Environmental Science Advances

ROYAL SOCIETY OF CHEMISTRY

CRITICAL REVIEW

View Article Online
View Journal | View Issue

Cite this: *Environ. Sci.: Adv.*, 2025, **4**, 1963

A comprehensive review of characterizing CO_2-brine interfacial tension in saline aquifers using machine learning

M. Saud Ul Hassan, Kashif Liaqat * and Laura Schaefer

The alarming increase in global warming, primarily driven by the rising CO_2 concentration in the atmosphere, has spurred the need for technological solutions to reduce CO_2 concentrations. One widely successful approach is geological sequestration, which involves pressurizing and injecting CO_2 into underground rock formations. Saline aquifers, containing saltwater, are often used for this purpose due to their large storage capacity and broad availability. However, to optimize CO_2 storage and reduce the risk of gas leakage, it is essential to account for capillary forces and the interfacial tension (IFT) between CO_2 and brine within the formation. Traditional methods for characterizing CO_2-brine IFT in saline aquifers, both experimental and theoretical, are well-documented in the literature. Experimental methods, though accurate, are labor-intensive, time-consuming, and require expensive equipment, while theoretical approaches rely on idealized models and computationally demanding simulations. Recently, machine learning (ML) techniques have emerged as a promising alternative for IFT characterization. These techniques allow models of CO_2-brine IFT to be automatically "learned" from data using optimization algorithms. The literature suggests that ML can achieve superior accuracy compared to traditional theoretical methods. However, in its current state, the literature lacks a comprehensive review of these emerging methods. This work addresses that gap by offering an in-depth survey of existing machine learning techniques for IFT characterization in saline aquifers, while also introducing novel, unexplored approaches to inspire future advancements. Our comparative analysis shows that simpler ML models, such as ensemble tree-based models and small multi-layer perceptrons, may be the most accurate and practical for estimating CO_2-brine IFT in saline aquifers.

Received 5th June 2025
Accepted 9th September 2025

DOI: 10.1039/d5va00163c

rsc.li/esadvances

Environmental significance

Addressing climate change requires effective carbon capture and storage (CCS), with geological sequestration in saline aquifers offering high potential. A key factor in CCS success is understanding interfacial tension (IFT) between CO_2 and brine, which affects storage efficiency and leakage risk. Traditional IFT methods are complex and resource-heavy. This study explores machine learning (ML) as a scalable, data-driven alternative for IFT prediction. Beyond performance, it examines why ML models work, their limitations, and future challenges. By enhancing IFT modeling, this work promotes safer, more efficient CO_2 storage and advances global sustainability through better climate mitigation strategies, uniting data science with environmental action.

1 Introduction

Earth's temperature is rising at an alarming rate of 0.2 °C per decade, and a main contributor to this is the increasing concentration of carbon dioxide (CO_2) in the atmosphere.[1] To limit the concentration of atmospheric CO_2, a process called geologic carbon sequestration is often employed, wherein CO_2 is pressurized and injected into porous underground formations for storage.[2] Typically, these formations are porous rocks that contain saltwater, termed saline aquifers, such as sandstone.[3] Another common option is to store CO_2 in hydrocarbon-bearing formations – such as oil and gas reservoirs, gas shales, and coal seams[4] – which can also be done as a part of enhanced oil recovery,[5–8] where CO_2 is injected into an oil-bearing formation to drive the oil out of it. However, saline aquifers are estimated to have a significantly larger carbon dioxide storage capacity than hydrocarbon formations,[9] and they are also much more ubiquitous,[10] making them a prime target for CO_2 storage.

To optimally utilize the CO_2 storage capacity of saline aquifers as well as reduce the danger of CO_2 leakage out of them, which can be damaging to the environment and harmful to animal and human life,[11] one must consider the capillary forces at play.[12,13] These forces are mainly governed by the interfacial

Energy Systems Lab, Dept. of Mechanical Engineering, Rice University, Houston, TX, 77005, USA. E-mail: Kashif.Liaqat@rice.edu; ms18ig@my.fsu.edu; Laura.Schaefer@rice.edu

tension (IFT) between CO_2 and the host fluid, brine.[13–15] Accurate characterization of CO_2-brine interfacial tension in saline aquifers is thus of prime importance to the success of geologic sequestration projects.

The methods presented in the literature for characterizing CO_2-brine IFT can broadly be classified into three categories: experimental, theoretical, and data-driven methods. The most widely used experimental methods to measure the CO_2-brine IFT are the pendant drop method[16,17] and the capillary rising method.[18] The pendant drop method measures surface or interfacial tension by analyzing the shape of a drop hanging from a needle in a surrounding fluid. The droplet's profile, captured as a shadow image, is used in drop shape analysis to calculate tension based on the balance between gravity and interfacial forces. The capillary rise method determines IFT by observing the height to which a liquid rises or falls in a narrow tube when in contact with another fluid. The balance between adhesive and cohesive forces allows calculation of IFT. The experimental procedures, though inherently accurate, are prone to inaccuracies introduced through measurement noise and experimental errors. Additionally, they can be time-consuming to carry out, require expensive equipment, and demand extensive experience with the equipment.[14,19] The theoretical approaches,[20–23] on the other hand, are mainly based on molecular dynamics models,[24] often demanding idealized conditions that are rarely satisfied closely in practice. Furthermore, they rely on computer simulations that introduce numerical inaccuracies.[14,19]

While experimental and theoretical methods have been invaluable to date for understanding CO_2-brine IFT, their practical application in large-scale carbon storage projects remains challenging. As geologic carbon sequestration projects expand globally, there is a growing need for predictive tools that are both accurate and scalable across diverse geologic settings for carbon capture and storage.[25–27] Machine learning (ML) offers a compelling alternative.[28–30] By leveraging existing experimental and simulation datasets, ML models can capture complex, nonlinear relationships between fluid properties, environmental conditions, and IFT, without requiring explicit physical simplifications. ML approaches can be rapidly retrained with new data, adapted to different brine compositions, and deployed for real-time prediction, making them attractive for both research and operational decision-making. Over the past decade, interest in applying ML to CO_2-brine IFT prediction has grown considerably, producing a scattered body of work across multiple disciplines. However, no comprehensive review currently exists to summarize these developments, compare methodologies, or identify open challenges. This article addresses that gap by providing the first systematic survey of ML-based approaches for IFT characterization in saline aquifers, alongside a comparative analysis to guide future research.

In Section 2, we introduce the problem of modeling CO_2-brine interfacial tension in saline aquifers using data-based methods, aiming to formulate the problem in a way that accommodates various machine learning techniques. Building on this formulation, we then present an overview of various machine learning models for IFT characterization, as reviewed in Section 3 and summarized in Fig. 1. We also present novel approaches for modeling IFT as a time series using state-of-the-art sequential machine learning models, which present

M. Saud Ul Hassan

Saud is a visiting researcher at Rice University working on the application of deep learning methods to problems in energy systems and climate science. Concurrently, he is a senior software engineer at Advanced Micro Devices (AMD). Saud has a master's degree in mechanical engineering from Florida State University.

Kashif Liaqat

Kashif is a PhD Candidate in the Department of Mechanical Engineering at Rice University. He holds a master's degree in mechanical engineering from Florida State University. His areas of research include applying interdisciplinary approaches to energy systems optimization, which encompasses thermal modeling, simulations, artificial intelligence, and software/tool development.

Laura Schaefer

Laura is the Burton J. and Ann M. McMurtry Chair in Engineering and a Professor of Mechanical Engineering at Rice University. She received a BS in Mechanical Engineering and a BA in English from Rice, and her MS and PhD in Mechanical Engineering from the Georgia Institute of Technology. Her research centers on the analysis, design and optimization of energy systems, with an emphasis on improving energy efficiency and diversification for increased sustainability.

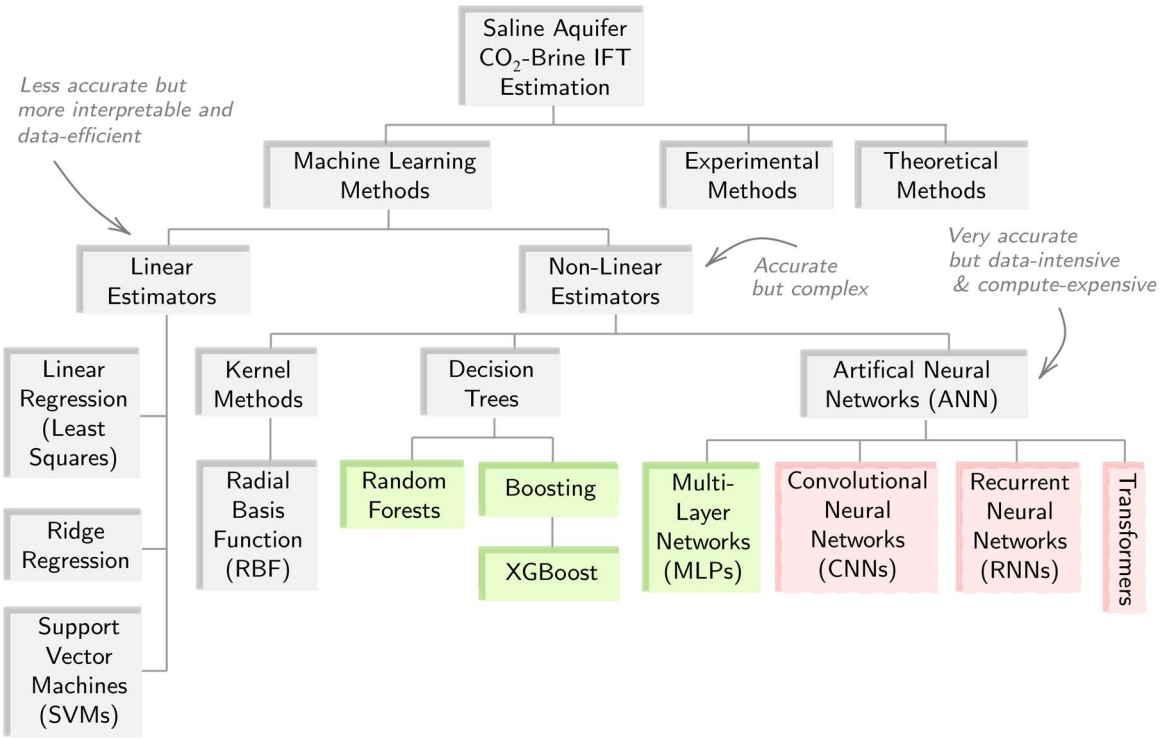

Fig. 1 Overview of machine learning approaches for estimating CO_2-brine interfacial tension in saline aquifers, as discussed in Section 2. The methods shown in red-shaded blocks – namely CNNs, RNNs, and Transformers – are promising based on their success in time-series prediction tasks but have not yet been applied to CO_2-brine IFT estimation, which can also be framed as a time-series prediction problem (see Section 2.2.3). Additionally, the green-shaded blocks denote ensemble tree-based models (random forests and gradient boosting) and MLPs, which the literature survey in Section 3 identifies as currently offering the strongest predictive performance for this application. Nonetheless, the inconsistent use of datasets across studies underscores the need for caution when generalizing these findings.

promising directions for future research. Throughout this section, we strive to establish a standardized mathematical notation for describing the different models, as the absence of a standardized approach in the existing literature has resulted in works that are difficult to compare. In Section 4, we provide a brief overview of how IFT relates to different physical parameters. Finally, in Section 5, we critique the current state of the literature and propose recommendations to incorporate into future research in the field.

2 Machine learning methods for CO_2-brine IFT characterization in saline aquifers

Different studies have opted for different sets of features to characterize CO_2-brine interfacial tension in saline aquifers. Some studies have opted for pressure, temperature, and salinity, while others have opted for larger feature sets. For example,[31,32] consider CO_2-brine IFT characterization in saline aquifers based on six features: pressure, temperature, molalities of the monovalent cations (Na^+, K^+) and bivalent cations (Ca^{2+}, Mg^{2+}), and mole fractions of CH_4 and N_2 in the CO_2 stream. This feature set is the most commonly adopted across studies, based on our review in Section 3. However, the specific choice of feature set does not impact the exposition below; simply denote $\mathcal{X} \subseteq \mathbb{R}^d$ as the feature space, where d is the number of features, also called the dimension of the feature space.

The CO_2-brine interfacial tension in saline aquifers may generally be modeled as $f: \mathcal{X} \to \mathcal{Y} \subseteq \mathbb{R}$.† Though this function is not known analytically, one may experimentally obtain data:

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) | \mathbf{x}_n \in \mathcal{X}, y_n \in \mathbb{R}\},$$

where the sample points $(\mathbf{x}_n, y_n) \in \mathcal{D}$ are picked independently such that $\mathbf{x}_n \sim P(\mathcal{X})$ and $y_n = f(\mathbf{x}_n) + \varepsilon_n$, to characterize the input–output behaviour of f under noise $\varepsilon_n \in \mathbb{R}$, and subsequently employ machine learning algorithms to construct an approximation $\hat{f} \in \mathcal{H}$ to f from a hypothesis set $\mathcal{H}$ using $\mathcal{D}$ (Fig. 2).‡ In the following, we formally present the machine learning algorithms for CO_2-brine IFT characterization reviewed in Section 3 using this problem formulation. Additionally, we introduce advanced methods for sequential data processing, which can be adapted for CO_2-brine IFT modeling through

† While our review focuses on saline aquifers, machine learning-based IFT characterization can also be applied to other rock formations.

‡ We adopt vector notation to formulate machine learning models, as opposed to the scalar notation commonly used in much of the literature we reviewed. Vector notation offers a more compact and elegant way to express machine learning models, which are high-dimensional objects, and aligns with the conventions of contemporary machine learning research.

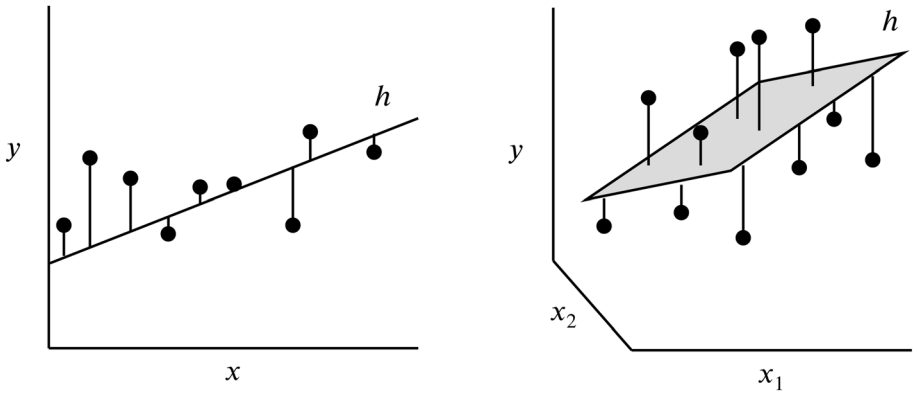

Fig. 2 A depiction of linear hypotheses $h \in \mathscr{H}_{\text{lin}}$ in one-dimensional ($d = 1$) and two-dimensional ($d = 2$) feature spaces.[33] Each black circle represents a data-point $(\mathbf{x}_n, y_n) \in \mathcal{D}$ in $\mathbb{R}^d$, and the vertical lines depict the error $\|h(\mathbf{x}_n) - y_n\|_2$. The linear regression algorithm designs the hyperplane to be such that the error is minimal on average.

a straightforward reformulation of the problem, as we later demonstrate.

2.1 Linear estimators

Many studies on data-driven modeling of CO_2-brine IFT have considered linear estimation techniques. These methods construct a linear hypothesis explaining f, despite mathematical and empirical models of CO_2-brine IFT showing that f is a nonlinear function.[34–37] While this approach sacrifices accuracy, the simplicity and analytical tractability of the linear hypothesis class make it more interpretable and computationally efficient to search over.

Formally, the hypothesis class of linear estimators, $\mathcal{H}_{\text{lin}}$, is the set of hyperplanes in $\mathbb{R}^d \times \mathbb{R}$ with a surface normal $\mathbf{a}^T$ and offset b from the origin:

$$\mathcal{H}_{\text{lin}} = \left\{ h : \mathbb{R}^d \to \mathbb{R} \middle| h(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b, \mathbf{a} \in \mathbb{R}^d, b \in \mathbb{R} \right\}.$$

This set can be represented more succinctly by embedding $\mathbb{R}^d$ in $\mathbb{R}^{d+1}$ as $\tilde{\mathbb{R}}^{d+1} = \mathbb{R}^d \times \{1\} \subset \mathbb{R}^{d+1}$. Then:

$$\tilde{\mathcal{H}}_{\text{lin}} = \left\{ \tilde{h} : \tilde{\mathbb{R}}^{d+1} \to \mathbb{R} \middle| \tilde{h}(\tilde{\mathbf{x}}) = \tilde{\mathbf{a}}^T\tilde{\mathbf{x}}, \tilde{\mathbf{a}} \in \mathbb{R}^{d+1} \right\},$$

denoting the set of hyperplanes in $\mathbb{R}^{d+1}$ passing through the origin, is equivalent to $\mathcal{H}_{\text{lin}}$, in that a hypothesis $\tilde{h}(\tilde{\mathbf{x}}) = \tilde{\mathbf{a}}^T\tilde{\mathbf{x}} \in \tilde{\mathcal{H}}_{\text{lin}}$ can be converted to a equivalent hypothesis $h(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b \in \mathcal{H}_{\text{lin}}$, where $\mathbf{a} = (a_1, \ldots, a_d)$, by choosing the surface normal as $\tilde{\mathbf{a}} = (a_1, \ldots, a_d, b)$. We will use both these representations of the linear hypotheses set in the exposition below – covering linear regression, ridge regression, and support vector machine regression – depending on whichever is mathematically convenient.

2.1.1 Linear regression. The linear regression algorithm chooses a hypothesis $\tilde{h} \in \tilde{\mathcal{H}}_{\text{lin}}$ that minimizes the error:

$$E_{\text{in}}\left(\tilde{h}, \tilde{\mathcal{D}}\right) = \frac{1}{N}\sum_{n=1}^{N} \|\tilde{h}(\tilde{\mathbf{x}}_n) - y_n\|_2^2, \quad \text{for} (\tilde{\mathbf{x}}_n, y_n) \in \tilde{\mathcal{D}}.$$

Here, $\|.\|_2$ is the ℓ^2 norm, and $\tilde{\mathcal{D}}$ represents the dataset $\mathcal{D}$ embedded in $\tilde{\mathcal{X}} \times \mathcal{Y}$, where $\tilde{\mathcal{X}} = \mathcal{X} \times \{1\} \subset \tilde{\mathbb{R}}^{d+1}$:

$$\tilde{\mathcal{D}} = \{(\tilde{\mathbf{x}}_1, y_1), \ldots, (\tilde{\mathbf{x}}_N, y_N) | \tilde{\mathbf{x}}_i \in \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \times \{1\}\}.$$

Using standard matrix calculus, the hypothesis $\tilde{h}_{\text{lr}} \in \tilde{\mathcal{H}}_{\text{lin}}$ that minimizes E_{in} can be found to be given by $\tilde{h}_{\text{lr}}(\tilde{\mathbf{x}}) = \tilde{\mathbf{a}}_{\text{lr}}{}^T\tilde{\mathbf{x}}$, where $\tilde{\mathbf{a}}_{\text{lr}}$ is given by the roots of $\nabla_{\tilde{\mathbf{a}}} E_{\text{in}}$:[38]

$$\tilde{\mathbf{a}}_{\text{lr}} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T\mathbf{y},$$

where:

$$\tilde{X} = \begin{bmatrix} — & \tilde{\mathbf{x}}_1{}^T & — \\ — & \tilde{\mathbf{x}}_2{}^T & — \\ & \vdots & \\ — & \tilde{\mathbf{x}}_n{}^T & — \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

and $\tilde{X}^T\tilde{X}$ is assumed to be invertible; that is, $\det\tilde{X}^T\tilde{X} \neq 0$. Linear regression is one of the few machine learning algorithms where the analytical formula describing the optimal hypothesis is known: $\tilde{h}_{\text{lr}}(\tilde{\mathbf{x}}) = \tilde{\mathbf{a}}_{\text{lr}}{}^T\tilde{\mathbf{x}}$.[33] It is important to realize that this hypothesis is optimal with respect to the in-sample error, E_{in}, while what is of interest is the out-of-sample error, E_{out}, which is a proxy for how well the model would generalize to real-world data. However, E_{out} cannot be computed since f is unknown, and thus one has no recourse but to work with E_{in}. This is a theme common to all machine learning methods, but some methods, particularly linear estimators, are special in that we can often find exact bounds on the out-of-sample error for the optimal hypothesis.

2.1.2 Ridge regression. The linear regression algorithm relies on $\tilde{X}^T\tilde{X}$ being invertible. If $\tilde{X}^T\tilde{X}$ is singular, the optimal hypothesis $\tilde{h}_{\text{lr}}$ is undefined. An ad-hoc solution to this problem is to define the optimal hypothesis as $\tilde{h}_{\text{rr}}(\tilde{\mathbf{x}}) = \tilde{\mathbf{a}}_{\text{rr}}{}^T\tilde{\mathbf{x}}$, where

$$\tilde{\mathbf{a}}_{\text{rr}} = \left(\tilde{X}^T\tilde{X} + \lambda I\right)^{-1}\tilde{X}^T\mathbf{y}, \quad \text{for } \lambda \in \mathbb{R}^+.$$

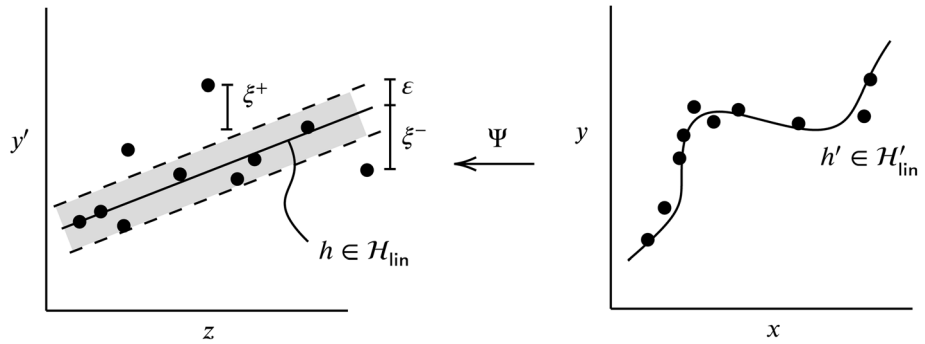

Fig. 3 Depiction of a nonlinear hypothesis in $\mathcal{X}$-space designed by transforming the problem into feature space $\mathcal{Z}$ through a non-linear transform Ψ, and using SVM (a linear method) in the feature space.

This technique is called ridge regression, where λ, called the regularization rate, is a hyperparameter, *i.e.*, a parameter whose value is either determined through trial and error or by using a meta-optimization scheme, such as Particle Swarm Optimization (PSO).[39] Note that ridge regression is also more numerically stable than linear regression.§ However, unlike linear regression, ridge regression is a biased estimator.[40] This increase in bias is counterbalanced by a reduction in variance – a phenomenon known as the bias-variance tradeoff[38,41] – making ridge regression less prone to the problem of overfitting, wherein a model fits the dataset too closely, thus compromising how well it fits the desired function f.[42]

2.1.3 Support vector machines. Support vector machines (SVMs) are linear classifiers, originally used to solve binary classification problems by finding the maximal margin hyperplane[43] separating the space $\mathbb{R}^d$ of all data points into two half-spaces. Here, margin is defined as the distance of a hyperplane to the data point(s) closest to it (called support vectors), and one can ascertain that it is inversely related to the flatness of the hyperplane.[44] SVMs can be extended to regression problems,[45] where the goal becomes to find a hyperplane $h(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b \in \mathcal{H}_{\text{lin}}$ that is as flat as possible and does not deviate from the targets y_n by more than ε.[46] However, as is, the problem can be infeasible if no function $h \in \mathcal{H}_{\text{lin}}$ exists that approximates all points $(\mathbf{x}_n, y_n) \in \mathcal{D}$ to within ε. In order to address this issue, the constraints are made "soft" through the introduction of slack variables ${\xi_n}^+$ and ${\xi_n}^-$,[46] thus yielding the following constrained optimization problem:

$$\min_{\mathbf{a},b} \quad \frac{1}{2}\|\mathbf{a}\|_2{}^2 + C\sum_{n=1}^{N}({\xi_n}^+ + {\xi_n}^-)$$

$$\text{s.t.} \quad -\varepsilon - {\xi_n}^- \le y_n - \left(\mathbf{a}^T\mathbf{x}_n + b\right) \le \varepsilon + {\xi_n}^+, \forall (\mathbf{x}_n, y_n) \in \mathcal{D}$$

Here, $C \in \mathbb{R}$ controls the ε-insensitivity, *i.e.*, the degree to which one is willing to allow data points to fall outside the ε allowance (Fig. 3 – left). For details on how to solve this problem using quadratic programming, refer to ref. 46 and 47.

2.2 Nonlinear estimators

Linear methods are supported by a well-developed body of mathematical theory,[38,44] which makes them reliable and robust machine learning techniques. However, linear methods, by their very construction, are limited to linear hypotheses, and since interfacial tension is a non-linear phenomenon, linear approximations to f might not be desirable, unless one is specifically interested in understanding linear patterns in f, or if computational resources are severely limited, *e.g.*, in the case of a real-time controller deployed on an edge device.

While previous research has explored nonlinear methods for CO_2-brine IFT estimation in saline aquifers, the coverage has been restricted. In particular, advanced neural estimators like convolutional neural networks, recurrent neural networks, and transformers, which have demonstrated cutting-edge performance in time-series prediction tasks across diverse scientific and engineering domains, remain untapped in the context of CO_2-brine IFT prediction in saline aquifers. In the following sections, we not only formally present the nonlinear estimators previously applied in the literature but also present some yet-to-be-applied modern time series prediction methods.

2.2.1 Kernel methods. A common way to augment the hypotheses class $\mathcal{H}_{\text{lin}}$ is by transforming the input space $\mathcal{X}$ through a non-linear transform $\Psi : \mathcal{X} \to \mathcal{Z} \subseteq \mathbb{R}^{d'}$ and constructing a hypotheses set over $\mathcal{Z}$:[38,44]

$$\mathcal{H}'_{\text{lin}} = \left\{ h : \mathcal{Z} \to \mathbb{R} \,\middle|\, h(\mathbf{z}) = \mathbf{a}^T\mathbf{z}, \mathbf{a} \in \mathbb{R}^{d'}, \mathbf{z} = \Psi(\mathbf{x}) \right\}.$$

The d'-dimensional space $\mathcal{Z}$ is commonly referred to as the feature space; and since it is related to the input space, $\mathcal{X}$, through a non-linear mapping, employing linear estimators in the feature space gives non-linear hypotheses in the input space (Fig. 3).[33]

Designing linear estimators in the feature space is computationally expensive if it is high-dimensional, as is usually the case.¶ Take, for example, ridge regression in a feature space $\mathcal{Z}$ of dimension d'. To compute the estimator, one must solve the linear system $\mathbf{a}_{\text{rr}} = (Z^TZ + \lambda I)^{-1}Z^T\mathbf{y}$, which requires $\mathcal{O}(d'^3)$

§ The condition number of $\tilde{X}^T\tilde{X}$ is $\sigma_{\max}/\sigma_{\min}$, where $\sigma_{\max}$ and $\sigma_{\min}$ represent the largest and smallest singular values of $\tilde{X}^T\tilde{X}$, respectively. Notice that $\sigma_{\max}/\sigma_{\min} \to \infty$ as $\sigma_{\min} \to 0$, making linear regression numerically unstable. On the contrary, the condition number of $\tilde{X}^T\tilde{X} + \lambda I$ is $(\sigma_{\max} + \lambda)/(\sigma_{\min} + \lambda)$, which remains finite as $\sigma_{\min} \to 0$.

¶ Many commonly used feature transformations are, in fact, infinite-dimensional. For instance, the popular Gaussian Radial Basis Function kernel induces an infinite-dimensional mapping.

operations. In such cases, the dual solution can often be cheaper to compute. For example, the dual solution to ridge regression is $\mathbf{a}_{\text{rr}} = (ZZ^T + \lambda I)^{-1}\mathbf{y}$, which requires $\mathcal{O}(N^3)$ operations, where N is the number of data points. Thus, for $N < d'$, the dual solution is computationally cheaper. Additionally, the entries $\langle \Psi(\mathbf{x}_i), \Psi(\mathbf{x}_j)\rangle$ of the matrix ZZ^T, known as the Gram matrix, are inner products, which can often be computed as a direct function of the inputs, thus further reducing the computational cost.[48]

Kernel methods capitalize on this property by defining a function $K : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$, called a kernel, that enables direct computation of these inner products. For instance, the quadratic transform $\Psi : \mathbf{z} = (z_1, z_2) \mapsto \Psi(\mathbf{z}) = ({z_1}^2, {z_2}^2, \sqrt{2}\, x_1 x_2)$ leads to the quadratic kernel $K(\mathbf{z}_i, \mathbf{z}_j) = \langle \mathbf{z}_i, \mathbf{z}_j\rangle^2$. However, it is not necessary to explicitly define the transformation Ψ; one can specify the kernel function K directly, provided that K meets Mercer's conditions.[49] For example,

$$K(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{z}_i - \mathbf{z}_j\|_2^{\,2}\right)$$

is a valid kernel, called the Gaussian Radial Basis Function (RBF) kernel, since it satisfies Mercer's conditions. Any machine learning algorithm that can be restated such that the input vectors $\mathbf{x}_n$ appear as inner-products only is amenable to the "kernel trick." That is, the inner product can be replaced by a kernel function, thus improving the computational efficiency of the algorithm.[50]

2.2.2 Decision trees. The methods discussed thus far (as well as neural networks, discussed in the upcoming section) result in global models, *i.e.*, models that describe the whole of input space $\mathcal{X}$ using a single rule. However, instead of using a complicated and difficult-to-interpret model to describe all of $\mathcal{X}$, an alternative is to partition $\mathcal{X}$ into subsets, and then partition those subsets into further subsets, and keep going in this fashion until the subsets are small enough that they can be described by simple models. Decision trees are machine learning methods that follow this approach to data modeling.[51]

Typically, decision trees are constructed in a top-down manner using a recursive partitioning strategy,[52] where the input space is greedily divided into subsets.[53] This process continues until the resulting subsets are sufficiently small to be represented by constant values.

While decision trees are expressive and powerful, they are also highly prone to overfitting. To control the degree of overfitting, one usually constrains the maximum depth of the tree (called top-down pruning) or removes leaves from the tree after it has been built (called bottom-up pruning). Another common strategy is to use an ensemble learning method,[54] such as bagging or boosting.

2.2.2.1 Bagging and random forests. The prediction error of an estimator is generally a function of its bias and variance– the lower the bias and variance, the lower the prediction error. However, as alluded to earlier, these two factors are linked by the bias-variance tradeoff,[38,41] meaning that estimators with low bias tend to have high variance. Bagging seeks to take several deep but largely uncorrelated trees (*i.e.*, estimators with low bias but high variance), and average out their predictions, which reduces the overall variance without changing the bias, thus reducing the overall prediction error.[38]

Formally, bagging is an ensemble learning method where M decision trees are constructed on independent subsets of the dataset $\mathcal{D}$, and at inference time, the final prediction on a given input $\mathbf{x} \in \mathcal{X}$, $\hat{f}(\mathbf{x})$, is obtained by averaging the predictions $\hat{f}_{\text{m}}(\mathbf{x})$ from the individual trees:

$$\hat{f}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M}\hat{f}_m(\mathbf{x}).$$

Note that even though bagged trees are grown on independent subsets of the dataset, the inputs to the trees can still be correlated, reducing the benefit that bagging brings. The Random forests approach aims to address this problem by employing bagging along with feature bagging, wherein each tree is constructed on a subset of the dataset using only a subset of the possible feature splits.[38]

2.2.2.2 Boosting. Boosting is also an ensemble learning method, combining multiple models to reduce the overall prediction error. However, unlike bagging and random forests, which seek to reduce the prediction error by reducing variance, boosting seeks to reduce the prediction error by reducing bias.[38] Boosting iteratively constructs an ensemble of shallow decision trees (*i.e.*, estimators with low variance but high bias) where each subsequent tree attempts to correct the error in its predecessor's prediction,[38] thus reducing the overall prediction error.

2.2.3 Artificial neural networks. Artificial neural networks (ANNs) owe their design to inspirations from the biological brain,[56] however, they are now understood as mathematical functions of the form:

$$\Gamma(\mathbf{x};\Theta) = A_L\Phi(A_{L-1}(\ldots\Phi(A_1\mathbf{x} + \mathbf{b}_1)\ldots) + \mathbf{b}_{L-1}) + \mathbf{b}_L$$

where $L \in \mathbb{N}$, $\Theta = (A_\ell, \mathbf{b}_\ell)_{\ell=1}^{\,L}$, and $\Phi : \mathbb{R}^k \to \mathbb{R}^k$ is a non-linear function, often referred to as the activation function.[57] Common choices for the activation function include the rectified linear unit (ReLU), wavelet basis functions, and radial basis functions.

In neural networks literature, $\mathbf{x}$ is called the input layer, and each application of $T_\ell(\mathbf{u}) = \Phi(A_\ell\mathbf{u} + \mathbf{b}_\ell)$ is called a hidden layer, where $A_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $\mathbf{b}_l \in \mathbb{R}^{n_\ell}$ for $n_\ell \in \mathbb{N}$ and $n_0 = d$, are called the parameters or weights of the ℓ-th layer. A layer is thought of as made of neurons (Fig. 4), and the output $T_\ell(\mathbf{u}) \in \mathbb{R}^{n_\ell}$ of a layer is called a neural activation. It is common to define the depth of a network as the number of layers, L, and its size as the total number of neurons in those layers: $\sum_{\ell=0}^{L-1} n_\ell$.[58]

The hypotheses set of neural networks is very expressive: under minor conditions on the activation function Φ, it can be shown that every continuous function $g : \mathcal{K} \to \mathbb{R}$ on a compact set $\mathcal{K}$ can be arbitrarily well-approximated by a fixed-size neural network.[59] One can pose the task of approximating the IFT function, f, from the dataset $\mathcal{D}$, using a neural network Γ as finding parameters Θ such that:

$$\arg\min_{\Theta}\mathcal{L}(\Theta) = \mathcal{L}_{\text{in}}(\Gamma(\cdot;\Theta), \mathcal{D}) + \lambda\mathcal{R}(\Theta),$$

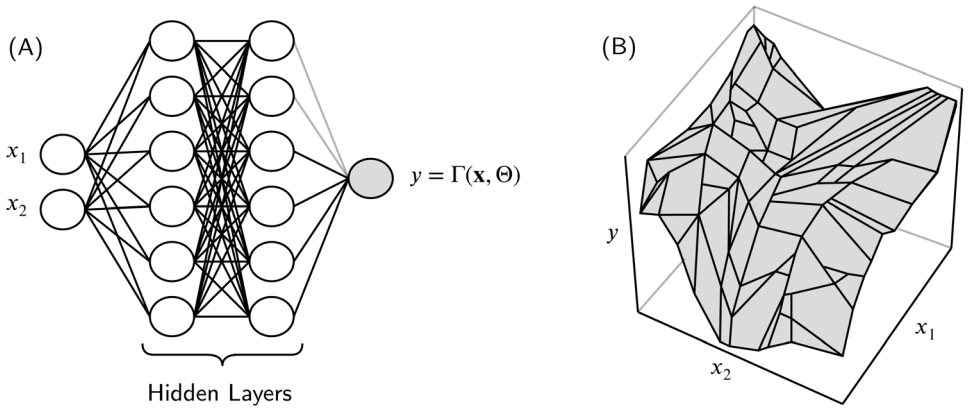

Fig. 4 (A) Visualization of a neural network with three layers: one input layer, and two hidden layers. To keep the diagram simple, the connections are shown as undirected, and the input $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ is depicted for $d = 2$. The white circles depict neurons, where each neuron in the hidden layers is an affine mapping followed by a nonlinearity Ψ. The grey circle is the output. Though a typical neural network can have several outputs, we only require a single output ($n_L = 1$), that is, the interfacial tension. (B) An example hypothesis of a single-output ReLU neural network visualized as piece-wise continuous linear function over $\mathcal{X} \subseteq \mathbb{R}^2$.[55]

where $\mathcal{L}$, known as the loss function, is composed of the in-sample loss $\mathcal{L}_{\text{in}}(\Gamma(\cdot;\Theta), \mathcal{D})$, measuring how well a given neural network parameterization $\Gamma(\cdot;\Theta)$ approximates f over $\mathcal{D}$, and a regularization term $\lambda\mathcal{R}(\Theta)$, modulated by a hyperparameter $\lambda \in \mathbb{R}^+$, which constrains the possible parameterizations of Γ.[58]

It is important to point out here that even if $\mathcal{L}$ is a "simple" function of Γ, obtaining an analytical expression for $\Theta^* = \arg\min_\Theta \mathcal{L}(\Theta)$ is generally not possible. Instead, one approximates Θ^* as Θ_t using an iterative procedure, such as gradient descent:[60]

$$\Theta_t = \Theta_{t-1} - \gamma \nabla_\Theta \mathcal{L}(\Theta_{t-1}),$$

where $t \in \{1, \ldots, T\}$, for some $T \in \mathbb{N}$, identifies a step in the algorithm's execution, and $\gamma \in \mathbb{R}$ is a hyperparameter, called the step size.‖ In practice, Θ_0 is chosen to be random non-zero values or set using a weight initialization scheme,[61] and $\nabla_\Theta \mathcal{L}$ is computed using the backpropagation algorithm,[62] which automatically computes gradients using an efficient graph-based implementation of the chain rule. However, note that $\nabla_\Theta \mathcal{L}(\Theta_{t-1})$ is evaluated over all data points $(\mathbf{x}, y) \in \mathcal{D}$, which might be computationally expensive. Therefore, it is common to use a variation of gradient descent called stochastic gradient descent,[63] where $\mathcal{D}$ is divided into a set of batches, and $\nabla_\Theta \mathcal{L}(\Theta_{t-1})$ is computed only over one batch in a given gradient descent step. It can thus take several steps to update Θ over all of $\mathcal{D}$, and once that happens, an epoch of training is said to have been completed.[58] It usually takes several epochs of training to get a good approximation to Θ^*.

Neural networks have seen wide success in tasks such as computer vision,[64,65] natural language understanding,[66,67] and reasoning and control.[68,69] This success has mainly been driven by an increase in compute power[70] and the availability of large datasets,[71–74] which have made training deep networks possible,[75] thus starting the field that is now known as deep learning.[76] Over the years, a number of improvements to the training of deep networks have been proposed and adopted, *e.g.*, gradient descent with momentum,[77] adaptive gradient descent,[78] and regularization techniques.[79] Also, different variations to the base neural network architecture, commonly known as multi-layer perceptron (MLP) or fully connected network (FCN), have been proposed and adopted to better tackle practical problems.[80–82] While a complete review of these various neural net architectures is beyond the scope of this paper, a brief introduction to three particularly important neural architectures follows.

2.2.3.1 Convolution neural networks. Convolutional Neural Networks (CNNs)[83] use a mathematical operation called convolution in at least one of the layers. This operation can be visualized as sliding a matrix A—referred to as a filter or kernel—across an input matrix U (Fig. 5). The entries of the filter represent the learnable weights of the convolutional layer. Because the same filter is applied across the entire input, these weights are shared spatially, substantially reducing the number of trainable parameters. This weight-sharing mechanism provides an implicit form of regularization, which makes CNNs

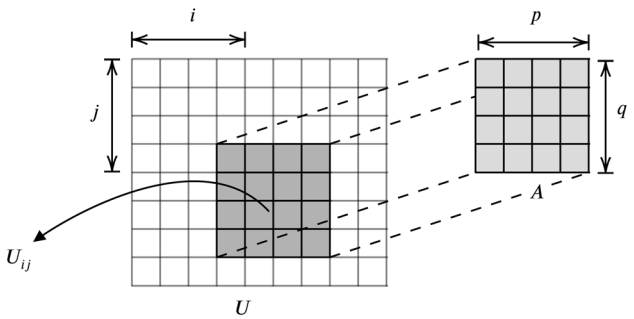

Fig. 5 Convolution visualized as sliding a filter (A) over an input (U).

‖ Hyperparameters of a neural network, such as λ, T, and γ, are often selected through trial and error or by using optimization techniques, such as Grid Search.

less prone to overfitting compared to multilayer perceptrons, where neurons are densely connected.

CNNs owe their design to the animal visual cortex,[84] and have become a foundation of modern computer vision.[85] Apart from that, CNNs, particularly 1-dimensional CNNs,[86] where $p = m$, *i.e.*, $C \in \mathbb{R}^{1\times(n-q+1)}$, have also found immense application in modeling time series data.[87] Since the CO_2-brine IFT function f is implicitly a function of time, one can create a sequence of points $\mathcal{S} = \langle(\mathbf{x}^{(0)}, y^{(0)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\rangle$ such that $(\mathbf{x}^{(t)}, y^{(t)}) \in \mathcal{D}$ represents the sample obtained at (normalized) time t. The dataset $\mathcal{D}$ casted as a time series $\mathcal{S}$ can thus be used to model f as a 1D-CNN. However, to the best of our knowledge, no prior work exists investigating this approach.

2.2.3.2 Recurrent neural networks. Recurrent neural networks (RNNs)[88] are a family of network architectures that share parameters across layers. Mathematically, RNNs are defined as the recurrence:

$$\mathbf{h}^{(t)} = \zeta(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}; \Theta),$$

where $\mathbf{h}^{(t)} \in \mathbb{R}^k$ is called the hidden state of the RNN (at time step t). Various choices are available for the function ζ; one being:

$$\zeta(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}; \Theta) = \tanh(A_{\mathrm{hx}}\mathbf{x}^{(t)} + A_{\mathrm{hh}}\mathbf{h}^{(t-1)} + \mathbf{b}_{\mathrm{h}}),$$

where $A_{\mathrm{hx}} \in \mathbb{R}^{k\times d}$, $A_{\mathrm{hh}} \in \mathbb{R}^{k\times k}$, $\mathbf{b}_{\mathrm{h}} \in \mathbb{R}^h$ make up the parameters Θ of the network. Recurrent networks are trained using backpropagation through time, which works by unrolling the compute graph through time (Fig. 6) and computing gradients using backpropagation.[89] However, backpropagation through the above choice of ζ is numerically unstable, leading to gradients exploding and/or vanishing as they flow to earlier time steps, which hinders the modeling of long-term dependencies.[90] One way to address the problem is to define ζ as the long short-term memory (LSTM) cell:[91]

$$\zeta(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}; \Theta) = \tanh\mathbf{c}^{(t)} \odot \mathbf{o}^{(t)},$$

where:

$$\mathbf{o}^{(t)} = \sigma\left(A_{\mathrm{ox}}\mathbf{x}^{(t)} + A_{\mathrm{oh}}\mathbf{h}^{(t-1)} + \mathbf{b}_{\mathrm{o}}\right),$$
$$\mathbf{c}^{(t)} = \mathbf{g}^{(t)} \odot \mathbf{i}^{(t)} + \mathbf{c}^{(t-1)} \odot \mathbf{f}^{(t)},$$

called output gate and cell state, respectively, are defined in terms of the following gates:

$$\mathbf{g}^{(t)} = \tanh\left(A_{\mathrm{gx}}\mathbf{x}^{(t)} + A_{\mathrm{gh}}\mathbf{h}^{(t-1)} + \mathbf{b}_{\mathrm{g}}\right) \quad \text{cell gate}$$
$$\mathbf{i}^{(t)} = \sigma\left(A_{\mathrm{ix}}\mathbf{x}^{(t)} + A_{\mathrm{ih}}\mathbf{h}^{(t-1)} + \mathbf{b}_{\mathrm{i}}\right) \quad \text{input gate}$$
$$\mathbf{f}^{(t)} = \sigma\left(A_{\mathrm{fx}}\mathbf{x}^{(t)} + A_{\mathrm{fh}}\mathbf{h}^{(t-1)} + \mathbf{b}_{\mathrm{f}}\right) \quad \text{forget gate}$$

Here, $A_{\mathrm{ox}}, A_{\mathrm{gx}}, A_{\mathrm{ix}}, A_{\mathrm{fx}} \in \mathbb{R}^{k\times d}$, $A_{\mathrm{oh}}, A_{\mathrm{gh}}, A_{\mathrm{ih}}, A_{\mathrm{fh}} \in \mathbb{R}^{k\times k}$, and $\mathbf{b}_{\mathrm{o}}, \mathbf{b}_{\mathrm{g}}, \mathbf{b}_{\mathrm{i}}, \mathbf{b}_{\mathrm{f}} \in \mathbb{R}^k$, define the parameters of the network, and $\sigma : \mathbb{R}^k \to (0,\ 1)^k$ is the softmax function, defined such that:

$$\sigma(\mathbf{u})_i = \frac{e^{u_i}}{\sum_{j=1}^{k} e^{u_j}}.$$

RNNs have enjoyed immense success in processing sequential data.[92,93] One can train an RNN on the sequence $\mathcal{S}$ defining the IFT data as well, for example, by formulating the loss at each time step t to be $\mathcal{L}_{\mathrm{in}}(\hat{y}^{(t)}, y^{(t)}) = \|\hat{y}^{(t)} - y^{(t)}\|_2^2$, where $\hat{y}^{(t)} = A\zeta(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}; \Theta) + b\varsigma$, for $A \in \mathbb{R}^{1\times k}$ and $b \in \mathbb{R}$, and $(\mathbf{x}^{(t)}, y^{(t)}) \in \mathcal{S}$. However, to the best of our knowledge, this line of work remains unexplored in the literature.

2.2.3.3 Transformers. Transformers[94] have largely superseded RNNs as the go-to architecture for modeling complex sequential data, especially natural language.[66,95–99] Instead of operating on a sequence one element at a time, as RNNs do, transformers operate on the whole sequence at once using a mechanism called attention.[93] At its core, an attention module is a parameterized function that takes in a sequence of inputs $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(T)}$, where $\mathbf{u}^{(t)} \in \mathbb{R}^k$, represented as rows of a matrix $U \in \mathbb{R}^{T\times k}$, and computes a weighted representation of the inputs:[67]

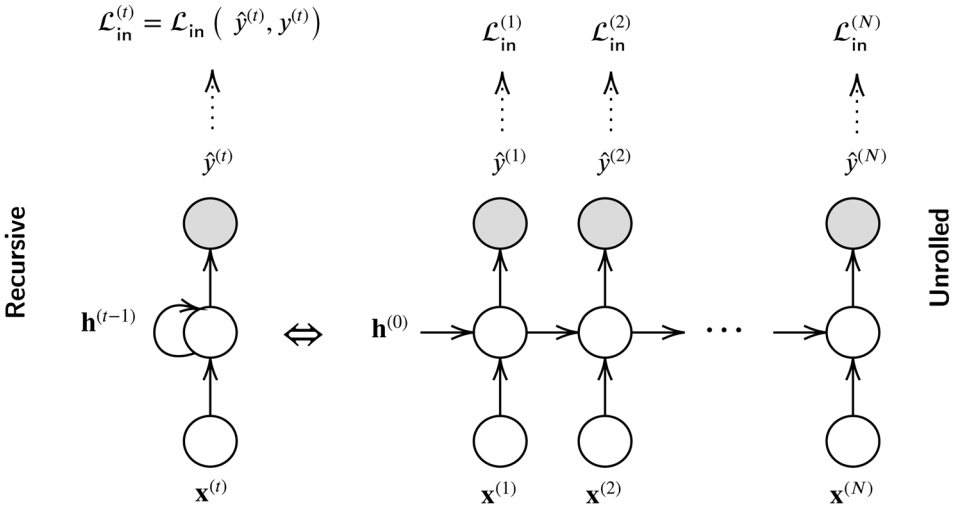

Fig. 6 A recurrent neural network in its recursive form (left) and unrolled form (right), where $\mathscr{L}_{\mathrm{in}}{}^{(t)}$ denotes the in-sample loss at the t-th time step. In practice, the initial hidden state of a RNN, $\mathbf{h}^{(0)}$, is usually set to the zero vector, and the expected loss is usually computed as the average loss over all time steps: $\frac{1}{N}\sum_{t=1}^{N}\mathscr{L}_{\mathrm{in}}{}^{(t)}$.

$$\text{Attention}(U; A_Q, A_K, A_V) = \underbrace{\sigma\left(\frac{UA_Q(UA_K)^T}{\sqrt{p}}\right)}_{\text{attention scores}} UA_V \in \mathbb{R}^{T\times k}.$$

In the literature, the rows of the matrices $Q = UA_Q \in \mathbb{R}^{T\times p}$, $K = UA_K \in \mathbb{R}^{T\times p}$ and $V = UA_V \in \mathbb{R}^{T\times k}$ are called queries, keys, and values, respectively.[80] Intuitively, the attention module produces a representation of each input $\mathbf{u}^{(i)}$ by weighing the value ($\mathbf{v}_j^T$) of each input $\mathbf{u}^{(j)}$ towards the input $\mathbf{u}^{(i)}$ according to how much the j-th input's key ($\mathbf{k}_j^T$) matches the i-th input's query ($\mathbf{q}_i^T$). This may be viewed as a mechanism for allowing the network to selectively focus on the inputs.

By doing away with recurrent processing in favor of attention modules, transformers allow for faster training through parallelization. They also support a much better gradient flow through their compute graph, thus leading to better learning of long-term dependencies.[67] However, despite the revolution that transformers have brought about in neural computing, the authors have not come across any publication featuring transformers to model CO_2-brine IFT.

3 Review of machine learning for IFT characterization

A number of machine learning algorithms have been employed in the literature to construct data-driven models of CO_2-brine IFT in saline aquifers.[100] Designed a 6-input multilayer perceptron (MLP) with tanh activations,[101] and trained it against mean-squared error *via* gradient descent over a dataset of 1202 samples. They tried networks with one and two hidden layers, with the number of neurons varying between 10 to 20 in each layer, to determine the best network topology. Table 1 summarizes the performance of their model, as well as other models from the literature, against three different standard statistical metrics: MAPE, RMSE, and R^2.** Where applicable/available, the table reports these metrics based on the different dataset splits: the training set, used to fit the model; the validation set, used as a proxy for real-world data during training; and the test set, used to evaluate the model's real-world performance. Additionally, the table includes the performance of a correlation method by ref. 102 for comparison with the machine learning approaches.

In 2017,[103] proposed a radial basis function network (RBFN) – essentially an MLP with RBF activations – for estimating IFT between CO_2 and brine in saline aquifers. Their model, which includes three input neurons for pressure, temperature, and brine salinity, and three hidden layers with 80 neurons each, outperformed the tanh-based MLP proposed by.[100] Despite being trained on a smaller dataset of 302 data points compared to the 1202 points used by,[100] the RBFN demonstrated superior performance. However, it is important to note that[103] utilized a different dataset than,[100] which complicates direct comparisons since model performance is highly dataset-dependent. We speculate that the performance gain 103 achieved over[100] is primarily from their use of a wider and deeper model, and less so from their use of the RBF activation over tanh.

Another RBFN model was proposed by[104] in the same year, and though the performance of their model appears competitive, they only report the performance metrics aggregated over the whole dataset, and not individually for the train and test datasets, making it difficult to draw a proper comparison to other works. Along with a RBFN model,[104] also proposes an adaptive neuro-fuzzy inference system (ANFIS),[121] which is a hybrid of MLPs and fuzzy inference, and is useful to model complex systems. They employed Subtractive Clustering[122] – a clustering algorithm to select representative data points (cluster centers) from the training data – to determine the membership functions in the fuzzy rule base of ANFIS. Based on the overall statistics provided in the paper, the ANFIS model performs better than the RBFN, as it builds on the strengths of both neural networks and fuzzy inference systems.

In 2017,[105] used classical machine learning, particularly, Least-Squares Support Vector Machines (LSSVM)[123] – a variant of regular Support Vector Machines (SVMs) that formulates the optimization problem as a least-squares problem, which is computationally cheaper to solve than a quadratic program – to model the interfacial tension of CO_2-brine systems. To this end, they developed and analyzed two LSSVM models – one with three inputs and the other with eight, as described in Table 1 – and optimized their hyperparameters using an algorithm called Coupled Simulated Annealing (see ref. 124 and 125).†† Though the predictive performance of these LSSVM models does not measure up to MLPs, it should be noted that LSSVMs are generally faster to train and less data-intensive than neural networks. Additionally, they are more interpretable and faster in terms of inference time.[106] also analyzed LSSVMs, along with other machine learning algorithms (namely decision trees and gene expression programming, see ref. 107), and their conclusion too remains that MLPs are more accurate than the classical techniques. However, their results show that decision trees also

** Machine learning models are typically evaluated on a test dataset, which is not used in their training, in order to obtain a reasonable estimate of the models' performance on out-of-sample data. If $(\mathbf{x}_n, y_n)$ is a point in the test dataset $\mathcal{D}_{\text{test}}$, and $\hat{y}_n = \hat{f}(\mathbf{x}_n)$ is the approximation to y_n obtained from a machine learning model $\hat{f}$, then the performance of the model according to the standard error metrics – mean absolute error (MAE), mean absolute percent error (MAPE), mean squared error (MSE), root mean squared error (RMSE), and the coefficient of determination (R^2) – is: $\text{MAE} = \frac{1}{N'}\sum_{n=1}^{N'}|y_n - \hat{y}_n|$, $\text{RMSE} = \sqrt{\frac{1}{N'}\sum_{n=1}^{N'}(y_n - \hat{y}_n)^2}$, $R^2 = 1 - \frac{\sum_{n=1}^{N'}(y_n - \hat{y}_n)^2}{\sum_{n=1}^{N'}(y_n - \bar{y})^2}$, $\text{MAPE} = \frac{100}{N'}\sum_{n=1}^{N'}\frac{|y_n - \hat{y}_n|}{y_n}$, $\text{MSE} = \frac{1}{N'}\sum_{n=1}^{N'}(y_n - \hat{y}_n)^2$. Here, $N' = |\mathcal{D}_{\text{test}}|$. It is important to note that not all studies reviewed reported their models' performance on a test set.

†† The optimization algorithms used to train and fine-tune the models (such as CSA, PSO, FFA, *etc.*) are discussed in the referenced studies. Given the wide range of available techniques, an exhaustive review is beyond the scope of this paper. Moreover, these algorithms are typically bundled with popular optimization and machine learning software packages, making them easy to deploy for model optimization through simple API calls.

Table 1 A comparative summary of the literature on ML based characterization of CO_2-brine IFT

Reference	Year	Dataset size	Inputs	Method	Split	MAPE	RMSE	R^2
Li *et al.*[102]	2014	1716 samples	Pressure Temperature N_2 molarity CH_4 molarity Na^+, K^+ molality Ca^{2+}, Mg^{2+} molality	Correlation	n.a.	7.81%	4.51	0.857
Zhang *et al.*[100]	2016	1716 samples	Pressure Temperature	Multi-layer perceptron	Train	2.58%	1.47	0.985
		70% train	N_2 molarity		Val	2.52%	1.34	0.985
		15% val	CH_4 molarity					
		15% test	Na^+, K^+ molality Ca^{2+}, Mg^{2+} molality		Test	3.39%	2.04	0.970
Niroomand-Toomaj *et al.*[103]	2017	378 samples	Pressure	Radial basis function network	Train	2.10%	1.139	0.994
		80% train	Temperature		Val	n.a.	n.a.	n.a.
		20% test	Brine salinity		Test	3.36%	1.567	0.986
Partovi *et al.*[104]	2017	1716 samples	Pressure Temperature N_2 molarity	Radial basis function network w/particle swarm optimization (PSO)	Whole	2.07%	1.400	0.986
		80% train	CH_4 molarity	Adaptive neuro fuzzy inference system w/ subtractive clustring	Whole	1.96%	1.240	0.989
		20% test	Na^+, K^+ molality Ca^{2+}, Mg^{2+} molality					
Rashid *et al.*[105]	2017	1019 samples	Pressure	Least squares support vector machine w/ coupled simulated annealing (CSA)	Train	2.08%	1.354	0.986
			Temperature		Val	4.03%	2.472	0.950
			Salinity		Test	3.54%	2.425	0.941
		75% train	Pressure	Least squares support vector machine w/ coupled simulated annealing (CSA)	Train	2.20%	n.a.	0.986
		75% val	Temperature		Val	4.03%	n.a.	0.958
		20% test	Na^+, K^+ concentration Ca^{2+}, Mg^{2+} concentration SO_4^{2-} concentration HCO^{3-} concentration		Test	3.21%	n.a.	0.964
Kamari *et al.*[106]	2017	1716 samples	Pressure Temperature	Decision trees	Whole	3.15%	n.a.	0.978
		80% train	N_2 molarity	Least squares support vector machine	Whole	3.75%	n.a.	0.972
		20% test	CH_4 molarity Na^+, K^+ molality Ca^{2+}, Mg^{2+} molality	Gene Expression Programming[107]	Whole	9.30%	n.a.	0.640
Dehaghani Saeedi *et al.*[108]	2019	378 samples	Pressure	Stochastic gradient boosting	Train	0.30%	0.179	1.000
		80% train	Temperature		Test	1.64%	0.947	0.995
		20% test	Salinity					
Amooie *et al.*[13]	2019	2517 samples	Pressure	Least squares support vector machine w/ CSA	Train	5.13%	2.51	n.a.
					Test	5.32%	2.71	n.a.
			Temperature	Radial basis function network w/PSO	Train	6.19%	2.85	n.a.
					Test	6.68%	3.38	n.a.
		80% train	Na^+, K^+ molality	Multi-layer perceptron w/Levenberg–Marquardt	Train	3.02%	1.66	n.a.
					Test	3.79%	2.09	n.a.

Table 1 (*Contd.*)

Reference	Year	Dataset size	Inputs	Method	Split	MAPE	RMSE	R^2
			Ca^{2+}, Mg^{2+} molality	Multi-layer perceptron w/Bayesian regularization	Train	3.16%	1.72	n.a.
					Test	3.87%	2.09	n.a.
		20% test	Critical temperature of gaseous mixture	Multi-layer perceptron w/scaled conjugate gradient	Train	3.56%	1.83	n.a.
					Test	4.66%	2.41	n.a.
				Multi-layer perceptron w/resilient backpropagation	Train	5.00%	2.30	n.a.
					Test	5.17%	2.61	n.a.
				Committee machine intelligent system	Train	2.99%	1.69	n.a.
					Test	3.35%	1.75	n.a.
				Group method of data handling	Train	8.53%	3.81	n.a.
					Test	8.32%	3.80	n.a.
Zhang *et al.*[14]	2020	2346 samples[a]	Pressure Temperature	Gradient boosted trees w/shrinkage strategy & column subsampling	Train	0.32%	0.04	1
		80% train	N_2 molarity CH_4 molarity		Test	1.71%	0.95	0.993
		20% test	Na^+, K^+ molality Ca^{2+}, Mg^{2+} molality					
Zhang *et al.*[109]	2020	1716 samples	Pressure	Gaussian process regression	Train	0.21%	0.14	1
					Test	7.87%	4.06	0.881
			Temperature	Multi-layer perceptron	Train	3.65%	1.92	0.973
					Test	4.48%	2.22	0.967
		80% train	N_2 molarity	Support vector machine	Train	1.63%	0.91	0.994
					Test	4.16%	2.24	0.967
			CH_4 molarity	Ridge regression w/radial basis function	Train	4.63%	2.38	0.959
					Test	4.41%	2.38	0.967
		20% test	Na^+, K^+ molality	Decision trees	Train	0.05%	0.1	1.000
					Test	4.77%	2.73	0.951
			Ca^{2+}, Mg^{2+} molality	Random forest	Train	1.63%	0.92	0.994
					Test	4.17%	2.26	0.967
				Adaptive boosting (Adaboost)	Train	2.19%	1.06	0.992
					Test	4.43%	2.28	0.966
				Gradient boosted decision tree	Train	1.29%	0.72	0.996
					Test	2.61%	1.4	0.987
				Gradient boosting (XGBoost)	Train	0.56%	0.31	0.999
					Test	2.37%	1.28	0.989
Hosseini *et al.*[110]	2020	1716 samples	Temperature	Multi-layer perceptron w/Levenberg–Marquardt	Train	1.23%	0.527	0.991
					Val	2.53%	1.057	0.971
		80% train	Pressure		Test	1.55%	0.786	0.986
		20% test	Salinity	Multi-layer perceptron w/Bayesian regularization	Train	1.15%	0.483	0.994
					Val	1.70%	0.578	0.983
					Test	1.73%	0.680	0.983
			Brine density	Multi-layer perceptron w/scaled conjugate gradient	Train	1.15%	0.479	0.993
					Val	1.91%	1.832	0.971
					Test	2.03%	0.859	0.983
			CO_2 density	Radial basis function w/differential evolution	Train	1.11%	0.500	0.991
					Val	1.40%	0.598	0.991
					Test	1.06%	0.516	0.995
				Radial basis function w/partical swarm optimization	Train	0.94%	0.478	0.993
					Val	1.73%	0.610	0.989

Table 1 (*Contd.*)

Reference	Year	Dataset size	Inputs	Method	Split	MAPE	RMSE	R^2
					Test	1.35%	0.511	0.988
				Radial basis function w/farmland fertility algorithm (FFA)	Train	1.12%	0.543	0.991
					Val	0.96%	0.455	0.991
					Test	1.10%	0.544	0.993
Liu *et al.*[19]	2021	974 samples	Pressure	Multi-layer perceptron w/back propagation	Train	n.a.	1.799	0.981
			Temperature		Test	n.a.	1.954	0.926
		70% train	N_2 molarity	Wavelet neural network (WNN)	Train	n.a.	3.549	0.993
		30% test	CH_4 molarity		Test	n.a.	3.837	0.976
			Na^+, K^+ molality	Radial basis function	Train	n.a.	1.112	0.908
			Ca^{2+}, Mg^{2+}		Test	n.a.	76.04	35.2
			Molality	Optimized wavelet neural network (I-WNN)	Train	n.a.	2.689	0.958
					Test	n.a.	2.782	0.952
Amar *et al.*[111]	2021	2346 samples[a]	Pressure	Genetic programming for temperature ≤ 313.15 K	Train	n.a.	4.806	0.943
			Temperature		Test	n.a.	4.643	0.935
		80% train	N_2 molarity	Genetic programming for temperature > 313.15 K	Train	n.a.	2.193	0.960
		20% test	CH_4 molarity		Test	n.a.	2.052	0.959
			Na^+, K^+ molality	Average of both models	Train	n.a.	3.500	0.952
			Ca^{2+}, Mg^{2+} molality		Test	n.a.	3.348	0.947
Safaei-Farouji *et al.*[112]	2022	2184 samples	Pressure	Random forest	Train	1.54%	0.590	0.995
			Temperature		Test	2.99%	1.108	0.980
		80% train	N_2 molarity	Gaussian process regression	Train	1.56%	0.557	0.994
		20% test	CH_4 molarity		Test	3.64%	1.282	0.970
			Na^+, K^+ molality	Radial basis function	Train	5.05%	1.828	0.951
			Ca^{2+}, Mg^{2+} molality		Test	5.01%	1.894	0.946
Mouallem *et al.*[113]	2024	2896 samples	Pressure	Least-squares boosting	Train	1.95%	1.009	0.994
			Temperature		Test	4.82%	1.009	0.941
		80% train	Salinity of monovalentsalts (NaCl, KCl, Na_2HCO_3, Na_2SO_4)	Extreme gradient boosting	Train	1.84%	1.115	0.992
		20% test			Test	3.77%	2.532	0.961
			Salinty of bivalent salts ($MgCl_2$, $CaCl_2$, $MgSO_4$)	Gradient boosting	Train	0.93%	0.719	0.997
					Test	3.38%	2.434	0.964
			Impurities CH_4 and N_2	Genetic programming	Train	9.12%	4.583	0.871
					Test	9.02%	4.284	0.886
				Artificial neural network	Train	5.86%	3.124	0.939
					Test	8.99%	3.124	0.921
				Gaussian process regression	Train	6.63%	3.416	0.929
					Test	6.78%	3.416	0.925
Vakili-Nezhaad *et al.*[114]	2024	549 samples	Pressure	Deep neural network w/Group method of data handling (GMDH)	Train	1.3%	n.a.	0.99
		80% train	Temperature		Test	2.95%	n.a.	0.97
		20% test						
Mutailipu *et al.*[115]	2024	1717 samples	Pressure	Particle swarm optimization random forest (PSO-RF)	Whole	2.68%	1.916	0.967
			Temperature					
		80% train	CH_4 molarity	Improved gray wolf optimization random forest (IGWO-RF)	Whole	2.70%	1.900	0.969
		10% val	N_2 molarity					
		10% test	Bulk carbon molecules	Sparrow search algorithm random forest (SSA-RF)	Whole	2.67%	1.900	0.969
			Microporous carbon					
			Molecules	Bayesian optimization random forest (BO-RF)	Whole	2.07%	1.770	0.973

Table 1 (*Contd.*)

Reference	Year	Dataset size	Inputs	Method	Split	MAPE	RMSE	R^2
Shen *et al.*[116]	2024	1716 samples	Pressure Temperature	Extreme gradient boosting (XGBoost)	Test	2.31%	1.24	0.989
		80% train	CH_4 molarity	Light gradient boosting machine	Test	2.45%	1.32	0.987
		20% test	N_2 molarity Monovalent molality	(LightGBM)				
			Bivalent molality Density differences	Ensemble learning	Test	2.01%	1.11	0.991
Nsiah Turkson *et al.*[117]	2024	1570 samples	Pressure	Gradient boosting (GradBoost)	Train	0.20%	0.126	1.000
			Temperature		Val	2.51%	1.400	0.986
		70% train	CH_4 molarity		Test	2.23%	1.227	0.990
		15% val	N_2 molarity	Light gradient boosting machine	Train	0.91%	0.504	0.998
		15% test	Monovalent molality	(LightGBM)	Val	3.00%	1.979	0.971
			Bivalent molality		Test	2.66%	1.650	0.982
Li *et al.*[118]	2024	1823 samples	Pressure Temperature	Grey wolf optimizer-back propagation neural network	Whole	3.38%	1.682	0.971
		80% train	CH_4 molarity	Dung beetle optimizer-back propagation	Whole	3.35%	1.678	0.972
		20% test	N_2 molarity Monovalent molality	neural network Particle swarm optimization – back	Whole	3.61%	1.899	0.962
			Bivalent molality	propagation neural network (PSO-BPNN)				
Fan *et al.*[119]	2025	1716 samples	Pressure	Multibranch structure convolutional	Whole	1.34%	1.06	0.992
			Temperature	neural network (MBCNN)	Train	1.05%	0.89	0.994
		80% train			Test	2.49%	1.54	0.982
		20% test	CH_4 molarity	Random forest regressor (RFR)	Whole	2.09%	1.26	0.983
					Train	1.63%	0.92	0.994
			N_2 molarity		Test	3.93%	2.13	0.967
			Monovalent molality	Gene expression programming (GEP)	Whole	13.4%	6.54	0.699
					Train	13.5%	6.62	0.695
					Test	13.0%	6.22	0.715
			Bivalent molality	Support vector regressor (SVR)	Whole	13.6%	7.56	0.597
					Train	13.6%	7.59	0.598
					Test	13.5%	7.45	0.590
Liaqat *et al.*[120]	2025	1254 samples	Pressure	Linear regression (LR)	Train	3.88%	2.03	0.94
			Temperature		Test	4.25%	2.22	0.94
		80% train	NaCl molality	Support vector machine (SVM)	Train	0.71%	0.42	0.99
		20% test			Test	0.97%	0.57	0.99
				Decision tree regressor (DTR)	Train	0.28%	0.20	1.00
					Test	1.56%	0.85	0.99
				Random forest regressor (RFR)	Train	0.57%	0.30	1.00
					Test	1.16%	0.62	0.99
				Multilayer perceptron (MLP)	Train	0.73%	0.40	1.00
					Test	0.99%	0.52	0.99

[a] They started with 2517 samples but reduced the sample space down to 2346 samples have removing inconsistent sample points.

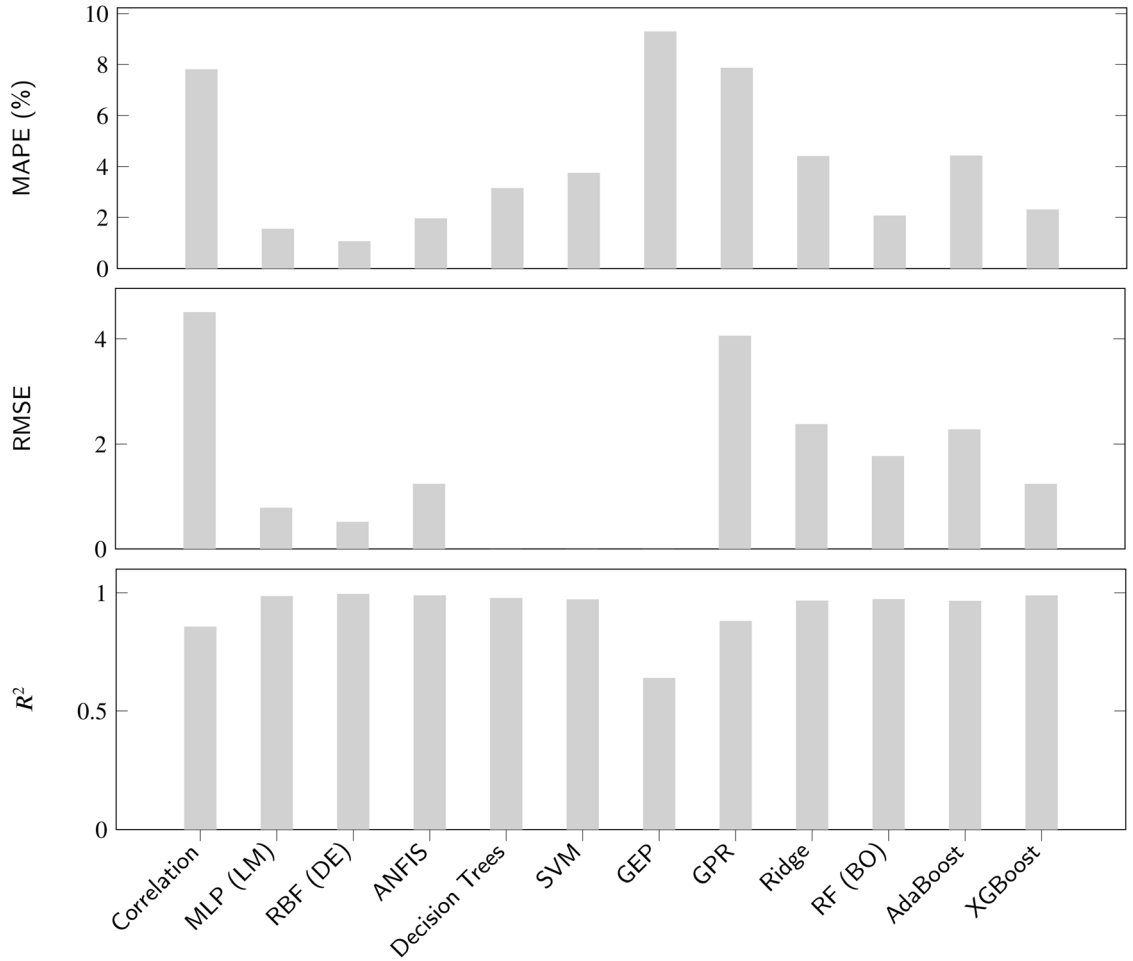

Fig. 7 A comparison of methods reported in the literature. For consistency, only studies based on the 1716-sample dataset are included. Reported metrics correspond to test sets where available, or to the full dataset otherwise. Differences in feature sets, data processing, and optimization schemes contribute to some variability. Full details are provided in Table 1 and the accompanying text.

perform admirably, and as shown in the work of,[108] decision trees in an ensemble can even outpace MLPs. In particular,[108] used an ensemble of 2707 decision trees constructed using Stochastic Gradient Boosting,[126] where they used 302 data points in the training process, in contrast to the 1372 training data points used by ref. 106.

Ref. 13, in 2019, conducted an extensive study on seven different machine learning models. They developed eight machine learning models in total: LSSVMs optimized with CSA; RBFNs optimized with PSO; MLPs with two hidden layers and sigmoid/tanh activations, optimized using Levenberg–Marquardt (LM),[127] Bayesian Regularization (BR),[128,129] Scaled Conjugate Gradient (SCG),[130] and Resilient Backpropagation (RB)[131] algorithms; and models based on Group Method of Data Handling (GMDH).[132,133] GMDH is a self-organizing neural network that optimizes both structural and parametric aspects of the model. Each of these models were designed to take in the same set of five inputs – namely, pressure, temperature, molalities of Na^+ and K^+, molalities of Ca^{2+} and Mg^{2+}, and the critical temperature of the mixture – and they were all trained on the same dataset of 2013 data samples. Based on the statistical performance reported in the paper, the authors ranked the models as follows: MLP-LM > MLP-BR > MLP-SCG > MLP-RB > LSSVM-CSA > RBF-PSO > GMDH. These results show that MLPs have the best predictive performance of all the models tested. Moreover,[13] proposes a committee machine intelligence system (CMIS) – an ensemble that weights the top three performing models, namely, the MLPs optimized with LM, BR, and SCG algorithms. The ensemble aggregates the performance of the three MLPs and generally outdoes the individual MLPs. Later,[14] took the same dataset as,[13] and cleaned out any inconsistent data entries. They then trained an XGBoost model,[134] an ensemble of gradient-boosted decision trees optimized for scalability and efficiency on large datasets, on the cleaned dataset, with hyperparameters optimized using 5-fold cross-validation (see ref. 135) with exhaustive grid search. In line with the findings of[108] in regards to gradient-boosted decision tree ensembles,[14] achieved remarkably low statistical error with their predictions, outperforming MLPs (Fig. 7).

In 2020, the works of ref. 110 and 109 sought to draw a detailed statistical picture of how various machine learning algorithms perform on the task of modeling the CO_2-brine IFT.

The former work compares MLPs and RBFNs optimized using various methods, and the latter compares a whole range of techniques, including MLPs, SVMs, ridge regression with RBF kernels (RR-RBF), decision trees, random forests, adaptive boosting, Gaussian process regression (GPR),[136,137] and gradient-boosted trees. Results from ref. 110 were obtained against a dataset of 91 points only; however, they show that RBFNs optimized with either PSO, Differential Evolution (DE),[138] or the Farmland Fertility Algorithm (FFA)[139] perform better than MLPs optimized with LM, BR, and SCG. And results from ref. 109 show that MLPs come second only to gradient-boosted trees. Later in 2021,[19] again conducted a comparative study of neural network models, including MLPs, RBFNs, and Wavelet Neural Networks (WNNs). They concluded that MLPs with sigmoid activations perform the best, and MLPs with wavelet activations (WNNs) perform the worst. The same year,[111] proposed another classical learning technique for the problem – namely, genetic programming (GP).[140] To that end, they divided the dataset into two subsets: one with data points where the temperature was less than or equal to 313.15 K, and the other with data points where the temperature was greater than 313.15 K. After creating the two subsets, they trained a separate model on each subset using genetic programming. However, as with most other non-ensemble classical techniques, the performance of their GP models does not stack up to the performance that neural networks with sigmoid activations have been shown to achieve. Another work comparing different machine learning methods for IFT was published in 2022 by ref. 112. They analyzed random forests (RF), GPR, and RBFNs, and reached the conclusion that random forests perform the best.

The application of AI to IFT prediction gained significant traction in 2024, reflected in the publication of six research papers: ref. 113, 114, 115, 116, 117 and 118. Ref. 113 uses multiple machine learning algorithms for IFT estimation, including Gradient Boosting, Extreme Gradient Boosting, Least Squares Boosting, Artificial Neural Networks, and Genetic Programming. Like most previous studies,[113] employed six input features: pressure, temperature, the salinity of both monovalent (NaCl, KCl, Na_2HCO_3, Na_2SO_4) and bivalent salts ($MgCl_2$, $CaCl_2$, $MgSO_4$), and the presence of impurities such as CH_4 and N_2. Among their models, the Gradient Boosting approach demonstrated the lowest MAPE (3.38%) for testing data, outperforming other models, whereas the Genetic Programming model exhibited the poorest performance. The ANN model achieved a relatively high MAPE of 8.99%, which is significantly higher compared to similar studies available in the literature. The discrepancy could be due to differences in the underlying dataset or a poor choice of hyperparameters. As a practical application of their models, the paper uses predicted IFT to determine the optimal storage depth for a real carbonate saline aquifer located onshore in the UAE.

Ref. 114 proposed a novel deep learning-based approach to estimate the IFT, specifically focusing on solutions containing divalent salts ($MgCl_2$ and $CaCl_2$), where GMDH was used to model IFT. The proposed GMDH-based model yielded a MAPE of 2.95% for test data, demonstrating high accuracy. A key advantage of the GMDH approach is its ability to optimize the network structure automatically, thus requiring less hyperparameter tuning.

Ref. 115 used an RF model coupled with a Bayesian Optimization algorithm (BO-RF) to predict IFT. The BO-RF model was compared against three other RF models, which were optimized using Sparrow Search Algorithm (SSA-RF), Particle Swarm Optimization (PSO-RF), and Improved Grey Wolf Optimization (IGWO-RF), respectively. Among these, the BO-RF model demonstrated the best performance, achieving a MAPE of 2.07% when evaluated on the entire dataset. The predicted IFT values were then utilized to determine the CO_2 sequestration capacity of saline aquifers in the Tarim Basin of Xinjiang, China.

Ref. 116 introduced heterogeneous ensemble learning to predict IFT by combining XGBoost and Light Gradient Boosting Machine (LightGBM). The performance of the ensemble learning model was compared to the individual performances of XGBoost and LightGBM. The results showed that the ensemble learning model achieved a lower MAPE of 2.01%, compared to 2.31% for XGBoost and 2.45% for LightGBM.[117] also investigated the use of Gradient Boosting and LightGBM with a slightly smaller dataset compared to ref. 116. The gradient boosting model achieved the best performance, reporting an error of 2.23% on the test data. While the gradient boosting model in this study outperformed the individual models of Shen *et al.*, it still underperformed compared to the ensemble learning model proposed by ref. 116.

Ref. 118 introduced a dung beetle optimization-based backpropagation neural network (DBO-BPNN) for IFT modeling. The model's performance was compared to particle swarm optimization-based BPNN (PSO-BPNN) and grey wolf optimizer-based BPNN (GWO-BPNN). DBO-BPNN achieved the best accuracy, with an error of 3.35% on the whole dataset, outperforming PSO-BPNN, which had the next best performance with an error of 3.61%. However, despite its improved accuracy, DBO-BPNN has higher computational complexity and requires a larger dataset to perform optimally, making it less suitable for IFT applications. Moreover, previous studies have demonstrated that less complex models can achieve even better results.

So far in 2025, at the time of writing, we have identified two research papers published this year on AI-driven IFT modeling.[119] introduced a multibranch convolutional neural network (MBCNN) for predicting CO_2-brine IFT across varying temperature and pressure conditions. Unlike conventional single-branch machine learning models such as Random Forests and Support Vector Regression, their proposed MBCNN architecture integrates multiple convolutional layers and fully connected layers to capture inter-attribute relationships. While the MBCNN achieved a MAPE of 2.49%, outperforming the RF, GEP (Gene Expression Programming), and SVR models used for comparison in this study, previous research has demonstrated that simpler models can achieve similar performance under comparable conditions. For example,[109] reported a MAPE of 2.37% using XGBoost, while[116] achieved 2.31% with XGBoost and 2.01% with an ensemble learning approach combining XGBoost and LightGBM.

Deviating from the oft-used set of six input features, the work by[120] in 2025 utilized three input features – temperature, pressure, and NaCl salinity – to predict IFT. The study explored a range of models, from simple linear regression to more complex architectures such as Multilayer Perceptron (MLP), striking a balance between accuracy and interpretability. Among the five models evaluated, Support Vector Machine (SVM) and MLP performed the best, achieving MAPE values of 0.97% and 0.99%, respectively, on the test data. These findings demonstrated that even relatively simple ML models with good data processing and hyperparameter tuning could accurately predict IFT, outperforming several complex models examined in previous studies.

From Table 1, several trends can be noticed. Among all ML algorithms surveyed, gradient-boosting consistently achieves the highest performance metric, such as high R^2 scores, often outperforming more complex architectures such as deep neural networks in this domain. Support vector machines also show competitive performance, where they sometimes match or exceed the accuracy of gradient boosting. Among the models evaluated, gradient boosting variants (*e.g.*, XGBoost, LightGBM) consistently show minimal signs of overfitting, with train and test metrics remaining very close in terms of R^2, MAPE, and RMSE. In contrast, models such as Gaussian Process Regression and Decision Trees tend to exhibit larger discrepancies between training and testing performance, reflecting susceptibility to overfitting. Neural network–based models also show signs of overfitting in some cases.

Most studies, as seen in Table 1, focus on common input variables such as pressure, temperature, and bulk salinity. However, several important conditions remain underexplored. High-salinity brines rich in divalent ions (Ca^{2+}, Mg^{2+}, SO_4^{2-}), which are typical of deep saline aquifers, are only sparsely represented. Similarly, datasets covering extreme pressures and temperatures relevant to supercritical CO_2 storage are limited, reducing model generalizability. While some studies have included impurities such as CH_4 and N_2, other common impurities (*e.g.*, H_2S, O_2) are rarely considered. Furthermore, most models rely solely on fluid-phase properties, leaving out potentially important features related to rock–fluid interactions, such as mineral composition and wettability.

4 Feature selection and understanding IFT from ML modeling

The ML modeling approach primarily focuses on the accuracy of prediction, and these models (*e.g.*, neural networks) are often considered "black-box" when it comes to understanding and interpreting IFT behavior. However, IFT datasets, whether derived from laboratory measurements or molecular simulations, are subject to uncertainties stemming from measurement noise, instrument limitations, operator variability, and idealized modeling assumptions. These uncertainties can propagate through ML models, potentially leading to misleading predictions if not accounted for or understood. For geologic sequestration applications, where IFT predictions may influence large-scale storage design and risk assessment, it is crucial for domain experts to comprehend not only the predicted values but also the underlying physical relationships between IFT and controlling parameters such as temperature, pressure, and brine composition. This need for physical interpretability motivates the use of model explanation techniques. To this end, various methods for model interpretation or for analyzing the influence of input features on the predicted IFT have been studied in existing research. Some of the key methods used for investigating the impact of different parameters on IFT include ML-based techniques such as Feature Importance Analysis and Shapley Values, and statistical methods such as the Akaike Information Criterion and Pearson coefficients.[13,80,118,120]

Feature analysis not only provides interpretability by offering insights into both the model and the underlying physical process, but it also improves predictive performance by guiding feature selection. Well-chosen features eliminate redundancy and reduce the influence of irrelevant or highly correlated variables, which can otherwise introduce noise and degrade a model's generalization performance. Moreover, high-dimensional input spaces exacerbate variance and increase the risk of overfitting, particularly in data-constrained settings.

In practice, the sensitivity of a model to feature selection depends on its underlying learning mechanism. For example, Linear Regression and Least Squares models are especially vulnerable to irrelevant or collinear features, making manual feature screening and dimensionality reduction techniques such as Principal Component Analysis critical for stable and accurate predictions. SVMs are also sensitive to feature quality, as irrelevant or noisy features dilute the kernel similarity measure and reduce the model's ability to identify meaningful decision boundaries. RBFNs are particularly susceptible to the "curse of dimensionality," since their performance depends on distance-based similarity; irrelevant or redundant features can therefore severely impair accuracy unless carefully pruned. Models based on non-linear architectures such as tree ensembles (*e.g.*, Random Forests and Gradient Boosting) and deep neural networks are generally more robust to feature redundancy, as they can implicitly down-weight or ignore uninformative inputs. Nevertheless, even for these models, careful feature selection can improve accuracy, accelerate training, and mitigate overfitting—particularly when the dataset size is limited.

We use the most commonly used input features for IFT prediction to analyze and understand their influence. First, we plot the trend analysis of IFT with respect to each of the input features: pressure, temperature, monovalent cation molality, and bivalent cation molality. The dataset used for this analysis is obtained from the study by Li *et al.*[118] Fig. 8 shows the trend analysis for each input feature. The IFT appears to decrease with increasing pressure. Regarding temperature, IFT increases until approximately 100 °C, after which it begins to decline. For the cations, while the overall trend suggests a direct relationship with increasing IFT, there is significant variation in IFT values for some cation concentrations. Bivalent cations show a more pronounced nonlinear effect at higher concentrations. The

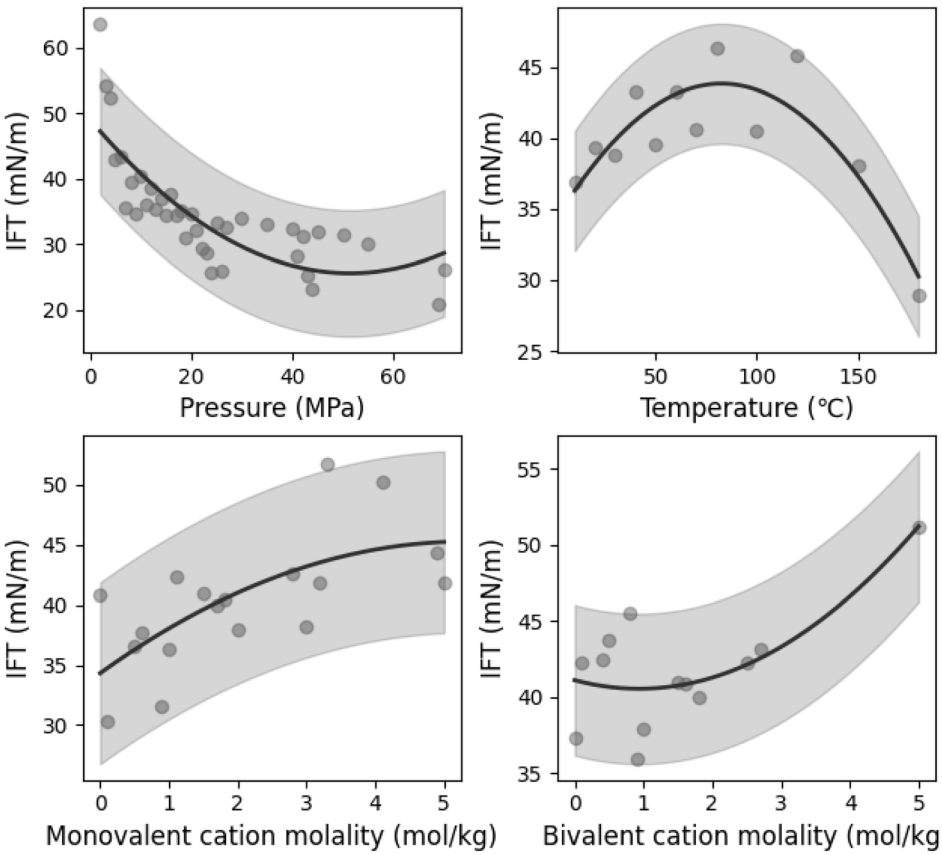

Fig. 8 IFT trend analysis with respect to different input features. A confidence interval (95%) around the trend line is also shown to indicate the spread of data.

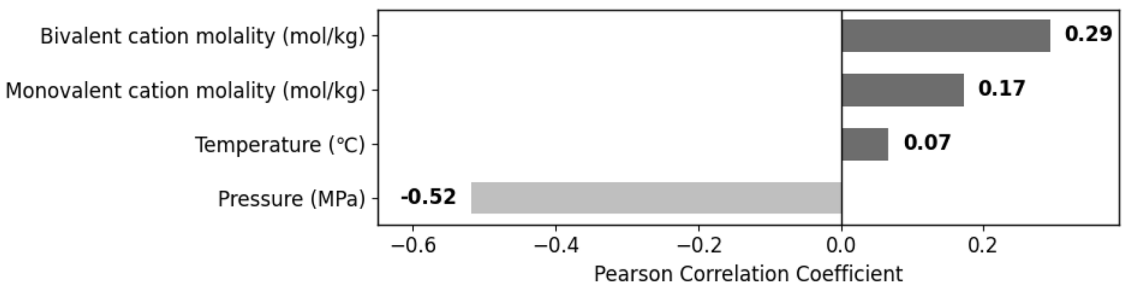

Fig. 9 Pearson correlation coefficients for feature importance analysis.

increase in IFT for monovalent cations tends to plateau, while for bivalent cations, it accelerates beyond 2 mol kg^{-1}.

Fig. 9 presents the Pearson correlation coefficients for the same set of input features. This analysis indicates that pressure is the most influential feature affecting the ML model's predictions, followed by cation molality, with bivalent cations having a more dominant impact. Temperature has the least effect on IFT. The feature importance rankings observed here align with the findings reported in the literature using methods discussed earlier.[13,80,118,120] While commonly used trend analysis and feature importance methods provide some insight into the underlying physics, the interpretability of ML-based IFT modeling remains a challenge. This will be explored in greater depth in the next section.

5 Challenges, critique and future directions

The literature demonstrates that, similar to other fields, data-driven modeling is an effective approach for characterizing IFT. The studies reviewed in Section 3 show that ML models can reliably predict IFT across different scenarios by leveraging diverse input features. However, several challenges and limitations persist in applying ML to IFT prediction, and many critical

questions remain unanswered. This section provides an in-depth discussion on these challenges and explores potential future directions for improving ML-based IFT modeling.

5.1 Data limitations

The accuracy and reliability of ML-based IFT prediction depend significantly on the availability of large, high-quality datasets. Achieving generalizability requires diverse and extensive data, yet a key challenge in IFT modeling is the limited and often incomplete nature of experimental datasets. Many existing datasets lack sufficient variation, which can lead to overfitting, where models perform well on training data but fail to generalize to new conditions.

There is also a lack of a standardized, publicly available, high-quality, and expansive data set for interfacial tension that ML models can reliably be trained on. It's crucial to recognize that machine learning methods, particularly neural networks, are highly dependent on the quality and quantity of data available. We believe that efforts to collect a comprehensive, high-quality IFT dataset – at least for saline aquifers, but ideally covering multiple underground rock formations to leverage common patterns‡‡ – would significantly advance the adoption of modern machine learning techniques for modeling interfacial tension. As shown in Table 1, the available datasets are limited in size. For studies aimed at developing and testing novel ML models for IFT prediction, we recommend adopting the most widely used dataset, as employed in studies,[100,102,104,106] as a baseline for standardized model performance comparison, and subsequently extending the analysis using larger datasets when available.

5.2 Model complexity

The landscape of ML models for IFT prediction encompasses a broad spectrum, ranging from simpler statistical approaches to highly complex architectures. This variety raises a fundamental question: which model is best suited for IFT prediction? The choice of an optimal model depends on several factors, including model complexity, data availability, computational efficiency, and the trade-off between accuracy and interpretability. Given that existing IFT datasets are relatively small and lack stochastic variability, simpler models may not only suffice but could also outperform complex architectures. Unlike deep learning models, simpler approaches are less prone to overfitting in data-limited scenarios and provide greater interpretability—an important consideration in IFT modeling. Furthermore, as shown in Table 1, a comparison of different models suggests that simpler architectures often generalize better on unseen testing data, whereas more complex models tend to overfit due to the lack of diverse and extensive datasets. This underscores the need for carefully balancing model complexity with dataset size to ensure both accuracy and robustness in ML-based IFT predictions. It is important to note, however, that "simple" ML models are not simplistic. Even the less complex architectures are capable of capturing intricate, nonlinear interactions among multiple input features—going beyond what traditional single-parameter correlations can offer. Thus, ML methods retain their superiority by learning richer representations of the underlying physics, even when model complexity is deliberately constrained.

While simpler models have demonstrated strong generalization in data-limited scenarios, the potential of modern deep learning architectures for IFT prediction remains unexplored. If large, high-quality datasets with sufficient variability become available, more complex models—such as CNNs, RNNs, and transformers—could offer state-of-the-art performance, particularly when framing CO_2-brine IFT prediction as a time-series modeling task. Given the increasing complexity and scale of CO_2 storage projects, there is an urgent need to explore advanced architectures capable of capturing long-range dependencies and intricate parameter interactions in IFT prediction. Transformer-based models, originally developed for natural language processing, have demonstrated promising performance in diverse sequence modeling domains. Leveraging transformers with data availability and computational resources could accelerate predictive capabilities and reduce uncertainty in large-scale sequestration planning. Furthermore, modern techniques to aid the training of deep neural networks also remain unexplored. For example, ReLU activations have shown great promise in improving the performance of deep networks in a variety of tasks, yet, to the best of our knowledge, they remain unappreciated for the task of CO_2-brine IFT modeling. Similarly, transfer learning,[141–143] also known as domain adaptation, has proven revolutionary towards several applications concerning deep learning,[66,144–147] and one can employ transfer learning for the problem at hand, too, where one would pre-train a network to model the general IFT function, and then adapt that network to model the IFT of CO_2-brine systems. However, to date, there has been no work on this approach.

5.3 Hybrid models

In scientific applications of ML, it is crucial to ensure that models adhere to fundamental physical principles to prevent unrealistic predictions. Integrating domain-specific physics into data-driven models enhances both their accuracy and reliability. A prominent method in this context is the use of Physics-Informed Neural Networks (PINNs),[148] which embed physical laws, typically represented by partial differential equations, into the neural network's loss function. This approach constrains the model's outputs to align with known physical behavior, thereby improving generalization, interpretability, and trust. PINNs have demonstrated significant success across various engineering and physics domains. Despite the extensive application of purely data-driven ML models in predicting the CO_2-brine IFT, the integration of physics-informed approaches remains underexplored in this area. Incorporating physical constraints specific to IFT phenomena into ML models could enhance their predictive performance and ensure

‡‡ One can make rock type an input feature, or one can employ techniques such as transfer learning to allow for physics-induced common data patterns across rock formations to be learned by a neural network and improve its prediction.

consistency with established physical laws. Future research should prioritize the development of hybrid models that seamlessly combine physics-based information with advanced ML techniques.

5.4 Standardization of evaluation methods

There is a need to adopt a standardized notation in the literature: when different studies use different symbols or terms for the same concepts, it makes it harder to compare and integrate findings from the various sources. But more critically, we feel that there is a need to adopt a systematic methodology to test and evaluate different machine learning models. Currently, in most of the literature, the datasets used for evaluation between the various publications are not the same, making it difficult to compare the models proposed in these publications against each other. Moreover, variations in training datasets further contribute to uncertainty, making it challenging to discern whether observed performance improvements are due to the models themselves or simply a result of using more or higher-quality data. Additionally, most publications compare their models against other data-based models only and lack a comparison of how these models fare against analytical and numerical models. Adding such comparisons in any future work would prove helpful.

5.5 Practical relevance for geologic storage

While most ML-based IFT models have been developed and tested on laboratory-scale datasets, their potential implications for large-scale carbon capture and storage projects are significant. Accurate and computationally efficient prediction of CO_2-brine IFT can directly inform reservoir simulation workflows, wellbore integrity assessments, and leakage risk analysis. In particular, ML models can provide rapid sensitivity analyses under varying pressure, temperature, and salinity conditions, allowing engineers to explore a wider range of operational scenarios than would be practical with experiments or molecular simulations alone. Moreover, integrating ML-based IFT prediction into existing carbon capture and storage decision-support tools could improve estimates of capillary trapping capacity and residual saturation, thereby reducing uncertainty in storage efficiency forecasts. The ability to retrain models with site-specific data also offers adaptability for different geologic settings, enabling more tailored project designs. Ultimately, bridging the gap between laboratory ML models and field-scale carbon storage engineering will be essential to ensure that predictive accuracy translates into safe, reliable, and cost-effective CO_2 storage.

6 Conclusions

The data-driven approach to modeling CO_2-brine IFT in saline aquifers offers a cost-effective alternative to traditional methods. Numerous studies have demonstrated the feasibility of this approach using a range of ML models, from simple to advanced architectures. In this work, we provided a comprehensive review of the existing literature and critically examined the opportunities and challenges associated with this data-driven approach.

Due to variations in training and evaluation datasets, as well as a lack of information on computational efficiency—such as time and memory footprint—we refrain from making definitive claims about the best model in the reviewed literature. However, based on predictive performance metrics, it seems reasonable to suggest that simpler ML models, such as gradient-boosted decision trees and support vector machines, may be the most accurate and practical for estimating CO_2-brine IFT in saline aquifers. While previous studies have explored advanced and complex neural network architectures, the currently available datasets appear to be a limiting factor, preventing these models from achieving more robust performance, thus giving an advantage to simpler ML approaches.

Nonetheless, we emphasize that multilayer perceptrons (MLPs) warrant further evaluation to fully assess their potential, as they have demonstrated state-of-the-art performance in similar tasks, often surpassing classical methods like decision trees. We believe that the MLPs reviewed in this study may have been constrained by their size, and that deeper MLP architectures trained on larger datasets could potentially yield even better results.

Future work could extend this review by conducting a meta-analysis on a substantially larger, standardized CO_2-brine IFT dataset collected across diverse saline aquifer conditions. This dataset will enable the benchmarking of more advanced architectures such as transformers and physics-informed neural networks, which may capture complex, nonlinear relationships beyond the capabilities of current models. Lastly, we propose that efforts should be made by the research community to make source codes and datasets openly accessible. This would facilitate the practical adoption of the proposed methods and provide a foundation upon which future research can be more easily built.

Conflicts of interest

There are no conflicts to declare.

Data availability

Data sharing is not applicable to this article, as no new data was created or analyzed in this study.

Acknowledgements

The authors would like to acknowledge financial support from the National Science Foundation (NSF) under grant No. CBET-2223078.

References

1 on Climate Change IP, Global warming of 1.5 °C: An IPCC special report on the impacts of global warming of 1.5 °C above pre-industrial levels and related global greenhouse gas emission pathways, in *The Context of Strengthening the*

Global Response to the Threat of Climate Change, Sustainable Development, and Efforts to Eradicate Poverty, Intergovernmental Panel on Climate Change, 2018.

2 D. W. Duncan and E. A. Morrissey, *The Concept of Geologic Carbon Sequestration*, US Department of the Interior, US Geological Survey, 2011.

3 Agency IE. CO2 Capture and Storage: A Key Carbon Abatement Option; 2008, Available from: **https://www.oecd-ilibrary.org/content/publication/9789264041417-en**.

4 C. R. Jenkins, P. J. Cook, J. Ennis-King, J. Undershultz, C. Boreham, T. Dance, *et al.*, Safe storage and effective monitoring of CO2 in depleted gas fields, *Proc. Natl. Acad. Sci. U. S. A.*, 2012, **109**(2), E35–E41.

5 F. M. Orr, Carbon Capture, Utilization, and Storage: An Update, *SPE J.*, 2018, **23**(06), 2444–2455.

6 H. Chen, X. Liu, C. Zhang, X. Tan, R. Yang, S. Yang, *et al.*, Effects of miscible degree and pore scale on seepage characteristics of unconventional reservoirs fluids due to supercritical CO2 injection, *Energy*, 2022, **239**, 122287.

7 X. Liu, H. Chen, W. Cheng, Y. Li, Y. Zhao, Y. Zhu, *et al.*, Occurrence states and transport behavior of crude oil in different permeability oil reservoirs during depletion development, *Geoenergy Sci. Eng.*, 2025, **252**, 213944.

8 X. Liu, H. Chen, Y. Li, Y. Zhu, H. Liao, Q. Zhao, *et al.*, Oil production characteristics and CO2 storage mechanisms of CO2 flooding in ultra-low permeability sandstone oil reservoirs, *Pet. Explor. Dev.*, 2025, **52**(1), 196–207.

9 B. Metz, O. Davidson, H. De Coninck, M. Loos and L. Meyer, *IPCC Special Report on Carbon Dioxide Capture and Storage*, Cambridge University Press, Cambridge, 2005.

10 M. A. Celia, S. Bachu, J. M. Nordbotten and K. W. Bandilla, Status of CO2 storage in deep saline aquifers with emphasis on modeling approaches and practical simulations, *Water Resour. Res.*, 2015, **51**(9), 6846–6892.

11 M. A. De Figueiredo, *The Liability of Carbon Dioxide Shortage*, Massachusetts Institute of Technology, 2007.

12 K. Damen, A. Faaij and W. Turkenburg, Health, safety and environmental risks of underground CO2 storage–overview of mechanisms and current knowledge, *Clim. Change*, 2006, **74**(1–3), 289–318.

13 M. A. Amooie, A. Hemmati-Sarapardeh, K. Karan, M. M. Husein, M. R. Soltanian and B. Dabir, Data-driven modeling of interfacial tension in impure CO2-brine systems with implications for geological carbon storage, *Int. J. Greenh. Gas Control*, 2019, **90**, 102811. Available from: **https://www.sciencedirect.com/science/article/pii/S1750583618305371**.

14 J. Zhang, Q. Feng, X. Zhang, C. Shu, S. Wang and K. Wu, A supervised learning approach for accurate modeling of CO2–brine interfacial tension with application in identifying the optimum sequestration depth in saline aquifers, *Energy Fuels*, 2020, **34**(6), 7353–7362.

15 C. A. Aggelopoulos, M. Robin and O. Vizika, Interfacial tension between CO2 and brine (NaCl+CaCl2) at elevated pressures and temperatures: The additive effect of different salts, *Adv. Water Resour.*, 2011, **34**(4), 505–511.

16 W. B. Tucker, *Surfact Tension by Pendant Drops*, Massachusetts Institute of Technology, 1938.

17 B. Song and J. Springer, Determination of Interfacial Tension from the Profile of a Pendant Drop Using Computer-Aided Image Processing: 2. Experimental, *J. Colloid Interface Sci.*, 1996, **184**(1), 77–91. Available from: **https://www.sciencedirect.com/science/article/pii/S0021979796905986**.

18 T. W. Richards and E. K. Carver, A critical study of the capillary rise method of determining surface tension, with data for water, benzene, toluene, chloroform, carbon tetrachloride, ether and dimethyl aniline, *J. Am. Chem. Soc.*, 1921, **43**(4), 827–847, DOI: **10.1021/ja01437a012**.

19 X. Liu, M. Mutailipu, J. Zhao and Y. Liu, Comparative Analysis of Four Neural Network Models on the Estimation of CO2–Brine Interfacial Tension, *ACS Omega*, 2021, **6**(6), 4282–4288.

20 S. Iglauer, M. S. Mathew and F. Bresme, Molecular dynamics computations of brine–CO2 interfacial tensions and brine–CO2–quartz contact angles and their effects on structural and residual trapping mechanisms in carbon geo-sequestration, *J. Colloid Interface Sci.*, 2012, **386**(1), 405–414.

21 S. Wang, Q. Feng, F. Javadpour, M. Zha and R. Cui, Multiscale Modeling of Gas Transport in Shale Matrix: An Integrated Study of Molecular Dynamics and Rigid-Pore-Network Model, *SPE J.*, 2020, **25**(03), 1416–1442. Available from: **https://onepetro.org/SJ/article-pdf/25/03/1416/2341431/spe-187286-pa.pdf**.

22 S. Wang, Q. Feng, F. Javadpour, Q. Hu and K. Wu, Competitive adsorption of methane and ethane in montmorillonite nanopores of shale at supercritical conditions: A grand canonical Monte Carlo simulation study, *Chem. Eng. J.*, 2019, **355**, 76–90. Available from: **https://www.sciencedirect.com/science/article/pii/S1385894718315419**.

23 X. Wang, C. Chen, K. Binder, U. Kuhn, U. Pöschl, H. Su, *et al.*, Molecular dynamics simulation of the surface tension of aqueous sodium chloride: from dilute to highly supersaturated solutions and molten salt, *Atmos. Chem. Phys.*, 2018, **18**(23), 17077–17086.

24 A. Ghoufi, P. Malfreyt and D. J. Tildesley, Computer modelling of the surface tension of the gas–liquid and liquid–liquid interface, *Chem. Soc. Rev.*, 2016, **45**, 1387–1409, DOI: **10.1039/C5CS00736D**.

25 H. Zhang, H. V. Thanh, M. Rahimi, W. J. Al-Mudhafar, S. Tangparitkul, T. Zhang, *et al.*, Improving predictions of shale wettability using advanced machine learning techniques and nature-inspired methods: Implications for carbon capture utilization and storage, *Sci. Total Environ.*, 2023, **877**, 162944.

26 H. Vo Thanh, Q. Yasin, W. J. Al-Mudhafar and K. K. Lee, Knowledge-based machine learning techniques for accurate prediction of CO2 storage performance in underground saline aquifers, *Appl. Energy*, 2022, **314**, 118985.

27 M. A. A. Al-qaness, A. A. Ewees, H. V. Thanh, A. M. AlRassas and M. Abd Elaziz, An optimized neuro-fuzzy system using advance nature-inspired Aquila and Salp swarm algorithms for smart predictive residual and solubility carbon trapping efficiency in underground storage formations, *J. Energy Storage*, 2022, **56**, 106150.
28 S. Davoodi, H. Vo Thanh, D. A. Wood, M. Mehrad, V. S. Rukavishnikov and Z. Dai, Machine-learning predictions of solubility and residual trapping indexes of carbon dioxide from global geological storage sites, *Expert Syst. Appl.*, 2023, **222**, 119796. Available from: https://www.sciencedirect.com/science/article/pii/S095741742300297X.
29 H. B. Abdulkhaleq, I. K. Ibraheem, W. J. Al-Mudhafar, Z. T. Mohammed and M. S. Abd, Harnessing the power of machine learning for the optimization of CO2 sequestration in saline aquifers: Applied on the tensleep formation at teapot dome in Wyoming, *Geoenergy Sci. Eng.*, 2025, **245**, 213522.
30 W. J. Al-Mudhafar and D. A. Wood, Development and simulation of the gas-downhole water sink-assisted gravity drainage (GDWS-AGD) process to reduce carbon footprint and improve clean oil production by injecting CO2 and petroleum-associated gas, *J. Clean. Prod.*, 2024, **464**, 142792.
31 X. Li, E. Boek, G. C. Maitland and J. M. Trusler, Interfacial Tension of (Brines+ CO2):(0.864 NaCl+ 0.136 KCl) at Temperatures between (298 and 448) K, Pressures between (2 and 50) MPa, and Total Molalities of (1 to 5) mol· kg–1, *J. Chem. Eng. Data*, 2012, **57**(4), 1078–1088.
32 X. Li, E. S. Boek, G. C. Maitland and J. M. Trusler, Interfacial Tension of (Brines+ CO2): CaCl2 (aq), MgCl2 (aq), and Na2SO4 (aq) at Temperatures between (343 and 423) K, Pressures between (2 and 50) MPa, and Molalities of (0.5 to 5) mol· kg–1, *J. Chem. Eng. Data*, 2012, **57**(5), 1369–1375.
33 Y. S. Abu-Mostafa, M. Magdon-Ismail and H. T. Lin, *Learning from Data*, AMLBook, New York, 2012, vol. 4.
34 Y. Liu, H. A. Li and R. Okuno, Measurements and modeling of interfacial tension for CO2/CH4/brine systems under reservoir conditions, *Ind. Eng. Chem. Res.*, 2016, **55**(48), 12358–12375.
35 L. M. Pereira, A. Chapoy, R. Burgass and B. Tohidi, Interfacial tension of CO2+ brine systems: Experiments and predictive modelling, *Adv. Water Resour.*, 2017, **103**, 64–75.
36 L. Ang, L. Yongming, C. Xi, Z. Zhongyi and P. Yu, Review of CO2 sequestration mechanism in saline aquifers, *Nat. Gas Ind. B*, 2022, **9**(4), 383–393.
37 G. R. Jerauld and A. Kazemi, An improved simple correlation for accurate estimation of CO2-Brine interfacial tension at reservoir conditions, *J. Petrol. Sci. Eng.*, 2022, **208**, 109537.
38 T. Hastie, R. Tibshirani, J. H. Friedman and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009, vol. 2.
39 R. Eberhart and J. Kennedy, Particle swarm optimization, in *Proceedings of the IEEE International Conference on Neural Networks*, Citeseer, 1995, vol. 4, pp. 1942–1948.
40 T. Hastie, Ridge regularization: An essential concept in data science, *Technometrics*, 2020, **62**(4), 426–433.
41 S. Geman, E. Bienenstock and R. Doursat, Neural networks and the bias/variance dilemma, *Neural Comput.*, 1992, **4**(1), 1–58.
42 X. Ying, An overview of overfitting and its solutions, *J. Phys.: Conf. Ser.*, 2019, **1168**, 022022.
43 J. Gareth, W. Daniela, H. Trevor and T. Robert, *An Introduction to Statistical Learning: with Applications in R*, Spinger, 2013.
44 C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*, Springer, 2006, vol. 4.
45 V. Vapnik, *The Nature of Statistical Learning Theory*, Springer science & business media, 1999.
46 A. J. Smola and B. Schölkopf, A tutorial on support vector regression, *Stat. Comput.*, 2004, **14**, 199–222.
47 C. J. Burges, A tutorial on support vector machines for pattern recognition, *Data Min. Knowl. Discov.*, 1998, **2**(2), 121–167.
48 J. Shawe-Taylor, N. Cristianini, *et al.*, *Kernel Methods for Pattern Analysis*, Cambridge university press, 2004.
49 M. Mohri, A. Rostamizadeh and A. Talwalkar, *Foundations of Machine Learning*, MIT press, 2018.
50 C. Campbell, *An Introduction to Kernel Methods*, 2001.
51 O. Z. Maimon and L. Rokach, *Data Mining with Decision Trees: Theory and Applications*, World scientific, 2014, vol. 81.
52 J. R. Quinlan, Induction of decision trees, *Mach. Learn.*, 1986, **1**, 81–106.
53 L. Hyafil and R. L. Rivest, Constructing optimal binary decision trees is NP-complete, *Inf. Process. Lett.*, 1976, **5**(1), 15–17. Available from: https://www.sciencedirect.com/science/article/pii/0020019076900958.
54 J. Mendes-Moreira, C. Soares, A. M. Jorge and J. F. D. Sousa, Ensemble approaches for regression: A survey, *ACM Comput. Surv.*, 2012, **45**(1), 1–40.
55 B. Hanin and D. Rolnick, Deep relu networks have surprisingly few activation patterns, *Adv. Neural Inf. Process. Syst.*, 2019, **32**, 361–370.
56 W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys.*, 1943, **5**, 115–133.
57 T. Szandała, Review and comparison of commonly used activation functions for deep neural networks, *Bio-inspired neurocomputing*, 2021, pp. 203–224.
58 I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016, http://www.deeplearningbook.org.
59 K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximators, *Neural Netw.*, 1989, **2**(5), 359–366.
60 P. Baldi, Gradient descent learning algorithm overview: A general dynamical systems perspective, *IEEE Trans. Neural Network.*, 1995, **6**(1), 182–195.
61 X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the Thirteenth International Conference on Artificial*

Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
62 D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning representations by back-propagating errors, *nature*, 1986, **323**(6088), 533–536.
63 L. eon Bottou, Online learning and stochastic approximations, *Online Learn. Neural Netw.*, 1998, **17**(9), 142.
64 O. Ronneberger, P. Fischer and T. Brox, U-net: Convolutional networks for biomedical image segmentation, in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, Springer, 2015, pp. 234–241.
65 K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, *arXiv*, 2014, preprint, arXiv:1409.1556, DOI: **10.48550/arXiv.1409.1556**.
66 J. Devlin, M. Chang, K. Lee and K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, CoRR, *arXiv*, 2018, preprint, arXiv:1810.04805, DOI: **10.48550/arXiv.1810.04805**.
67 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez *et al.*, Attention is All you Need, in *Advances in Neural Information Processing Systems*, ed. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan et al., Curran Associates, Inc., 2017, vol. 30, **https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf**.
68 J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, *et al.*, Highly accurate protein structure prediction with AlphaFold, *Nature*, 2021, **596**(7873), 583–589.
69 D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, *et al.*, Mastering the game of Go with deep neural networks and tree search, *nature*, 2016, **529**(7587), 484–489.
70 Y. LeCun, 1.1 deep learning hardware: past, present, and future, in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2019, pp. 12–19.
71 A. Krizhevsky, G. Hinton, *et al.*, *Learning Multiple Layers of Features from Tiny Images*, 2009.
72 J. Deng, W. Dong, R. Socher, L. J. Li, K. Li and L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Ieee, 2009, pp. 248–255.
73 T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, *et al.*, Microsoft coco: Common objects in context, in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755.
74 M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, *et al.*, The cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.
75 I. Daubechies, R. DeVore, S. Foucart, B. Hanin and G. Petrova, Nonlinear approximation and (deep) ReLU networks, *Constr. Approx.*, 2022, **55**(1), 127–172.
76 R. Dechter, *Learning while Searching in Constraint-Satisfaction Problems*, 1986.
77 B. T. Polyak, Some methods of speeding up the convergence of iteration methods, *USSR Comput. Math. Math. Phys.*, 1964, **4**(5), 1–17.
78 J. Duchi, E. Hazan and Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, *J. Mach. Learn. Res.*, 2011, **12**(7), 2121–2159.
79 N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *J. Mach. Learn. Res.*, 2014, **15**(56), 1929–1958. Available from: **http://jmlr.org/papers/v15/srivastava14a.html**.
80 A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, Dive into Deep Learning, *arXiv*, 2021, preprint, arXiv:2106.11342, DOI: **10.48550/arXiv.2106.11342**.
81 M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, *et al.*, The history began from alexnet: A comprehensive survey on deep learning approaches, *arXiv*, 2018, preprint, arXiv:1803.01164, DOI: **10.48550/arXiv.1803.01164**.
82 M. Saud Ul Hassan, K. Liaqat, L. Schaefer and A. J. Zolan, Modern deep neural networks for Direct Normal Irradiance forecasting: A classification approach. e-Prime - Advances in Electrical Engineering, *Electron. Energy*, 2024, **10**, 100853.
83 A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Commun. ACM*, 2017, **60**(6), 84–90.
84 K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biol. Cybern.*, 1980, **36**(4), 193–202.
85 C. Farabet, C. Couprie, L. Najman and Y. LeCun, Learning hierarchical features for scene labeling, *IEEE Trans. Pattern Anal. Mach. Intell.*, 2012, **35**(8), 1915–1929.
86 A. Waibel, Modular construction of time-delay neural networks for speech recognition, *Neural Comput.*, 1989, **1**(1), 39–46.
87 H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar and P. A. Muller, Deep learning for time series classification: a review, *Data Min. Knowl. Discov.*, 2019, **33**(4), 917–963.
88 D. E. Rumelhart, P. Smolensky, J. L. McClelland and G. Hinton, Sequential thought processes in PDP models, *Parallel distributed processing: explorations in the microstructures of cognition*, 1986, vol. 2, pp. 3–57.
89 J. L. Elman, Finding structure in time, *Cogn. Sci.*, 1990, **14**(2), 179–211.
90 Y. Bengio, P. Simard and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Trans. Neural Network.*, 1994, **5**(2), 157–166.
91 S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.*, 1997, **9**(8), 1735–1780.

92 A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke and J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, *IEEE Trans. Pattern Anal. Mach. Intell.*, 2008, **31**(5), 855–868.
93 D. Bahdanau, K. Cho and Y. Bengio, Neural machine translation by jointly learning to align and translate, *arXiv*, 2014, preprint, arXiv:1409.0473, DOI: **10.48550/arXiv.1409.0473**.
94 M. S. Ul Hassan, K. Liaqat and L. Schaefer, Adarmer: An Adaptive Transformer for Direct Normal Irradiance Forecasting, in *2024 International Conference on Machine Learning and Applications (ICMLA)*, 2024, pp. 222–229.
95 K. Clark, M. T. Luong, Q. V. Le and C. D. Manning, Electra: Pre-training text encoders as discriminators rather than generators, *arXiv*, 2020, preprint, arXiv:2003.10555, DOI: **10.48550/arXiv.2003.10555**.
96 Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, *et al.*, Roberta: A robustly optimized bert pretraining approach, *arXiv*, 2019, preprint, arXiv:1907.11692, DOI: **10.48550/arXiv.1907.11692**.
97 I. Beltagy, M. E. Peters and A. Cohan, Longformer: The long-document transformer, *arXiv*, 2020, preprint, arXiv:2004.05150, DOI: **10.48550/arXiv.2004.05150**.
98 T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, *et al.*, Language models are few-shot learners, *Adv. Neural Inf. Process. Syst.*, 2020, **33**, 1877–1901.
99 A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, Language models are unsupervised multitask learners, *OpenAI blog*, 2019, **1**(8), 9.
100 J. Zhang, Q. Feng, S. Wang, X. Zhang and S. Wang, Estimation of CO2–brine interfacial tension using an artificial neural network, *J. Supercrit. Fluids*, 2016, **107**, 31–37.
101 Y. LeCun, L. Bottou, G. B. Orr and K. R. Müller, Efficient backprop, in *Neural Networks: Tricks of the Trade*, Springer, 2002, pp. 9–50.
102 Z. Li, S. Wang, S. Li, W. Liu, B. Li and Q. C. Lv, Accurate Determination of the CO2–Brine Interfacial Tension Using Graphical Alternating Conditional Expectation, *Energy Fuels*, 2014, **28**(1), 624–635, DOI: **10.1021/ef401815q**.
103 E. Niroomand-Toomaj, A. Etemadi and A. Shokrollahi, Radial basis function modeling approach to prognosticate the interfacial tension CO2/Aquifer Brine, *J. Mol. Liq.*, 2017, **238**, 540–544. Available from: **https://www.sciencedirect.com/science/article/pii/S016773221633286X**.
104 M. Partovi, M. Mosalanezhad, S. Lotfi, A. Barati-Harooni, A. Najafi-Marghmaleki and A. H. Mohammadi, On the estimation of CO2-brine interfacial tension, *J. Mol. Liq.*, 2017, **243**, 265–272.
105 S. Rashid, B. Harimi and E. Hamidpour, Prediction of CO2-Brine interfacial tension using a rigorous approach, *J. Nat. Gas Sci. Eng.*, 2017, **45**, 108–117.
106 A. Kamari, M. Pournik, A. Rostami, A. Amirlatifi and A. H. Mohammadi, Characterizing the CO2-brine interfacial tension (IFT) using robust modeling approaches: A comparative study, *J. Mol. Liq.*, 2017, **246**, 32–38.
107 C. Ferreira, Gene expression programming: a new adaptive algorithm for solving problems, *arXiv*, 2001, preprint, arXiv:cs/0102027, DOI: **10.48550/arXiv.cs/0102027**.
108 S. A. H. Dehaghani and R. Soleimani, Estimation of interfacial tension for geological CO2 storage, *Chem. Eng. Technol.*, 2019, **42**(3), 680–689.
109 J. Zhang, Q. Feng and X. Zhang, The use of machine learning methods for fast estimation of CO2-brine interfacial tension: A comparative study, in *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, 2020, pp. 1–5.
110 A. H. Hosseini, H. Ghadery-Fahliyany, D. Wood and A. Choubineh, Artificial Intelligence-based Modeling of Interfacial Tension for Carbon Dioxide Storage, *Gas Process. J.*, 2020, **8**(1), 83–92. Available from: **https://gpj.ui.ac.ir/article_24625.html**.
111 M. N. Amar, Towards improved genetic programming based-correlations for predicting the interfacial tension of the systems pure/impure CO2-brine, *J. Taiwan Inst. Chem. Eng.*, 2021, **127**, 186–196.
112 M. Safaei-Farouji, H. Vo Thanh, D. Sheini Dashtgoli, Q. Yasin, A. E. Radwan, U. Ashraf, *et al.*, Application of robust intelligent schemes for accurate modelling interfacial tension of CO2 brine systems: Implications for structural CO2 trapping, *Fuel*, 2022, **319**, 123821. Available from: **https://www.sciencedirect.com/science/article/pii/S0016236122006834**.
113 J. Mouallem, A. Raza, G. Glatz, M. Mahmoud and M. Arif, Estimation of CO2-Brine interfacial tension using Machine Learning: Implications for CO2 geo-storage, *J. Mol. Liq.*, 2024, **393**, 123672. Available from: **https://www.sciencedirect.com/science/article/pii/S0167732223024790**.
114 G. R. Vakili-Nezhaad, R. Yousefzadeh, A. Kazemi, A. A. Shaaili and A. Al Ajmi, Application of deep learning through group method of data handling for interfacial tension prediction in brine/CO2 systems: MgCl2 and CaCl2 aqueous solutions, *Int. J. Greenh. Gas Control*, 2024, **135**, 104147. Available from: **https://www.sciencedirect.com/science/article/pii/S1750583624000902**.
115 M. Mutailipu, Y. Yang, K. Zuo, Q. Xue, Q. Wang, F. Xue, *et al.*, Estimation of CO2-Brine Interfacial Tension Based on an Advanced Intelligent Algorithm Model: Application for Carbon Saline Aquifer Sequestration, *ACS Omega*, 2024, **9**(35), 37265–37277, DOI: **10.1021/acsomega.4c04888**.
116 B. Shen, S. Yang, J. Hu, Y. Gao, H. Xu, X. Gao, *et al.*, Application of Heterogeneous Ensemble Learning for CO2–Brine Interfacial Tension Prediction: Implications for CO2 Storage, *Energy Fuels*, 2024, **38**(5), 4401–4416, DOI: **10.1021/acs.energyfuels.3c05092**.
117 J. Nsiah Turkson, M. Aslam, M. Yusof, I. Fjelde, Y. Adams Sokama-Neuyam, V. Darkwah-Owusu and B. Nii Tackie-Otoo, *Harnessing Ensemble Learning Techniques for Accurate Interfacial Tension Estimation in Aqueous CO2*

Systems, GOTECH, Dubai, UAE, May 2024, DOI: 10.2118/219176-MS.

118 J. Q. Li, X. Q. Bian, J. Chen, Y. B. Liu and A. Matthews, Improved neural network model based on dung beetle algorithm to predict CO2-brine interfacial tension, *Geoenergy Sci. Eng.*, 2024, **239**, 212957.

119 J. Fan, Y. Jiang, Z. Fan, C. Yang, K. He and D. Wang, Enhanced Prediction of CO2–Brine Interfacial Tension at Varying Temperature Using a Multibranch-Structure-Based Neural Network Approach, *Langmuir*, 2025, **41**(3), 1587–1600, DOI: 10.1021/acs.langmuir.4c03366.

120 K. Liaqat, D. J. Preston and L. Schaefer, Predicting the interfacial tension of CO2 and NaCl aqueous solution with machine learning, *Sci. Rep.*, 2025, **15**, 1–16.

121 D. Karaboga and E. Kaya, Adaptive network based fuzzy inference system (ANFIS) training approaches: a comprehensive survey, *Artif. Intell. Rev.*, 2019, **52**, 2263–2293.

122 U. M. Rao, Y. R. Sood and R. K. Jarial, Subtractive clustering fuzzy expert system for engineering applications, *Procedia Comput. Sci.*, 2015, **48**, 77–83.

123 J. A. Suykens and J. Vandewalle, Least squares support vector machine classifiers, *Neural Process. Lett.*, 1999, **9**, 293–300.

124 S. Xavier-de Souza, J. A. K. Suykens, J. Vandewalle and D. Bolle, Coupled Simulated Annealing, *IEEE Trans. Syst. Man Cybern. B Cybern.*, 2010, **40**(2), 320–335.

125 V. Fabian, Simulated annealing simulated, *Comput. Math. Appl.*, 1997, **33**(1–2), 81–94.

126 J. H. Friedman, Stochastic gradient boosting, *Comput. Stat. Data Anal.*, 2002, **38**(4), 367–378.

127 Ö. Kisi, Multi-layer perceptrons with Levenberg-Marquardt training algorithm for suspended sediment concentration prediction and estimation/Prévision et estimation de la concentration en matières en suspension avec des perceptrons multi-couches et l'algorithme d'apprentissage de Levenberg-Marquardt, *Hydrol. Sci. J.*, 2004, **49**(6), 1040.

128 D. J. MacKay, Bayesian interpolation, *Neural Comput.*, 1992, **4**(3), 415–447.

129 F. D. Foresee and M. T. Hagan, Gauss-Newton approximation to Bayesian learning, in *Proceedings of International Conference on Neural Networks (ICNN'97)*, IEEE, 1997, vol. 3, pp. 1930–1935.

130 M. F. Møller, A scaled conjugate gradient algorithm for fast supervised learning, *Neural Netw.*, 1993, **6**(4), 525–533.

131 M. Riedmiller and H. Braun, A direct adaptive method for faster backpropagation learning: The RPROP algorithm, in *IEEE International Conference on Neural Networks*, IEEE, 1993, pp. 586–591.

132 A. G. Ivakhnenko, Polynomial theory of complex systems, *IEEE Trans. Syst. Man Cybern.*, 1971, (4), 364–378.

133 A. Ivakhnenko and G. Ivakhnenko, The review of problems solvable by algorithms of the group method of data handling (GMDH), *Pattern Recogn. Image Anal.*, 1995, **5**, 527–535.

134 T. Chen and C. Guestrin, Xgboost: A scalable tree boosting system, in *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

135 S. Raschka, Model evaluation, model selection, and algorithm selection in machine learning, *arXiv*, 2018, preprint, arXiv:1811.12808, DOI: 10.48550/arXiv.1811.12808.

136 E. Schulz, M. Speekenbrink and A. Krause, A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions, *J. Math. Psychol.*, 2018, **85**, 1–16.

137 M. Kanagawa, P. Hennig, D. Sejdinovic and B. K. Sriperumbudur, Gaussian processes and kernel methods: A review on connections and equivalences, *arXiv*, 2018, preprint, arXiv:1807.02582, DOI: 10.48550/arXiv.1807.02582.

138 K. Fleetwood, An introduction to differential evolution, in *Proceedings of Mathematics and Statistics of Complex Systems (MASCOS) One Day Symposium, 26th November*, Brisbane, Australia, 2004, pp. 785–791.

139 H. Shayanfar and F. S. Gharehchopogh, Farmland fertility: A new metaheuristic algorithm for solving continuous optimization problems, *Appl. Soft Comput.*, 2018, **71**, 728–746.

140 J. R. Koza, Genetic programming as a means for programming computers by natural selection, *Stat. Comput.*, 1994, **4**, 87–112.

141 S. J. Pan and Q. Yang, A survey on transfer learning, *IEEE Trans. Knowl. Data Eng.*, 2010, **22**(10), 1345–1359.

142 D. Hendrycks, X. Liu, E. Wallace, A. Dziedzic, R. Krishnan and D. Song. Pretrained transformers improve out-of-distribution robustness, *arXiv*, 2020, preprint, arXiv:2004.06100, DOI: 10.48550/arXiv.2004.06100.

143 A. Tamkin, T. Singh, D. Giovanardi and N. Goodman, Investigating transferability in pretrained language models, *arXiv*, 2020, preprint, arXiv:2004.14975, DOI: 10.48550/arXiv.2004.14975.

144 Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov and Q. V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, *Adv. Neural Inf. Process. Syst.*, 2019, **32**, 5753–5763.

145 C. Sun, A. Shrivastava, S. Singh and A. Gupta, Revisiting unreasonable effectiveness of data in deep learning era, in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 843–852.

146 D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, *et al.*, Exploring the limits of weakly supervised pretraining, in *Proceedings of the European Conference on Computer Vision*, ECCV, 2018, pp. 181–196.

147 P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, *et al.*, Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning, *arXiv*, 2017, preprint, arXiv:1711.05225, DOI: 10.48550/arXiv.1711.05225.

148 S. Ganga and Z. Uddin, Exploring Physics-Informed Neural Networks: From Fundamentals to Applications in Complex Systems, *arXiv*, 2024, preprint, arXiv:2410.00422, DOI: 10.48550/arXiv.2410.00422.