IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 January 2025, accepted 12 February 2025, date of publication 19 February 2025, date of current version 28 February 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3543475

RESEARCH ARTICLE

A Computational Intelligence Framework Integrating Data Augmentation and Meta-Heuristic Optimization Algorithms for Enhanced Hybrid Nanofluid Density Prediction Through Machine and Deep Learning Paradigms

PRIYA MATHUR[1], HAMMAD SHAIKH[2], FARHAN SHETH[2], DHEERAJ KUMAR[2], AND AMIT KUMAR GUPTA[2]

[1]Poornima Institute of Engineering and Technology, Jaipur, Rajasthan 302022, India
[2]Department of Computer Science and Engineering, Manipal University Jaipur, Jaipur, Rajasthan 303007, India

Corresponding author: Amit Kumar Gupta (amit.gupta@jaipur.manipal.edu)

This work was supported by the Manipal University Jaipur, Jaipur, Rajasthan, India.

ABSTRACT This research presents a robust and comprehensive framework for predicting the density of hybrid nanofluids using state-of-the-art machine learning and deep learning techniques. Addressing the limitations of conventional empirical approaches, the study used a curated dataset of 436 samples from the peer-reviewed literature, which includes nine input parameters such as the nanoparticle, base fluid, temperature (°C), volume concentration (ϕ), base fluid density (ρ_{bf}), density of primary and secondary nanoparticles (ρ_{np1} and ρ_{np2}), and volume mixture ratios of primary and secondary nanoparticles. Data preprocessing involved outlier removal via the Interquartile Range (IQR) method, followed by augmentation using either autoencoder-based or Gaussian noise injection, which preserved statistical integrity and enhanced dataset diversity. The research analyzed fourteen predictive models, employing advanced hyperparameter optimization methods facilitated by Grey Wolf Optimization (GWO) and Particle Swarm Optimization (PSO). In particular, autoencoder-based augmentation combined with hyperparameter optimization consistently improved predictive accuracy across all models. For machine learning models, Gradient Boosting achieved the most remarkable performance, with R^2 scores of 0.99999 and minimal MSE values of 0.00091. Among deep learning models, Recurrent Neural Networks (RNN) stacked with Linear Regression achieved superior performance with an R^2 of 0.9999, MSE of 0.0014, and MAE of 0.012. The findings underscore the synergy of advanced data augmentation, meta-heuristic optimization, and modern predictive algorithms in modelling hybrid nanofluid density with unprecedented precision. This framework offers a scalable and reliable tool for advancing nanofluid-based applications in thermal engineering and related domains.

INDEX TERMS Density prediction, hybrid nanofluids, machine learning, deep learning, data augmentation, meta-heuristic optimization, optimization algorithms, thermal engineering.

I. INTRODUCTION

The exploration of hybrid nanofluids has garnered considerable attention due to their exceptional thermophysical characteristics, which make them ideal for use in energy systems, electronics cooling, and advanced material manufacturing. These fluids are created by dispersing multiple types of nanoparticles into a base fluid, resulting in improved thermal conductivity, viscosity, and density compared to single-component nanofluids. Among these properties, density

The associate editor coordinating the review of this manuscript and approving it for publication was Tyson Brooks.

plays a pivotal role in determining flow dynamics and heat transfer performance in various engineering applications. Consequently, accurately predicting the density of hybrid nanofluids is crucial to improve their effectiveness in real-world systems [1], [2]. Machine learning (ML) and deep learning (DL) have emerged as transformative tools in the field of nanofluid research, offering the ability to model complex relationships between input features and output properties. Unlike traditional empirical or theoretical approaches, ML and DL methods utilize data-driven algorithms to capture non-linear interactions, yielding predictions with high accuracy and reliability. Recent advances have demonstrated the potential of these approaches in predicting thermophysical properties, including viscosity, thermal conductivity, and density, in various nanofluid systems [3], [4]. For example, studies by Ali et al. and Ullah et al. highlighted the effectiveness of these tools in modeling the rheological and thermal transport behaviors of hybrid nanofluids, respectively [5], [6]. Using these techniques can accelerate the design and deployment of hybrid nanofluids by providing robust predictive frameworks. This study aims to develop and evaluate ML and DL models for predicting the density of hybrid nanofluids. By integrating experimental data sets and advanced algorithms, this research aims to provide insights into the predictive accuracy of various approaches and identify optimal models for practical implementation. The following sections discuss the current state of knowledge in this domain, highlighting key contributions and gaps in the literature.

The primary contribution of this work is the development of a dataset containing 436 samples for predicting the density (ρ) of hybrid nanofluids. The density prediction relies on several input features, including the type of nanoparticle, base fluid, temperature (°C), volume concentration (ϕ), base fluid density (ρ_{bf}), density of the first nanoparticle (ρ_{np1}), density of the second nanoparticle (ρ_{np2}), volume mixture of the first nanoparticle, and volume mixture of the secondary nanoparticle. An extensive workflow is employed, involving data augmentation techniques (autoencoder-based and Gaussian noise injection) and metaheuristic optimization strategies for hyperparameter tuning, specifically Grey Wolf Optimization (GWO) and Particle Swarm Optimization (PSO). Furthermore, the study integrates a wide array of machine learning and deep learning models—fourteen in total—to enhance predictive accuracy. An ensemble version of these fourteen models is also proposed, where a stacking linear regression approach is applied on top of each individual model to leverage their collective strengths. In addition, a comprehensive analysis and discussion of the underlying methods are presented, supported by an in-depth graphical examination of the results. The investigation also addresses computational performance by considering training time, testing time, per-sample inference time, and model size, alongside standard performance metrics. This holistic workflow ultimately establishes a robust computational intelligence framework for improved prediction of hybrid nanofluid density.

The structure of the paper is organized as follows. Section II reviews previous research conducted in related areas. Section III provides a detailed explanation of the materials and methods employed in this study. Section IV presents the experimental results and includes a discussion that lays the foundation for the conclusions, which are outlined in Section V.

II. LITERATURE REVIEW

The use of machine learning techniques to predict the thermophysical properties of nanofluids has advanced notably in recent years. Specifically, for density prediction, Singh et al. introduced a GBR-GSO-based model to estimate the density of nanofluids containing Al_2N_3, Si_3N_4, and TiN nanoparticles. Their findings highlighted the significance of feature selection and optimization in improving model performance [1]. Adun et al. investigated the density characteristics of a novel Al_2O_3-ZnO-Fe_3O_4 ternary hybrid nanofluid using both experimental techniques and ML frameworks, providing a comprehensive understanding of hybrid nanofluid properties [2]. Dehury et al. explored the prediction of thermophysical properties of deep eutectic solvent-based organic nanofluids using ML models, showcasing their capability in capturing the complex interactions of nanofluid constituents [3]. AbuShanab et al. focused on the dynamic viscosity prediction of polyalpha-olefin boron nitride nanofluids, demonstrating the efficacy of ML models in achieving high accuracy [4]. Similarly, Ali et al. utilized ML algorithms to predict the rheological behavior of BN-diamond/thermal oil hybrid nanofluids, emphasizing the potential of these tools in enhancing the understanding of hybrid nanofluids [5]. Ullah et al. applied deep neural networks to analyze the influence of dissipative forces on thermal transport in hybrid nanofluid flows, underscoring the robustness of DL techniques in handling complex fluid dynamics [6]. Deep learning approaches have also been increasingly employed in nanofluid studies. Changdar et al. developed a DL-based model for viscosity prediction, achieving superior performance over traditional ML methods [7]. Kanti et al. provided an experimental and explainable ML approach to investigate the thermal conductivity and viscosity of graphene oxide-based mono and hybrid nanofluids. This study emphasized the importance of interpretable models for understanding the underlying mechanisms driving property enhancements [8]. Similarly, Said et al. integrated experimental data with data-driven models to analyze the thermophysical properties of hybrid nanofluids containing nanodiamond and Fe_3O_4, showcasing the synergy between experimental and computational methods [9]. Furthermore, The predictive capabilities of artificial neural networks (ANNs) were highlighted by Alqaed et al. in their study on oxide-MWCNT water hybrid nanofluids [10]. Despite these advancements, the accurate prediction of hybrid nanofluid

density remains an open challenge, particularly for systems involving complex nanoparticle interactions and diverse base fluids. This study aims to address these gaps by employing state-of-the-art ML and DL techniques, leveraging extensive datasets, and exploring innovative model architectures to enhance predictive performance.

III. MATERIALS AND METHODS

The present research establishes a comprehensive framework for predicting hybrid nanofluid density [11] through the implementation of advanced machine learning and deep learning methodologies, addressing the inherent limitations of conventional empirical approaches. The collected experimental dataset [12] encompasses crucial thermophysical parameters, including temperature variations, volume concentration distributions, and constituent particle densities. The study explored various data preparation approaches, with particularly effective results from two key methods: autoencoder-based data augmentation [13] and Gaussian noise injection [14]. Both techniques successfully expanded the training split of the dataset while maintaining its essential statistical properties and preserving the fundamental physical relationships within the data. The research framework incorporated fourteen distinct machine learning and deep learning architectures, with model selection criteria prioritizing prediction accuracy, computational resource utilization, and generalization robustness across diverse operating conditions. Following augmentation, for systematic hyperparameter optimization, both Grey Wolf Optimization (GWO) [15] and Particle Swarm Optimization (PSO) [16] algorithms were implemented, with comparative analysis revealing their complementary strengths in navigating the complex hyperparameter landscape. The methodological framework integrates iterative validation procedures, sophisticated data augmentation techniques, and meta-heuristic optimization approaches to establish reliable predictive models for hybrid nanofluid density estimation. The comprehensive methodology implementation pipeline is illustrated in Figure 1, depicting the systematic workflow from data collection to model deployment.

A. DATASET

The experimental density dataset, compiled from peer-reviewed literature [12], consists of 436 distinct samples, each characterized by five input parameters along with density as the output parameter. For this study, an updated version of the dataset was made, also derived from the literature [12], now featuring nine input parameters and the corresponding density as the output. The density can be expressed using equation (1) as follows:

$$\rho_{hybrid} = \phi_1 * \rho_{np1} + \phi_2 * \rho_{np2} + (1 - \phi_1 - \phi_2) * \rho_{bf} \quad (1)$$

The density of the hybrid nanofluid (ρ_{hybrid}), which is the target column in the dataset, can be expressed in terms of various parameters. Here, ϕ_1 represents the volume concentration of the first nanoparticle, ϕ_2 denotes the volume concentration of the second nanoparticle, ρ_{np1} is the density of the first nanoparticle, ρ_{np2} is the density of the second nanoparticle, and ρ_{bf} represents the density of the base fluid.

The input features encompass both categorical variables (the nanoparticle and base fluid) and numerical parameters (temperature, volume concentration, density of the primary and secondary particles, base fluid density, and volume mixture ratios of the primary and secondary particles). Table 1 presents the various hybrid nanofluids analyzed in this research. These nanofluids encompass various nanoparticle combinations, including Al_2O_3/SiO_2, TiO_2/SiO_2, Fe_3O_4/MWCNT, Al_2O_3/CNT, Al_2O_3/MWCNT, TiO_2/MWCNT, CeO_2/MWCNT, ZnO/MWCNT, MgO/MWCNT, CuO/MWCNT, Co_3O_4/rGO, TiO_2/MgO, Ag/GNP, and ND/Fe_3O_4.

The base fluids used in this study are water, GB, distilled water (DW), and W-EG, 60:40%. Table 2 gives the range of values for each attribute in the dataset. The temperature range spans from 16 °C to 70 °C, while the volume concentration (ϕ) varies from 0 to 2. The densities of the first (ρ_{np1}) and second (ρ_{np2}) nanoparticles range from 3100–10490 kg/m^3 and 1910–5810 kg/m^3, respectively, with the base fluid density (ρ_{bf}) ranging from 977.60–1063.00 kg/m^3. The volume mixtures of the first and second particles are represented by discrete values (17, 20, 28, 50, 74, 80) and (20, 26, 50, 72, 80, 83), respectively. The hybrid nanofluid density (ρ_{hnf}) spans from 983.01 to 1093.43 kg/m^3. These ranges effectively cover the typical operating conditions and material properties encountered in hybrid nanofluid applications.

Table 3 provides a statistical overview of the variables analyzed in this study, capturing essential measures of central tendency, dispersion, and distribution. It includes the mean, mode, and median to represent central values, along with variance and standard deviation to indicate variability within the data. The table also highlights the minimum and maximum values, reflecting the range of the dataset, while skewness and kurtosis provide insights into the asymmetry and peakiness of the distributions. Together, these statistics offer a comprehensive summary of the dataset, enabling a deeper understanding of the variability, consistency, and distributional characteristics of the studied parameters.

Statistical analysis reveals significant variability across the thermophysical parameters in Table 3. Temperature distributions exhibit a mean of 39.70 °C with a standard deviation of 12.44 °C, spanning from 16 °C to 70 °C, demonstrating mild positive skewness (0.329) and platykurtic behavior (−0.442). Volume concentration ranges from 0% to 2%, with a mean of 0.595% and variance of 0.411, indicating considerable dispersion in nanoparticle loading. The particle density parameters show distinct characteristics, with first particle density (ρ_{np1}) averaging 5380.23 kg/m^3 (SD = 1858.17 kg/m^3) and second particle density (ρ_{np2}) averaging 2470.64 kg/m^3 (SD = 895.04 kg/m^3), both exhibiting positive skewness and leptokurtic distributions. Base fluid density (ρ_{bf}) demonstrates relatively lower variability (mean = 998.24 kg/m^3, SD = 17.72 kg/m^3) compared to particle

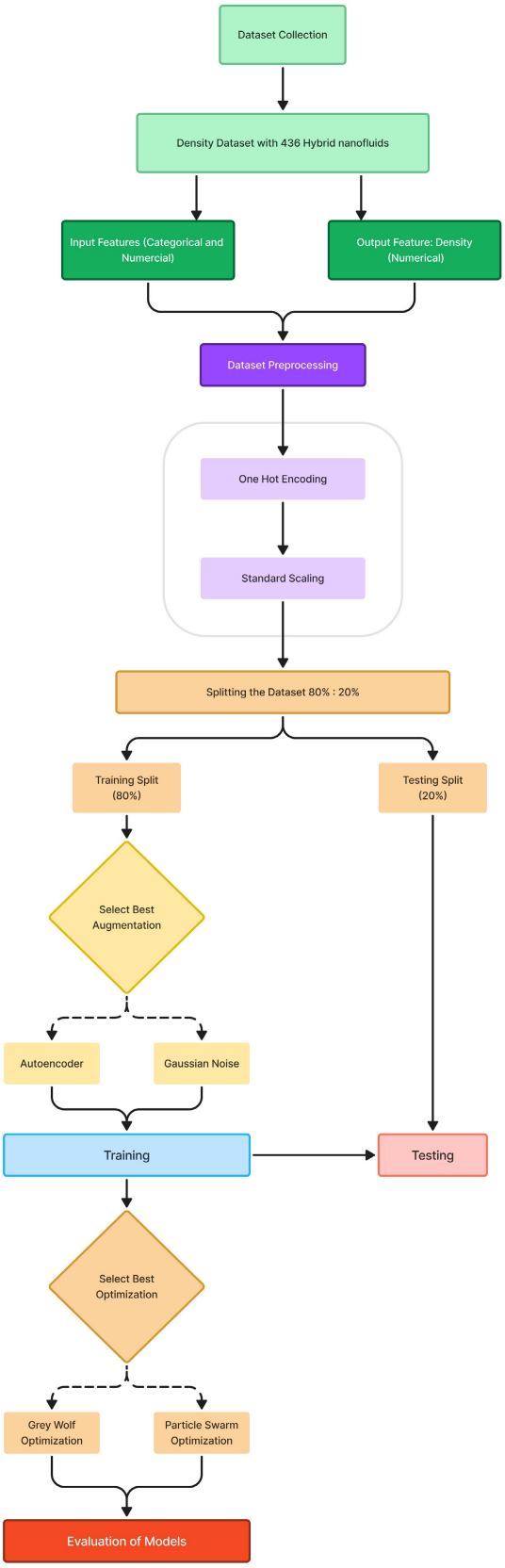

FIGURE 1. General workflow of the methodology.

TABLE 1. Hybrid nanofluid description from the dataset.

Nano Fluid	Base Fluid	Sample	Reference
Al_2O_3/SiO_2	Water	15	[18]
TiO_2/SiO_2	GB	30	[19]
Fe_3O_4/MWCNT	Water	45	[20]
Al_2O_3/CNT	Water	10	[21]
Al_2O_3/MWCNT	Water	48	[22]
TiO_2/MWCNT	Water	48	[22]
CeO_2/MWCNT	Water	48	[22]
ZnO/MWCNT	Water	48	[22]
MgO/MWCNT	Water	07	[23]
CuO/MWCNT	Water	07	[23]
Co_3O_4/rGO	Water	20	[24]
TiO_2/MgO	DW	60	[25]
Ag/GNP	Water	30	[25]
ND-Fe_3O_4	W-EG (60:40%), Water	20	[26]

TABLE 2. Range of the features in the dataset.

Features	Range
Temperature (°C)	16.00–70.00
Volume concentration (Φ)	0.000–2.000
ρ_{np1} (kg/m^3)	3100.00–10490.00
ρ_{np2} (kg/m^3)	1910.00–5810.00
ρ_{bf} (kg/m^3)	977.60–1063.00
Volume mixture of First Particle	17, 20, 28, 50, 74, 80
Volume mixture of Second Particle	20, 26, 50, 72, 80, 83
ρ_{hnf} (kg/m^3)	983.01–1093.43

densities. Volume mixture ratios for both particles show complementary distributions, with means of 57.52% and 42.48%, respectively, displaying opposite skewness patterns but identical kurtosis values (−1.634). The output variable, hybrid nanofluid density, ranges from 983.01 kg/m^3 to 1093.43 kg/m^3, with a mean of 1016.37 kg/m^3 and positive skewness (1.092), indicating a right-tailed distribution.

Categorical variables underwent binary vector encoding [17] to facilitate numerical processing, while continuous variables were standardized to ensure uniform scale distributions across the feature space.

Hybrid nanofluids are widely utilized in cutting-edge applications, including heat exchangers, solar thermal collectors, automotive engine cooling, microprocessor and chip cooling, data center cooling systems, and battery cooling systems. These diverse applications underscore the importance of understanding the density of hybrid nanofluids, making it a critical area for scientific exploration.

B. PREPROCESSING

Data preprocessing represents a crucial phase in developing machine learning models, encompassing systematic procedures to enhance data quality and ensure optimal model performance. The preprocessing pipeline implemented in this study addresses several critical aspects of data preparation for hybrid nanofluid density prediction.

The initial preprocessing step involved outlier detection and removal using the Interquartile Range (IQR) method [27]. This technique identifies and eliminates anomalous data

TABLE 3. Statistical overview of input and output variables of the dataset.

Dataset Features	Mean	Mode	Median	Variance	Standard Deviation	Minimum	Maximum	Skewness	Kurtosis
Temperature (°C)	39.70183	40	40	154.8764	12.44494	16	70	0.329594	-0.44196
Volume concentration (Φ)	0.594828	0.1	0.3	0.4111	0.641171	0	2	0.938571	-0.50486
ρ_{np1} (kg/m^3)	5380.229	4320	4320	3452784	1858.167	3100	10490	1.398068	1.423411
ρ_{np2} (kg/m^3)	2470.642	2100	2100	801104	895.0441	1910	5810	2.587184	6.286153
ρ_{bf} (kg/m^3)	998.2439	990.22	994.08	314.1077	17.72308	977.6	1063	2.241501	3.914028
Volume mixture of First Particle	57.52294	80	80	785.0454	28.01866	17	80	-0.554	-1.63436
Volume mixture of Second Particle	42.47706	20	20	785.0454	28.01866	20	83	0.554004	-1.63436
ρ_{hnf} (kg/m^3)	1016.367	990	1010.319	515.0617	22.69497	983.01	1093.432	1.092438	0.76729

points that could potentially skew the model's learning process. The study implements binary vector encoding for categorical variables, specifically the particle types and base fluid compositions, as studied earlier [17], converting categorical variables into a format suitable for machine learning algorithms by creating binary columns for each unique category.

Feature scaling was applied through standardization [28], normalizing all numerical features to zero mean and unit variance. This standardization is particularly important for our dataset, where features like temperature (16 °C to 70 °C) and particle density (1910 to 10490 kg/m^3) exist on substantially different scales. The standardization process ensures that all features contribute equally during model training, preventing features with larger numerical ranges from overpowering those with smaller ranges.

After preprocessing, the dataset was divided into training and testing sets with an 80:20 split [29]. This allocation dedicates 80% of the standardized data (349 samples) for training the model, while the remaining 20% (87 samples) is used as an independent testing set to evaluate the model's performance on unseen data, offering an unbiased measure of its generalization ability.

C. AUGMENTATION

Data augmentation is a crucial preprocessing technique in machine learning, particularly for improving model generalization and robustness. By artificially expanding the training split of the dataset, augmentation introduces variations that help the model better capture underlying patterns in the data. In the initial phase of the study, eleven data augmentation techniques were evaluated to enhance the dataset and improve model generalization. These methods included Gaussian Noise, the Synthetic Minority Over-sampling Technique (SMOTE), Autoencoder-based augmentation, Interpolation, Synthetic Bootstrapping, Scaling Augmentation, Polynomial and Fourier Series Expansion inspired augmentation, Local Outlier Factor (LOF) Sampling, Multivariate Perturbation, Kernel Density Estimation (KDE) Sampling, and Conditional Variational Autoencoder (CVAE). After assessing the impact and consistency of each approach, two augmentation techniques were selected for application on the training split: Autoencoder-based augmentation [13] and Gaussian noise augmentation [14].

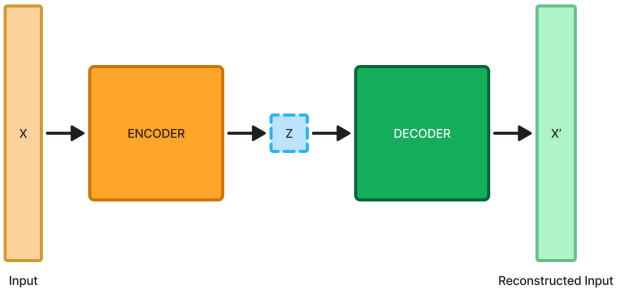

FIGURE 2. Architecture of autoencoder augmentation technique.

1) AUTOENCODER-BASED AUGMENTATION

Autoencoder-based [13] augmentation was employed to enhance the dataset by generating synthetic samples through a reconstruction process. Autoencoder-based augmentation enriches the training split of the dataset by generating new, realistic samples. This process helps improve the effectiveness and robustness of machine learning models, especially when the original dataset is small or imbalanced.

Figure 2 illustrates the autoencoder architecture. The autoencoder works by first compressing the data through the encoder and then reconstructing it through the decoder, with the aim of minimizing the difference between the original data X and the reconstructed data X'. The process is mathematically expressed in Equation (2).

$$Z = f(X)\,[\text{Encoding}] \quad \text{and} \quad X' = g(Z)\,[\text{Decoding}] \tag{2}$$

where X represents the input data, Z the compressed latent representation, X' the reconstructed output, and $f(X)$, $g(Z)$ the encoder and decoder functions, respectively. The autoencoder minimizes the reconstruction error $||X - X'||^2$ [30], ensuring that Z captures meaningful representations for data augmentation through sampling.

2) GAUSSIAN NOISE AUGMENTATION

Gaussian noise augmentation [14] involves strategic addition of controlled random variations to existing samples through a normal distribution mechanism. This technique leverages statistical principles to generate synthetic data points while preserving the dataset's underlying characteristics and physical constraints. The method applies normally distributed perturbations following the probability density function in

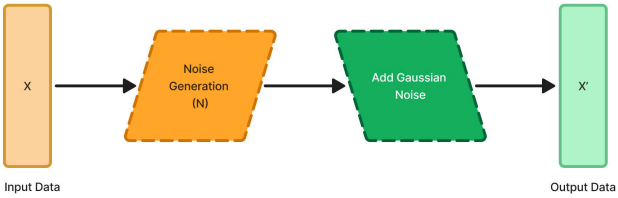

FIGURE 3. Workflow of gaussian noise augmentation technique.

Equation (3)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3}$$

Here in the normal distribution, μ represents the mean, determining the center of the distribution, and σ denotes the standard deviation, controlling the magnitude of variations. The augmented sample X' is generated through the mathematical formulation given in Equation (4):

$$X' = X + N(\mu, \sigma^2) \tag{4}$$

where X represents the original sample and $N(\mu, \sigma^2)$ denotes Gaussian noise with mean μ and variance σ^2. The selection of noise parameters requires careful calibration to maintain physical feasibility while introducing sufficient variability for model generalization. Key considerations include the scale of features, their physical bounds, and the preservation of meaningful correlations between variables [31]. Figure 3 illustrates the flowchart for Gaussian Noise augmentation. The procedure begins with the input data X, which is subjected to a noise generation step where Gaussian noise N is generated. This noise is then incorporated into the training split of the original data, creating the augmented output X'. By preserving the statistical properties of the original dataset while adding slight variations, this method improves the robustness of predictive models during training, helping them perform better in the face of real-world data fluctuations.

D. OPTIMIZATION

Optimization algorithms play a vital role in enhancing the performance of machine learning models by efficiently tuning their hyperparameters and minimizing prediction errors. These algorithms systematically explore the parameter space to find optimal configurations that maximize model accuracy while maintaining computational efficiency. These algorithms were systematically applied following the augmentation. In the preliminary phase of the study, several nature-inspired optimization techniques were considered, including Grey Wolf Optimization (GWO), Particle Swarm Optimization (PSO), Cuckoo Search, Ant Colony Optimization (ACO), Artificial Bee Colony, and the Firefly Algorithm. Based on baseline results obtained from a select group of models, the study narrowed the focus to two methods for further analysis: the Grey Wolf Optimizer (GWO) [15] and Particle Swarm Optimization (PSO) [32]. Both methods demonstrate robust capabilities in handling the complex, non-linear relationships inherent in hybrid nanofluid density prediction.

1) GREY WOLF OPTIMIZATION

Grey Wolf Optimization (GWO) is a powerful optimization technique inspired by the social structure and hunting strategy of grey wolves. It is used to solve complex optimization problems by simulating natural behaviors such as encircling and cooperative hunting. GWO [15] was employed to enhance the hyperparameter tuning of the predictive models. This metaheuristic algorithm, first introduced in [33], categorizes the population into four groups: alpha (α), beta (β), delta (δ), and omega (ω), representing a hierarchical optimization structure.

The position of a wolf in the search space is mathematically represented as given in Equation (5):

$$\begin{aligned} D_\alpha &= |C_1 \cdot X_\alpha - X|, \\ D_\beta &= |C_2 \cdot X_\beta - X|, \\ D_\delta &= |C_3 \cdot X_\delta - X|, \end{aligned} \tag{5}$$

where $\vec{X}$ is the current position of a wolf, and X_α, X_β, and X_δ represent the positions of the alpha, beta, and delta wolves, respectively. The coefficients C_1, C_2, and C_3 are dynamically adjusted to control the search process.

Figure 4 illustrates the workflow of the Grey Wolf Optimization (GWO) technique. GWO was selected for its proven ability to balance exploration and exploitation effectively, enabling a thorough exploration of the hyperparameter space while maintaining convergence efficiency. This characteristic made GWO particularly suitable for fine-tuning the hyperparameters of complex machine learning models.

2) PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) [32] is a population-based optimization technique inspired by the social behavior of bird flocking and fish schooling. It works by iteratively improving candidate solutions to find the optimal parameters for a given problem. The algorithm begins by initializing a swarm of particles, with each particle representing a potential solution in the search space. These particles explore the space by updating their positions and velocities. The position update is influenced by two factors:

- *Personal experience:* Each particle remembers its own best position (solution) found so far, known as *pbest*.
- *Social experience:* Each particle is influenced by the best position found by its neighbors, called *gbest*.

The velocity of each particle is updated according to Equation (6):

$$v_i^{(t+1)} = \omega v_i^{(t)} + c_1 r_1 (p_i^{best} - x_i^{(t)}) + c_2 r_2 (g^{best} - x_i^{(t)}) \tag{6}$$

In this equation, $v_i^{(t)}$ represents the velocity of the i-th particle at iteration t, while $x_i^{(t)}$ denotes its corresponding position in the search space. The term ω is the inertia weight, which balances the exploration and exploitation capabilities

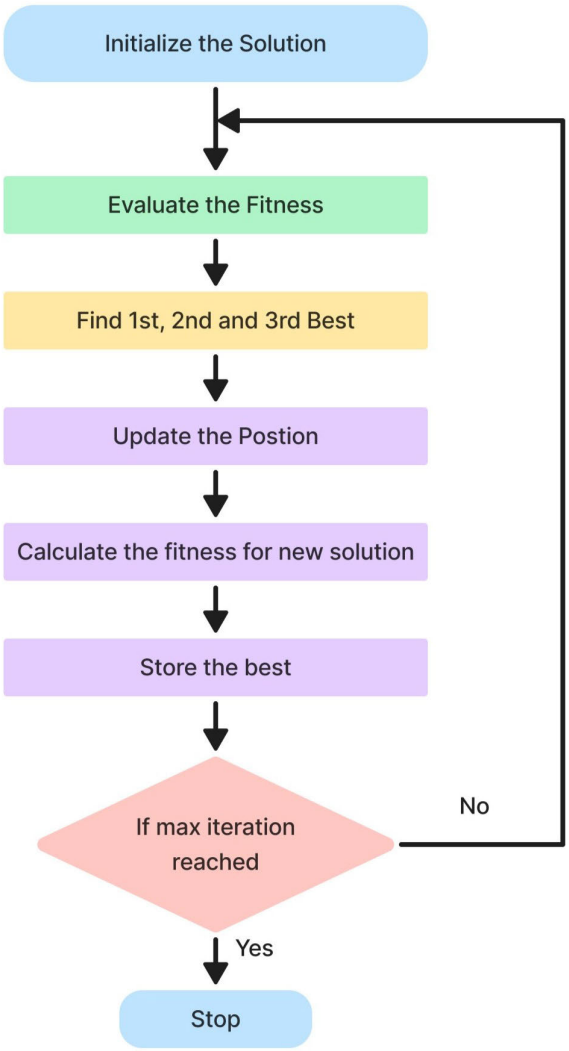

FIGURE 4. Workflow of grey wolf optimization algorithm.

of the swarm. The cognitive and social components, r_1 and r_2, control the influence of a particle's own best-known position p_i^{best} and the global best position of the swarm g^{best}, respectively. ω is a constant inertia weight, which could remain static or vary dynamically based on the iteration number (e.g., as a decaying function). Random variables r_1 and r_2 are uniformly distributed within the range [0, 1] to introduce stochasticity in the movement, thereby promoting diverse exploration of the search space.

The position of each particle is then updated as given in Equation (7):

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \tag{7}$$

This iterative process continues until a stopping criterion, such as a maximum number of iterations or convergence to an acceptable error threshold, is met. Figure 5 illustrates the workflow of the PSO technique. PSO is particularly advantageous for optimization problems involving complex,

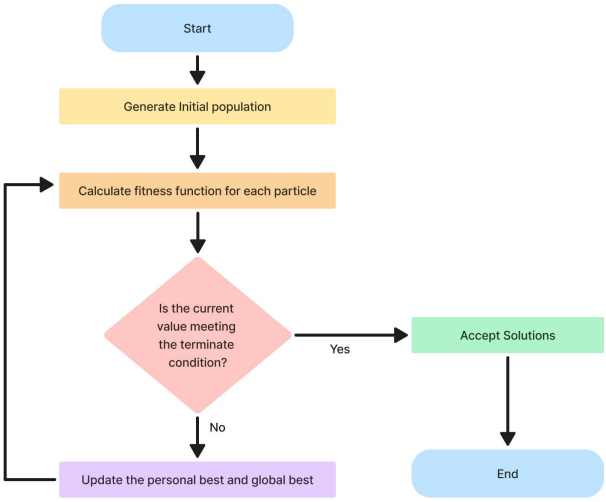

FIGURE 5. Workflow of the particle swarm optimization (PSO) technique.

multidimensional, and non-differentiable objective functions [34].

Its simplicity, ability to escape local optima, and fewer hyperparameters make it a suitable choice for optimizing machine learning models. In this study, PSO was utilized to optimize the hyperparameters of predictive models, ensuring improved accuracy and performance by finding an optimal balance between exploration and exploitation within the search space.

E. MODELS

In this study, fourteen estimation models were developed and implemented to estimate the density of hybrid nanofluids, leveraging a combination of machine learning [36] and deep learning [36] models. The machine learning models, with some being regression-oriented [37], were designed to capture intricate relationships in the data and provide precise predictions. In contrast, the Deep learning models employed direct learning approaches combined with regression outputs, enabling them to model complex, non-linear patterns inherent in the dataset.

To enhance predictive accuracy and interpretability, a second variant of all models were also engineered, here the models were ensembled by stacking [38] them with linear regression [39], [62]. This stacking approach leveraged the strengths of both the base learner (e.g., capturing complex patterns) and the linear regression layer (e.g., ensuring simplicity and generalization in the final output). This combination not only improved predictive performance but also reduced the risk of overfitting [40] by balancing model complexity and interpretability.

Optimization techniques were employed exclusively with the machine learning models to fine-tune their hyperparameters [41] and enhance their performance. The deep learning models, owing to their inherent capability to self-learn optimal weights through backpropagation [42] and large

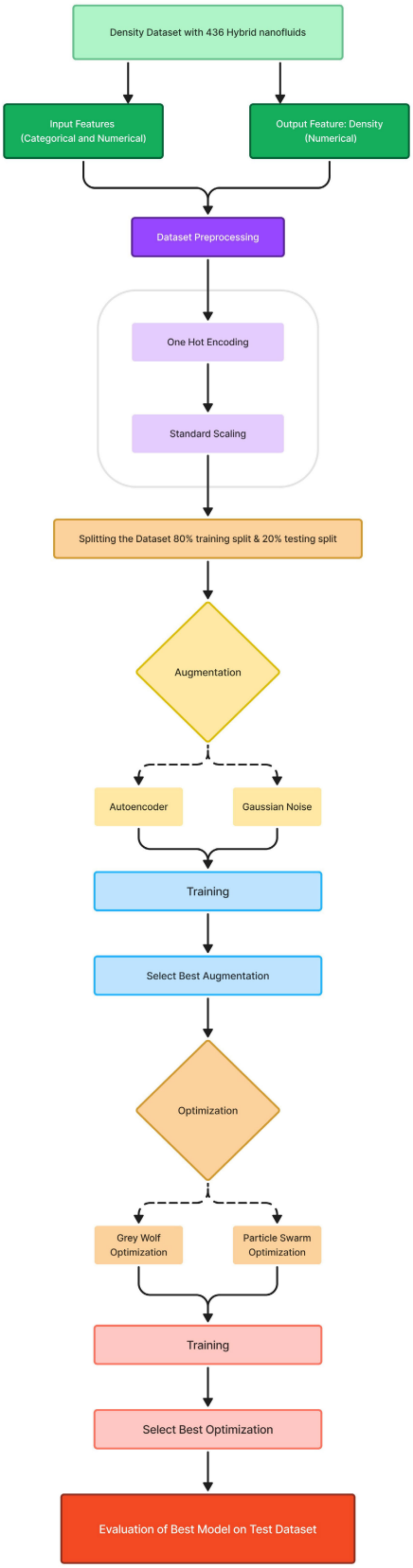

FIGURE 6. Generalized workflow of the machine learning models in the study.

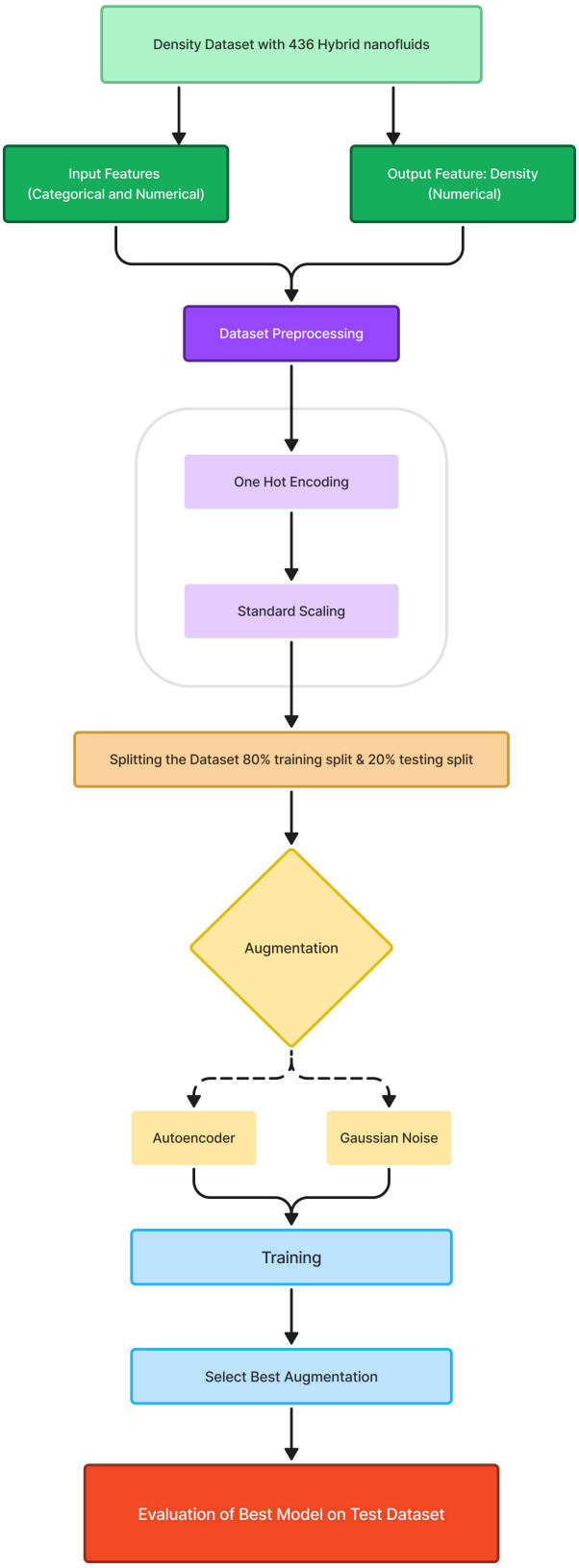

FIGURE 7. Generalized workflow of deep learning models implemented in the study.

training times, were trained without the use of external optimization techniques.

To further assess and mitigate potential overfitting, the models were evaluated using K-fold cross validation. This technique involves dividing the dataset into k equally sized subsets, allowing the model to be trained and validated across multiple partitions. By doing so, it becomes possible to ensure that the model's performance is consistent across all subsets rather than being tailored to a specific portion of the data.

Figure 6 illustrates the generalized workflow adopted for machine learning models, highlighting the preprocessing, optimization, and training stages. Figure 7 presents the generalized workflow for deep learning models, showcasing their architectural design and direct learning-to-regression mechanism. Together, these workflows provide a comprehensive understanding of the methodologies employed for predictive modelling in this study.

1) DECISION TREE

A non-parametric supervised learning method called the Decision Tree algorithm [43] divides data into subsets according to feature values in order to generate predictions. A tree structure is used to represent the model, with each internal node standing for a feature-based decision rule, each branch for a decision's result, and each leaf node for a prediction. Decision trees work especially well with datasets that have non-linear feature interactions and linkages. The Decision Tree algorithm splits the dataset iteratively by selecting features and thresholds that minimize a chosen impurity metric, such as Gini Impurity or Entropy. For regression tasks, the objective is to minimize the Mean Squared Error (MSE) [44] of the target variable, defined as given in Equation (8)

$$\text{MSE} = \sum_{i=1}^{n} (\alpha_i - \hat{\alpha}_i)^2 \tag{8}$$

where, α_i is the observed value, $\hat{\alpha}_i$ is the predicted value, and n is the number of samples in the node. The splitting process continues until a stopping criterion, such as a maximum tree depth or a minimum number of samples required to split a node, is met. The decision-making process in Decision Trees is intuitive and interpretable, making it a widely adopted method for structured datasets.

2) RANDOM FOREST

Random Forest [45] is an ensemble learning method that constructs multiple decision trees and combines their outputs, using averaging for regression tasks or majority voting for classification tasks. To ensure diversity and minimize correlations among the trees, the technique integrates random feature selection with bootstrap aggregation, commonly referred to as bagging. Each tree is trained on a bootstrap sample of the data using a random subset of features at each split, mathematically expressed as Equation (9):

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x) \tag{9}$$

where B represents the number of trees, $T_b(x)$ denotes the prediction of the b-th tree, and x is the input vector. The variance of the random forest predictor can be expressed as Equation (10):

$$\text{Var}(\hat{f}_{rf}^{B}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{10}$$

where ρ represents the correlation between trees, and σ^2 is the variance of individual trees. The randomness in both observation and feature selection reduces overfitting and inter-tree correlation, enhancing model robustness and generalization capability. This dual randomization strategy makes Random Forest particularly effective for handling complex, non-linear relationships in the data.

3) RIDGE REGRESSION

Ridge Regression [46], also known as Tikhonov regularization, is a linear regression technique that includes a penalty term to prevent overfitting and improve the model's generalization. It is particularly useful when there is multicollinearity among the predictor variables or when the number of predictors exceeds the number of observations. The optimization objective is formulated in Equation (11):

$$\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{11}$$

where λ controls the regularization strength, β represents the model coefficients, and (x_i, y_i) are the input-output pairs [47]. The closed-form solution for the ridge estimator is given by Equation (12):

$$\hat{\beta}_{ridge} = (X^T X + \lambda I)^{-1} X^T y \tag{12}$$

where X is the design matrix, y is the response vector, and I is the identity matrix [48]. The L_2 penalty term prevents coefficient values from becoming excessively large, particularly useful when dealing with multicollinearity in the feature space. This regularization approach helps stabilize the model and improve its generalization performance by finding a balance between bias and variance [49].

4) POISSON REGRESSION

Poisson Regression [50] models the relationship between predictors and a response variable using the Poisson distribution, making it particularly suitable for count data and non-negative continuous outcomes. The model's core Equation is expressed as Equation (13):

$$\log(\mu) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i \tag{13}$$

where μ represents the expected value of the response variable, β_0 is the intercept, and β_i are the regression coefficients [51]. The probability mass function of the Poisson distribution is given by Equation (14):

$$P(Y = y \mid \mu) = \frac{e^{-\mu}\mu^{y}}{y!} \tag{14}$$

where Y is the response variable and μ is the mean parameter [52]. The logarithmic link function ensures non-negative predictions, making it particularly suitable for density prediction tasks where negative values are physically impossible. The model parameters are typically estimated using maximum likelihood estimation, optimizing the log-likelihood function [53] in Equation (15):

$$l(\beta) = \sum_{i=1}^{n} \left[y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) - e^{\left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\right)} \right] \tag{15}$$

5) GRADIENT BOOSTING

Gradient Boosting [54] is a powerful ensemble learning method that builds a predictive model by combining a series of simpler models, known as weak learners (commonly decision trees). Each weak learner focuses on improving the performance of the previous ones in a sequential manner. The key idea is to minimize a chosen loss function by iteratively refining the model. The fundamental principle of Gradient Boosting can be expressed through its additive modelling framework, where each new weak learner is added to the ensemble according to Equation (16):

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x) \tag{16}$$

Provided that at iteration m, $F_m(x)$ represents the model, $F_{m-1}(x)$ is the model from the previous iteration, η is the learning rate, and $h_m(x)$ is the weak learner fitted to the negative gradient of the loss function. This iterative process continues until the specified number of estimators is reached or convergence criteria are met.

6) LightGBM

LightGBM [55] is a sophisticated gradient boosting framework that introduces innovative techniques such as Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). Unlike conventional tree-based models, LightGBM grows trees in a leaf-wise manner rather than level-wise, prioritizing the leaf that offers the most significant reduction in the loss function. The leaf-wise growth pattern is mathematically expressed as in Equation (17):

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F \tag{17}$$

where $\hat{y}_i$ represents the predicted value for the i-th instance, K denotes the number of trees, f_k represents the k-th tree, and F is the space of possible regression trees [56]. The objective function for optimization follows Equation (18):

$$L = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{18}$$

where l represents the loss function, y_i is the true value, and $\Omega(f_k)$ is the regularization term controlling model complexity.

7) CatBoost

CatBoost [57] (Categorical Boosting) is a gradient boosting algorithm optimized for handling categorical features directly, eliminating the need for extensive preprocessing like one-hot encoding [40]. Unlike conventional boosting algorithms, CatBoost introduces Ordered Boosting, a technique that addresses overfitting by maintaining an unbiased estimate of gradients during training. The CatBoost objective function for a regression task is defined as in Equation (19):

$$\text{Obj}(\Theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \lambda \cdot \Omega(f) \tag{19}$$

where $L(y_i, \hat{y}_i)$ represents the loss function (e.g., mean squared error) measuring the difference between observed y_i and predicted $\hat{y}_i$ values, and $\Omega(f)$ is the regularization term that controls model complexity to prevent overfitting.

8) XGBoost

XGBoost [58], short for Extreme Gradient Boosting, is a powerful ensemble learning algorithm known for its high efficiency and predictive performance on structured datasets. The regularized objective function is defined as in Equation (20):

$$\text{Obj}(\Theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{20}$$

Here, $L(y_i, \hat{y}_i)$ measures the loss between the observed y_i and predicted $\hat{y}_i$ values, while $\Omega(f_k) = \gamma T + 0.5\lambda \|w\|^2$ penalizes model complexity to prevent overfitting [40]. The term T represents the number of leaves in a tree, w refers to leaf weights, and Θ denotes all model parameters, including tree structures and weights.

9) NEURAL NETWORK

Neural Networks [59] process information through multiple interconnected layers of neurons, with each layer transforming its input through weighted connections and non-linear activations. The forward propagation in each layer follows Equation (21):

$$a^{[l]} = g^{[l]}(W^{[l]} a^{[l-1]} + b^{[l]}) \tag{21}$$

where $a^{[l]}$ represents the activations at layer l, $W^{[l]}$ and $b^{[l]}$ are the weights and biases, and $g^{[l]}$ is the activation function [60]. The loss function for the neural network

component is given by Equation (22):

$$L_{NN} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{l=1}^{L} \|W^{[l]}\|^2 \tag{22}$$

where λ controls the L_2 regularization strength [61].

10) CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks (CNNs) [63] have demonstrated remarkable effectiveness in sequence modelling through 1D convolution operations. The primary operation in a 1D CNN can be expressed as in the Equation (23):

$$y_l = f\left(\sum_{k=1}^{K} w_k \odot x_{l-k} + b\right) \tag{23}$$

where y_l represents the output at position l, w_k represents the k-th filter weight, x_{l-k} is the input at position $l-k$, K is the filter size, and f is the activation function [64].

11) RECURRENT NEURAL NETWORK

Recurrent Neural Networks (RNNs) [65] are designed to handle sequential data by maintaining a hidden state that encodes temporal dependencies. Unlike standard feedforward neural networks, RNNs include recurrent connections that enable information from previous time steps to influence the current output. This makes RNNs especially well-suited for data with sequential or contextual relationships, such as time series, natural language, or speech data [67]. The fundamental operation of an RNN is expressed as in Equation (24):

$$h_t = f(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1} + b_h) \tag{24}$$

where h_t is the hidden state at time step t, x_t is the input at t, and h_{t-1} is the hidden state from the previous step, $W_x h$ and $W_h h$ are weight matrices, b_h is the bias, and f is the activation function (commonly ReLU [66]). The hidden state, h_t encodes both current and past information, allowing RNNs to model sequential dependencies effectively.

12) GATED RECURRENT UNIT

Gated Recurrent Units (GRUs) [68] are a simplified variant of LSTMs that provides a similar performance while using a more streamlined architecture. GRUs consolidate the forget and input gates into a single 'update gate' and unify the cell state and hidden state into one representation. This reduces the number of parameters and computations, making GRUs more computationally efficient while retaining their ability to capture long-term dependencies in sequential data. The fundamental Equations governing GRU operations are given in Equation (25),(26),(27) and (28)

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{25}$$
$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{26}$$
$$\tilde{h_t} = \tanh(W[r_t \odot h_{t-1}, x_t] + b_h) \tag{27}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t} \tag{28}$$

where z_t is the update gate, r_t is the reset gate, ht is the hidden state, and W terms and b terms represent weights and biases respectively. The update gate z_t determines how much of the previous state should be retained, while the reset gate r_t controls access to prior information.

13) LONG SHORT-TERM MEMORY (LSTM)

Long Short-Term Memory networks (LSTMs) [69] are designed to overcome the vanishing gradient problem and effectively capture long-term dependencies in sequential data. LSTMs feature a complex gating mechanism, comprising input, forget, and output gates, which control the flow of information through the network. These gates enable LSTMs to selectively retain or discard information, allowing them to maintain relevant context over extended sequences while discarding irrelevant details. The core operations of an LSTM cell can be expressed through the following Equation (29),(30),(31),(32),(33) and (34):

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{29}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{30}$$
$$\tilde{c_t} = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{31}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \tag{32}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{33}$$
$$h_t = o_t \odot \tanh(c_t) \tag{34}$$

where f_t, i_t, and o_t represent the forget, input, and output gates respectively, c_t is the cell state, h_t is the hidden state, W terms are weight matrices, b terms are bias vectors, and σ represents the sigmoid activation function [70]. The cell state c_t acts as a memory highway, allowing information to flow through the network with minimal degradation.

14) AUTOENCODER

Autoencoders [71] are a type of neural network designed to learn compact and efficient representations of data through an encoder-decoder structure. The encoder maps the input data into a lower-dimensional latent space, effectively compressing it into a more concise form. The decoder then reconstructs the original input from this compressed representation. This process allows autoencoders to capture the essential features of the data while discarding noise or less important details, making them useful for tasks such as dimensionality reduction, anomaly detection, and data denoising. The encoding process follows Equation (35):

$$h = f(W_e x + b_e) \tag{35}$$

where h is the latent representation, W_e represents encoder weights, b_e is the encoder bias, and f is an activation function [72]. The decoding process is expressed as Equation (36):

$$\hat{x} = g(W_d h + b_d) \tag{36}$$

where W_d and b_d are decoder weights and bias, and g is the decoder's activation function.

F. PERFORMANCE METRICS

The model's performance is assessed using a variety of complementary metrics to provide a thorough evaluation of its predictive capabilities. These metrics include:

- **Coefficient of Determination (R^2 Score)** [73]: Measures the proportion of the variance in the dependent variable that is predictable from the independent variables, giving an indication of the model's goodness of fit. Equation (37) gives the formulation of the metric.
- **Mean Squared Error (MSE)** [35]: Calculates the average of the squared differences between actual and predicted values, highlighting the prediction accuracy and giving more weight to larger errors. Equation (38) gives the formulation of the metric.
- **Root Mean Squared Error (RMSE)** [74]: The square root of MSE, providing a measure of the error in the same units as the target variable and emphasizing larger errors. Equation (39) gives the formulation of the metric.
- **Mean Absolute Error (MAE)** [75]: Represents the average of the absolute differences between actual and predicted values, providing a straightforward measure of error magnitude without exaggerating outliers. Equation (40) gives the formulation of the metric.
- **Mean Absolute Percentage Error (MAPE)** [76]: Quantifies the percentage difference between the actual and predicted values, offering a scale-independent error metric. Equation (41) gives the formulation of the metric.
- **Symmetric Mean Absolute Percentage Error (SMAPE)** [77]: A variant of MAPE that treats over- and under-predictions symmetrically, making it more robust to data with varying scales. Equation (42) gives the formulation of the metric.
- **Maximum Error** [78]: Identifies the worst-case prediction error, highlighting the largest discrepancy between actual and predicted values. Equation (43) gives the formulation of the metric.
- **Explained Variance Score (EVS)** [79]: Measures the proportion of the variance in the actual data that is captured by the model, indicating how well the model accounts for variability. Equation (44) gives the formulation of the metric.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{37}$$

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2 \tag{38}$$

$$RMSE = \sqrt{\frac{1}{n}\sum(y_i - \hat{y}_i)^2} \tag{39}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{40}$$

$$MAPE = \frac{100}{n}\sum\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{41}$$

$$SMAPE = \frac{200}{n}\sum\frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|} \tag{42}$$

$$\text{Max Error} = \max(|y_i - \hat{y}_i|) \tag{43}$$

$$\text{Explained Variance} = 1 - \frac{\text{Var}(y_i - \hat{y}_i)}{\text{Var}(y_i)} \tag{44}$$

where:

- y_i represents the actual values,
- $\hat{y}_i$ represents the predicted values,
- $\bar{y}$ represents the mean of actual values,
- n represents the number of samples.

These various metrics collectively provide a comprehensive view of the model's predictive performance, ensuring that different aspects of the model's accuracy, robustness, and reliability are assessed.

G. SETUP

The development, training, and testing were carried out on a personal GPU system equipped with an NVIDIA RTX 3080 Ti GPU, an Intel® Core™ i9-10900K CPU, 64 GB of RAM, and 1 TB of SSD storage. Data was read and processed in the Pandas library [80]. Data preprocessing was performed using the Scikit-learn library [83]. Machine Learning models were also implemented using the Scikit-learn library. Deep learning models were implemented using either the PyTorch [81] or the TensorFlow framework [82]. Evaluation metrics were computed with the Scikit-learn library [83], while graphical visualizations were created using the matplotlib [84] and seaborn libraries [85]. This setup facilitated efficient model training and evaluation, leveraging the capabilities of both hardware and software to achieve high-performance results.

IV. RESULTS AND DISCUSSION

A. DENSITY DATASET ANALYSIS

The comprehensive dataset analysis of hybrid nanofluids is presented through multiple visualizations that reveal intricate relationships between key parameters. The nanoparticle and base fluid combination frequencies are quantified in Figure 8, which reveals that TiO_2-MgO represents the most extensively available system with 60 samples, followed by several MWCNT-based combinations (Al_2O_3-MWCNT, TiO_2-MWCNT, CeO_2-MWCNT, and ZnO-MWCNT) with approximately 48 samples each. The analysis encompasses four base fluids: Water, GB, DW, and W-EG(16)-40%), with varying utilization frequencies across different nanoparticle systems.

Figure 9 presents a correlation matrix that quantitatively demonstrates the interconnections between principal variables. A notable finding is the robust positive correlation (0.77) between base fluid density and overall system density, emphasizing the base fluid's crucial role in determining final nanofluid properties. The matrix also reveals that temperature exhibits relatively modest correlations with other parameters, ranging from −0.22 to 0.10, suggesting that compositional

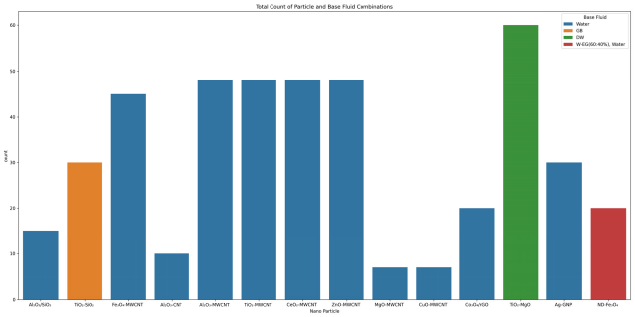

FIGURE 8. Distribution of nanoparticles and base fluids.

factors exert stronger influence on density variations than thermal conditions.

The distribution characteristics of the dataset are captured in Figure 10 through a series of histograms. Temperature measurements exhibit a relatively uniform distribution across the 20 °C to 70 °C range, with a slight concentration in the 30 °C-50 °C interval. Volume concentration (ϕ) distributions display a pronounced right-skew, with most measurements falling below 0.5%, though the experimental range extends to 2.0%. The density distributions for both nanoparticles and base fluids show material-specific patterns that reflect their inherent physical properties.

Figure 11 provides detailed heat map visualizations of feature correlations for different nanoparticle combinations, offering deeper insights into system-specific relationship patterns. Figure 12 shows the thermal response of fourteen hybrid nanoparticle systems, highlighting density variations across 25 °C-60 °C. The plots indicate characteristic density reductions with rising temperature, reflecting each nanoparticle combination's thermal expansion behavior. MWCNT-based hybrids (Fe_3O_4-MWCNT, TiO_2-MWCNT) exhibit sharper density changes, while metal oxides (Al_2O_3/SiO_2, TiO_2-SiO_2) show more moderate responses.

Figure 13 illustrates concentration-dependent density profiles for volume fractions from 0.25% to 2.00%, capturing particle-fluid interactions where higher nanoparticle concentrations increase bulk density. Noble metal hybrids (Ag-GNP) show steep density rises, while graphene-based systems (Co_3O_4/rGO) display gradual increases. Together, these figures provide crucial insights into hybrid nanofluid density behavior under different conditions.

Figure 14 presents three-dimensional scatter plots illustrating the complex interplay between temperature, volume concentration, and density across various nanoparticle combinations. These plots demonstrate the multifaceted nature of parameter interactions, highlighting system-specific behavioral patterns that necessitate sophisticated modeling approaches.

B. BASELINE MODEL PERFORMANCE ANALYSIS

The direct implementation results of fourteen distinct machine learning and deep learning models for hybrid

FIGURE 9. Correlation matrix of all the thermophysical and compositional features.

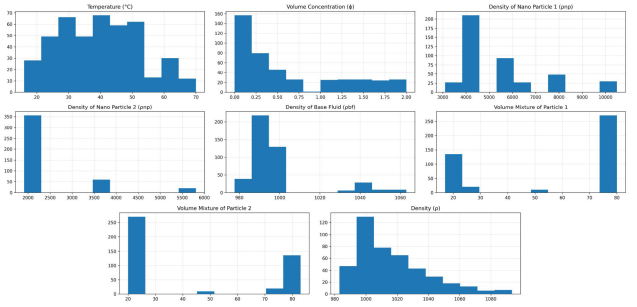

FIGURE 10. Histogram for all the features of the dataset.

nanofluid density prediction are presented in Table 4. The detailed results table, encompassing twenty-eight models, including both direct and ensembled variants, is provided in the Appendix (Table 16). These results represent the baseline performance of each model without any augmentation or optimization techniques, evaluated using standard metrics, including R^2 Score, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Explained Variance Score, Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (SMAPE), and Maximum Error.

Among the implemented models, the RNN stacked with linear regression demonstrated superior performance, achieving an R^2 score of 0.99803 and the lowest MSE (1.25213), indicating exceptional accuracy in density predictions. The model also achieved the lowest MAE (0.60977) and MAPE (0.00059), suggesting consistent performance across the dataset. Following closely, the Neural Network embedded with linear regression exhibited strong predictive capabilities with an R^2 score of 0.99543 and RMSE of 1.70794, while XGBoost showed robust performance with an R^2 score of 0.99345 and relatively low error metrics.

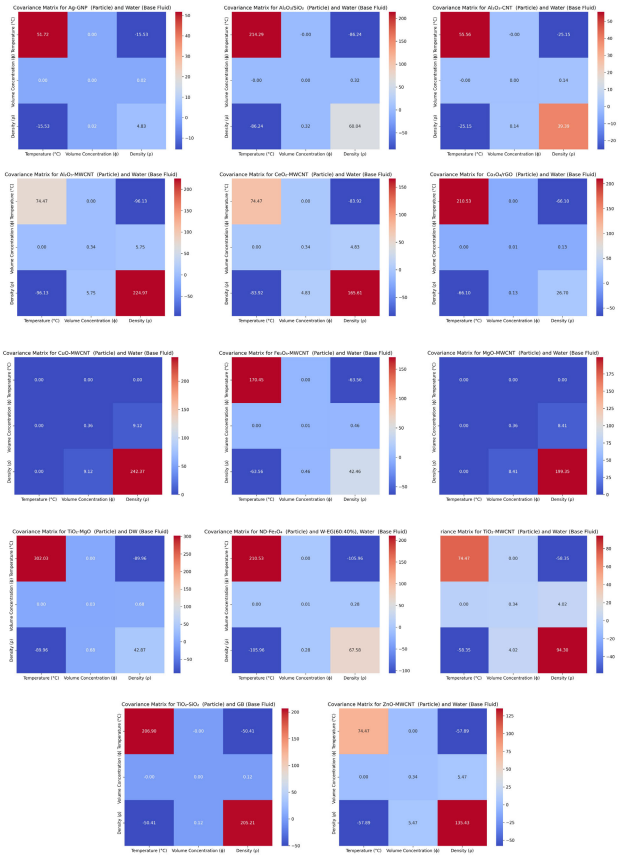

FIGURE 11. Covariance matrix analysis of temperature, volume concentration, and density interactions across multiple nanoparticle-base fluid systems.

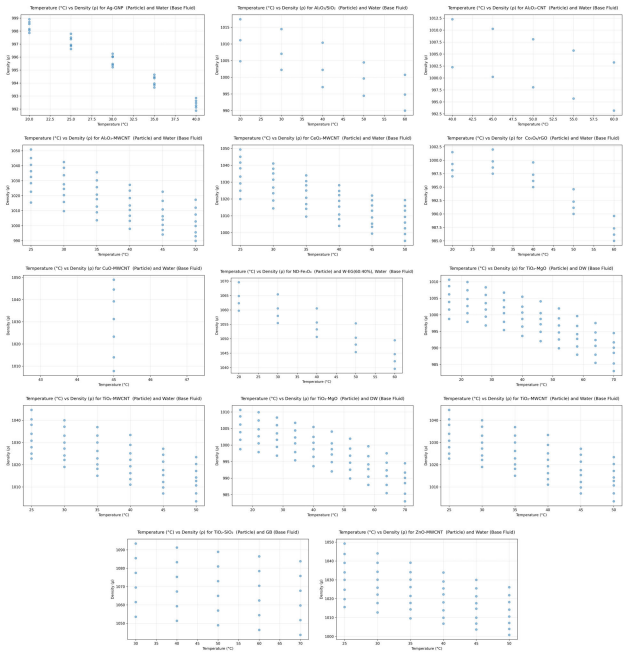

FIGURE 12. Temperature-density relationships across each nanofluid.

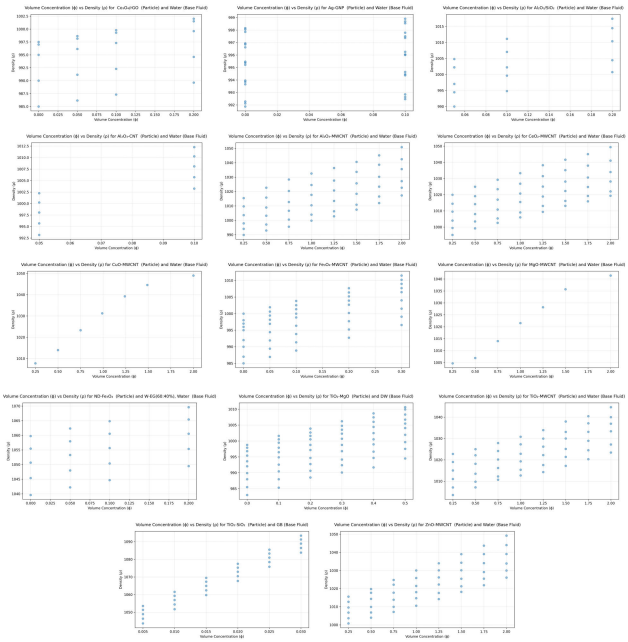

FIGURE 13. Volume concentration-density relationships across each nanofluid.

Traditional machine learning approaches, including Decision Tree and Random Forest, showed moderate performance with R^2 scores of 0.95963 and 0.976, respectively. However, these models exhibited higher error metrics compared to the hybrid neural network approaches. Ridge Regression and Poisson Regression demonstrated relatively lower performance among all models, with R^2 scores around 0.940, indicating limitations in capturing complex relationships within the dataset.

Deep learning models augmented with linear regression generally outperformed their traditional counterparts. The LSTM and GRU, both stacked with linear regression, achieved R^2 scores of 0.98699 and 0.95957, respectively, demonstrating the effectiveness of recurrent architectures in capturing temporal patterns within the data. The CNN and linear regression model combination showed competitive performance with an R^2 score of 0.95315, though with slightly higher error metrics compared to other deep learning approaches.

These baseline results establish RNN stacked along with linear regression as the most promising architecture for hybrid nanofluid density prediction, with Neural Network combined with linear regression and XGBoost as strong alternatives.

C. AUGMENTATION

The use of autoencoder-based and Gaussian noise augmentation significantly improved model predictions across the dataset. Augmented data maintained key physical

TABLE 4. Results of baseline implementation of the predictive models.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.95963	25.77969	5.07737	3.37809	0.96115	0.00328	0.32930	26.32067
Random Forest	0.97600	15.31988	3.91406	2.48769	0.97694	0.00240	0.24126	20.53880
Ridge Regression	0.94048	38.00359	6.16470	4.23374	0.94059	0.00411	0.41062	26.36109
Poisson Regression	0.94035	38.08850	6.17158	4.21203	0.94049	0.00409	0.40845	27.11793
Gradient Boosting	0.98618	8.82500	2.97069	2.28362	0.98618	0.00222	0.22271	7.99069
LightGBM	0.99381	3.94800	1.98695	1.09364	0.99382	0.00105	0.10581	10.31864
CatBoost	0.97027	18.98128	4.35675	2.75896	0.97034	0.00266	0.26637	17.86869
XGBoost	0.99345	4.17769	2.04394	1.03149	0.99351	0.00099	0.09975	13.84403
Neural Network + LR	0.99543	2.91708	1.70794	0.64008	0.99550	0.00062	0.06211	14.60286
CNN + LR	0.95315	29.91203	5.46918	3.17013	0.95340	0.00306	0.30584	24.59039
RNN + LR	0.99803	1.25213	1.11898	0.60977	0.99804	0.00059	0.05953	6.85347
GRU + LR	0.95957	25.81558	5.08090	1.34701	0.95976	0.00130	0.13170	44.29007
LSTM + LR	0.98699	8.30460	2.88177	1.81083	0.98749	0.00176	0.17660	17.01468
Autoencoders + LR	0.98667	8.50618	2.91653	0.86026	0.98675	0.00083	0.08279	24.18599

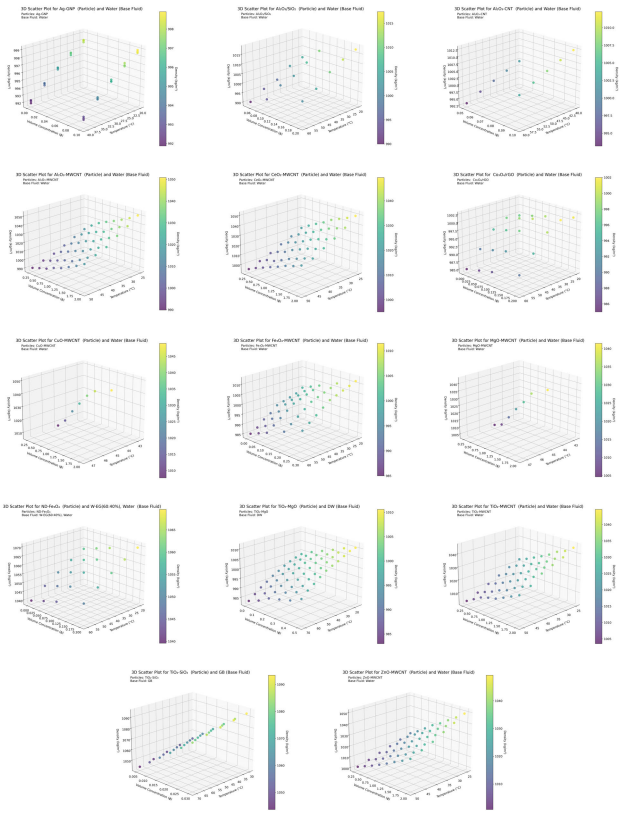

FIGURE 14. Three-dimensional scatter plot showing the relationships between temperature (°C), volume concentration (ϕ), and density (kg/m^3) for various nanoparticle-base fluid combinations.

relationships while expanding training samples, ensuring statistical consistency with the original data. These techniques were applied to fourteen machine learning models, ranging from traditional to advanced deep learning-based models. The comparative analysis highlights their effectiveness in enhancing model performance and robustness for nanofluid density prediction. The comprehensive tables of augmentation results, encompassing twenty-eight models, including both direct and ensembled variants, are provided in the Appendix (Table 17 and Table 18.

1) AUTOENCODER AUGMENTATION ANALYSIS

The implementation of autoencoder-based data augmentation demonstrated unique and distinguishable patterns across the evaluated models. A detailed summary of the results obtained from applying this augmentation technique to all models is presented in Table 5.

The Decision Tree algorithm emerged as the superior performer, achieving an exceptional R^2 score of 0.99999 and demonstrating remarkable precision with the lowest MSE (0.00132) among all tested models. This outstanding performance suggests that the autoencoder-augmented data particularly complements the hierarchical decision-making process inherent to tree-based algorithms.

The gradient boosting family of models exhibited strong performance, with XGBoost achieving an R^2 score of 0.99997 and maintaining low error metrics (MSE: 0.01796, RMSE: 0.13404). CatBoost showed similar robustness with an R^2 score of 0.99975, though with slightly higher error metrics (MSE: 0.15911, RMSE: 0.39889). This pattern indicates that ensemble boosting methods effectively capture the complex relationships in autoencoder-augmented nanofluid density data.

Among the neural network architectures, the RNN and linear regression combination demonstrated exceptional accuracy ($R^2 = 0.9999$, MSE $= 0.0014$), outperforming both standard neural networks and CNN approaches. GRU stacked with linear regression also showed strong performance ($R^2 = 0.9999$, MSE $= 0.0417$), suggesting that recurrent architectures are particularly well-suited for processing autoencoder-augmented nanofluid data. Conversely, CNN with linear regression stacking showed relatively lower performance ($R^2 = 0.9657$), indicating that convolutional architectures might be less optimal for this specific augmentation approach.

Traditional regression models, particularly Ridge Regression and Poisson Regression, exhibited the highest error metrics (MSE: 33.37189 and 40.03929 respectively), sug-

TABLE 5. Autoencoder-based Augmentation results on all the predictive models.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.0084	0.99999	0	0.00084	0.19
Random Forest	0.99827	1.103	1.05023	0.67279	0.99835	0.00065	0.06544	5.22739
Ridge Regression	0.94774	33.37189	5.77684	4.13048	0.94803	0.004	0.40024	16.5062
Poisson Regression	0.9373	40.03929	6.32766	4.43851	0.93737	0.0043	0.4307	22.81524
Gradient Boosting	0.99004	6.35673	2.52125	1.97525	0.99005	0.00192	0.19277	7.32955
LightGBM	0.99227	4.93188	2.22078	1.40012	0.9923	0.00135	0.13556	8.57836
CatBoost	0.99975	0.15911	0.39889	0.30735	0.99975	0.0003	0.03006	1.30951
XGBoost	0.99997	0.01796	0.13404	0.09434	0.99997	0.00009	0.00925	0.50286
Neural Network + LR	0.99872	0.81431	0.90239	0.51181	0.99872	0.00049	0.04988	5.53633
CNN + LR	0.9657	21.8847	4.6781	2.7732	0.9658	0.0027	0.268	19.2691
RNN + LR	0.9999	0.0014	0.037	0.012	0.9999	0	0.0012	0.189
GRU + LR	0.9999	0.0417	0.2042	0.1484	0.9999	0.0001	0.0146	0.683
LSTM + LR	0.9998	0.1117	0.3342	0.2716	0.9998	0.0003	0.0267	0.8133
Autoencoders + LR	0.9995	0.3302	0.5746	0.3779	0.9995	0.0004	0.037	2.0666

TABLE 6. Gaussian Noise-based augmentation results on all the predictive models.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.00841	0.99999	0.00001	0.00084	0.19
Random Forest	0.99787	1.35898	1.16575	0.85358	0.9979	0.00083	0.08318	4.65284
Ridge Regression	0.94269	36.59759	6.04959	4.18539	0.94278	0.00407	0.40596	26.00603
Poisson Regression	0.94282	36.51496	6.04276	4.15183	0.94293	0.00403	0.40264	26.43655
Gradient Boosting	0.98769	7.86066	2.80369	2.0626	0.98809	0.00201	0.20069	9.97304
LightGBM	0.99571	2.73765	1.65459	0.95656	0.99572	0.00092	0.09252	8.95397
CatBoost	0.99913	0.55591	0.74559	0.55165	0.99913	0.00053	0.05344	2.25047
XGBoost	0.99954	0.29112	0.53956	0.39765	0.99955	0.00039	0.0389	1.66325
Neural Network + LR	0.99216	5.00108	2.23631	1.03348	0.9922	0.00099	0.0989	11.34499
CNN + LR	0.95804	26.79626	5.17651	2.80065	0.95813	0.0027	0.26934	24.00616
RNN + LR	0.9941	3.76879	1.94134	1.22394	0.99414	0.00119	0.11862	6.7996
GRU + LR	0.99112	5.66879	2.38092	1.1218	0.99119	0.00108	0.10744	11.81081
LSTM + LR	0.99201	5.10528	2.25949	0.99489	0.99209	0.00095	0.09493	10.60134
Autoencoders + LR	0.99064	5.97641	2.44467	1.21602	0.99078	0.00117	0.11641	11.14089

TABLE 7. Performance metrics of deep learning models with the best augmentation.

Final Models	Aug.	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Neural Network + LR	A1	0.99872	0.81431	0.90239	0.51181	0.99872	0.00049	0.04988	5.53633
CNN + LR	A1	0.9657	21.8847	4.6781	2.7732	0.9658	0.0027	0.268	19.2691
RNN + LR	A1	0.9999	0.0014	0.037	0.012	0.9999	0	0.0012	0.189
GRU + LR	A1	0.9999	0.0417	0.2042	0.1484	0.9999	0.0001	0.0146	0.683
LSTM + LR	A1	0.9998	0.1117	0.3342	0.2716	0.9998	0.0003	0.0267	0.8133
Autoencoders + LR	A1	0.9995	0.3302	0.5746	0.3779	0.9995	0.0004	0.037	2.0666

gesting that these simpler models may struggle to capture the complexity introduced by autoencoder augmentation.

2) GAUSSIAN NOISE AUGMENTATION ANALYSIS

The application of Gaussian noise augmentation yielded distinctly varied performance patterns across the range of models. A comprehensive summary of the results obtained from this augmentation technique is provided in Table 6. The Decision Tree model maintained its superior performance ($R^2 = 0.99999$, MSE $= 0.00132$), demonstrating remarkable resilience to noise-based augmentation. This consistency across both augmentation techniques underscores the robustness of tree-based approaches in nanofluid density prediction.

The ensemble methods showed varying degrees of adaptation to Gaussian noise. XGBoost maintained high accuracy ($R^2 = 0.99954$) with relatively low error metrics (MSE $=$ 0.29112, RMSE $= 0.53956$), while CatBoost demonstrated similar resilience ($R^2 = 0.99913$, MSE $= 0.55591$). However, these error metrics were notably higher compared to their performance under autoencoder augmentation, suggesting some sensitivity to noise-based data expansion.

A significant performance shift was observed in neural network architectures under Gaussian noise augmentation. The RNN stacking with linear regression experienced a considerable increase in error metrics (MSE $= 3.76879$, RMSE $= 1.94134$) compared to its autoencoder augmentation performance, while maintaining a respectable R^2 score of 0.9941. This suggests that recurrent neural architectures may be more sensitive to Gaussian noise compared to autoencoder-generated variations.

The regression-based models showed similar patterns of higher error metrics under Gaussian noise, with Ridge Regression and Poisson Regression exhibiting the highest MSE values (36.59759 and 36.51496, respectively). This consistent pattern across both augmentation techniques

suggests that these models might be fundamentally less suitable for augmented nanofluid density prediction tasks, regardless of the augmentation method employed.

Notably, the deep learning models (Neural Network+LR, CNN+LR, GRU+LR) all showed increased error metrics under Gaussian noise compared to autoencoder augmentation, suggesting that these architectures might be more sensitive to random noise-based perturbations in the training data. This observation could inform future choices in data augmentation strategies for similar prediction tasks.

3) COMPARATIVE ANALYSIS OF DEEP LEARNING MODELS WITH OPTIMAL AUGMENTATION

This subsection presents a comprehensive analysis of the deep learning models' performance metrics with the best augmentations (Aug.). Autoencoder-based augmentation (A1) demonstrated superior results compared to Gaussian noise augmentation (A2). Table 7 summarizes the performance metrics across various deep learning architectures.

The empirical results indicate that recurrent neural networks (RNN) coupled with linear regression achieved exceptional performance, demonstrating the highest R^2 score of 0.9999 and minimal error metrics (MSE: 0.0014, RMSE: 0.037, MAE: 0.012). This architecture also exhibited remarkable precision with a MAPE of 0 and SMAPE of 0.0012, suggesting near-perfect prediction capabilities. The GRU and LSTM architectures similarly demonstrated robust performance, with R^2 scores of 0.9999 and 0.9998 respectively, indicating strong model fitness. However, their error metrics were marginally higher than the RNN implementation, with the GRU showing slightly better performance (MSE: 0.0417) compared to LSTM (MSE: 0.1117). While the CNN-based model achieved a respectable R^2 score of 0.9657, it displayed notably higher error metrics (MSE: 21.8847, RMSE: 4.6781) compared to other architectures, suggesting potential limitations in capturing the underlying patterns in the data. The neural network with linear regression and autoencoder-based implementation showed strong performance with an R^2 score of 0.99872, though it exhibited higher maximum error (5.53633) compared to other architectures, indicating occasional significant deviations in predictions. These results demonstrate the superior performance of recurrent architectures (RNN, GRU, LSTM) when combined with autoencoder-based augmentation for this application, with the RNN combined with linear regression model emerging as the optimal choice based on comprehensive metric evaluation.

D. OPTIMIZATION

Model optimization plays a vital role in enhancing machine learning algorithms' predictive capabilities through hyperparameter tuning. This study implemented two nature-inspired optimization techniques: Grey Wolf Optimization (GWO) and Particle Swarm Optimization (PSO). Following the analysis of augmentation results, autoencoder-based augmentation demonstrated the best performance. Consequently, it was selected, and optimization using either GWO or PSO was subsequently applied. These methods were exclusively applied to traditional machine learning models, as deep learning architectures inherently utilize gradient-based optimization through their backpropagation training process. The comparative analysis of GWO and PSO provides valuable insights into their effectiveness in enhancing model performance for nanofluid density prediction. The comprehensive tables of optimization results, encompassing sixteen models, including both direct and ensembled variants, are presented in the Appendix (Table 19 and Table 20).

1) GREY WOLF OPTIMIZATION ANALYSIS

Grey Wolf Optimization implementation demonstrated substantial performance improvements across the machine learning algorithms, as evidenced in Table 8. The gradient boosting family of models exhibited exceptional results, with XGBoost, CatBoost, and Gradient Boosting all achieving near-perfect R^2 scores of 0.99999. These models also demonstrated remarkably low error metrics, with MSE values of 0.00134, 0.00133, and 0.00091 respectively, indicating the effectiveness of GWO in optimizing their hyperparameters. The Decision Tree model showed equally impressive performance under GWO optimization, achieving an R^2 score of 0.99999 and the lowest MAE (0.0084) among all models. This exceptional performance suggests that GWO effectively identified optimal decision boundaries and tree structure parameters. The LightGBM and Random Forest models also showed strong performance with R^2 scores above 0.998, though with slightly higher error metrics compared to other ensemble methods. Notably, the traditional regression models (Ridge and Poisson Regression) showed relatively lower performance even after GWO optimization, with R^2 scores of 0.95131 and 0.94914 respectively. Their higher error metrics (MSE: 31.09297 and 32.47437) suggest that these simpler models might have fundamental limitations in capturing complex nanofluid density relationships, regardless of optimization.

2) PARTICLE SWARM OPTIMIZATION ANALYSIS

Particle Swarm Optimization (PSO) produced results comparable to GWO, highlighting the robustness of both approaches. Table 9 presents the outcomes of applying PSO to the machine learning models. The ensemble methods maintained their superior performance, with XGBoost and CatBoost achieving identical R^2 scores of 0.99999 and very similar error metrics (MSE: 0.00133 and 0.00134 respectively). PSO optimization showed effectiveness with the Gradient Boosting model, achieving a slightly better MSE of 0.00093 compared to GWO's 0.00091. The Decision Tree model maintained its strong performance under PSO, replicating the exact metrics achieved under GWO ($R^2 = 0.99999$, MSE $= 0.00132$), suggesting that both optimization techniques converged to similar optimal solutions for this model architecture. The consistency in performance metrics between GWO and PSO extended to the LightGBM and

TABLE 8. Grey wolf optimization results on machine learning models.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Random Forest	0.99842	1.00668	1.00333	0.68275	0.99849	0.00066	0.06639	4.75172
Ridge Regression	0.95131	31.09297	5.5761	4.07102	0.95136	0.00394	0.39436	19.10887
Poisson Regression	0.94914	32.47437	5.69862	4.02523	0.9492	0.0039	0.39041	22.65105
Gradient Boosting	0.99999	0.00091	0.03022	0.01372	0.99999	0.00001	0.00136	0.15689
LightGBM	0.99892	0.68737	0.82908	0.59521	0.99892	0.00057	0.05788	2.76293
CatBoost	0.99999	0.00133	0.03654	0.01155	0.99999	0.00001	0.00115	0.18961
XGBoost	0.99999	0.00134	0.03664	0.0113	0.99999	0.00001	0.00112	0.1896

Random Forest models, with nearly identical R^2 scores and error metrics. This pattern suggests that both optimization techniques effectively explored the hyperparameter space to find optimal configurations. The regression models similarly showed consistent performance across both optimization techniques, maintaining the same limitations observed under GWO.

3) COMPARATIVE ANALYSIS OF MACHINE LEARNING MODELS WITH OPTIMAL OPTIMIZATION TECHNIQUES

This subsection provides a detailed analysis of the performance metrics of machine learning models enhanced through optimization techniques (OT) following augmentation, with a comparative evaluation of Grey Wolf Optimization (O1) and Particle Swarm Optimization (O2). Table 10 illustrates the comprehensive performance metrics across various machine learning architectures with their respective optimal optimization approaches.

E. OVERALL RESULTS ANALYSIS

The empirical results demonstrate exceptional performance metrics across several models when optimized using Grey Wolf Optimization (O1). Decision Tree, Gradient Boosting, CatBoost, and XGBoost algorithms achieved remarkably high R^2 scores of 0.99999, with Gradient Boosting exhibiting the lowest error metrics (MSE: 0.00091, RMSE: 0.03022), indicating superior prediction accuracy. The ensemble-based methods, particularly the boosting algorithms, demonstrated consistently strong performance when optimized with GWO, maintaining extremely low error rates (MAPE: 0.00001, SMAPE $<$ 0.002). LightGBM, optimized through Particle Swarm Optimization (O2), achieved strong results (R^2 score: 0.99892) but showed marginally higher error metrics compared to GWO-optimized models. Traditional algorithms like Ridge Regression and Poisson Regression exhibited relatively higher error metrics (MSE: 31.09297 and 32.47437 respectively) despite optimization, while Random Forest achieved a strong R^2 score (0.99842) but showed higher error metrics compared to other ensemble methods. These findings indicate Grey Wolf Optimization's superior performance over Particle Swarm Optimization, with Gradient Boosting emerging as the optimal choice when considering the comprehensive set of performance metrics.

The comprehensive evaluation of model performance across different experimental configurations reveals significant insights into the effectiveness of various machine learning and deep learning approaches for nanofluid density prediction. Our analysis encompasses two distinct scenarios: the performance evaluation of all models with optimal enhancements (IV-E1), and a detailed comparison between baseline and enhanced configurations of the top 5 performing models (IV-E2). The comprehensive table of the models enhanced with best augmentation and optimizations, encompassing all twenty-eight utilized models, are presented in the Appendix (Table 21).

1) PERFORMANCE ANALYSIS OF MACHINE LEARNING AND DEEP LEARNING MODELS WITH ENHANCED CONFIGURATIONS

The integration of autoencoder augmentation for the deep learning models and augmentation combined with Grey Wolf Optimization (GWO) or Particle Swarm Optimization (PSO) for machine learning models demonstrated significant performance enhancements as examined in this study. The ensemble learning approaches exhibited exceptional predictive accuracy for the deep learning models and the machine learning models like XGBoost, CatBoost, and Gradient Boosting achieved R^2 scores of 0.99999. These models demonstrated notably low error metrics, with MSE values of 0.00133, 0.00133, and 0.00091 respectively, validating their robust predictive capabilities under the enhanced conditions. In the domain of deep learning architectures, the GRU with linear regression stacking emerged as the superior performer ($R^2 = 0.9999$, MSE $= 0.0417$) among the neural network variants. The LSTM and linear regression configuration similarly demonstrated strong predictive capabilities ($R^2 = 0.9998$, MSE $= 0.1117$), while the CNN and linear regression combination exhibited comparatively lower performance metrics ($R^2 = 0.9657$, MSE $= 21.8847$). The comprehensive implementation results and performance metrics for all models are presented in Table 11.

2) COMPARATIVE ANALYSIS OF BASELINE IMPLEMENTATION VERSUS ENHANCED CONFIGURATIONS OF TOP-PERFORMING MODELS

Based on the comprehensive performance metrics presented in Table 11, the five most effective models were selected for detailed comparative analysis in their direct implementation and enhanced configurations. These models—Decision Tree, Gradient Boosting, CatBoost, XGBoost, and

TABLE 9. Particle swarm optimization results on machine learning models.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Random Forest	0.99838	1.03182	1.01578	0.67754	0.99844	0.00065	0.06592	4.78449
Ridge Regression	0.95131	31.09297	5.5761	4.07102	0.95136	0.00394	0.39436	19.10887
Poisson Regression	0.94914	32.47437	5.69862	4.02523	0.9492	0.0039	0.39041	22.65105
Gradient Boosting	0.99999	0.00093	0.03062	0.01177	0.99999	0.00001	0.00117	0.14902
LightGBM	0.99892	0.68733	0.82905	0.5952	0.99892	0.00057	0.05787	2.76282
CatBoost	0.99999	0.00134	0.03664	0.01173	0.99999	0.00001	0.00116	0.19045
XGBoost	0.99999	0.00133	0.03651	0.01045	0.99999	0.00001	0.00104	0.1896

TABLE 10. Performance metrics of machine learning models with best optimization.

Final Models	OT	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	O1	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Random Forest	O1	0.99842	1.00668	1.00333	0.68275	0.99849	0.00066	0.06639	4.75172
Ridge Regression	O1	0.95131	31.09297	5.5761	4.07102	0.95136	0.00394	0.39436	19.10887
Poisson Regression	O1	0.94914	32.47437	5.69862	4.02523	0.9492	0.0039	0.39041	22.65105
Gradient Boosting	O1	0.99999	0.00091	0.03022	0.01372	0.99999	0.00001	0.00136	0.15689
LightGBM	O2	0.99892	0.68733	0.82905	0.5952	0.99892	0.00057	0.05787	2.76282
CatBoost	O1	0.99999	0.00133	0.03654	0.01155	0.99999	0.00001	0.00115	0.18961
XGBoost	O1	0.99999	0.00134	0.03664	0.0113	0.99999	0.00001	0.00112	0.1896

TABLE 11. Performance metrics of machine learning and deep learning models with enhanced configurations and techniques.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Random Forest	0.99842	1.00668	1.00333	0.68275	0.99849	0.00066	0.06639	4.75172
Ridge Regression	0.95131	31.09297	5.5761	4.07102	0.95136	0.00394	0.39436	19.10887
Poisson Regression	0.94914	32.47437	5.69862	4.02523	0.9492	0.0039	0.39041	22.65105
Gradient Boosting	0.99999	0.00091	0.03022	0.01372	0.99999	0.00001	0.00136	0.15689
LightGBM	0.99892	0.68733	0.82905	0.5952	0.99892	0.00057	0.05787	2.76282
CatBoost	0.99999	0.00133	0.03654	0.01155	0.99999	0.00001	0.00115	0.18961
XGBoost	0.99999	0.00133	0.03651	0.01045	0.99999	0.00001	0.00104	0.1896
Neural Network + LR	0.99872	0.81431	0.90239	0.51181	0.99872	0.00049	0.04988	5.53633
CNN + LR	0.9657	21.8847	4.6781	2.7732	0.9658	0.0027	0.268	19.2691
RNN + LR	0.9999	0.0014	0.037	0.012	0.9999	0	0.0012	0.189
GRU + LR	0.9999	0.0417	0.2042	0.1484	0.9999	0.0001	0.0146	0.683
LSTM + LR	0.9998	0.1117	0.3342	0.2716	0.9998	0.0003	0.0267	0.8133
Autoencoders + LR	0.9995	0.3302	0.5746	0.3779	0.9995	0.0004	0.037	2.0666

TABLE 12. Comparative performance metrics of top-5 baseline models.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.95963	25.77969	5.07737	3.37809	0.96115	0.00328	0.3293	26.32067
Gradient Boosting	0.98618	8.825	2.97069	2.28362	0.98618	0.00222	0.22271	7.99069
CatBoost	0.97027	18.98128	4.35675	2.75896	0.97034	0.00266	0.26637	17.86869
XGBoost	0.99345	4.17769	2.04394	1.03149	0.99351	0.00099	0.09975	13.84403
RNN + LR	0.99803	1.25213	1.11898	0.60977	0.99804	0.00059	0.05953	6.85347

TABLE 13. Comparative performance metrics of the top-5 enhanced and improved models.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Gradient Boosting	0.99999	0.00091	0.03022	0.01372	0.99999	0.00001	0.00136	0.15689
CatBoost	0.99999	0.00133	0.03654	0.01155	0.99999	0.00001	0.00115	0.18961
XGBoost	0.99999	0.00133	0.03651	0.01045	0.99999	0.00001	0.00104	0.1896
RNN + LR	0.9999	0.0014	0.037	0.012	0.9999	0	0.0012	0.189

RNN with linear regression stacking—demonstrated superior performance characteristics across multiple evaluation metrics. In their direct implementation without augmentation or optimization, these models exhibited varying degrees of predictive capability. XGBoost maintained the highest performance among directly implemented models (R^2 = 0.99345, MSE = 4.17769), though with notably higher error metrics compared to its enhanced counterpart. The RNN and linear regression embedding demonstrated remarkable resilience in direct implementation, achieving an R^2 score of

TABLE 14. Comparative performance metrics of the top-5 enhanced models using 5-Fold Cross-Validation.

Final Models	R^2 Score	MSE	RMSE	MAE	SMAPE
Decision Tree	0.999890 ± 0.000064	11.7313 ± 5.7428	3.3415 ± 0.7520	2.4404 ± 0.3480	0.2392 ± 0.0341
Gradient Boosting	0.999973 ± 0.000026	2.8331 ± 2.5207	1.5553 ± 0.6435	0.9111 ± 0.2520	0.0902 ± 0.0240
CatBoost	0.999968 ± 0.000022	3.5243 ± 2.1617	1.7967 ± 0.5442	0.9336 ± 0.1939	0.0916 ± 0.0189
XGBoost	0.999975 ± 0.000024	2.6019 ± 2.3227	1.4882 ± 0.6223	0.8414 ± 0.2055	0.0829 ± 0.0199
RNN + LR	0.999989 ± 0.000008	1.118467 ± 0.739541	1.006301 ± 0.325309	0.567196 ± 0.135853	0.081913 ± 0.012630

0.99803 with relatively low error metrics (MSE = 1.25213), suggesting inherent architectural advantages for nanofluid density prediction. These baseline implementation results are presented in Table 12.

The application of autoencoder augmentation combined with Grey Wolf Optimization (GWO) led to substantial improvements across all models, as shown in Table 13. The enhanced configurations achieved exceptional performance metrics, with R^2 scores of 0.99999 for the traditional machine learning models and 0.9999 for the RNN stacked with linear regression architecture. The Gradient Boosting model achieved the lowest MSE (0.00091) among all configurations, while the Decision Tree model demonstrated the lowest MAE (0.0084), indicating different strengths in error minimization.

The comparative analysis reveals dramatic improvements between direct and enhanced implementations. For instance, the Decision Tree model's performance improved significantly from an R^2 score of 0.95963 to 0.99999. Similarly, XGBoost demonstrated a remarkable 99.97% reduction in prediction error, with MSE improving from 4.17769 to 0.00133. CatBoost showed substantial improvement in RMSE, reducing from 4.35675 to 0.03654, highlighting the effectiveness of the enhancement strategies.

Looking at the K-fold cross-validation results (K = 5) presented in Table 14, the models demonstrate excellent stability and reliability in their performance, effectively ruling out overfitting concerns. All the top five models show remarkably consistent performance metrics with very small standard deviations across all evaluation criteria. The high R^2 values (all above 0.99) combined with their minimal standard deviations (in the order of 10^{-5}) indicate that the models are generalizing well across different data splits. These tight confidence intervals across all metrics strongly suggest that the models perform consistently across different subsets of the data, effectively avoiding overfitting.

The consistent superior performance of ensemble methods (XGBoost, CatBoost, Gradient Boosting) and the Decision Tree model in their enhanced configurations suggests these architectures are particularly well-suited for nanofluid density prediction when properly optimized and trained on augmented data. The strong performance of the RNN and linear regression stacking among deep learning architectures indicates that recurrent neural networks offer a viable alternative when gradient-based optimization is preferred, maintaining robust performance across both direct and enhanced implementations.

The visual performance analysis of the optimally enhanced models reveals distinct patterns in their predictive capabilities across different density ranges. Figure 15 illustrates the Actual versus Predicted density relationships, while Figure 16 presents the comparative density distribution analysis between actual and predicted values. The XGBoost implementation demonstrates superior prediction accuracy, exhibiting minimal deviation from the perfect prediction line across the entire density spectrum (1000-1080 kg/m^3). This consistent performance is characterized by exceptionally tight point clustering along the reference line. Similarly, the CatBoost implementation maintains high prediction fidelity, though displaying marginally increased variation in the upper density ranges (1060-1080 kg/m^3). The Gradient Boosting implementation preserves robust predictive capabilities, with slight deviations observed primarily in the mid-range densities (1020-1040 kg/m^3). The Decision Tree implementation, while maintaining strong overall performance, exhibits moderately increased scatter patterns compared to the ensemble-based implementations, particularly in the median density range. The hybrid implementation of RNN with linear regression demonstrates good alignment with the reference line but displays the most pronounced scatter among the enhanced implementations, particularly evident in higher density ranges.

The residual analysis presented in Figure 17 provides detailed insight into the error patterns of these enhanced implementations. Complementary performance metrics are visualized in Figures 18, 19, and 20, presenting R^2 scores and error metrics (MSE, MAPE, SMAPE, and RMSE) values respectively, offering a comprehensive quantitative assessment of each implementation's predictive capabilities.

F. COMPUTATIONAL PERFORMANCE ANALYSIS OF TOP ENHANCED MODELS

Table 15 presents the computational performance metrics for the five highest-performing models in terms of prediction accuracy, analyzing their training time, testing time, average inference time, and model size requirements. The analysis reveals significant variations in computational efficiency across different model architectures. The Decision Tree model demonstrates remarkable computational efficiency with a training time of 0.01584 seconds and a compact model size of 59.29 KB, making it the most lightweight solution among the evaluated models. Despite its modest resource requirements, it maintains competitive testing and inference times of 0.00433 and 0.00005 seconds respectively. Gradient Boosting exhibits moderate computational demands with a

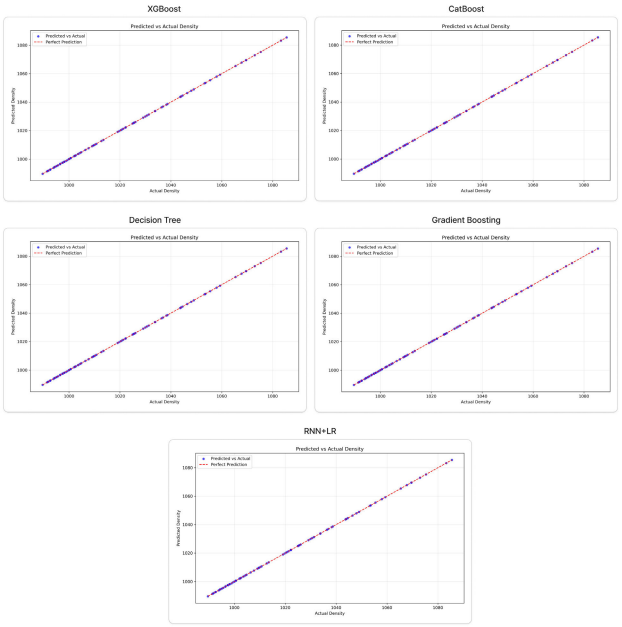

FIGURE 15. Actual vs Predicted density prediction line graph for the top five enhanced and improved models.

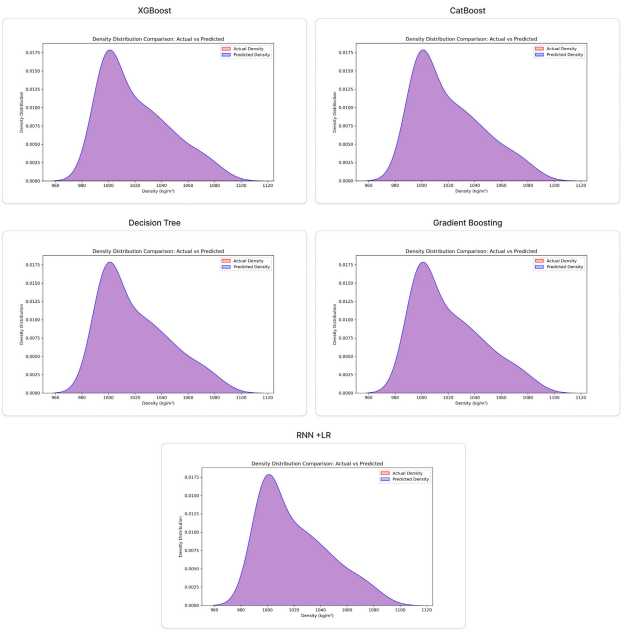

FIGURE 16. Density distribution comparison graph for the top five enhanced models.

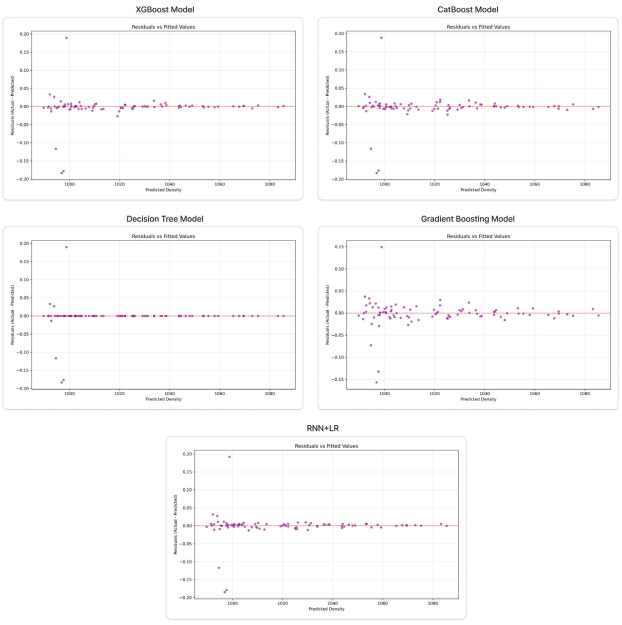

FIGURE 17. Residual vs Fitted prediction line graph for the top five enhanced models.

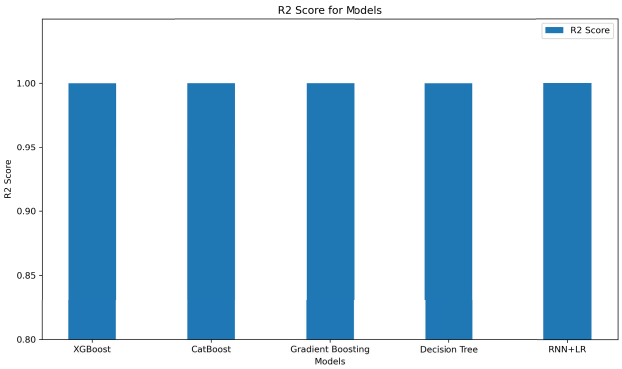

FIGURE 18. R^2 Scores for the top-5 enhanced and improved models.

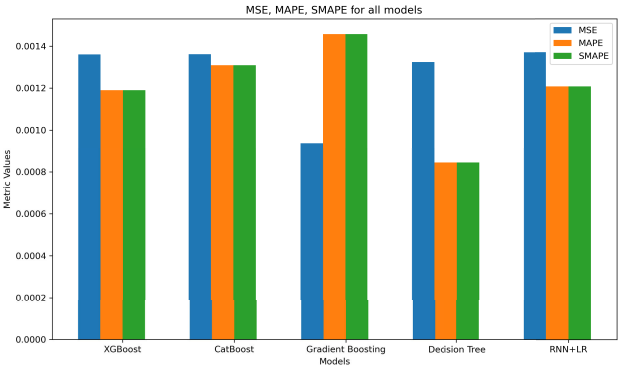

FIGURE 19. MSE, MAPE, and SMAPE for the top-5 enhanced and improved models.

training time of 0.36066 seconds and requires substantial storage at 4371.82 KB. However, it achieves efficient testing and inference times of 0.00013 and 0.00004 seconds respectively. Similarly, CatBoost shows increased training complexity at 0.82379 seconds with the largest model size of 4435.45 KB, while maintaining rapid inference capabilities at 0.00002 seconds. XGBoost achieves a balanced computational profile with a training time of 0.42796 seconds and a moderate model size of 2360.34 KB, demonstrating efficient inference performance at 0.00002 seconds. The hybrid RNN

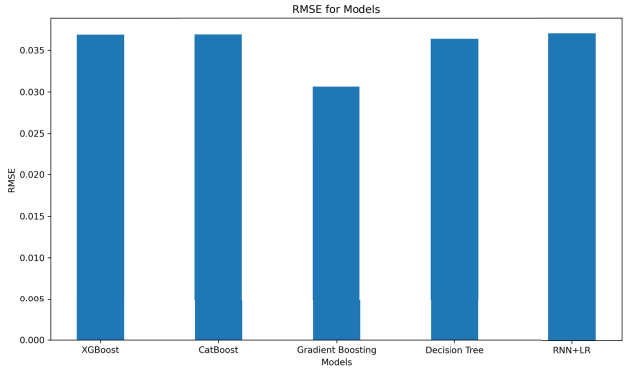

FIGURE 20. RMSE for the top-5 enhanced and improved models.

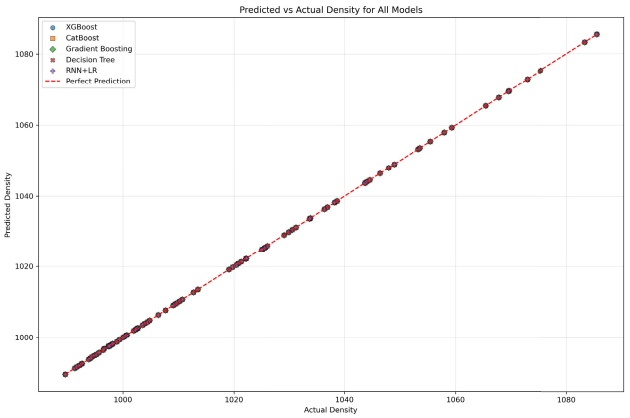

FIGURE 21. Actual vs Predicted density for the top-5 enhanced and improved models.

and linear regression architecture, despite its sophisticated nature, maintains reasonable resource requirements with a model size of 258.58 KB, though it requires the longest training time at 13.00958 seconds. Notably, it achieves the fastest testing and inference times at 10 microseconds (μs). These findings highlight the trade-offs between model complexity, computational requirements, and inference efficiency, providing valuable insights for practical implementation considerations in hybrid nanofluid density prediction applications.

G. COMPREHENSIVE ANALYSIS OF DENSITY PREDICTIONS

1) ACTUAL VS PREDICTED DENSITY ANALYSIS

The detailed performance evaluation of the top five enhanced models — XGBoost, CatBoost, Gradient Boosting, Decision Trees, and RNN with linear regression stacking — reveals remarkable accuracy in predicting hybrid nanofluid densities across all fourteen nanoparticle combinations. The actual versus predicted density plots demonstrate strong linear correlations, indicating excellent model performance after enhancement through autoencoder-based data augmentation and Grey Wolf Optimization (GWO). For metal oxide-based hybrid nanofluids (Al_2O_3/SiO_2, TiO_2-SiO_2, Al_2O_3-CNT), the predictions show particularly high accuracy with R^2 values approaching unity. The XGBoost and CatBoost models exhibit superior performance, with prediction points closely aligning along the ideal 45-degree line, suggesting minimal deviation between actual and predicted values. This high accuracy can be attributed to the models' ability to capture the complex interactions between the metal oxide nanoparticles and their influence on fluid density.

The carbon-based hybrid combinations (Fe_3O_4-MWCNT, Al_2O_3-MWCNT, TiO_2-MWCNT) display slightly more scattered predictions, though still maintaining excellent overall accuracy. The increased scatter likely results from the complex surface interactions between the carbon nanotubes and metal oxide particles. Nevertheless, the enhanced models successfully capture these intricate relationships, with the Gradient Boosting model showing strength in handling these

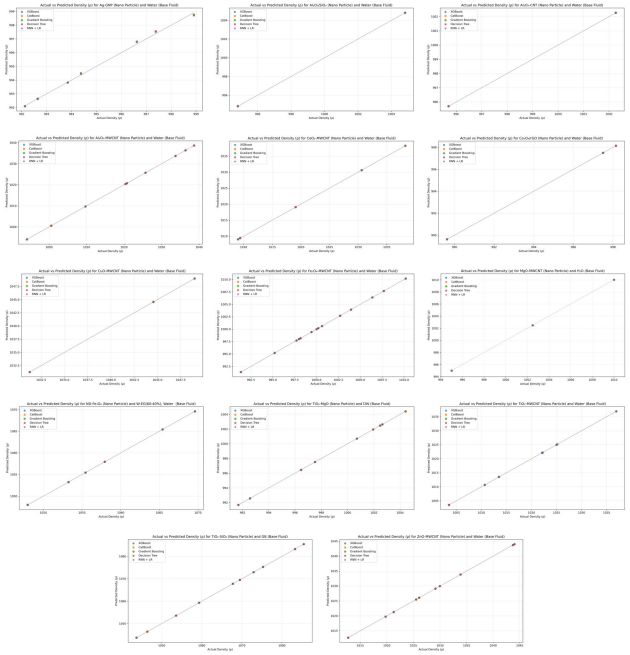

FIGURE 22. Experimental versus model-predicted densities using the top-5 enhanced and improved models for each nanofluid particle.

combinations. For noble metal and advanced material hybrids (Ag-GNP, ND-Fe_3O_4), the models demonstrate robust predictive capability despite the unique physicochemical properties of these materials. The RNN and linear regression stacked model, while generally performing well, shows slightly higher variance in predictions for these combinations, particularly at higher density values.

Figure 21 shows the overall Predicted vs Actual Densities of all the top 5 enhanced models. The corresponding graphs for each of the nanoparticles are in Figure 22.

2) TEMPERATURE DEPENDENCE ANALYSIS

The temperature dependence plots reveal crucial insights into the models' ability to capture thermal effects on hybrid

TABLE 15. Computational performance of the top-5 enhanced and improved models.

Final Models	Training Time (s)	Testing Time (s)	Avg. Inference Time (s)	Model Size (KB)
Decision Tree	0.01584	0.00433	0.00005	59.29
Gradient Boosting	0.36066	0.00013	0.00004	4371.82
CatBoost	0.82379	0.00017	0.00002	4435.45
XGBoost	0.42796	0.00200	0.00002	2360.34
RNN + LR	13.00958	0.00001	0.00001	258.58

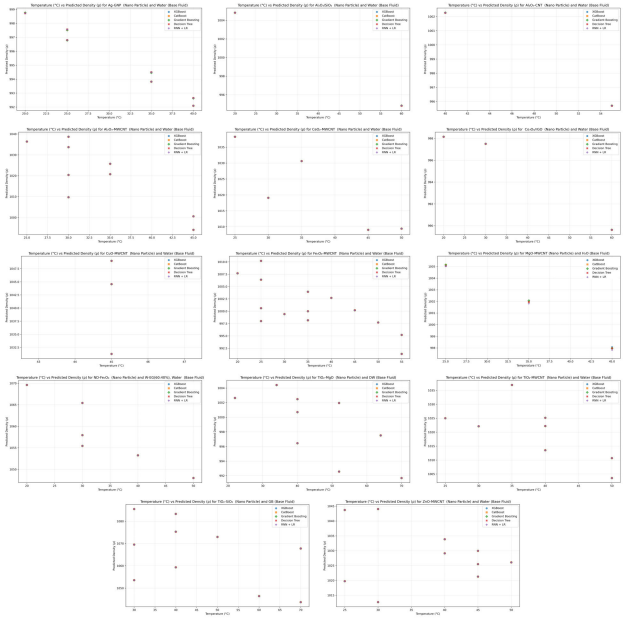

FIGURE 23. Temperature verses model-predicted densities using the top-5 enhanced and improved models for each nanofluid particle.

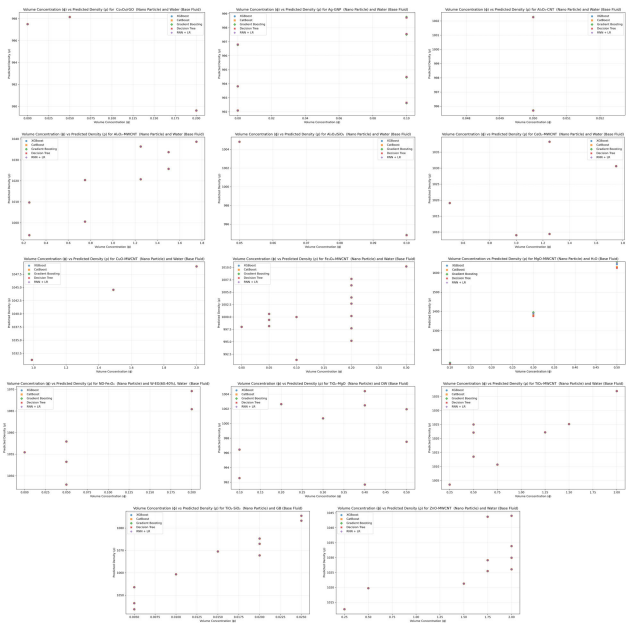

FIGURE 24. Volume Concentration verses model-predicted densities using the top five enhanced and improved models for each nanofluid particle.

nanofluid density. Across the temperature range of 20 °C–60 °C, all five enhanced models successfully predict the characteristic inverse relationship between temperature and density. For metal oxide hybrids, the predictions show a clear linear decrease in density with increasing temperature, with minimal scatter in the data points. The XGBoost and CatBoost models particularly excel in capturing this relationship, demonstrating consistent prediction accuracy across the entire temperature range. This suggests effective learning of the thermal expansion characteristics of these hybrid combinations.

Carbon nanotube-based hybrids (Fe_3O_4-MWCNT, TiO_2-MWCNT, CeO_2-MWCNT) exhibit more complex temperature-dependent behavior, with slightly nonlinear trends at higher temperatures. The enhanced models successfully capture these nuanced relationships, with the Gradient Boosting model showing strength in predicting the subtle variations in density response to temperature changes. The temperature sensitivity appears most pronounced in the noble metal hybrids (Ag-GNP) and advanced material combinations (ND-Fe_3O_4), where the density changes more rapidly with temperature. The models effectively predict this enhanced temperature sensitivity, though with slightly increased uncertainty at temperature extremes.

Figure 23 shows the graphs for temperature vs predicted densities for each of the nanoparticles.

3) VOLUME CONCENTRATION EFFECTS ANALYSIS

The volume concentration analysis reveals the models' capability to predict density variations across different nanoparticle loading levels. The relationship between volume concentration and density shows a predominantly linear trend, with some hybrid combinations exhibiting slight nonlinearity at higher concentrations. Metal oxide hybrid nanofluids demonstrate a strong linear increase in density with increasing volume concentration. All five enhanced models accurately capture this relationship, with the XGBoost and CatBoost models showing particularly tight prediction bands. This suggests excellent capability in modeling the particle loading effects on bulk fluid density.

The enhanced models successfully capture the slight deviation from linearity at higher concentrations, likely resulting from CNT agglomeration effects. The Gradient Boosting and Decision Tree models demonstrate robust

TABLE 16. Comprehensive model performance metrics.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.95963	25.77969	5.07737	3.37809	0.96115	0.00328	0.3293	26.32067
Decision Tree + LR	0.95963	25.77969	5.07737	3.37809	0.96115	0.00328	0.3293	26.32067
Random Forest	0.976	15.31988	3.91406	2.48769	0.97694	0.0024	0.24126	20.5388
Random Forest + LR	0.97628	15.14361	3.89148	2.48337	0.97707	0.0024	0.24083	20.40365
Ridge Regression	0.94048	38.00359	6.1647	4.23374	0.94059	0.00411	0.41062	26.36109
Ridge Regression + LR	0.36576	405.0146	20.12497	14.28601	0.39009	0.01375	1.38485	54.46312
Poisson Regression	0.94035	38.0885	6.17158	4.21203	0.94049	0.00409	0.40845	27.11793
Poisson Regression + LR	0.478	333.3423	18.25766	12.82248	0.52289	0.01233	1.24491	52.18415
Gradient Boosting	0.98618	8.825	2.97069	2.28362	0.98618	0.00222	0.22271	7.99069
Gradient Boosting + LR	0.98634	8.7213	2.95318	2.2758	0.98634	0.00221	0.22191	7.74093
LightGBM	0.99381	3.948	1.98695	1.09364	0.99382	0.00105	0.10581	10.31864
LightGBM + LR	0.88367	74.28694	8.61898	6.72632	0.89118	0.00659	0.65834	23.70509
CatBoost	0.97027	18.98128	4.35675	2.75896	0.97034	0.00266	0.26637	17.86869
CatBoost + LR	0.99383	3.93574	1.98387	1.08673	0.99383	0.00105	0.10512	10.27805
XGBoost	0.99345	4.17769	2.04394	1.03149	0.99351	0.00099	0.09975	13.84403
XGBoost + LR	0.95684	27.55938	5.2497	3.90528	0.9577	0.0038	0.38054	18.25614
Neural Network	0.6252	239.3416	15.47066	10.09043	0.69466	0.00981	0.97283	51.68853
Neural Network + LR	0.99543	2.91708	1.70794	0.64008	0.9955	0.00062	0.06211	14.60286
CNN	-0.70582	1089.324	33.0049	25.03799	-0.65856	0.02443	2.46182	70.36695
CNN + LR	0.95315	29.91203	5.46918	3.17013	0.9534	0.00306	0.30584	24.59039
RNN	0.79111	133.3948	11.54966	7.6847	0.79205	0.00747	0.7468	42.40411
RNN + LR	0.99803	1.25213	1.11898	0.60977	0.99804	0.00059	0.05953	6.85347
GRU	0.88222	75.20916	8.67232	4.98187	0.89006	0.00481	0.47923	39.85845
GRU + LR	0.95957	25.81558	5.0809	1.34701	0.95976	0.0013	0.1317	44.29007
LSTM	-894.4493	571826.9954	756.19243	755.77007	-6.67743	0.740348	117.55621	820.61567
LSTM + LR	0.98699	8.3046	2.88177	1.81083	0.98749	0.00176	0.1766	17.01468
Autoencoders	0.88477	73.57934	8.57784	4.7988	0.88891	0.00464	250.7969	38.54193
Autoencoders + LR	0.98667	8.50618	2.91653	0.86026	0.98675	0.00083	0.08279	24.18599

TABLE 17. Performance metrics of models after autoencoder data augmentation.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Decision Tree + LR	0.99999	0.00115	0.0339	0.0078	0.99999	0.00001	0.00081	0.185
Random Forest	0.99827	1.103	1.05023	0.67279	0.99835	0.00065	0.06544	5.22739
Random Forest + LR	0.9997	0.21	0.4581	0.3102	0.9997	0.00032	0.031	1.75
Ridge Regression	0.94774	33.37189	5.77684	4.13048	0.94803	0.004	0.40024	16.5062
Ridge Regression + LR	0.995	2.5	1.58	1.07	0.995	0.0011	0.11	7.1
Poisson Regression	0.9373	40.03929	6.32766	4.43851	0.93737	0.0043	0.4307	22.81524
Poisson Regression + LR	0.993	3	1.73	1.21	0.993	0.0014	0.14	8.2
Gradient Boosting	0.99004	6.35673	2.52125	1.97525	0.99005	0.00192	0.19277	7.32955
Gradient Boosting + LR	0.99998	0.0123	0.111	0.0805	0.99998	0.00007	0.0082	0.48
LightGBM	0.99227	4.93188	2.22078	1.40012	0.9923	0.00135	0.13556	8.57836
LightGBM + LR	0.99996	0.0179	0.134	0.0943	0.99996	0.00008	0.0092	0.5
CatBoost	0.99975	0.15911	0.39889	0.30735	0.99975	0.0003	0.03006	1.30951
CatBoost + LR	0.99978	0.12	0.3463	0.2605	0.99978	0.00028	0.0275	1.21
XGBoost	0.99997	0.01796	0.13404	0.09434	0.99997	0.00009	0.00925	0.50286
XGBoost + LR	0.99997	0.0156	0.1249	0.0863	0.99997	0.00008	0.0089	0.495
Neural Network	0.91548	53.9691	7.34636	4.94753	0.91565	0.00478	0.47782	25.90163
Neural Network + LR	0.99872	0.81431	0.90239	0.51181	0.99872	0.00049	0.04988	5.53633
CNN	-1.2696	1449.336	38.0701	28.2291	-1.2153	0.0274	2.7511	116.8406
CNN + LR	0.9657	21.8847	4.6781	2.7732	0.9658	0.0027	0.268	19.2691
RNN	0.7261	174.894	13.2247	10.459	0.881	0.0102	1.0164	34.7684
RNN + LR	0.9999	0.0014	0.037	0.012	0.9999	0.00001	0.0012	0.189
GRU	0.7314	171.5425	13.0974	9.6862	0.7461	0.0094	0.9365	38.7637
GRU + LR	0.9999	0.0417	0.2042	0.1484	0.9999	0.0001	0.0146	0.683
LSTM	-677.781	433464	658.3798	657.8947	-2.2204	0.6444	95.0827	722.7402
LSTM + LR	0.9998	0.1117	0.3342	0.2716	0.9998	0.0003	0.0267	0.8133
Autoencoders	0.9484	32.9428	5.7396	3.7897	0.9499	0.0037	0.24351	23.2364
Autoencoders + LR	0.9995	0.3302	0.5746	0.3779	0.9995	0.0004	0.037	2.0666

performance in predicting these nonlinear effects. The noble metal and advanced material hybrids exhibit unique concentration-dependent behavior, with more pronounced nonlinearity compared to other combinations. The enhanced models effectively predict these complex relationships, though with slightly higher uncertainty at extreme concentration values. The hybrid RNN and linear regression model, while generally accurate, shows increased prediction variance at higher concentrations for these specific combinations.

TABLE 18. Performance metrics of models after gaussian noise data augmentation.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.00841	0.99999	0.00001	0.00084	0.19
Decision Tree + LR	0.99999	0.00118	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Random Forest	0.99787	1.35898	1.16575	0.85358	0.9979	0.00083	0.08318	4.65284
Random Forest + LR	0.99789	1.34463	1.15958	0.86861	0.9979	0.00084	0.08456	4.40284
Ridge Regression	0.94269	36.59759	6.04959	4.18539	0.94278	0.00407	0.40596	26.00603
Ridge Regression + LR	0.68651	200.19127	14.14889	9.91234	0.68811	0.00958	0.96015	42.92968
Poisson Regression	0.94282	36.51496	6.04276	4.15183	0.94293	0.00403	0.40264	26.43655
Poisson Regression + LR	0.83435	105.77731	10.2848	8.43838	0.83861	0.00821	0.82297	25.33119
Gradient Boosting	0.98769	7.86066	2.80369	2.0626	0.98809	0.00201	0.20069	9.97304
Gradient Boosting + LR	0.98729	8.11102	2.84798	2.07252	0.98779	0.00201	0.20153	10.36647
LightGBM	0.99571	2.73765	1.65459	0.95656	0.99572	0.00092	0.09252	8.95397
LightGBM + LR	0.91552	53.94344	7.34462	5.49649	0.91556	0.00534	0.53449	20.86097
CatBoost	0.99913	0.55591	0.74559	0.55165	0.99913	0.00053	0.05344	2.25047
CatBoost + LR	0.99911	0.56817	0.75377	0.5549	0.99911	0.00053	0.05373	2.33991
XGBoost	0.99954	0.29112	0.53956	0.39765	0.99955	0.00039	0.0389	1.66325
XGBoost + LR	0.94679	33.9752	5.82882	4.05648	0.94725	0.00394	0.39498	23.30793
Neural Network	0.89875	64.65351	8.04074	6.46005	0.92088	0.0063	0.63211	17.62982
Neural Network + LR	0.99216	5.00108	2.23631	1.03348	0.9922	0.00099	0.0989	11.34499
CNN	0.72416	176.14494	13.27196	10.12404	0.88388	0.00988	0.98078	49.00909
CNN + LR	0.95804	26.79626	5.17651	2.80065	0.95813	0.0027	0.26934	24.00616
RNN	0.94759	33.46543	5.78493	2.65554	0.95273	0.00255	0.25442	29.10467
RNN + LR	0.9941	3.76879	1.94134	1.22394	0.99414	0.00119	0.11862	6.7996
GRU	0.92079	50.58136	7.11205	5.04109	0.92961	0.0049	0.48846	24.65142
GRU + LR	0.99112	5.66879	2.38092	1.1218	0.99119	0.00108	0.10744	11.81081
LSTM	-0.03835	663.08534	25.75044	21.00657	0	0.02034	2.05057	69.79459
LSTM + LR	0.99201	5.10528	2.25949	0.99489	0.99209	0.00095	0.09493	10.60134
Autoencoders	0.93147	43.76211	6.61529	4.24386	0.93457	0.00411	246.96276	29.51055
Autoencoders + LR	0.99064	5.97641	2.44467	1.21602	0.99078	0.00117	0.11641	11.14089

TABLE 19. Performance metrics of machine learning models after grey wolf optimization.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Decision Tree + LR	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Random Forest	0.99842	1.00668	1.00333	0.68275	0.99849	0.00066	0.06639	4.75172
Random Forest + LR	0.99846	0.97997	0.98993	0.69163	0.99847	0.00067	0.06734	4.54708
Ridge Regression	0.95131	31.09297	5.5761	4.07102	0.95136	0.00394	0.39436	19.10887
Ridge Regression + LR	0.95135	31.06479	5.57358	4.07792	0.95135	0.00395	0.39504	19.28557
Poisson Regression	0.94914	32.47437	5.69862	4.02523	0.9492	0.0039	0.39041	22.65105
Poisson Regression + LR	0.94914	32.47454	5.69864	4.02523	0.9492	0.0039	0.39041	22.65137
Gradient Boosting	0.99999	0.00091	0.03022	0.01372	0.99999	0.00001	0.00136	0.15689
Gradient Boosting + LR	0.99999	0.00091	0.03026	0.01372	0.99999	0.00001	0.00136	0.15735
LightGBM	0.99892	0.68737	0.82908	0.59521	0.99892	0.00057	0.05788	2.76293
LightGBM + LR	0.99885	0.72971	0.85423	0.61893	0.99885	0.0006	0.06015	3.09723
CatBoost	0.99999	0.00133	0.03654	0.01155	0.99999	0.00001	0.00115	0.18961
CatBoost + LR	0.99999	0.00134	0.0366	0.01184	0.99999	0.00001	0.00118	0.18991
XGBoost	0.99999	0.00134	0.03664	0.0113	0.99999	0.00001	0.00112	0.1896
XGBoost + LR	0.99999	0.00134	0.03667	0.01156	0.99999	0.00001	0.00115	0.18905

The models' successful prediction of concentration effects across all hybrid types validates the effectiveness of the autoencoder-based data augmentation and GWO optimization in enhancing model performance. The consistent accuracy across different concentration ranges demonstrates the models' robustness in capturing the fundamental physics of particle loading effects on nanofluid density.

Figure 24 shows the graphs for volume concentration vs predicted densities for each of the nanoparticles.

V. CONCLUSION

This study introduced a comprehensive framework for accurately predicting the density of hybrid nanofluids by leveraging machine learning and deep learning techniques. A robust dataset comprising 436 samples was curated and enhanced using advanced data augmentation methods, including autoencoders and Gaussian noise injection, which effectively improved the diversity and representativeness of the data. The integration of hyperparameter optimization techniques, such as Grey Wolf Optimization (GWO) and Particle Swarm Optimization (PSO), further refined the performance of the predictive models.

Among the tested approaches, Gradient Boosting, CatBoost, and XGBoost emerged as the top-performing machine learning models, achieving near-perfect R^2 scores and minimal error metrics. Similarly, deep learning

TABLE 20. Performance metrics of machine learning models after particle swarm optimization.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Decision Tree + LR	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Random Forest	0.99838	1.03182	1.01578	0.67754	0.99844	0.00065	0.06592	4.78449
Random Forest + LR	0.99847	0.97463	0.98723	0.69058	0.99848	0.00067	0.06724	4.69341
Ridge Regression	0.95131	31.09297	5.5761	4.07102	0.95136	0.00394	0.39436	19.10887
Ridge Regression + LR	0.95135	31.06479	5.57358	4.07792	0.95135	0.00395	0.39504	19.28557
Poisson Regression	0.94914	32.47437	5.69862	4.02523	0.9492	0.0039	0.39041	22.65105
Poisson Regression + LR	0.94914	32.47454	5.69864	4.02523	0.9492	0.0039	0.39041	22.65137
Gradient Boosting	0.99999	0.00093	0.03062	0.01177	0.99999	0.00001	0.00117	0.14902
Gradient Boosting + LR	0.99999	0.00103	0.03211	0.01257	0.99999	0.00001	0.00125	0.16788
LightGBM	0.99892	0.68733	0.82905	0.5952	0.99892	0.00057	0.05787	2.76282
LightGBM + LR	0.99881	0.75746	0.87032	0.63157	0.99881	0.00061	0.0615	3.13375
CatBoost	0.99999	0.00134	0.03664	0.01173	0.99999	0.00001	0.00116	0.19045
CatBoost + LR	0.99999	0.00134	0.03663	0.01175	0.99999	0.00001	0.00117	0.19044
XGBoost	0.99999	0.00133	0.03651	0.01045	0.99999	0.00001	0.00104	0.1896
XGBoost + LR	0.99999	0.00136	0.03697	0.01211	0.99999	0.00001	0.0012	0.18966

TABLE 21. Performance metrics of all the models enhanced with the best optimizations.

Final Models	R^2 Score	MSE	RMSE	MAE	EVS	MAPE	SMAPE	Max Error
Decision Tree	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Decision Tree + LR	0.99999	0.00132	0.03639	0.0084	0.99999	0.00001	0.00084	0.19
Random Forest	0.99842	1.00668	1.00333	0.68275	0.99849	0.00066	0.06639	4.75172
Random Forest + LR	0.99846	0.97997	0.98993	0.69163	0.99847	0.00067	0.06734	4.54708
Ridge Regression	0.95131	31.09297	5.5761	4.07102	0.95136	0.00394	0.39436	19.10887
Ridge Regression + LR	0.95135	31.06479	5.57358	4.07792	0.95135	0.00395	0.39504	19.28557
Poisson Regression	0.94914	32.47437	5.69862	4.02523	0.9492	0.0039	0.39041	22.65105
Poisson Regression + LR	0.94914	32.47454	5.69864	4.02523	0.9492	0.0039	0.39041	22.65137
Gradient Boosting	0.99999	0.00091	0.03022	0.01372	0.99999	0.00001	0.00136	0.15689
Gradient Boosting + LR	0.99999	0.00091	0.03026	0.01372	0.99999	0.00001	0.00136	0.15734
LightGBM	0.99892	0.68733	0.82905	0.5952	0.99892	0.00057	0.05787	2.76282
LightGBM + LR	0.99885	0.72971	0.85423	0.61893	0.99885	0.0006	0.06015	3.09723
CatBoost	0.99999	0.00133	0.03654	0.01155	0.99999	0.00001	0.00115	0.18961
CatBoost + LR	0.99999	0.00134	0.0366	0.01184	0.99999	0.00001	0.00118	0.18991
XGBoost	0.99999	0.00133	0.03651	0.01045	0.99999	0.00001	0.00104	0.1896
XGBoost + LR	0.99999	0.00134	0.03667	0.01156	0.99999	0.00001	0.00115	0.18905
Neural Network	0.91548	53.9691	7.34636	4.94753	0.91565	0.00478	0.47782	25.90163
Neural Network + LR	0.99872	0.81431	0.90239	0.51181	0.99872	0.00049	0.04988	5.53633
CNN	-1.26958	1449.33624	38.07014	28.2291	-1.21534	0.0274	2.75113	116.84058
CNN + LR	0.9657	21.8847	4.6781	2.7732	0.9658	0.0027	0.268	19.2691
RNN	0.72612	174.89401	13.22474	10.45896	0.88096	0.01024	1.0164	34.76839
RNN + LR	0.9999	0.0014	0.037	0.012	0.9999	0	0.0012	0.189
GRU	0.73137	171.54252	13.09742	9.68623	0.7461	0.0094	0.93652	38.76373
GRU + LR	0.9999	0.0417	0.2042	0.1484	0.9999	0.0001	0.0146	0.683
LSTM	-677.78053	433463.9858	658.37981	657.89466	0	0.64439	95.08271	722.74021
LSTM + LR	0.9998	0.1117	0.3342	0.2716	0.9998	0.0003	0.0267	0.8133
Autoencoders	0.94441	35.49366	5.95765	3.67296	0.94452	0.00353	230.03121	25.24987
Autoencoders + LR	0.9995	0.3302	0.5746	0.3779	0.9995	0.0004	0.037	2.0666

models, particularly RNNs and LSTMs, demonstrated exceptional accuracy, with the autoencoder-augmented RNN model achieving an R^2 score of 0.9999 and MSE of 0.0014. The consistent improvement in prediction accuracy using data augmentation techniques highlighted their effectiveness in mitigating overfitting and enhancing generalization.

The study's findings underscore the transformative potential of combining advanced data preprocessing, optimization algorithms, and state-of-the-art predictive models to address complex engineering challenges. The developed framework provides a scalable and reliable tool for predicting hybrid nanofluid properties, paving the way for optimized designs in thermal management systems and other applications. Future work could focus on expanding the dataset to include diverse nanofluid compositions and exploring real-time applications of these predictive models in industrial settings.

VI. DATA AND CODE AVAILABILITY

The dataset generated and analyzed during this study is publicly available at **Hybrid-Nanofluid-Density-Prediction-Dataset**. Additionally, the source code used in this study is accessible on GitHub at **Hybrid-ML-DL_Nanofluid-Density-Predictor**.

APPENDIX

A. COMPREHENSIVE MODEL PERFORMANCE METRICS

This section presents a detailed compilation of performance metrics for all developed models in this study, including both base models and their Linear Regression stacked variants. The comparative analysis utilizes eight key performance indicators: R^2 Score, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Explained Variance Score (EVS), Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (SMAPE), and Maximum Error. These metrics provide a comprehensive evaluation framework for assessing the predictive capabilities of each model configuration in estimating hybrid nanofluid densities. Table 16 shows the metrics of all the models and is an extension of Table 4.

B. COMPARATIVE ANALYSIS OF MODEL PERFORMANCE WITH DATA AUGMENTATION TECHNIQUES

This section presents a comprehensive compilation of performance metrics for all models implemented in this study. Tables 17 and 18 consolidate the evaluation results for 28 models—comprising 14 base models and their Linear Regression stacked variants—under different augmentation strategies. Each model's performance is assessed through eight standardized metrics: R^2 Score, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Explained Variance Score, Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (SMAPE), and Maximum Error. Table 17 presents the results obtained using autoencoder-based augmentation, while Table 18 contains the performance metrics achieved through Gaussian noise augmentation, providing a comprehensive basis for comparing the effectiveness of both augmentation techniques in hybrid nanofluid density prediction. Table 17 and Table 18 extends 5 and 6 respectively.

C. COMPARATIVE ANALYSIS OF MODEL PERFORMANCE WITH OPTIMIZATION TECHNIQUES

This section presents a comprehensive compilation of performance metrics for all machine learning models optimized through nature-inspired algorithms following the augmentation. Autoencoder-based augmentation was chosen for all models due to its superior overall performance compared to Gaussian noise data augmentation. Tables 19 and 20 consolidate the evaluation results for 16 models—comprising 8 base machine learning models and their Linear Regression stacked variants—under different optimization strategies. Each model's performance is assessed through eight standardized metrics: R^2 Score, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Explained Variance Score, Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (SMAPE), and Maximum Error. Table 19 presents the results obtained using Grey Wolf Optimization (GWO) and extends Table 8, while Table 20 contains the performance metrics achieved through Particle Swarm Optimization (PSO) and extends Table 9, providing a comprehensive basis for comparing the effectiveness of both optimization techniques in hybrid nanofluid density prediction.

D. COMPREHENSIVE PERFORMANCE ANALYSIS OF ALL THE OPTIMIZED MODELS

Table 21 presents a detailed compilation of performance metrics for all twenty-eight models after implementing the optimal data enhancement techniques identified in this study. The tabulated results showcase the predictive capabilities of each model through standard statistical metrics including MAE, RMSE, and R^2 values. This comprehensive performance analysis provides insights into the relative effectiveness of different modeling approaches when applied to hybrid nanofluid density prediction under optimized conditions. The results serve as a benchmark for future research in this domain and offer a quantitative basis for model selection based on specific application requirements. Table 21 extends Table 11.

REFERENCES

[1] H. M. Singh, D. P. Sharma, and I. O. Alade, "GBR-GSO based machine learning predictive model for estimating density of Al_2N_3, Si_3N_4, and TiN nanoparticles suspended in ethylene glycol nanofluids," *Eur. Phys. J. Plus*, vol. 137, no. 5, p. 587, May 2022.

[2] H. Adun, M. Abid, D. Kavaz, Y. Hu, and J. H. Zaini, "Exploring the density characteristics of a novel Al_2N_3-ZnO-Fe_3O_4 ternary hybrid nanofluid through experimental research and constructing a predictive machine learning framework," *Heliyon*, vol. 10, no. 12, 2024, Art. no. e32728.

[3] P. Dehury, S. Chaudhari, T. Banerjee, and S. K. Das, "Prediction of thermophysical properties of deep eutectic solvent-based organic nanofluids: A machine learning approach," *J. Mol. Liquids*, vol. 411, Oct. 2024, Art. no. 125809.

[4] Y. AbuShanab, W. A. Al-Ammari, S. Gowid, and A. K. Sleiti, "Accurate prediction of dynamic viscosity of polyalpha-olefin boron nitride nanofluids using machine learning," *Heliyon*, vol. 9, no. 6, Jun. 2023, Art. no. e16716.

[5] A. Ali, N. Noshad, A. Kumar, S. U. Ilyas, P. E. Phelan, M. Alsaady, R. Nasir, and Y. Yan, "Application of machine learning algorithms in predicting rheological behavior of BN-diamond/thermal oil hybrid nanofluids," *Fluids*, vol. 9, no. 1, p. 20, Jan. 2024.

[6] A. Ullah, E. A. Algehyne, A. Althobaiti, Waseem, and H. A. E.-W. Khalifa, "Influence of dissipative forces on thermal transport in hybrid nanofluid flows: A deep neural network approach," *Int. Commun. Heat Mass Transf.*, vol. 159, Dec. 2024, Art. no. 108085.

[7] S. Changdar, S. Saha, and S. De, "A smart model for prediction of viscosity of nanofluids using deep learning," *Smart Sci.*, vol. 8, no. 4, pp. 242–256, Oct. 2020.

[8] P. K. Kanti, P. Paramasivam, V. V. Wanatasanappan, S. Dhanasekaran, and P. Sharma, "Experimental and explainable machine learning approach on thermal conductivity and viscosity of water based graphene oxide based mono and hybrid nanofluids," *Sci. Rep.*, vol. 14, no. 1, p. 30967, Dec. 2024.

[9] Z. Said, M. Jamei, L. Syam Sundar, A. K. Pandey, A. Allouhi, and C. Li, "Thermophysical properties of water, water and ethylene glycol mixture-based nanodiamond + Fe_3O_4 hybrid nanofluids: An experimental assessment and application of data-driven approaches," *J. Mol. Liquids*, vol. 347, Feb. 2022, Art. no. 117944.

[10] S. Alqaed, J. Mustafa, S. M. Sajadi, and M. Sharifpur, "Enhancing thermal conductivity of water/CeO2-MWCNTs hybrid nanofluid: Experimental insights and artificial neural network modeling," *J. Thermal Anal. Calorimetry*, vol. 149, no. 9, pp. 4019–4031, May 2024.

[11] J. Sarkar, P. Ghosh, and A. Adil, "A review on hybrid nanofluids: Recent research, development and applications," *Renew. Sustain. Energy Rev.*, vol. 43, pp. 164–177, Mar. 2015.

[12] A. K. Gupta, P. Mathur, M. O. Oyedeji, I. O. Alade, T. F. Qahtan, and S. Gupta, "Development of predictive models for density of hybrid nanofluids using different machine learning techniques," *Proc. Inst. Mech. Eng., E, J. Process Mech. Eng.*, vol. 237, no. 5, pp. 1722–1739, Oct. 2023.

[13] Z. Islam, M. Abdel-Aty, Q. Cai, and J. Yuan, "Crash data augmentation using variational autoencoder," *Accident Anal. Prevention*, vol. 151, Mar. 2021, Art. no. 105950.

[14] M. Arslan, M. Guzel, M. Demirci, and S. Ozdemir, "SMOTE and Gaussian noise based sensor data augmentation," in *Proc. 4th Int. Conf. Comput. Sci. Eng. (UBMK)*, Sep. 2019, pp. 1–5.

[15] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Jan. 2014.

[16] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: An overview," *Soft Comput.*, vol. 22, no. 2, pp. 387–408, Jan. 2018.

[17] S. Okada, M. Ohzeki, and S. Taguchi, "Efficient partition of integer optimization problems with one-hot encoding," *Sci. Rep.*, vol. 9, no. 1, p. 13036, Sep. 2019.

[18] S. Kannaiyan, C. Boobalan, F. C. Nagarajan, and S. Sivaraman, "Modeling of thermal conductivity and density of alumina/silica in water hybrid nanocolloid by the application of artificial neural networks," *Chin. J. Chem. Eng.*, vol. 27, no. 3, pp. 726–736, Mar. 2019.

[19] S. N. M. Zainon and W. H. Azmi, "Stability and thermo-physical properties of green bio-glycol based TiO_2-SiO_2 nanofluids," *Int. Commun. Heat Mass Transf.*, vol. 126, Jul. 2021, Art. no. 105402.

[20] B. Saleh and L. S. Sundar, "Thermal efficiency, heat transfer, and friction factor analyses of MWCNT + Fe_3O_4/Water hybrid nanofluids in a solar flat plate collector under thermosyphon condition," *Processes*, vol. 9, no. 1, p. 180, Jan. 2021.

[21] M. Devarajan, N. Parasumanna Krishnamurthy, M. Balasubramanian, B. Ramani, S. Wongwises, K. A. El-Naby, and R. Sathyamurthy, "Thermophysical properties of CNT and CNT/Al_2O_3 hybrid nanofluid," *Micro Nano Lett.*, vol. 13, no. 5, pp. 617–621, May 2018.

[22] V. Kumar, A. Pare, A. K. Tiwari, and S. K. Ghosh, "Efficacy evaluation of oxide-MWCNT water hybrid nanofluids: An experimental and artificial neural network approach," *Colloids Surf. A, Physicochemical Eng. Aspects*, vol. 620, Jul. 2021, Art. no. 126562.

[23] S. K. Verma, A. K. Tiwari, S. Tiwari, and D. S. Chauhan, "Performance analysis of hybrid nanofluids in flat plate solar collector as an advanced working fluid," *Sol. Energy*, vol. 167, pp. 231–241, Jun. 2018.

[24] Z. Said, M. Ghodbane, L. S. Sundar, A. K. Tiwari, M. Sheikholeslami, and B. Boumeddane, "Heat transfer, entropy generation, economic and environmental analyses of linear Fresnel reflector using novel rGO-Co_3O_4 hybrid nanofluids," *Renew. Energy*, vol. 165, pp. 420–437, Mar. 2021.

[25] L. S. Sundar, S. Mesfin, E. V. Ramana, Z. Said, and A. C. M. Sousa, "Experimental investigation of thermo-physical properties, heat transfer, pumping power, entropy generation, and exergy efficiency of nanodiamond + Fe_3O_4/60:40% water-ethylene glycol hybrid nanofluid flow in a tube," *Thermal Sci. Eng. Prog.*, vol. 21, Mar. 2021, Art. no. 100799.

[26] H. Yarmand, S. Gharehkhani, G. Ahmadi, S. F. S. Shirazi, S. Baradaran, E. Montazer, and M. Dahari, "Graphene nanoplateletssilver hybrid nanofluids for enhanced heat transfer," *Energy Convers. Manage.*, vol. 100, pp. 419–428, Jul. 2015.

[27] H. P. Vinutha, B. Poornima, and B. M. Sagar, "Detection of outliers using interquartile range technique from intrusion dataset," in *Proc. 6th Int. Conf. Inf. Decis. Sci. (FICTA)*, Jan. 2018, pp. 511–518.

[28] M. S. Gal and D. L. Rubinfeld, "Data standardization," *NYUL Rev.*, vol. 94, pp. 737–758, Jan. 2018.

[29] G. Muhammad, S. Naveed, L. Nadeem, T. Mahmood, A. R. Khan, Y. Amin, and S. A. O. Bahaj, "Enhancing prognosis accuracy for ischemic cardiovascular disease using k nearest neighbor algorithm: A robust approach," *IEEE Access*, vol. 11, pp. 97879–97895, 2023.

[30] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lect. IE*, vol. 2, no. 1, pp. 1–18, Jan. 2015.

[31] F. J. Moreno-Barea, F. Strazzera, J. M. Jerez, D. Urda, and L. Franco, "Forward noise adjustment scheme for data augmentation," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 728–734.

[32] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN'95)*, vol. 4, Nov. 2002, pp. 1942–1948.

[33] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Metaheuristic algorithms: A comprehensive review," in *Computational Intelligence for Multimedia Big Data on the Cloud With Engineering Applications* (Intelligent Data-Centric Systems), A. K. Sangaiah, M. Sheng, and Z. Zhang, Eds. Academic Press, 2018, pp. 185–231. [Online]. Available: https://doi.org/10.1016/B978-0-12-813314-9.00010-4

[34] M. Song and G. Gu, "Research on particle swarm optimization: A review," in *Proc. Int. Mach. Learn. Cybern.*, vol. 4, Feb. 2005, pp. 2236–2241.

[35] B. Mahesh, "Machine learning algorithms: A review," *Int. J. Sci. Res.*, vol. 9, no. 1, pp. 381–386, 2020.

[36] C. Bhatt, I. Kumar, V. Vijayakumar, K. U. Singh, and A. Kumar, "The state of the art of deep learning models in medical science and their challenges," *Multimedia Syst.*, vol. 27, no. 4, pp. 599–613, Aug. 2021.

[37] S. Rong and Z. Bao-Wen, "The research of regression model in machine learning field," in *Proc. MATEC Web Conferences*, vol. 176, 2018, p. 01033.

[38] F. Divina, A. Gilson, F. Goméz-Vela, M. García Torres, and J. Torres, "Stacking ensemble learning for short-term electricity consumption forecasting," *Energies*, vol. 11, no. 4, p. 949, Apr. 2018.

[39] X. Su, X. Yan, and C.-L. Tsai, "Linear regression," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 4, no. 3, pp. 275–294, 2012.

[40] X. Ying, "An overview of overfitting and its solutions," *J. Phys., Conf. Ser.*, vol. 1168, Feb. 2019, Art. no. 022022.

[41] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020.

[42] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[43] B. De Ville, "Decision trees," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 5, no. 6, pp. 448–455, 2013.

[44] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *PeerJ Comput. Sci.*, vol. 7, p. 623, Jul. 2021.

[45] R. Genuer, "Contributions to random forests methods for several data analysis problems," Ph.D. dissertation, Dept. Statist., Univ. de Bordeaux, Bordeaux, France, 2021. [Online]. Available: https://theses.hal.science/tel-03111020/document

[46] D. W. Marquardt and R. D. Snee, "Ridge regression in practice," *Amer. Statistician*, vol. 29, no. 1, pp. 3–20, Feb. 1975.

[47] M. Stock, T. Pahikkala, A. Airola, B. De Baets, and W. Waegeman, "A comparative study of pairwise learning methods based on kernel ridge regression," *Neural Comput.*, vol. 30, no. 8, pp. 2245–2283, Aug. 2018.

[48] G. C. McDonald, "Ridge regression," *Wiley Interdiscipl. Reviews: Comput. Statist.*, vol. 1, no. 1, pp. 93–100, Jul. 2009.

[49] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, "A high-bias, low-variance introduction to machine learning for physicists," *Phys. Rep.*, vol. 810, pp. 1–124, May 2019.

[50] P. C. Consul and F. Famoye, "Generalized Poisson regression model," *Commun. Stat.-Theory Methods*, vol. 21, no. 1, pp. 89–109, Jan. 1992.

[51] C. C. Clogg, E. Petkova, and A. Haritou, "Statistical methods for comparing regression coefficients between models," *Amer. J. Sociol.*, vol. 100, no. 5, pp. 1261–1293, Mar. 1995.

[52] M. J. Hayat and M. Higgins, "Understanding Poisson regression," *J. Nursing Educ.*, vol. 53, no. 4, pp. 207–215, Apr. 2014.

[53] E. L. Frome, "The analysis of rates using Poisson regression models," *Biometrics*, vol. 39, no. 3, p. 665, Sep. 1983.

[54] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers Neurorobotics*, vol. 7, p. 21, Jul. 2013.

[55] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Dec. 2017, pp. 1–12.

[56] W. Y. Loh, "Classification and regression trees," *Inst. for Math. Sci.*, vol. 10, p. 19, Mar. 2014.

[57] J. T. Hancock and T. M. Khoshgoftaar, "CatBoost for big data: An interdisciplinary review," *J. Big Data*, vol. 7, no. 1, p. 94, Dec. 2020.

[58] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[59] N. Kriegeskorte and T. Golan, "Neural network models and deep learning," *Current Biol.*, vol. 29, no. 7, pp. R231–R236, Apr. 2019.
[60] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.
[61] C. Cortes, M. Mohri, and A. Rostamizadeh, "L_2 regularization for learning kernels," 2012, *arXiv:1205.2653*.
[62] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction To Linear Regression Analysis,*. Hoboken, NJ, USA: Wiley, 2021.
[63] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, pp. 1–74, Mar. 2021.
[64] W. Hao, W. Yizhou, L. Yaqin, and S. Zhili, "The role of activation function in CNN," in *Proc. 2nd Int. Conf. Inf. Technol. Comput. Appl. (ITCA)*, Dec. 2020, pp. 429–432.
[65] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Phys. D: Nonlinear Phenomena*, vol. 404, Mar. 2020, Art. no. 132306.
[66] C. Banerjee, T. Mukherjee, and E. Pasiliao, "An empirical study on generalizations of the ReLU activation function," in *Proc. ACM Southeast Conf.*, Apr. 2019, pp. 164–167.
[67] W. Li, Z. Fang, and Y. Wang, "Stacking ensemble of deep learning methods for landslide susceptibility mapping in the three Gorges reservoir area, China," *Stochastic Environ. Res. Risk Assessment*, vol. 36, no. 8, pp. 2207–2228, Aug. 2022.
[68] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (GRU) neural networks," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 1597–1600.
[69] A. Graves and A. Graves, "Long short-term memory," in *Supervised Sequence Labelling With Recurrent Neural Networks*. Berlin, Germany: Springer, 2012, pp. 37–45. [Online]. Available: https://doi.org/10.1007/978-3-642-24797-2_4
[70] S. Narayan, "The generalized sigmoid activation function: Competitive supervised learning," *Inf. Sci.*, vol. 99, nos. 1–2, pp. 69–82, Jun. 1997.
[71] W. H. L. Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, "Autoencoders," in *Machine Learning*. New York, NY, USA: Academic, 2020, pp. 193–208.
[72] L. Vu and Q. U. Nguyen, "An ensemble of activation functions in AutoEncoder applied to IoT anomaly detection," in *Proc. 6th NAFOSTED Conf. Inf. Comput. Sci. (NICS)*, Dec. 2019, pp. 534–539.
[73] A. Ash and M. Shwartz, "R^2: A useful measure of model performance when predicting a dichotomous outcome," *Statist. Med.*, vol. 18, no. 4, pp. 375–384, Feb. 1999.
[74] T. O. Hodson, "Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not," *Geoscientific Model Develop. Discuss.*, vol. 2022, pp. 1–10, Mar. 2022.
[75] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Res.*, vol. 30, pp. 79–82, Jun. 2005.
[76] A. de Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, Jun. 2016.
[77] V. Kreinovich, H. T. Nguyen, and R. Ouncharoen, "How to estimate forecasting quality: A system-motivated derivation of symmetric mean absolute percentage error (SMAPE) and other similar characteristics," Departmental Tech. Rep. 865, 2014. [Online]. Available: https://scholarworks.utep.edu/cs_techrep/865
[78] M. Garofalakis and A. Kumar, "Deterministic wavelet thresholding for maximum-error metrics," in *Proc. 23rd ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, Jun. 2004, pp. 166–176.
[79] K. E. O'Grady, "Measures of explained variance: Cautions and limitations," *Psychol. Bull.*, vol. 92, no. 3, pp. 766–777, 1982.
[80] W. McKinney, "Pandas: A foundational Python library for data analysis and statistics," *Python High Perform. Sci. Comput.*, vol. 14, no. 9, pp. 1–9, 2011.
[81] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.
[82] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI 16)*, Nov. 2016, pp. 265–283.
[83] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. J. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Jan. 2011.
[84] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007.
[85] M. Waskom, "Seaborn: Statistical data visualization," *J. Open Source Softw.*, vol. 6, no. 60, p. 3021, Apr. 2021.

PRIYA MATHUR is a highly accomplished mathematics professor with more than 20 years of experience in teaching and research in engineering education. She has an impressive portfolio of 50 SCOPUS-indexed research papers, four authored academic books, and multiple awards for academic excellence. Her research interests include applied mathematics, computational fluid dynamics, nanofluid dynamics, entropy generation, and mathematical modeling, making significant contributions to both theoretical and practical aspects of these fields. She innovative work has been published in leading international journals and conferences, earning her recognition as a thought leader in her domain. She has successfully guided numerous postgraduate and doctoral students, fostering the next generation of researchers and professionals. She has received funding from TEQIP-III for research projects and Faculty Development Programs (FDPs), showcasing her ability to lead and execute impactful academic initiatives. She has organized two International FDPs and the International Conference on Mathematical Modeling and Statistics (ICMMS), in 2023, demonstrating her leadership in promoting global academic collaboration. She has also edited two SCOPUS-indexed books and managed four special issues of SCOPUS-indexed journals, further reflecting her significant contributions to academic publishing and research dissemination. A recipient of prestigious academic awards, she has been honored for her dedication to excellence in teaching, research, and mentoring. She has played an integral role in developing interdisciplinary research projects and establishing collaborations with national and international institutions. In addition to her academic contributions, she has actively participated in organizing seminars, workshops, and conferences, serving as a keynote speaker and panelist. Her leadership in promoting knowledge dissemination and fostering innovation has left a lasting impact on her field. Passionate about bridging the gap between academia and industry, she has worked on projects that address real-world challenges and drive technological advancements. Her dedication to education, innovation, and research continues to inspire her peers and students alike, solidifying her position as a beacon of academic excellence.

HAMMAD SHAIKH is currently pursuing the Bachelor of Technology degree in computer science and engineering with Manipal University Jaipur, Rajasthan, India. As an Undergraduate Researcher at his university, he has actively contributed to several innovative projects, showcasing a deep interest in advancing the fields of machine learning, deep learning, computer vision, and artificial intelligence. He is currently working as an Aspiring Researcher. In addition to his academic endeavors, he works part-time as a Research Assistant at the AI4A Laboratory at Manipal University Jaipur, where he collaborates with faculty and peers on cutting-edge research initiatives aimed at addressing real-world challenges. His passion for exploring emerging technologies and his dedication to bridging the gap between theoretical knowledge and practical applications underscore his commitment to the field of computer science.

FARHAN SHETH is currently pursuing the Bachelor of Technology (B.Tech.) degree in computer science and engineering with Manipal University Jaipur, India. He is currently working as a Distinguished Undergraduate Researcher. He is also working as a Lead Researcher with the AI4A Laboratory, where he spearheads cutting-edge artificial intelligence projects. He has contributed extensively as a Research Assistant on multiple interdisciplinary initiatives, collaborating with esteemed faculty and international institutions. With a stellar research portfolio, he has co-authored five peer-reviewed papers published in reputed Scopus and SCI-indexed journals. His accolades include multiple Student Excellence Awards for Research at Manipal University Jaipur, and his projects have garnered recognition for their societal impact. Beyond academics, he has demonstrated exceptional leadership and problem-solving skills, earning university-wide honors for his academic and extracurricular contributions. His research interests include artificial intelligence, machine learning, deep learning, computer vision, natural language processing, and trustworthy AI. Committed to advancing AI research, his vision is to create transformative technological solutions that address real-world challenges and inspire future innovation.

DHEERAJ KUMAR is currently pursuing the Bachelor of Technology in computer science and engineering with Manipal University Jaipur, Rajasthan, India. He is currently working as a Research Assistant with AI4A Laboratory, Manipal University Jaipur, where he actively contributes to projects in machine learning, deep learning, and artificial intelligence. His research primarily focuses on leveraging data-driven techniques to address complex real-world challenges, with a strong emphasis on developing innovative and practical solutions. As an Undergraduate Researcher, he has gained valuable experience working on diverse projects, honing his skills in algorithm development, model optimization, and large-scale data processing. Beyond academics, he is deeply interested in exploring emerging technologies and their transformative potential in solving modern computational problems. He is committed to bridging the gap between theoretical knowledge and real-world applications, demonstrating a passion for impactful research and interdisciplinary collaboration.

AMIT KUMAR GUPTA received the B.E. degree in information technology in 2005 and the M.Tech. and Ph.D. degrees in computer science and engineering in 2011 and 2018, respectively. He is currently working as an Associate Professor with the Department of Computer Science and Engineering, Manipal University Jaipur, Rajasthan, India. With 20 years of experience in teaching and research, he has published more than 50 research papers in internationally reputed journals indexed in Scopus, Web of Science, and SCI. His research interests include machine learning, deep learning, artificial intelligence, and image processing. He has also worked as a Guest Editor of nine Scopus-indexed journals, edited a book with IGI Global, and organized three international conferences sponsored by AICTE and TEQIP-III.