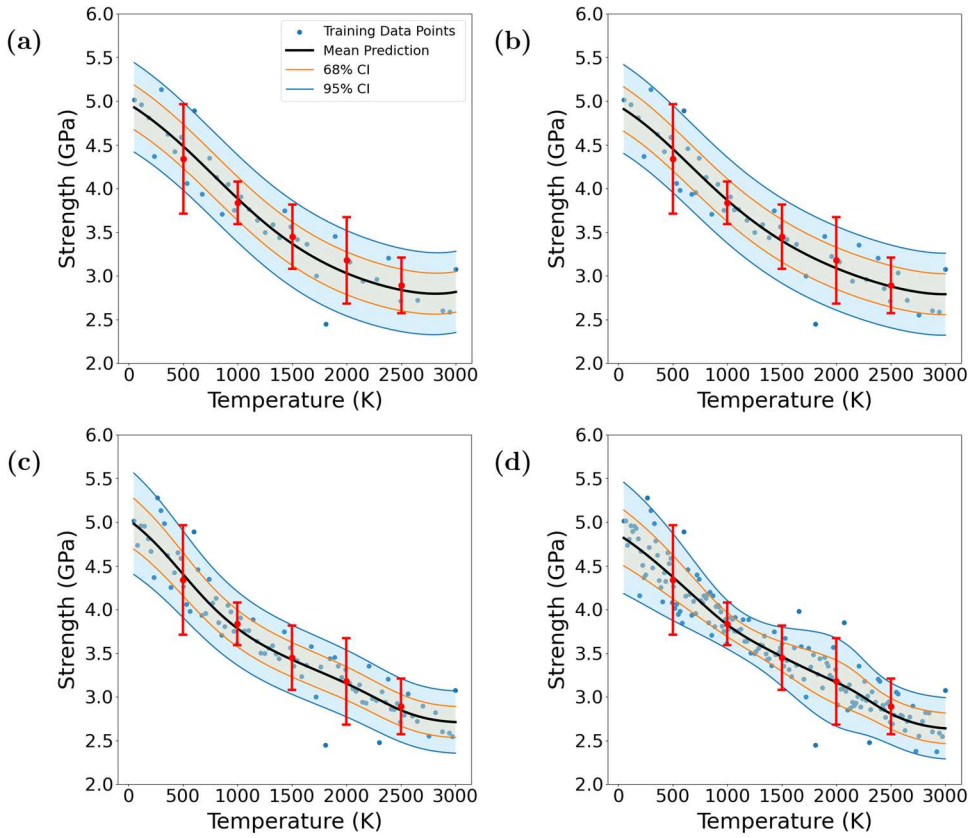

Fig. 5 The evolution of model performance using ALg with the IGS-KL query strategy with **a** 38, **b** 47, **c** 86, and **d** 169 training samples. The red dots, accompanied by error bars at specific temperatures, represent the reference distributions with 95% confidence intervals (CI), which are expected to align with the predicted distributions indicated by the blue lines

Table 4 Material properties of TiB and Ti [66]

Material	Density (Kg/m^3)	Young's modulus (GPa)	Poisson's ratio	UTS (GPa)
TiB	4520	428.4	0.245	28
Ti	4357	106.2	0.298	1.17

the sintering of Ti and TiB2 powders. Within the Ti matrix, TiB has been observed as whiskers or needles, typically oriented along its [010] direction [64]. As TiB is not naturally occurring, we conducted MD simulations using a recently developed potential function [65] to characterize its material properties. These properties, along with Ti's material properties outlined in Table 4, were subsequently utilized in peridynamics for microscale simulations.

The Ceramic Volume Fraction (CVF), which represents the volume percentage of the TiB component relative to the overall composite volume, significantly influences the properties of the composite. In this example, the CVF was considered within the range of 0% to 25% [67]. Moreover, the spatial arrangement of TiB whiskers within the composite is crucial. Microstructures with identical CVF values may exhibit varying properties due to the distribution and alignment of TiB whiskers [68]. Additionally, literature [64] suggested that the diameters of TiB whiskers vary from 1μm to 3μm, with an aspect ratio of 12.2 ± 4.3.

The Peridynamics model utilized numerous discrete, equally spaced material points throughout the simulation domain to handle discontinuous interfaces in composite materials. In this study, the simulation domain was configured as a cubic microscale measuring 42 μm × 42 μm × 42 μm. Each material point, representing either Ti or TiB particles, has a diameter of 2 μm, resulting in a total of 9261 material points within the simulation domain.

To incorporate TiB whiskers into the model, we randomly selected a material point to serve as the whisker's starting point (referred to as the "head"). Subsequently, one of its neighboring points was randomly designated as TiB. Those two points established a straight line, which was extended by selecting corresponding material points until the length fell within the range of 15.8μm and 33 μm. Notably, to prevent intersection, two whiskers were not allowed to intersect in our models. In addition, the number of whiskers generated was determined by the specified CVF. Figure 6 illustrates two different configurations with the same CVF. This 3D model accurately depicts the dispersion of TiB whiskers within the composite material.

In our study, we utilized Peridigm [69, 70], an open-source peridynamics software developed by Sandia National Laboratories, to conduct simulations of Ti-TiB composite materials. To define the failure criterion, we required the critical stretch s_C, which could be derived by the critical energy release rate G_{0C}, as indicated in Eq. (10). Given the limited research available on the properties of bulk TiB material, determining a reliable value for G_{0C} was challenging. In our research, we calibrated the critical stretch s_C in peridynamics simulations of pure TiB to align with the known UTS from MD simulations, thus determining an appropriate s_C. This methodology was consistently applied to the Ti material as well.

4.2.2 Predicting ultimate tensile strength

In this regression task, each microscale configuration represented one unlabeled data sample. When labeling this sample was required, a peridynamics simulation was conducted on this configuration subject to uniaxial tension, and then the UTS was determined from the resulting stress–strain curve. As the input feature, the configuration was converted into a 3D binary tensor with "1" and "0" representing Ti and TiB, respectively. We employed a CNN as the regression ML model to process the 3D binary tensor as input and predict the UTS.

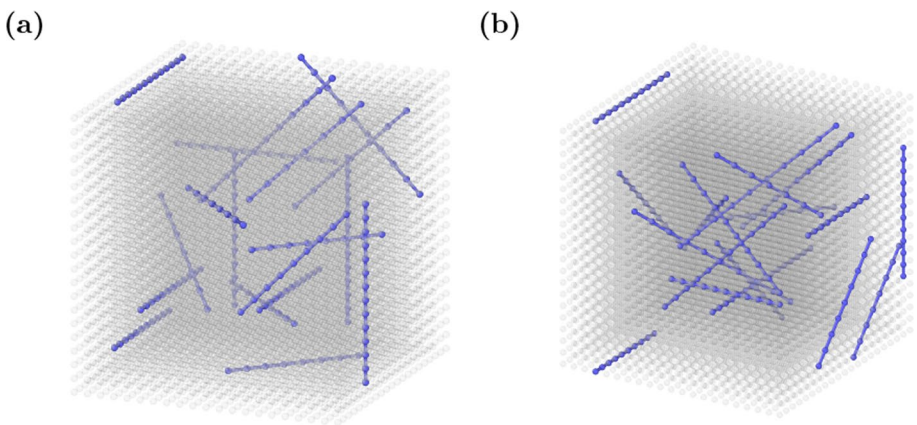

Fig. 6 Two configurations of Ti-TiB composite materials with 2% CVF for Peridynamics simulations. Transparent spheres represent Ti particles, and blue spheres and their connecting lines represent TiB whiskers

The CNN architecture consisted of two convolutional layers, each followed by a max pooling layer. The convolutional layers utilized a 3D kernel sized $3 \times 3 \times 3$, with 16 filters for the first layer and 32 filters for the second layer. After each convolutional operation, max pooling with a $2 \times 2 \times 2$ filter was applied to reduce the spatial dimensions. Dropout layers with a rate of 0.25 and batch normalization were introduced after each max pooling layer to prevent overfitting and promote model generalization. Additionally, the network comprised two dense hidden layers with 32 and 16 neurons, respectively, with dropout layers added after each dense hidden layer. The ReLU activation function was utilized in all layers except for the output layer.

Based on the average length of TiB whiskers observed in experiments [64], a simulation model comprising 235 whiskers represented the corresponding Ti-TiB composite with a 25% CVF. Consequently, a pool of 236 unlabeled samples or configurations was generated for AL to assist the CNN regression model. Each sample in the pool varies in the number of TiB whiskers, ranging from 0 to 235, leading to configurations with diverse CVFs between 0% and 25%. Each microscale configuration was randomly generated based on the specific number of TiB whiskers. Additionally, 55 configurations were randomly generated to form the testing set, with another 55 configurations to the validation set. The validation set is utilized to tune the hyperparameters of the CNN model, including the number and size of layers, dropout rate, learning rate, training epochs, and batch size. These samples were labeled using peridynamics simulations.

In our study on AL for UTS prediction, We implemented three strategies: GS, GSO, and IGS. Following the procedures outlined in Algorithm 2, we set the parameters $K = 5$ and $E = 1$ for all AL approaches. However, we made specific revisions to calculating the distance d_{nm} on input. Recognizing the significant influence of CVF on the material properties of the composite, we incorporated both 3D binary tensors and the corresponding CVFs into sample selection. Consequently, we redefined the distance d_{nm} in Eq. (17) for the GS strategy as follows:

$$d_{nm} = \|\mathbf{x}_n - \mathbf{x}_m\| \cdot |\mathrm{CVF}_n - \mathrm{CVF}_m| \tag{27}$$

Table 5 The numbers of training samples needed to achieve specific R2 scores

R2 score	0.85	0.90	0.95	0.96	0.97
AL with GS	13	15	26	32	47
AL with GSO	14	17	32	55	>80
AL with IGS	12	14	25	32	58
ML with random sampling	27	29	37	48	>80

Similarly, the distance calculation for the IGS strategy was adjusted:

$$d_{nm} = \|\mathbf{x}_n - \mathbf{x}_m\| \cdot |\mathrm{CVF}_n - \mathrm{CVF}_m| \cdot |f(\mathbf{x}_n) - y_m| \tag{28}$$

To highlight the efficacy of AL methodologies, we implemented the random sampling strategy as the baseline. This sampling procedure was divided into two steps during each iteration. Initially, we randomly selected a number of TiB whiskers between 0 and 235, mirroring the arbitrary selection of a CVF. Following this, we generated a microscale configuration incorporating the specified number of whiskers.

Continuing with our approach for regression ML models, we also consider the R2 score as the metric for evaluating model performance. Table 5 illustrates the number of training samples required by different AL approaches and the baseline method to achieve specific R2 scores. Overall, the GS and IGS approaches performed better. This superiority stemmed from their ability to leverage detailed microscale configurations as input features, providing ample information for selecting crucial training samples. Specifically, Fig. 7 demonstrates the enhancement of the ML model assessment on the testing set as the size of the training dataset increases, exemplifying the effectiveness of the GS approach. Due to the data's complexity, the overall model performances did not match those of the one-dimensional example depicted in Fig. 4. Achieving a higher R2 score necessitates a larger training set. However, in both examples, it's apparent that the number of training samples exponentially increased to enhance the ML model performance.

Conversely, GSO's performance exceeded the baseline only until the R2 score reached 0.95. Beyond this point, GSO's effectiveness declined in comparison. This decline could be attributed to the influence of CVF on the UTS of composite materials. A prior study [10] demonstrated that at low CVF, the UTS exhibits minimal variation with changes in CVF; conversely, at higher CVF, the UTS responds more significantly to changes in CVF. Consequently, as the volume of the training set increased, the GSO approach tended to concentrate samples with larger CVFs, thereby limiting the data diversity. As a result, after the R2 score reached 0.95, the performance of GSO lagged behind the baseline. A similar influence affected the IGS approach, which failed to match the performance of the GS approach when a high model accuracy was required.

4.2.3 Material failure classification

This classification task aimed to determine whether a specified composite material failed under a tension-induced deformation gradient. Each data sample consisted of two

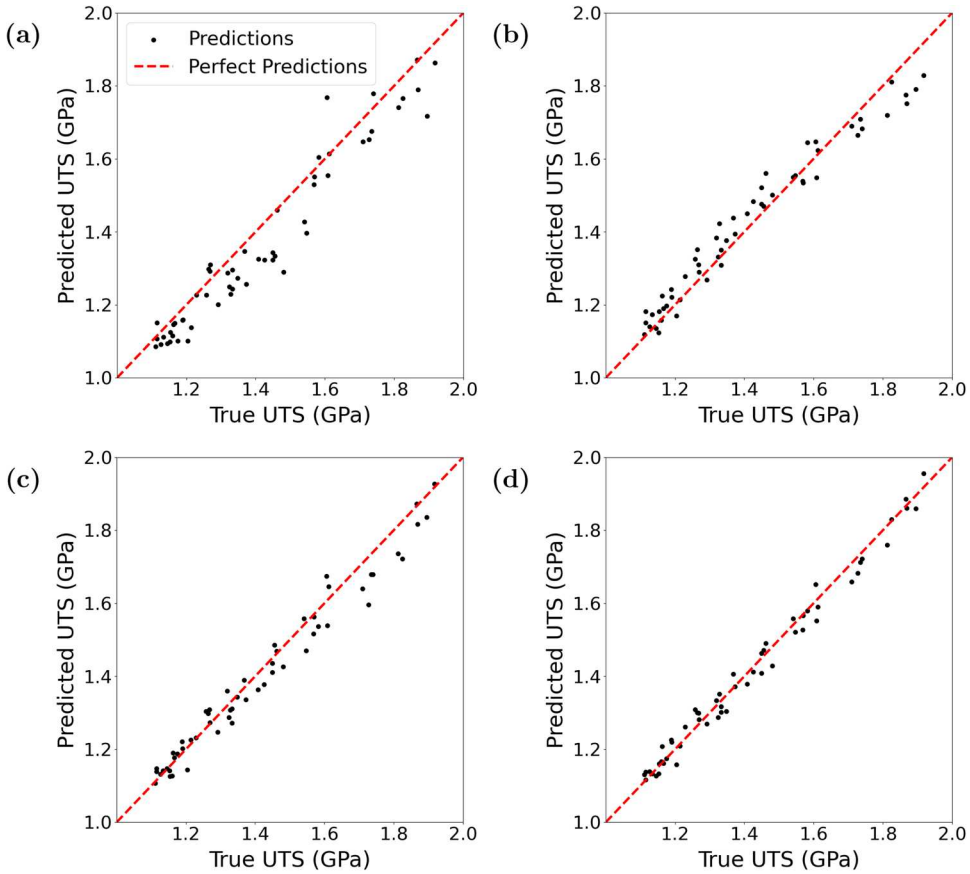

Fig. 7 The evolution of model performance on the testing set using AL with the GS approach. The numbers of data samples in the training set were **a** 15, **b** 26, **c** 32, and **d** 47

inputs: the microscale configuration and the deformation gradient. The microscale configurations were generated following the previously described procedure, while uniaxial deformation gradients were uniformly distributed in the range of 1.006 to 1.016. Consequently, the dataset comprised a total of 4720 unlabeled samples. For testing and validation, we randomly generated 55 samples for each set. To label a data sample, a peridynamics simulation was conducted on the composite. This simulation commenced from its undeformed microscale configuration and continued until the designated deformation gradient was attained. The resulting material failure status determined the sample's label: "1" indicating failure and "0" indicating non-failure.

We employed the similar CNN architecture described in Sect. 4.2.2 to input and process the 3D binary tensor encoded from a microscale configuration. The primary adaptation involved concatenating the deformation gradient with the flattened output before forwarding it through the fully connected layer for prediction. Furthermore, the ML model was compiled using binary cross-entropy as the loss function, with accuracy serving as the metric for evaluating model performance.

To implement AL in model training, we adopted the uncertainty sampling query strategy, as previously discussed. By using the 'sigmoid' activation function in the output layer, the ML model produced a value ranging from 0 to 1, representing the probability of material failure. A value near 0.5 indicated greater uncertainty in the prediction. Given the complexity of this classification task with two input features, ensuring the convergence of model performance necessitated a relatively larger training set compared to the one-dimensional example. Balancing the computational time and the efficiency of the AL process, we set $K = 10$ and $E = 10$ following Algorithm 1. Accordingly, our AL methodology utilized the trained model to select and label 10 new samples during each iteration.

We also assessed the baseline method for comparative purposes, which involved randomly selecting up to 350 samples. The effectiveness of the uncertainty sampling approach, along with the baseline results, is illustrated Fig. 8. This visualization showcases the relationship between the number of training data samples and the model's accuracy. The results unequivocally reveal a significant advantage of the uncertainty sampling approach over the baseline method. Specifically, the AL approach achieved nearly 100 % accuracy with 230 data samples, while the

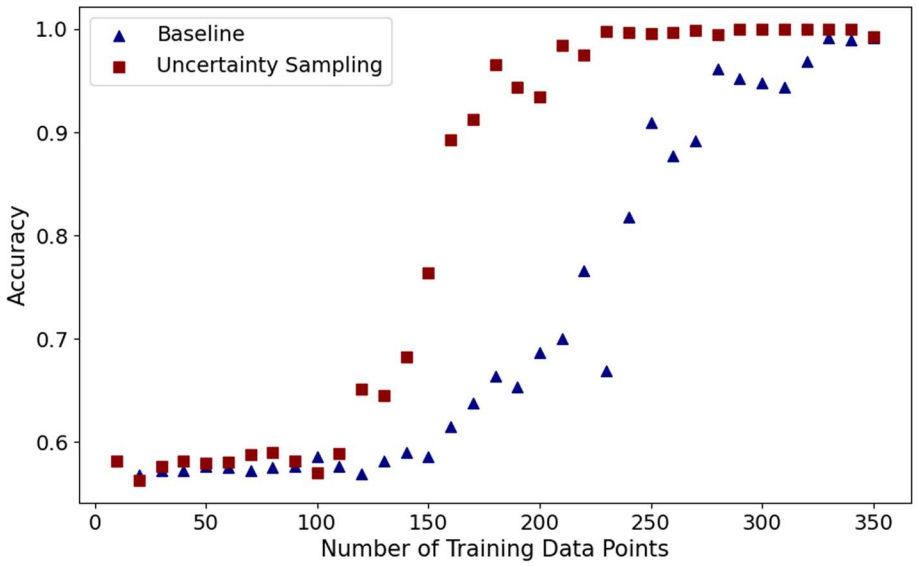

Fig. 8 Comparing the baseline and the active learning approaches in model accuracy with various numbers of training data samples

random sampling required 330 samples. When the number of training data samples was 110 or fewer, the training set became too small for the ML model to learn the underlying relationships effectively. At this early stage, the predictions made by the trained model using the AL approach did not hold much practical significance over random sampling.

5 Conclusion

This study extensively examined the application of AL strategies within computational materials science, particularly focusing on their role in enhancing the efficiency of ML models for predicting material properties and mechanical behaviors. We systematically evaluated the effectiveness of various AL strategies, including GS and its variations, uncertainty sampling, and several novel approaches tailored for probabilistic ML models. These strategies were applied across a spectrum of contexts, ranging from traditional ML tasks to advanced probabilistic models, encompassing one-dimensional molecule chains and three-dimensional metal-ceramic composites in both classification and regression tasks, which successfully demonstrated the application of AL across different complexities. Moreover, our AL strategies effectively adapted to varying dataset sizes, tested in our case studies from as few as 10 samples up to 230 samples.

Overall, our findings underscored the significant potential of AL in boosting ML model performance, often requiring substantially fewer training samples compared to conventional random sampling methods. This reduction in training samples significantly lowers labeling costs. In our study, across five examples, the labeling costs for AL compared to random sampling were 19.27%, 32.26%, 36.75%, 58.75%, and 69.70%, respectively, demonstrating substantial computational cost savings. Although AL typically requires a smaller training set, it involves iterative training sessions rather than the single-session traditional methods often need. In our study, data labeling involves time-consuming MD and peridynamics simulations. As a result, the small size of the training set makes the training time negligible compared to the labeling time, which further highlights the advantage of AL in reducing computational costs, particularly in material science where labeling demands significant computational resources. Conversely, in fields where labeled data is readily available and less costly to obtain, the necessity of AL might not be as pronounced. Specifically, uncertainty sampling for classification tasks demonstrated an intuitive selection of samples near decision boundaries for labeling, effectively bypassing irrelevant samples. However, the efficacy of regression ML models with GS, GSO, and IGS strategies exhibited variability, likely attributed to the diverse influences of input features or output targets on sample selection. Furthermore, our investigation into newly developed AL strategies for probabilistic ML regression models revealed the necessity of considering predicted probabilistic distributions as high entropy corresponding to high information content based on information theory.

Looking forward, future research endeavors should aim to refine these AL methodologies further and extend their applications into multiscale modeling and simulation, facilitating the efficient data-driven transmission of information across diverse scales. Additionally, while our study encompassed shallow ML and DL techniques such as ANN and CNN, there is considerable merit in exploring the applications of AL strategies in Recurrent Neural Networks (RNNs) for sequence-to-sequence predictions.

Acknowledgements The authors gratefully acknowledge the support from the National Science Foundation (#2104383) and the US Department of Education (ED#P116S210005).

Author Contributions Yingbin Chen: Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing—original draft. Phillip Deierling: Conceptualization, Formal analysis, Funding acquisition,Project administration, Writing—review & editing. Shaoping Xiao: Conceptualization, Formal analysis, Funding acquisition, Methodology, Project administration, Resources, Supervision, Writing—original draft, Writing—review and editing.

Data availibility The datasets and code generated during the current study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. S.P. Xiao, J. Li, S.P.A. Bordas, T.Y. Kim, Artificial neural networks and their applications in computational materials science: A review and a case study. Adv. Appl. Mech. 57, 1–33 (2023) https://doi.org/10.1016/bs.aams.2023.09.001
2. Z. Zhang, K. Friedrich, Artificial neural networks applied to polymer composites: a review. Compos. Sci. Technol. **63**(14), 2029–2044 (2003). https://doi.org/10.1016/S0266-3538(03)00106-4
3. H. El Kadi, Y. Al-Assaf, Prediction of the fatigue life of unidirectional glass fiber/epoxy composite laminae using different neural network paradigms. Compos. Struct. **55**(2), 239–246 (2002). https://doi.org/10.1016/S0263-8223(01)00152-0
4. H.N. Bar, M.R. Bhat, C.R.L. Murthy, Identification of failure modes in gfrp using pvdf sensors: Ann approach. Compos. Struct. **65**(2), 231–237 (2004). https://doi.org/10.1016/j.compstruct.2003.10.019
5. Z. Zhang, P. Klein, K. Friedrich, Dynamic mechanical properties of ptfe based short carbon fibre reinforced composites: experiment and artificial neural network prediction. Compos. Sci. Technol. **62**(7–8), 1001–1009 (2002). https://doi.org/10.1016/S0266-3538(02)00036-2
6. D. Heider, M.J. Piovoso, J.W. Gillespie, A neural network model-based open-loop optimization for the automated thermoplastic composite tow-placement system. Compos. Part A Appl. Sci. Manuf. **34**(8), 791–799 (2003). https://doi.org/10.1016/S1359-835X(03)00120-9
7. C.W. Ulmer II., D.A. Smith, B.G. Sumpter, D.I. Noid, Computational neural networks and the rational design of polymeric materials: the next generation polycarbonates. Comput. Theor. Polym. Sci. **8**(3–4), 311–321 (1998). https://doi.org/10.1016/S1089-3156(98)00035-X
8. S.P. Xiao, R. Hu, Z. Li, S. Attarian, K. Bjork, A. Lendasse, A machine-learning-enhanced hierarchical multiscale method for bridging from molecular dynamics to continua. Neural Comput. Appl. **32**(18), 14359–14373 (2020). https://doi.org/10.1007/S00521-019-04480-7
9. S.P. Xiao, P. Deierling, S. Attarian, A. Tuhami, Machine learning in multiscale modeling of spatially tailored materials with microstructure uncertainties. Comput. Struct. **249**, 106511 (2021). https://doi.org/10.1016/j.compstruc.2021.106511
10. A. Tuhami, S.P. Xiao, Multiscale modeling of metal-ceramic spatially tailored materials via gaussian process regression and peridynamics. Int. J. Comput. Methods **19**(10), 2250025 (2022). https://doi.org/10.1142/S0219876222500256
11. J.F. Rodrigues, L. Florea, M.C.F. Oliveira, D. Diamond, O.N. Oliveira, Big data and machine learning for materials science. Discov. Mater. **1**(1), 1–27 (2021). https://doi.org/10.1007/S43939-021-00012-0
12. D. Morgan, R. Jacobs, Opportunities and challenges for machine learning in materials science. Annu. Rev. Mater. **50**, 71–103 (2020). https://doi.org/10.1146/ANNUREV-MATSCI-070218-010015
13. K. Choudhary, B. DeCost, C. Chen, A. Jain, F. Tavazza, R. Cohn, C.W. Park, A. Choudhary, A. Agrawal, S.J.L. Billinge, E. Holm, S.P. Ong, C. Wolverton, Recent advances and applications of deep learning methods in materials science. NPJ Comput. Mater. **8**(1), 1–26 (2022). https://doi.org/10.1038/s41524-022-00734-6
14. S.P. Xiao, W. Hou, Studies of nanotube-based resonant oscillators through multiscale modeling and simulation. Phys. Rev. B **75**(12), 125414 (2007). https://doi.org/10.1103/PhysRevB.75.125414
15. S.P. Xiao, J. Ni, S.W. Wang, The bridging domain multiscale method and its high performance computing implementation. J. Comput. Theor. Nanosci. **5**(7), 1220–1229 (2008). https://doi.org/10.1166/jctn.2008.2557
16. B. Ren, J. Qiang, X. Zeng, A.K. Jha, S.P. Xiao, S. Li, Recent developments on thermo-mechanical simulations of ductile failure by meshfree method. Comput. Model. Eng. Sci. **71**(3), 253–277 (2011). https://doi.org/10.3970/cmes.2011.071.253
17. P. Kumar, A. Gupta, Active learning query strategies for classification, regression, and clustering: a survey. J. Comput. Sci. Technol. **35**, 913–945 (2020). https://doi.org/10.1007/s11390-020-9487-4
18. D.D. Lewis, J. Catlett, Heterogeneous Uncertainty Sampling for Supervised Learning. Proceedings of the 11th International Conference on Machine Learning, Rutgers University, New Brunswick, 148–156 (1994). https://doi.org/10.1016/B978-1-55860-335-6.50026-X
19. X. Zhu, P. Zhang, X. Lin, Y. Shi, Active learning from data streams. Proceedings of the 7th IEEE International Conference on Data Mining, Omaha, Nebraska, 757–762 (2007). https://doi.org/10.1109/ICDM.2007.101
20. B. Settles, M. Craven, An analysis of active learning strategies for sequence labeling tasks. Proceedings of the 2008 conference on empirical methods in natural language processing, Honolulu, Hawaii, 1070–1079 (2008)
21. S. Tong, D. Koller, Support vector machine active learning with applications to text classification. J. Mach. Learn. Res. 2(Nov), 45–66 (2001) https://doi.org/10.1162/153244302760185243
22. P. Melville, R.J. Mooney, Diverse ensembles for active learning. Proceedings of the 21st international conference on Machine learning, Banff, Alberta, 74 (2004). https://doi.org/10.1145/1015330.1015385
23. S.C. Hoi, R. Jin, M.R. Lyu, Large-scale text categorization by batch mode active learning. Proceedings of the 15th international conference on World Wide Web, Edinburgh, Scotland, 633–642 (2006). https://doi.org/10.1145/1135777.1135870
24. N. Roy, A. McCallum, Toward optimal active learning through monte carlo estimation of error reduction. Proceedings of the 18th international conference on machine learning, Williamstown, Massachusetts, 441–448 (2001)
25. J.J.R. Burbidge, K. R. D. Rowland, Active learning for regression based on query by committee. Proceedings of the 8th international conference on intelligent data engineering and automated learning, Birmingham, England, 209–218 (2007). https://doi.org/10.1007/978-3-540-77226-2_22

26. W. Cai, Y. Zhang, J. Zhou, Maximizing expected model change for active learning in regression. Proceedings of the IEEE 13th international conference on data mining workshops, Dallas, Texas, 51–60 (2013).https://doi.org/10.1109/ICDM.2013.104
27. D. Wu, V.J. Lawhern, S. Gordon, B.J. Lance, C.T. Lin, Offline EEG-based driver drowsiness estimation using enhanced batch-mode active learning (EBMAL) for regression. Proceedings of the IEEE international conference on systems, man, and cybernetics, Budapest, Hungary, 730–736 (2016). https://doi.org/10.1109/SMC.2016.7844328
28. H. Yu, S. Kim, Passive sampling for regression. Proceedings of the 10th IEEE International Conference on Data Mining, Sydney, Australia, 1151–1156 (2010). https://doi.org/10.1109/ICDM.2010.9
29. P. Ren, Y. Xiao, X. Chang, P.Y. Huang, Z. Li, B.B. Gupta, X. Chen, X. Wang, A survey of deep active learning. ACM Comput. Surv. **54**(9), 1–40 (2021). https://doi.org/10.1145/3472291
30. D. Yuan, X. Chang, Q. Liu, Y. Yang, D. Wang, M. Shu, Z. He, G. Shi, Active learning for deep visual tracking. IEEE Trans. Neural Netw. Learn. Syst., 1–13 (2023) https://doi.org/10.1109/TNNLS.2023.3266837
31. C. Schröder, A. Niekler, M. Potthast, Revisiting uncertainty-based query strategies for active learning with transformers. Preprint at arXiv:2107.05687 (2021)
32. S. Begum, R. Sarkar, D. Chakraborty, S. Sen, U. Maulik, Application of active learning in dna microarray data for cancerous gene identification. Expert Syst. Appl. 177, 114914 (2021) https://doi.org/10.1016/j.eswa.2021.114914
33. Z. Xiang, J. Chen, Y. Bao, H. Li, An active learning method combining deep neural network and weighted sampling for structural reliability analysis. Mech. Syst. Signal Process. 140, 106684 (2020) https://doi.org/10.1016/j.ymssp.2020.106684
34. T. Lookman, P.V. Balachandran, D. Xue, R. Yuan, Active learning in materials science with emphasis on adaptive sampling using uncertainties for targeted design. NPJ Comput. Mater. **5**(1), 21 (2019). https://doi.org/10.1038/s41524-019-0153-8
35. R. Yuan, Z. Liu, P.V. Balachandran, D. Xue, Y. Zhou, X. Ding, J. Sun, D. Xue, T. Lookman, Accelerated discovery of large electrostrains in batio3- based piezoelectrics using active learning. Adv. Mater. **30**(7), 1702884 (2018). https://doi.org/10.1002/adma.201702884
36. B. Rouet-Leduc, C. Hulbert, K. Barros, T. Lookman, C.J. Humphreys, Automatized convergence of optoelectronic simulations using active machine learning. Appl. Phys. Lett. **111**(4) (2017) https://doi.org/10.1063/1.4996233
37. D.E. Farache, J.C. Verduzco, Z.D. McClure, S. Desai, A. Strachan, Active learning and molecular dynamics simulations to find high melting temperature alloys. Comput. Mater. Sci. 209, 111386 (2022) https://doi.org/10.1016/j.commatsci.2022.111386
38. J. Allotey, K.T. Butler, J. Thiyagalingam, Entropy-based active learning of graph neural network surrogate models for materials properties. J. Chem. Phys. **155**(17), 174116 (2021). https://doi.org/10.1063/5.0065694
39. D. Wu, C.T. Lin, J. Huang, Active learning for regression using greedy sampling. Inf. Sci. **474**, 90–105 (2019). https://doi.org/10.1016/j.ins.2018.09.060
40. D.D. Lewis, A sequential algorithm for training text classifiers: Corrigendum and additional data. In Acm Sigir Forum, New York, USA **29**(2), 13–19 (1995). https://doi.org/10.1145/219587.219592
41. D. Khatamsaz, B. Vela, P. Singh, D.D. Johnson, D. Allaire, R. Arróyave, Bayesian optimization with active learning of design constraints using an entropy-based approach. NPJ Comput. Mater. **9**(1), 49 (2023). https://doi.org/10.1038/s41524-023-01006-7
42. W.Y. Hou, S.P. Xiao, Mechanical behaviors of carbon nanotubes with randomly located vacancy defects. J. Nanosci. Nanotechnol. **7**(12), 4478–4485 (2007). https://doi.org/10.1166/jnn.2007.862
43. M.A. Ghaffari, Y. Zhang, S.P. Xiao, Molecular dynamics modeling and simulation of lubricant between sliding solids. J. Micromechanics Mol. Phys. **2**(2), 1750009 (2017). https://doi.org/10.1142/S2424913017500096
44. D. Spoel, E. Lindahl, B. Hess, G. Groenhof, A.E. Mark, H.J.C. Berendsen, Gromacs: Fast, flexible, and free. J. Comput. Chem. **26**(16), 1701–1718 (2005). https://doi.org/10.1002/jcc.20291
45. S. Silling, Reformulation of elasticity theory for discontinuities and long-range forces. J. Mech. Phys. Solids **48**(1), 175–209 (2000). https://doi.org/10.1016/S0022-5096(99)00029-0
46. T. Belytschko, W.K. Liu, B. Moran, *Nonlinear Finite Elements for Continua and Structures* (Wiley, New York, 2000)
47. S.F. Li, W.K. Liu, Meshfree and particle methods and their applications. Appl. Mech. Rev. **55**(1), 1–34 (2002). https://doi.org/10.1115/1.1431547
48. T. Rabczuk, H. Ren, X. Zhuang, A nonlocal operator method for partial differential equations with application to electromagnetic waveguide problem. Comput. Mater. Contin. **59**(1), 31–55 (2019). https://doi.org/10.32604/cmc.2019.04567
49. H. Ren, X. Zhuang, T. Rabczuk, A higher order nonlocal operator method for solving partial differential equations. Comput. Methods Appl. Mech. Eng. **367**(1), 113132 (2020). https://doi.org/10.1016/j.cma.2020.113132
50. H. Ren, X. Zhuang, T. Rabczuk, Dual-horizon peridynamics: A stable solution to varying horizons. Comput. Methods Appl. Mech. Eng. **318**(1), 762–782 (2017). https://doi.org/10.1016/j.cma.2016.12.031
51. Q.Z. Zhu, T. Ni, Peridynamic formulations enriched with bond rotation effects. Int. J. Eng. Sci. **121**, 118–129 (2017). https://doi.org/10.1016/j.ijengsci.2017.09.004
52. S.A. Silling, M. Epton, O. Weckner, J. Xu, E. Askari, Peridynamic states and constitutive modeling. J. Elast. **88**(2), 151–184 (2007). https://doi.org/10.1007/s10659-007-9125-1
53. A.J. Smola, B. Scholkopf, A tutorial on support vector regression. Stat. Comput. **14**(3), 199–222 (2004). https://doi.org/10.1023/B:STCO.0000035301.49549.88
54. C.W. Hsu, C.J. Lin, A comparison of methods for multiclass support vector machines. IEEE Trans. Neural Netw. **13**(2), 415–425 (2002). https://doi.org/10.1109/72.991427
55. S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans. Neural Netw. **2**(2), 302–309 (1991). https://doi.org/10.1109/72.80341
56. L.S. Dreiseitl, andOhno-Machado: Logistic regression and artificial neural network classification models: A methodology review. J. Biomed. Inform. **35**(5–6), 352–359 (2002). https://doi.org/10.1016/S1532-0464(03)00034-0
57. V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th International Conference on International Conference on Machine Learning, 807–814, Haifa, Israel (2010)
58. T. Sainath, O. Vinyals, A. Senior, H. Sak, Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 4580–4584 (2015). https://doi.org/10.1109/ICASSP.2015.7178838
59. I.J. Myung, Tutorial on maximum likelihood estimation. J. Math. Psychol. **47**(1), 90–100 (2003). https://doi.org/10.1016/S0022-2496(02)00028-7
60. D. Wu, Pool-based sequential active learning for regression. IEEE Trans. Neural Netw. Learn. Syst. **30**(5), 1348–1359 (2019). https://doi.org/10.1109/TNNLS.2018.2868649

61. S. Kullback, R.A. Leiber, On information and sufficiency. Ann. Math. Statist. **22**(1), 79–86 (1951). https://doi.org/10.1214/aoms/1177729694
62. K. Morsi, V.V. Patel, Processing and properties of titanium-titanium boride (tibw) matrix composites. J. Mater. Sci. **42**, 2037–2047 (2007). https://doi.org/10.1007/s10853-006-0776-2
63. B.F. Decker, J.S. Kasper, The crystal structure of tib. Acta Crystallogr. **7**(1), 77–80 (1954). https://doi.org/10.1107/S0365110X5400014X
64. F.C. Ma, P. Liu, W. Li, X.K. Liu, X.H. Chen, K. Zhang, D. Pan, W.J. Lu, The mechanical behavior dependence on the tib whisker realignment during hot-working in titanium matrix composites. Sci. Rep. **6**(1), 1–9 (2016). https://doi.org/10.1038/srep36126
65. S. Attarian, S.P. Xiao, Development of a 2nn-meam potential for tib system and studies of the temperature dependence of the nanohardness of tib2. Comput. Mater. Sci. **201**, 11857 (2022). https://doi.org/10.1016/J.COMMATSCI.2021.110875
66. ASM (ed.): Properties and Selection-Nonferrous Alloys and Pure Metals Volume 1: Metals Park. American Society for Metal, Ohio (1979)
67. S.S. Sahay, K.S. Ravichandran, R. Atri, B. Chen, J. Rubin, Evolution of microstructure and phases in in situ processed ti-tib composites containing high volume fractions of tib whiskers. Mater. Res. **14**(11), 4214–4223 (1999). https://doi.org/10.1557/JMR.1999.0571
68. F. Ma, B. Zheng, P. Liu, W. Li, X. Liu, X. Chen, K. Zhang, D. Pan, W. Lu, Modeling of effects of thermomechanical processing on elevated-temperature mechanical properties of in situ (tib+ tic)/ti-1100 composite. J. Mater. Sci. **51**(16), 7502–7511 (2016). https://doi.org/10.1007/s10853-016-0029-y
69. D.J. Littlewood, M.L. Parks, J.T. Foster, J.A. Mitchell, P. Diehl, The peridigm meshfree peridynamics code. J. Peridynamics Nonlocal Model. **6**(1), 118–148 (2024). https://doi.org/10.1007/s42102-023-00100-0
70. M.L. Parks, D.J. Littlewood, J.A. Mitchell, S.A. Silling, Peridigm users' guide. V1. 0.0. Sandia National Laboratories (SNL), Albuquerque, New Mexico, and Livermore, California (2012)