

Article

Comparative Evaluation of ANN, LSTM, and 1D-CNN Models for Energy-Efficient Prediction of Low-Cost Gas Sensor Time-Series Data

Jelena Čulić Gambiroža ^{1,*} , Ana Čulić ² , Kristina Medić ¹  and Ana Grubišić ³

¹ Faculty of Maritime Studies, University of Split, Ruđera Boškovića 37, 21000 Split, Croatia; kzovko@pfst.hr

² Maritime Department, Dr. Franjo Tuđman Defense and Security University, Zrinsko Frankopanska 207A, 21000 Split, Croatia; ana.culic@sois-ft.hr

³ Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Ruđera Boškovića 32, 21000 Split, Croatia; agrubi00@fesb.hr

* Correspondence: jculicgam@pfst.hr

Abstract

This study investigates the application of Artificial Neural Network (ANN), Long Short-Term Memory network (LSTM) as a representative of Recurrent Neural Network (RNN), and one-dimensional Convolutional Neural Network (1D-CNN) for time series prediction, demonstrated through a use case of low-cost gas sensor readings from transient signals. Despite the widespread use of these architectures in IoT forecasting applications, there is a lack of systematic comparative studies that evaluate their performance under identical experimental conditions, particularly in energy-constrained sensing scenarios. The primary objective is to evaluate the trade-offs between model accuracy, computational cost, and memory requirements under energy-efficient data acquisition scenarios. A comprehensive experimental analysis was conducted using 186 recorded transient samples, where all models were trained and evaluated under consistent preprocessing, identical data splits, and uniform hyperparameter settings. Performance was assessed using RMSE, MAE, R^2 , training time, and model size as key evaluation metrics under varying input sequence lengths. The results show that the LSTM model achieved the highest accuracy, with an RMSE of 3.69%, R^2 of 0.85 and scaled MAE of 0.04, effectively capturing long-term temporal dependencies. The 1D-CNN exhibited a balanced compromise between accuracy and training efficiency, while the ANN provided the shortest training time but lower overall performance. Reducing the number of input readings from 186 to as few as 10–20 resulted in only a 2–4% increase in RMSE, with model size reductions of up to 50%, making such configurations particularly suitable for edge or embedded IoT devices. The findings demonstrate that artificial neural networks can maintain high prediction accuracy even under reduced data conditions, contributing to the development of low-power, resource-efficient sensing systems for intelligent and distributed IoT environments.

Keywords: neural networks; ANN; 1D-CNN; LSTM; time series data



Academic Editors: Chen Yang, Olufemi A. Omitaomu and Haoran Niu

Received: 8 June 2026

Revised: 9 July 2026

Accepted: 14 July 2026

Published: 16 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

With the increasing number of interconnected devices and sensors in the Internet of Things (IoT), efficient and accurate forecasting of sensor data has become a critical research focus [1]. In particular, gas level sensors are essential in monitoring environmental conditions, industrial safety, and energy systems. However, continuous high-frequency

data collection consumes significant energy and storage resources. Therefore, this work emphasizes energy-efficient data collection while maintaining high prediction accuracy, introducing a lightweight update mechanism for gas level prediction models [2].

Time series forecasting faces significant challenges due to the increasing complexity and scale of data across various fields, including data nonlinearity, noise, and complex temporal dependencies. Traditional statistical models often fail to capture these patterns effectively, leading to low prediction accuracy and poor generalization. Recent advances in machine learning (ML) and deep learning (DL) have demonstrated significant potential to overcome these limitations by learning complex, nonlinear relationships directly from raw data.

Artificial Neural Networks (ANNs) provide a flexible architecture for approximating nonlinear mappings, while Convolutional Neural Networks (CNNs) can effectively extract local and spatial dependencies from structured time series data. In contrast, Recurrent Neural Networks (RNNs), particularly architectures such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are well suited for modeling temporal dependencies and long-term dynamics due to their internal memory structures [3].

Liu et al. [3] present a comprehensive survey of forecasting methods for time series data, categorizing existing approaches and highlighting challenges such as scalability, low prediction accuracy, and limited generalization. Their study emphasizes the growing importance of hybrid and adaptive models capable of incremental learning and efficient updates, which align with the goals of this research. Similarly, Enssafi et al. [4] compare traditional forecasting techniques such as SARIMA and Exponential Smoothing with DL models including Prophet, CNN, and LSTM networks. Their results demonstrate that DL models, especially stacked LSTM, achieve superior performance in predictive accuracy, validating the advantage of neural network-based forecasting.

Despite these advances, most existing studies primarily focus on improving prediction accuracy, often neglecting the trade-off between accuracy, computational complexity, and efficiency as an important factor in resource-constrained IoT environments. Moreover, there is a lack of systematic evaluation of different neural network architectures under identical experimental conditions, particularly when varying input sequence lengths.

In addition to energy consumption caused by high-frequency data acquisition, deploying machine learning models in low-cost IoT and embedded systems introduces further challenges. These include limited computational resources, restricted memory capacity, high inference latency requirements, and constraints related to hardware adaptability [1,5]. Complex deep learning models with large parameter counts may not be suitable for real-time execution on edge devices, where efficient and lightweight architectures are required to ensure practical usability.

Therefore, understanding the trade-off between model complexity and deployment efficiency is essential when designing forecasting systems for resource-constrained environments.

To address these limitations, this study systematically evaluates three representative neural network architectures, ANN, 1D-CNN, and LSTM, under consistent experimental settings. Instead of proposing a new model, the focus is on a fair comparative analysis of these architectures in terms of prediction accuracy, computational cost, and model complexity.

The experiments are conducted on preprocessed gas sensor time-series data with multiple input sequence lengths. All models are trained under identical conditions to ensure a fair comparison. Performance is evaluated using RMSE, (scaled) MAE, and R^2 , along with training time and model size, enabling a multi-dimensional assessment of accuracy–efficiency trade-offs.

The results show that LSTM models achieve the highest prediction accuracy but require higher computational cost. ANN models offer the fastest training time but are

less effective for sequential dependencies. The 1D-CNN architecture provides a balanced trade-off between accuracy and efficiency. Additionally, reducing input sequence length significantly lowers computational cost while maintaining acceptable prediction accuracy, making it suitable for energy-constrained IoT applications.

The remainder of the paper is structured as follows. Section 2 provides an overview of related work. Section 3 presents the proposed methodology, while Section 4 discusses the results. The findings are discussed in Section 5. Section 6 concludes the paper with a summary of findings and directions for future work.

2. Related Work

With the rise of data availability and computational power, ML and DL models have emerged as powerful alternatives to traditional time series forecasting techniques due to their ability to capture complex, nonlinear patterns. Recent literature on time series forecasting [6] highlights a clear shift from classical models to DL approaches. While traditional methods such as ARIMA and Kalman filters have been widely used, recent studies emphasize the superior performance of DL architectures, particularly LSTMs, bidirectional and encoder–decoder networks, and CNNs, for multi-step forecasting. These models have demonstrated improved accuracy across diverse applications, including weather prediction, energy forecasting, and financial markets. Comparative evaluations consistently show that deep architectures outperform classical methods in capturing complex temporal patterns and long-term dependencies.

DL, a powerful subset of ML, has achieved remarkable success across various domains, including image processing, speech recognition, and natural language understanding. Its ability to learn complex nonlinear mappings without requiring scaled or stationary inputs, along with its support for multivariate and multi-step forecasting, makes it particularly well-suited for complex time series prediction tasks. This paper reviews common DL models for time series forecasting, discusses their relationship with traditional methods, and highlights the unique challenges associated with time series data. In addition, it surveys previous research applying DL to time series analysis [7].

2.1. Artificial Neural Networks(ANNs)/Feed-Forward Neural Networks (FFNNs)

Feedforward Neural Networks (FFNNs), often referred to as Artificial Neural Networks (ANNs), represent one of the earliest and most widely used neural architectures in time-series forecasting. Both terms are frequently used interchangeably in the literature, as they describe models composed of interconnected layers of neurons that process information in a forward direction, from input to output, without feedback loops. Their ability to approximate nonlinear relationships and generalize from complex data patterns makes them a strong foundation for predictive modeling tasks.

The application of neural networks to time-series prediction dates back to the early 1990s. In 1993, researchers demonstrated their potential for multi-step-ahead forecasting, highlighting ANNs as a promising alternative to traditional statistical methods for both short- and long-term predictions [8]. Since then, ANNs have been extensively applied to nonlinear time-series prediction, often alongside methods such as Support Vector Machines (SVMs). However, performance can be affected by noisy data and evolving system dynamics, motivating the integration of nonlinear filtering techniques such as Extended Kalman Filtering (EKF) and Unscented Kalman Filtering (UKF) with ANNs. Studies show that both EKF- and UKF-based ANN models provide accurate forecasts, with EKF-ANNs offering a favorable balance between accuracy and computational efficiency [9].

Recent research has further demonstrated the utility of ANNs in robust outlier detection for time series. Hybrid approaches combining multivariate outlier detection algorithms

with ANNs have proven effective at identifying both isolated and clustered anomalies, improving forecast reliability by filtering outliers before model training [10]. Similarly, novel two-stage ANN frameworks have been proposed for electrical load demand forecasting, incorporating error correction steps to model residual forecast errors. By leveraging temporal dependencies in the error series, these models improve initial predictions while maintaining computational efficiency, with applicability across industrial, residential, and service sectors [11].

More recent comparative studies have confirmed that ANN models remain competitive baseline approaches for time-series forecasting, particularly when low computational complexity and fast inference are required. The study demonstrated that although recurrent architectures achieved slightly higher prediction accuracy, ANN provided a favorable balance between computational efficiency and forecasting performance, making it suitable for resource-constrained applications [12]. Recent work has further demonstrated the applicability of ANN-based forecasting models for IoT sensor data. The proposed predictive analytics framework employed ANN as one of the baseline deep learning models for environmental sensor forecasting and confirmed its effectiveness in learning nonlinear relationships while maintaining relatively low computational complexity, making it suitable for practical sensor-based forecasting applications [13]. In applied energy and environmental forecasting, ANNs have also shown strong performance. For instance, Kolokythas et al. [14] developed and optimized a 24 h ahead wind speed prediction model across multiple wind farms. The study systematically evaluated different input configurations, averaging intervals, and time-series lengths using k-fold cross-validation. Benchmarking against traditional persistence forecasting revealed that the ANN was capable of capturing complex temporal patterns, thereby providing improved predictive accuracy essential for efficient wind farm management and grid operation.

Overall, ANNs remain a versatile and effective tool for time-series modeling, offering a balance of predictive power, flexibility, and computational efficiency. Their continued evolution, particularly in hybrid frameworks and multi-step forecasting applications, underscores their relevance in addressing the challenges posed by increasingly complex and variable time-dependent data.

2.2. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of ANN specifically designed to process sequential data by maintaining a hidden state that captures information from previous time steps. This architecture enables RNNs to effectively model temporal dependencies, making them well-suited for tasks such as time-series forecasting, speech recognition, and anomaly detection. Traditionally, classical forecasting methods like ETS and ARIMA have remained popular due to their simplicity, robustness, and efficiency, whereas early neural network approaches often underperformed because of short series lengths and high model complexity. The advent of big data, particularly datasets with many related time series rather than long individual sequences, has increased the applicability of RNNs by allowing cross-series learning and improved prediction accuracy [15].

RNNs have been applied successfully in both prediction and classification tasks. While prediction focuses on forecasting future values, and classification assigns labels to sequences, the internal features learned by RNNs in both contexts are closely related. Incorporating prediction as an auxiliary task during training can improve learning speed, generalization ability, and classification performance, resulting in richer internal representations and more reliable outcomes [16].

Among RNN architectures, the Long Short-Term Memory (LSTM) network is the most widely used and studied. LSTM networks address key limitations of traditional RNNs,

such as vanishing and exploding gradients, through specialized memory cells and gating mechanisms, enabling the network to capture both short- and long-term dependencies in sequential data. LSTM-based models have demonstrated superior performance in numerous domains, including meteorological forecasting [17], air temperature prediction [18], low-power gas sensing [2,19], and general nonlinear time-series modeling [20]. Variants such as hierarchical, attention-based, and sequence-to-sequence LSTM networks further enhance their capability to handle multidimensional inputs and multi-step predictions with minimal error propagation.

Recent work has further demonstrated the applicability of LSTM networks to gas sensing. The proposed LSTM–Pearson framework improved gas concentration prediction by combining temporal modeling with feature selection, highlighting the capability of LSTM architectures to capture nonlinear temporal relationships in gas sensor signals [21].

Another prominent RNN variant is the Gated Recurrent Unit (GRU), which provides a simpler architecture with fewer parameters while maintaining similar capabilities in capturing temporal dependencies. GRUs have been effectively applied in fault diagnosis and other applications requiring robustness to noise and efficient modeling of time-dependent features [22].

Recent work has also demonstrated the applicability of LSTM-based forecasting models to multivariate IoT sensor data for short-term solar power prediction, further confirming the capability of recurrent architectures to model complex temporal dependencies in resource-constrained sensing environments [23]. Recent developments have also shown the applicability of optimized LSTM architectures for natural gas consumption forecasting. The proposed incremental LSTM framework achieved improved forecasting accuracy while maintaining computational efficiency suitable for practical deployment [24].

Overall, RNNs, particularly LSTMs and GRUs, have become essential tools for modeling sequential and time-dependent data, offering significant advantages in handling nonlinear, chaotic, and multi-step patterns across a wide range of scientific and engineering domains.

2.3. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of DL models originally developed for image and video analysis, where they have demonstrated state-of-the-art performance in object detection, recognition, and classification tasks. The core principle of CNNs lies in their ability to extract hierarchical feature representations from raw input data using convolutional layers, pooling operations, and nonlinear activation functions. While classical 2D CNNs operate on two-dimensional spatial data, such as images, their architecture can be adapted for one-dimensional data sequences, such as time-series data, giving rise to 1D Convolutional Neural Networks (1D-CNNs).

1D-CNNs are particularly well-suited for time series and sequential data because they apply convolutional filters along the temporal dimension, enabling the model to automatically learn local patterns and higher-order features without explicit feature engineering. Recent research has further demonstrated the applicability of CNN architectures to environmental sensor data. The proposed CNN-based framework successfully detected anomalies in Healthcare-IoT sensor time series, confirming that convolutional models efficiently capture temporal patterns while maintaining computational efficiency suitable for embedded sensing applications [25]. This property makes 1D-CNNs highly effective for applications such as human activity recognition [26], network intrusion detection [27], and structural health monitoring, among others. Moreover, 1D-CNNs are computationally efficient and can be implemented in real-time, low-cost hardware due to their compact architecture, which requires only scalar multiplications and additions along a single dimension [28].

A critical aspect of designing effective 1D-CNNs for time series is the selection of kernel size, which determines the receptive field over which temporal patterns are captured. Improper kernel selection can result in missing salient features or overfitting to irrelevant details. Recent work has addressed this challenge through theoretical analyses and innovative architectures, such as the Omni-Scale 1D-CNN (OS-CNN), which dynamically determines optimal kernel sizes during training [29]. Furthermore, advanced multi-headed 1D-CNN architectures, where each convolutional layer uses a distinct kernel size, have demonstrated superior performance by simulating ensemble-like learning effects, as evidenced in human activity recognition studies [26].

Recent comparative evaluations have further shown that 1D-CNN architectures remain competitive alternatives to recurrent models for forecasting tasks requiring reduced computational complexity, particularly in energy-related sensor applications where efficient extraction of local temporal features is sufficient for accurate prediction [12]. Comparative evaluations have also shown that CNN-based forecasting models provide competitive prediction accuracy for IoT sensor time-series data while preserving efficient feature extraction and moderate computational requirements, making them attractive for practical forecasting applications in resource-constrained environments [13].

Overall, 1D-CNNs have emerged as a flexible and powerful tool for time-series modeling, capable of learning complex temporal dependencies, reducing preprocessing requirements, and achieving state-of-the-art performance across diverse scientific and engineering domains. Their simplicity, efficiency, and adaptability make them a compelling alternative to more complex recurrent architectures, especially in applications with high-dimensional, short-length, or resource-constrained sequential data.

2.4. Summary and Limitations of Existing Solutions

The selection of ANN, LSTM, and 1D-CNN architectures was motivated by their established roles in time-series forecasting and their complementary modeling capabilities. ANN models are commonly employed as computationally efficient baseline forecasting models [30]. LSTM networks have become one of the most widely adopted recurrent architectures for modeling long-term temporal dependencies in sequential data [31], while 1D-CNN architectures have demonstrated strong capability in extracting local temporal features with relatively low computational requirements [32]. Recent survey studies continue to recognize these architectures as fundamental reference models for evaluating forecasting methods and emerging edge-intelligence solutions [30–32]. Therefore, they were selected to investigate the trade-offs between prediction accuracy, computational cost, and model complexity in energy-efficient gas sensing systems, where both forecasting accuracy and resource consumption are critical deployment considerations.

Recent studies have demonstrated the effectiveness of ANN, LSTM, and CNN-based models for forecasting and analyzing sensor time-series data. Comparative evaluations have been reported in applications such as smart metering and energy forecasting, where different neural network architectures exhibit distinct trade-offs between prediction accuracy and computational requirements [12]. Similarly, LSTM-based approaches have shown promising results in gas concentration prediction and low-cost sensing applications due to their ability to model temporal dependencies in sensor signals [21]. CNN-based methods have also been successfully applied to environmental and IoT sensor data, demonstrating efficient extraction of local temporal patterns while maintaining relatively low computational complexity [25]. Furthermore, neural-network-based forecasting models continue to be widely adopted in IoT applications requiring accurate short-term prediction of sensor measurements [23].

However, several limitations remain in the existing literature. Most studies focus on a single neural network architecture or a specific application scenario, making direct comparisons between different architectures difficult. In addition, existing evaluations primarily emphasize prediction accuracy, while comparatively less attention is devoted to computational complexity, model size, training efficiency, and deployment constraints in resource-constrained IoT environments. Furthermore, only a limited number of studies investigate the impact of reduced-data acquisition and signal reduction strategies on forecasting performance in transient gas-sensing applications. Consequently, there remains a need for a systematic comparative analysis of different neural network architectures under reduced-data acquisition scenarios.

To address this gap, this paper provides:

- (1) A comparative evaluation of ANN, LSTM, and 1D-CNN models for transient gas sensor prediction;
- (2) An investigation of multiple sampling reduction strategies;
- (3) An analysis of accuracy, training time, and model size trade-offs.

Such an evaluation is particularly relevant for energy-efficient IoT deployments, where both prediction accuracy and computational requirements must be considered when designing practical gas-sensing systems.

3. Methodology of Implementing Energy Efficient Data Collection Methods

In this section, three different neural network models, ANN, LSTM, and 1D-CNN, are examined for the prediction of gas concentration levels. These models are compared in terms of training time, model size, and prediction accuracy, expressed as Root Mean Square Error (RMSE), R-squared (R^2) and Mean Absolute Error (MAE).

In our previous research [2], we developed and experimentally validated a data acquisition and prediction framework for energy-efficient operation of low-cost gas sensors. The study addressed a fundamental limitation of such sensors, the need for a preheating period before producing accurate readings by leveraging transient signal analysis. Instead of waiting for the sensor to fully stabilize, the approach used an LSTM neural network to predict the actual gas concentration from the early portion of the transient response.

The experimental setup consisted of two identical sensors: one continuously online (used as the ground truth) and one intermittently active, controlled via a real-time clock (DS3231) and MOSFET-based power switching on an Arduino Uno microcontroller. To ensure stable environmental conditions, the entire system was enclosed in a sealed chamber.

A detailed preliminary analysis was performed to identify the optimal balance between energy savings and predictive accuracy. Transient behavior was examined under different online/sleep ratios ranging from 10/60 s to 60/120 s. Results indicated that shorter online periods (<20 s) produced unstable readings, while longer ones (>40 s) offered minimal accuracy improvement but significantly increased power consumption. Consequently, an online period of 20 s and a sleep period of 120 s were selected as the optimal configuration, achieving over 80% energy savings compared to always-online operation while maintaining high correlation (>95%) with ground truth values.

Data were collected from 381 transients sampled at 10 Hz, normalized between 0 and 1, and categorized into eight gas concentration classes to ensure balanced training. The LSTM network was trained on 304 transients out of 381, using a threefold stratified validation approach that selects one third of transients from each gas level class, with hyperparameters optimized through testing various neuron counts, learning rates, and epochs. This leaves 77 out of 381 transients for the test set, which is around 20% of the whole data set. For simplicity and finite number of parameters for comparison, a single layer is considered for each neural network.

Furthermore, several input configurations were tested to assess the impact of signal length and sampling rate on performance. It was observed that using 20 evenly spaced values over the 20 s transient (sampling at 1 Hz) achieved better generalization than denser sampling, confirming that a compact feature set reduces overfitting and computational cost without sacrificing accuracy.

Building upon this methodology (Figure 1), the present study extends the analysis by comparing different DL architectures, ANN, LSTM, and 1D-CNN, using the same dataset and optimal online/sleep configuration. The goal is to evaluate model efficiency, scalability, and predictive accuracy for low-cost gas sensing applications under energy-constrained IoT conditions.

METHODOLOGY

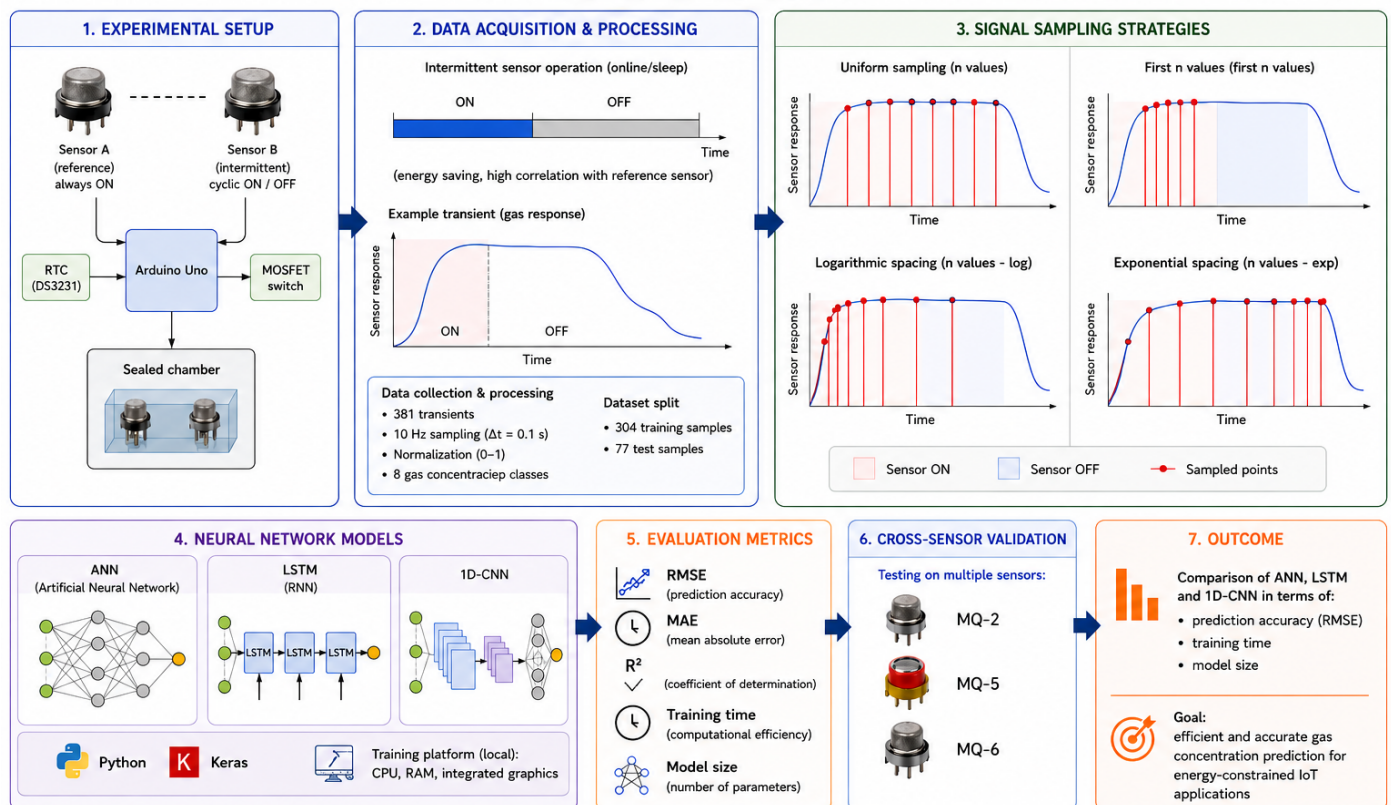


Figure 1. Methodology overview of implementing energy efficient data collection methods. Created with help of ChatGPT (GPT-5.5).

After completing the data collection process with the optimal online/sleep configuration of 20/120 s, the next step involved exploring how different portions of the transient signal could be used as input for neural network training. The goal was to determine the minimal signal length that still preserves sufficient information for accurate gas concentration prediction while reducing computational complexity and data transmission requirements.

To this end, several configurations of input signal length and sampling strategies were evaluated. Each configuration represents a subset of the 186 samples recorded during the 20 s active period (approximately nine readings per second). This systematic approach enables the assessment of model performance under varying data densities and temporal resolutions.

Figure 2 illustrates the process of dividing the signal into n samples according to four distinct strategies: n uniformly spaced readings (Figure 2a), the first n consecutive

readings (Figure 2b), n logarithmically spaced readings (Figure 2c), and n exponentially spaced readings (Figure 2d). These sampling schemes are applied across different values of n to investigate how the number and temporal distribution of samples affect model performance and generalization capability.

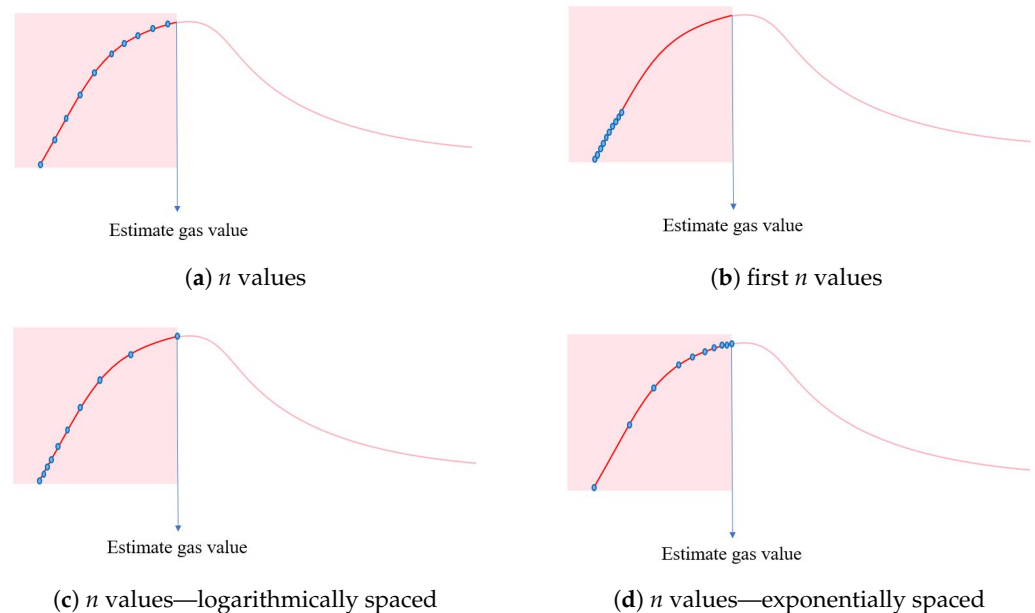


Figure 2. Gas sensor transient sampling options.

The following combinations of signal sizes were analyzed:

1. **Full signal:** All 186 values collected during the 20 s transient period (~ 9 readings per second—Figure 2a, $n = 20$, *sampling* = 100 ms).
2. **Reduced signal—20 samples:** 20 values collected over the 20 s interval using three different sampling strategies:
 - (a) Uniformly sampled data collected every second (Figure 2a, $n = 20$, *sampling* = 1000 ms);
 - (b) Logarithmically spaced readings from the first 20 s (Figure 2c, $n = 20$);
 - (c) Exponentially spaced readings from the first 20 s (Figure 2d, $n = 20$).
3. **Reduced signal—10 samples:** 10 values collected during the 20 s period (the sensor must remain online for 20 s to ensure proper heating), with five sampling strategies tested:
 - (a) Uniform sampling every two seconds (Figure 2a, $n = 10$);
 - (b) Data from the first 10 s, collected every second (Figure 2b, $n = 10$, *sampling* = 1000 ms);
 - (c) Data from the first second, collected every 100 ms (Figure 2b, $n = 10$, *sampling* = 100 ms);
 - (d) Logarithmically spaced readings over the 20 s interval (Figure 2c, $n = 10$);
 - (e) Exponentially spaced readings over the 20 s interval (Figure 2d, $n = 10$).
4. **Reduced signal—5 samples:** 5 values collected during 20 s, obtained as:
 - (a) Logarithmically spaced readings (Figure 2c, $n = 5$);
 - (b) Exponentially spaced readings (Figure 2d, $n = 5$).
5. **Reduced signal—3 samples:** 3 values collected during the 20 s interval:
 - (a) Logarithmically spaced readings (Figure 2c, $n = 3$);
 - (b) Exponentially spaced readings (Figure 2d, $n = 3$).

6. **Minimal signal:** Only two readings—the first and the last measurement collected during the 20 s period (Figure 2a, $n = 2$).

This systematic variation of signal size and temporal spacing allows for evaluating how the amount and distribution of input data influence the predictive accuracy and efficiency of different neural network architectures, providing a foundation for optimizing both model performance and sensor energy consumption in IoT-based gas monitoring systems.

After defining the signal segmentation and sampling strategies, three different neural network architectures were implemented and evaluated for gas concentration prediction: an ANN, an LSTM, and a 1D-CNN. These models were selected to compare different approaches to feature extraction and temporal pattern recognition in transient sensor data.

Each model was developed and trained using the Python 3.9.13 programming language with the Keras DL framework, allowing for rapid prototyping and fine-tuning of network architectures. The initial hyperparameter optimization was performed using data collected with the optimal 20/120 online/sleep period. The goal of this step was to identify the most suitable configuration for each model type, ensuring stable training dynamics and accurate predictions.

The final sets of optimized hyperparameters including the number of layers, neurons, learning rates, activation functions, batch sizes, and training epochs are summarized in Tables 1, 2 and 3, corresponding, respectively, to the ANN, LSTM, and 1D-CNN architectures. For consistency across evaluations, all three architectures employed the Adam optimizer, Rectified Linear Unit (ReLU) activation functions, and Mean Squared Error (MSE) as the loss function. Once determined, these optimal configurations were applied to all remaining signal segmentation variants (uniform, logarithmic, exponential, and reduced-length signals) to evaluate the robustness and generalization of the trained models under different input conditions. All experiments were executed 10 times and the average results are presented.

All models were trained locally on a personal computer equipped with an Intel Core i5-8500 CPU (3.0 GHz), 16 GB RAM, and integrated Intel UHD 630 graphics. This setup provided sufficient computational resources to train all three architectures within reasonable time frames without the use of dedicated GPU acceleration.

After training and validation on data obtained from the MQ-2 gas sensor, the best-performing model configurations were subsequently evaluated on MQ-5 and MQ-6 sensors to assess their generalization capability across similar sensor types with different sensitivities and response characteristics. This cross-sensor evaluation aims to verify the robustness and transferability of the developed neural network models in practical IoT-based gas sensing scenarios.

To ensure an objective and consistent comparison between the three neural network architectures, several performance and efficiency metrics were employed. The Root Mean Square Error (RMSE) served as the primary measure of prediction accuracy, quantifying the deviation between predicted and ground-truth gas concentration values. Additionally, the training time and model size (number of parameters) were recorded for each configuration to assess computational efficiency and scalability.

These metrics collectively provide insight into the trade-offs between accuracy, complexity, and resource requirements of each neural network model. The following section presents a detailed comparison of results across the ANN, LSTM, and 1D-CNN architectures using the defined sampling strategies and evaluation criteria.

Table 1. Parameter combinations for building ANN (the best in **bold**).

Parameter	Values
Units	20, 50, 80
Number of Epochs	250, 500, 750, 1000
Learning Rate	0.001 , 0.0001, 0.00001

Table 2. Parameter combinations for building LSTM (the best in **bold**).

Parameter	Values
Number of neurons	20, 50 , 80
Number of Epochs	250, 500 , 750, 1000
Learning Rate	0.001, 0.0001 , 0.00001

Table 3. Parameter combinations for building CNN (the best in **bold**).

Parameter	Values
Filters/Kernels	16 , 32, 64
Number of Epochs	250, 500, 750, 1000
Learning Rate	0.001 , 0.0001, 0.00001

4. Results

This section presents the results of time-series data prediction for gas sensor readings for all three considered neural networks, namely, ANN, LSTM and 1D-CNN. More details for results of each neural network are available in the following subsections.

4.1. ANN Results Overview

Results for the 20/120 online/sleep period ratio using the ANN model are presented in Table 4, summarizing the results for all combinations of hyperparameters, including the model training time, saved model size, and RMSE. The best-performing configuration is highlighted in red, with an RMSE of 6.74%. The corresponding model size is 2211 kB, and the required training time is 44.06 s. The optimal model features the lowest number of units and the smallest learning rate among all tested combinations, while the number of epochs is the highest. Furthermore, Table 4 indicates that the model size primarily depends on the number of units, whereas the training time is influenced by the number of epochs and the learning rate as well. Consequently, the model with the lowest RMSE also has the smallest size among all configurations, while about half of the models exhibit a shorter training time.

Table 4. Comparison of ANN architectures by model size, execution time, and RMSE (Optimal results in red).

Units	LR	Epoch	Time [s]	Model Size [kB]	RMSE [%]
20	0.001	250	11.38	2211	10.51%
20	0.001	500	21.82	2211	8.66%
20	0.001	750	32.25	2211	7.34%
20	0.001	1000	44.06	2211	6.74%
20	0.0001	250	11.40	2211	10.70%
20	0.0001	500	22.36	2211	12.85%
20	0.0001	750	32.56	2211	13.49%
20	0.0001	1000	42.80	2211	12.40%
20	0.00001	250	11.50	2211	17.64%
20	0.00001	500	21.83	2211	16.62%

Table 4. *Cont.*

Units	LR	Epoch	Time [s]	Model Size [kB]	RMSE [%]
20	0.00001	750	32.38	2211	15.95%
20	0.00001	1000	42.66	2211	15.78%
50	0.001	250	20.14	5481	15.51%
50	0.001	500	39.27	5481	12.28%
50	0.001	750	58.36	5481	13.96%
50	0.001	1000	83.06	5481	9.25%
50	0.0001	250	20.07	5481	15.78%
50	0.0001	500	39.52	5481	15.42%
50	0.0001	750	58.47	5481	14.98%
50	0.0001	1000	78.15	5481	15.23%
50	0.00001	250	20.22	5481	16.88%
50	0.00001	500	39.70	5481	16.61%
50	0.00001	750	59.69	5481	15.92%
50	0.00001	1000	77.74	5481	16.44%
80	0.001	250	29.40	8751	14.61%
80	0.001	500	58.09	8751	16.66%
80	0.001	750	86.96	8751	13.79%
80	0.001	1000	120.68	8751	9.30%
80	0.0001	250	29.30	8751	16.06%
80	0.0001	500	58.15	8751	15.09%
80	0.0001	750	86.52	8751	15.31%
80	0.0001	1000	115.73	8751	14.86%
80	0.00001	250	29.48	8751	17.40%
80	0.00001	500	58.61	8751	16.87%
80	0.00001	750	87.18	8751	16.61%
80	0.00001	1000	117.03	8751	16.01%

4.2. LSTM Results Overview

Results for the 20/120 online/sleep period ratio using the LSTM model are presented in Table 5, summarizing the results for all combinations of hyperparameters, including the model training time, saved model size, and RMSE. The best-performing configuration is highlighted in red, with an RMSE of 3.69%. The corresponding model size is 143 kB, and the required training time is 169.8 s. Please note that the NaN values observed for certain LSTM configurations were caused by training instability and convergence failure. Large weight updates, particularly when using higher learning rates, can lead to exploding gradients and numerical overflow during optimization, resulting in undefined (NaN) values. In our experiments, NaN values occurred in 5 out of the 36 evaluated LSTM hyperparameter combinations. These configurations failed to converge and therefore could not produce valid prediction results. Consequently, they were considered non-convergent and were excluded from further performance analysis and comparison.

The optimal model features a medium number of units, a medium learning rate, and a medium number of epochs among all tested configurations. Similarly to the ANN results, the model size primarily depends on the number of units (neurons), while the training time is influenced by all three considered hyperparameters. Consequently, the model with the lowest RMSE exhibits medium size and training time.

It is worth noting that the following combination of hyperparameters (*units* = 20, *LR* = 0.001, *epochs* = 750) achieves a comparable RMSE value of 4.09%. This configuration uses the lowest number of units and the smallest learning rate but includes 50% more

epochs than the model selected as optimal. However, its training time is approximately 30% higher compared to the optimal model.

Table 5. Comparison of LSTM architectures by model size, execution time, and RMSE (Optimal results in red).

Units	LR	Epoch	Time [s]	Model Size [kB]	RMSE [%]
20	0.001	250	73.80	41	22.87%
20	0.001	500	144.93	41	12.05%
20	0.001	750	216.91	41	4.09%
20	0.001	1000	289.30	41	12.41%
20	0.0001	250	73.01	41	12.21%
20	0.0001	500	145.23	41	11.08%
20	0.0001	750	220.00	41	5.83%
20	0.0001	1000	288.71	41	NaN
20	0.00001	250	72.61	41	44.67%
20	0.00001	500	146.56	41	25.89%
20	0.00001	750	216.64	41	14.74%
20	0.00001	1000	290.07	41	20.34%
50	0.001	250	85.96	143	12.91%
50	0.001	500	168.83	143	NaN
50	0.001	750	254.10	143	11.12%
50	0.001	1000	336.84	143	4.41%
50	0.0001	250	86.72	143	5.88%
50	0.0001	500	169.89	143	3.69%
50	0.0001	750	251.47	143	8.55%
50	0.0001	1000	336.26	143	5.30%
50	0.00001	250	83.41	143	31.88%
50	0.00001	500	167.01	143	18.62%
50	0.00001	750	250.44	143	12.33%
50	0.00001	1000	334.39	143	11.65%
80	0.001	250	111.95	329	7.92%
80	0.001	500	223.94	329	NaN
80	0.001	750	340.09	329	NaN
80	0.001	1000	451.69	329	NaN
80	0.0001	250	111.60	329	NaN
80	0.0001	500	227.73	329	12.21%
80	0.0001	750	334.95	329	6.01%
80	0.0001	1000	454.21	329	4.43%
80	0.00001	250	111.81	329	17.49%
80	0.00001	500	223.28	329	13.10%
80	0.00001	750	336.39	329	12.84%
80	0.00001	1000	462.79	329	9.62%

4.3. CNN Results Overview

Results for the 20/120 online/sleep period ratio using the CNN model are presented in Table 6, summarizing the results for all combinations of hyperparameters, including the model training time, saved model size, and RMSE. The best-performing configuration is highlighted in red, with an RMSE of 5.48%. The corresponding model size is 1756 kB, and the required training time is 44.14 s. The optimal model features the lowest number of units and the smallest learning rate among all tested combinations, while the number of epochs is the highest. Furthermore, Table 6 indicates that the model size primarily depends on the number of units, whereas the training time is influenced by the number of epochs

and the learning rate. Consequently, the model with the lowest RMSE also has the smallest size among all configurations, with 20 models exhibiting a shorter training time. None of those models achieves a comparable RMSE.

Table 6. Comparison of CNN architectures by model size, execution time, and RMSE (Optimal results in red).

Kernels	LR	Epoch	Time [s]	Model Size [kB]	RMSE [%]
16	0.001	250	12.02	1756	7.92%
16	0.001	500	22.83	1756	9.81%
16	0.001	750	33.46	1756	6.36%
16	0.001	1000	44.14	1756	5.48%
16	0.0001	250	11.71	1756	13.99%
16	0.0001	500	23.95	1756	14.63%
16	0.0001	750	33.70	1756	13.33%
16	0.0001	1000	45.38	1756	12.50%
16	0.00001	250	11.66	1756	19.53%
16	0.00001	500	22.69	1756	15.76%
16	0.00001	750	37.97	1756	15.97%
16	0.00001	1000	45.01	1756	15.69%
32	0.001	250	15.58	3482	12.62%
32	0.001	500	30.12	3482	12.61%
32	0.001	750	43.92	3482	10.00%
32	0.001	1000	60.79	3482	7.65%
32	0.0001	250	15.52	3482	16.13%
32	0.0001	500	30.31	3482	15.56%
32	0.0001	750	44.07	3482	15.06%
32	0.0001	1000	58.51	3482	14.49%
32	0.00001	250	15.91	3482	16.74%
32	0.00001	500	30.86	3482	16.21%
32	0.00001	750	44.65	3482	17.38%
32	0.00001	1000	58.89	3482	15.78%
64	0.001	250	24.65	6934	14.24%
64	0.001	500	48.82	6934	11.19%
64	0.001	750	71.93	6934	11.25%
64	0.001	1000	96.16	6934	12.29%
64	0.0001	250	24.98	6934	16.15%
64	0.0001	500	52.08	6934	15.85%
64	0.0001	750	73.93	6934	15.61%
64	0.0001	1000	96.50	6934	14.99%
64	0.00001	250	24.64	6934	17.26%
64	0.00001	500	48.23	6934	17.36%
64	0.00001	750	72.16	6934	16.01%
64	0.00001	1000	117.03	8751	16.01%

4.4. Segment Size Analysis

After selecting the best combination of hyperparameters for each neural network trained with the 20/120 online/sleep ratio using all 186 available readings, this configuration was used to build 11 additional models. These new models maintained the same online/sleep ratio; however, the number of readings within that period was varied to investigate whether comparable or improved performance could be achieved with reduced input data.

The results are shown in Table 7 and in Figures 3–5. Figures 3–5 illustrate the relationship between time (x-axis) and RMSE (y-axis). Model architectures are differentiated by markers: circles for ANN, squares for 1D-CNN, and triangles for LSTM, while the various configurations are color-coded according to the legend. As shown in Table 7 and in Figures 3–5, feeding the models with all 186 readings yields the highest accuracy. Under this configuration, the LSTM model achieved the best overall performance with an RMSE of 3.69%, an MAE of 0.04, and an R^2 value of 0.85. The 1D-CNN model followed with an RMSE of 5.48%, MAE of 0.04, and R^2 of 0.81, while the ANN model achieved an RMSE of 6.74%, MAE of 0.06, and R^2 of 0.71. However, if the objective is to reduce the amount of data processed through the network while tolerating slightly lower accuracy, selecting 20 values, either fixed or logarithmically spaced provides the best RMSE with lower model size and/or training time.

Furthermore, reducing the input from 20 to 10 readings also offers a reasonable trade-off.

For the ANN and 1D-CNN architectures, exponentially spaced samples produced the best performance, achieving RMSE values of 8.02% and 7.30%, MAE values of 0.06 and 0.05, and R^2 values of 0.65 and 0.71, respectively.

Such a reduction slightly worsens the performance, but the model size decreases by up to 50%, while the training time remains largely unaffected. For the LSTM model, 10 fixed readings spaced every 10 s should be used. This configuration achieves an RMSE of 7.17%. Although the model size remains constant, the training time is reduced by approximately 25%.

Overall, the results are about 2–4% less accurate; however, the LSTM model requires 50% fewer input points, which contributes to moderate energy savings whether the model is executed on a microcontroller or deployed in the cloud.

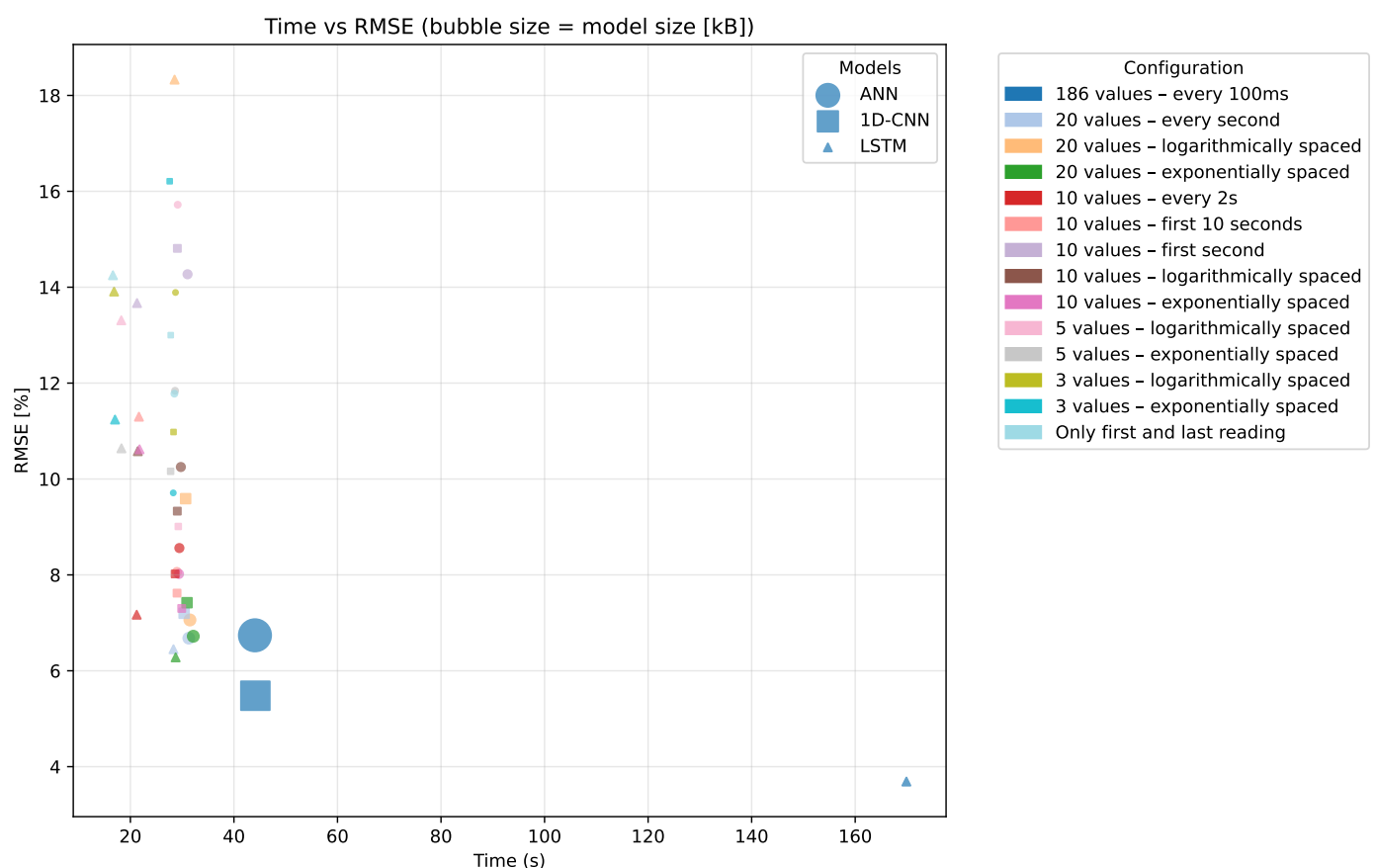


Figure 3. Results for different transient sizes—All algorithms (RMSE).

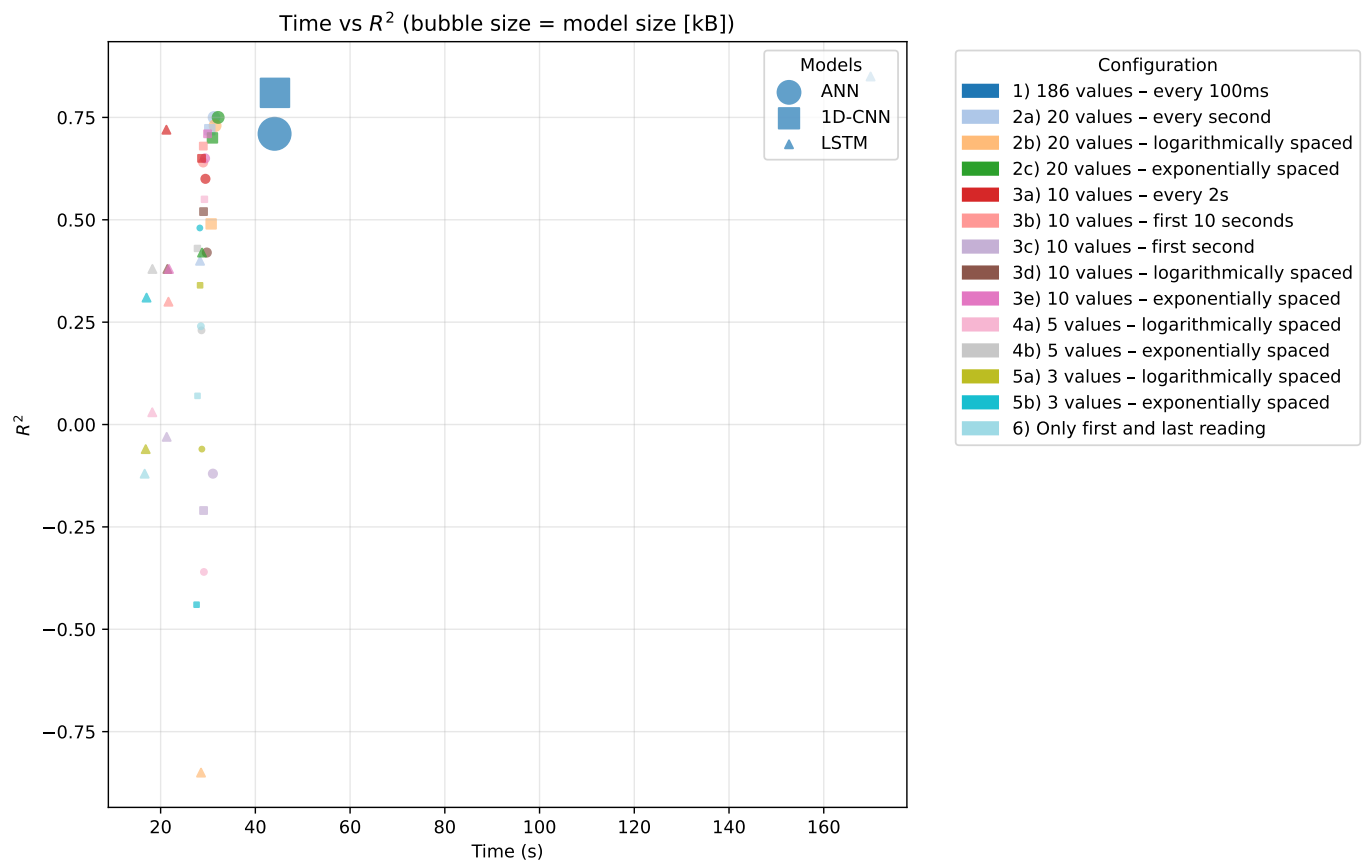


Figure 4. Results for different transient sizes—All algorithms (R^2).

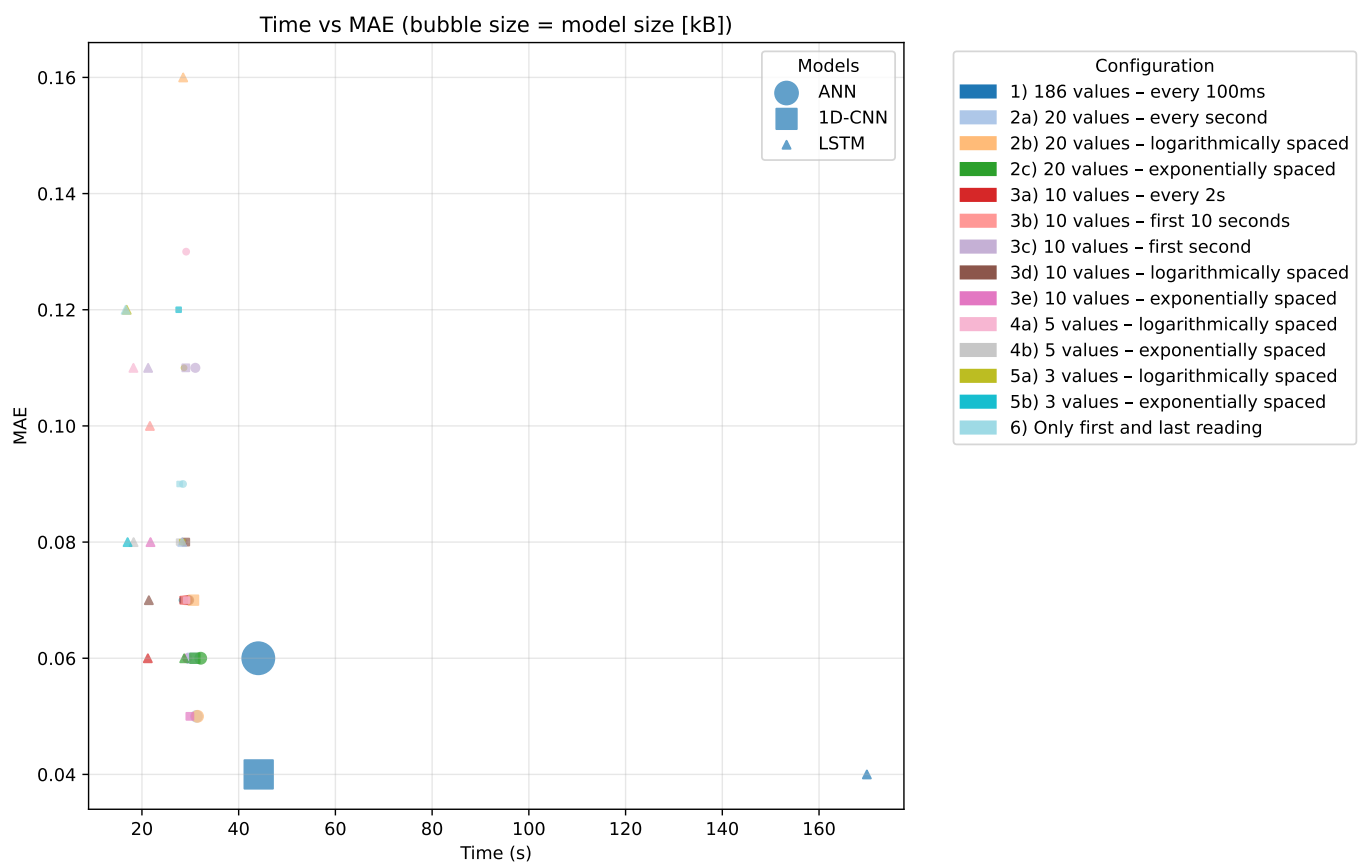


Figure 5. Results for different transient sizes—All algorithms (MAE).

Table 7. Results for different transient sizes—All algorithms.

	RMSE [%]	R ² [-]	MAE [-]	TIME [s]	Size [kB]	RMSE [%]	R ² [-]	MAE [-]	TIME [s]	Size [kB]	RMSE [%]	R ² [-]	MAE [-]	TIME [s]	Size [kB]
(1) 186 values—every 100 ms	6.74	0.71	0.06	44.06	2211	5.48	0.81	0.04	44.14	1756	3.69	0.85	0.04	169.89	143
(2a) 20 values—every second	6.68	0.75	0.05	31.28	256	7.20	0.72	0.06	30.38	200	10.45	0.40	0.08	28.32	143
(2b) 20 values—logarithmically spaced	7.06	0.73	0.05	31.50	265	9.59	0.49	0.07	30.67	200	18.33	−0.85	0.16	28.53	143
(2c) 20 values—exponentially spaced	6.72	0.75	0.06	32.15	265	7.42	0.70	0.06	30.94	200	10.28	0.42	0.06	28.74	143
(3a) 10 values—every 2 s	8.56	0.60	0.07	29.46	148	8.02	0.65	0.07	28.64	106	7.17	0.72	0.06	21.22	143
(3b) 10 values—first 10 s	8.06	0.64	0.07	28.98	148	7.62	0.68	0.07	29.01	106	11.30	0.30	0.10	21.65	143
(3c) 10 values—first second	14.27	−0.12	0.11	31.04	148	14.81	−0.21	0.11	29.07	106	13.67	−0.03	0.11	21.28	143
(3d) 10 values—logarithmically spaced	10.25	0.42	0.07	29.75	148	9.33	0.52	0.08	29.06	106	10.58	0.38	0.07	21.43	143
(3e) 10 values—exponentially spaced	8.02	0.65	0.06	29.38	148	7.30	0.71	0.05	29.91	106	10.62	0.38	0.08	21.77	143
(4a) 5 values—logarithmically spaced	15.72	−0.36	0.13	29.14	78	9.01	0.55	0.07	29.25	69	13.31	0.03	0.11	18.24	143
(4b) 5 values—exponentially spaced	11.84	0.23	0.08	28.63	78	10.16	0.43	0.08	27.77	69	10.64	0.38	0.08	18.27	143
(5a) 3 values—logarithmically spaced	13.89	−0.06	0.11	28.71	54	10.98	0.34	0.08	28.33	50	13.91	−0.06	0.12	16.85	143
(5b) 3 values—exponentially spaced	9.71	0.48	0.07	28.28	54	16.21	−0.44	0.12	27.59	50	11.24	0.31	0.08	17.03	143
(6) Only first and last reading	11.78	0.24	0.09	28.50	78	13.00	0.07	0.09	27.79	50	14.25	−0.12	0.12	16.63	143

Finally, training models using only the first and last readings—thus reducing the input to merely two values—results in an RMSE of approximately 12–14%, with training time as low as 16.63 s and model size as small as 50 kB. This configuration may also represent an acceptable compromise in scenarios where data size is a critical constraint.

5. Discussion, Limitations and Future Work

This section provides a comprehensive discussion of the experimental findings, focusing on the trade-offs between model architectures in terms of computational efficiency, model complexity, and prediction accuracy. The results are interpreted beyond numerical comparisons to highlight their practical implications for real-world deployment scenarios. In addition, the key limitations of the study are explicitly addressed, and directions for future research are outlined to guide further improvements in this domain.

5.1. Model Performance Trade-Offs

Table 8 compares the three neural network types in terms of computational time. The LSTM model demonstrates the longest training duration due to its recursive structure and ability to capture long-term temporal dependencies. In contrast, the ANN model is the fastest, as it is optimized for tabular data and lacks temporal context modeling capabilities. The 1D-CNN represents a middle ground faster than LSTM but slower than ANN, while being effective in capturing short-term temporal patterns.

Model size results, presented in Table 9, indicate that LSTM networks are least sensitive to hyperparameter and dataset size variations, maintaining relatively compact models. Conversely, ANN architectures exhibit higher model size sensitivity due to their dense connections.

The overall comparison of results presented in Table 10 confirms that LSTM models generally yield the smallest file sizes but at the cost of longer training times. The ANN model remains the fastest to train but produces the largest models and is less efficient for sequential data. Interestingly, as the input sequence length decreases, the LSTM becomes faster, while the 1D-CNN tends to produce smaller models (case 6).

These differences can be attributed to the inherent architectural characteristics of the models in relation to the temporal structure of gas sensor signals. LSTM networks benefit from explicit memory mechanisms that enable the capture of long-term dependencies present in transient sensor responses, whereas ANN models lack sequential modeling capability and therefore cannot effectively exploit temporal correlations. The 1D-CNN model provides intermediate performance by extracting localized temporal patterns through convolutional filters, but its receptive field limits its ability to fully model long-range dependencies.

In addition, cross-sensor variations observed across MQ-2, MQ-5, and MQ-6 sensors further highlight the influence of sensor-specific characteristics on model performance. Differences in sensitivity, response time, and noise levels affect the complexity of the prediction task; however, the overall performance trends among ANN, CNN, and LSTM models remain consistent across sensors, indicating good generalization capability of the proposed comparative framework.

Table 8. Time comparison.

Model	Training Time (per Epoch)	Speed Notes	Best For
ANN (Dense)	Fastest	Matrix multiplications only; fully parallelizable on GPU/CPU	Low-latency tasks, tabular data
1D CNN	Fast	Can be parallelized; depends on kernel size and depth	Short-term sequence modeling
LSTM	Slowest	Sequential—time-step by time-step; less GPU-efficient	Capturing long-term temporal dependencies

Table 9. Size comparison.

Model Type	Key Layer	Param Dependency on n_features	File Size Sensitivity	Notes
LSTM	LSTM (units)	Affects only the input weight matrix of the first LSTM layer	Moderate	Weight matrices grow, but LSTM reuses hidden states; only the first layer depends on n_features
1D CNN	Conv1D (filters, kernel_size)	Every Conv1D layer's kernel depends on n_features	High	Filters operate across all n_features—size increases linearly with features
ANN (Dense)	Dense (units)	Each Dense layer's size fully scales with n_features	Highest	First Dense layer has n_features × units parameters

Table 10. Comparison between training time and model size.

Model	File Size Sensitivity	Training Speed	Best Use Case
LSTM	Low–Moderate	Slow	Long-term dependencies (e.g., sequences, forecasting)
1D CNN	High	Fast	Short-term patterns in time series
ANN	Highest	Very Fast	Tabular data or static features

5.2. Practical Implications

For applications where maximum accuracy is the primary objective, such as navigation safety systems or predictive maintenance frameworks, the full LSTM model trained on all 186 readings is the most suitable. However, in real-world deployments with computational or memory constraints, such as embedded or real-time maritime systems, a trade-off between accuracy, model size, and training time becomes necessary.

Models trained on shorter input segments, such as the LSTM (10 values, case 3a) or the 1D-CNN (10 values, case 3e), demonstrate significant efficiency gains while maintaining acceptable prediction accuracy. Training models on only the first and last values yields an RMSE between 11.75% and 14.25%, with training times reduced to 16.6–28.5 s and model sizes as small as 50 kB. This behavior can be explained by the structure of gas sensor transient responses, where the most informative variations occur in the early dynamic phase, while later measurements contribute less additional information. These lightweight models are suitable for low-power, edge-computing environments or systems requiring rapid retraining with limited onboard resources.

Gas sensor transient responses are characterized by a rapid initial change followed by a slower convergence to a steady-state value. This non-uniform temporal structure implies that early-time measurements carry more discriminative information about the gas concentration dynamics. Sampling strategies that allocate higher temporal resolution to the early transient phase (such as logarithmic or exponential sampling) are therefore better aligned with the physical behavior of the sensor signal compared to uniform sampling [2].

5.3. Study Limitations

Several limitations of the present study should be acknowledged. First, the dataset consists of experimentally collected gas sensor transient responses and is relatively modest in size compared with datasets commonly used in large-scale deep learning applications. Nevertheless, all evaluated models were trained and tested under identical conditions, ensuring a fair comparative assessment.

Second, the study focuses on representative single-layer ANN, LSTM, and 1D-CNN architectures in order to maintain comparable model complexity and facilitate the analysis

of accuracy-resource trade-offs. The investigation of deeper architectures may provide additional performance improvements and should be explored in future research.

Third, the experimental evaluation was performed on a desktop computing platform and did not include deployment on embedded hardware devices. Although model size and training time were considered, future studies should investigate inference latency, energy consumption, and memory requirements on resource-constrained IoT hardware platforms.

5.4. Future Work

Future research should expand the hyperparameter optimization process and explore automated tuning techniques for shorter transient sequences. While this study focused on single-layer architectures, further performance improvements could be achieved through multi-layer or hybrid neural networks [3], such as LSTM–CNN combinations, which can integrate both long- and short-term feature learning. Additionally, model compression techniques such as pruning, quantization, and knowledge distillation may be investigated to further reduce computational cost and improve deployability in resource-constrained IoT environments.

Moreover, emerging architectures like Graph Neural Networks (GNNs) represent a promising direction for time-series prediction. Unlike conventional models that treat temporal data as independent sequences, GNNs model data as graph structures, where nodes represent sensors or variables, and edges capture spatial or relational dependencies. This allows GNNs to jointly learn spatial and temporal relationships, making them particularly effective systems characterized by networked sensors and correlated measurements.

Integrating GNNs with temporal models such as Temporal Convolutional Networks (TCNs) or LSTMs could enable spatio-temporal forecasting frameworks that are more robust, scalable, and capable of handling dynamic, multi-source sensor data. Such hybrid approaches could significantly enhance predictive accuracy and resilience in complex, data-driven environments.

Furthermore, future work may involve extending the proposed evaluation to real-world embedded deployments, including inference latency, memory footprint, and energy consumption analysis on edge IoT hardware to further assess practical applicability.

6. Conclusions

This study evaluated the performance of three neural network architectures, ANN, 1D-CNN, and LSTM, for predicting transient responses of low-cost gas sensors under energy-efficient data acquisition conditions. The experimental framework considered multiple configurations with varying input sizes and hyperparameter combinations, emphasizing the trade-offs between model accuracy, training time, and memory requirements.

The results demonstrate that the LSTM model achieved the highest prediction accuracy, with a minimum RMSE of 3.69%, R^2 of 0.85 and scaled MAE of 0.04, effectively capturing long-term temporal dependencies. The ANN exhibited the shortest training time but showed higher sensitivity to input length and model size. The 1D-CNN offered a balanced compromise between computational efficiency and prediction accuracy, capturing short-term temporal patterns with moderate resource usage.

Reducing the number of input readings from 186 to 10–20 values resulted in only a 2–4% increase in RMSE, while significantly reducing model size (up to 50%) and training time. Such configurations are highly suitable for resource-constrained or edge-based IoT systems, where computational efficiency and energy consumption are critical factors.

Future work will investigate automated hyperparameter optimization, hybrid deep learning architectures, embedded deployment, and GNN-based spatio-temporal models for distributed sensor networks.

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

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset available on request from the authors, the same dataset used in previous research is available at doi: 10.1109/JIOT.2020.2990526.

Acknowledgments: During the preparation of this work, the authors used ChatGPT (GPT-5.5) to improve the readability and language of the manuscript, as well as to assist in the generation of Figure 1. After using this tool, the authors reviewed and edited the content as needed and assume full responsibility for the content of the published article.

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

References

1. Čulić Gambiroža, J.; Mastelić, T. Big Data Challenges and Trade-offs in Energy Efficient Internet of Things systems. In Proceedings of the 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2018), Split, Croatia, 13–15 September 2018; pp. 1–6. [\[CrossRef\]](#)
2. Čulić Gambiroža, J.; Mastelić, T.; Kovačević, T.; Čagalj, M. Predicting Low-Cost Gas Sensor Readings From Transients Using Long Short-Term Memory Neural Networks. *IEEE Internet Things J.* **2020**, *7*, 8451–8461. [\[CrossRef\]](#)
3. Liu, Z.; Zhu, Z.; Gao, J.; Xu, C. Forecast Methods for Time Series Data: A Survey. *IEEE Access* **2021**, *9*, 91896–91912. [\[CrossRef\]](#)
4. Ensafi, Y.; Amin, S.H.; Zhang, G.; Shah, B. Time-series forecasting of seasonal items sales using machine learning—A comparative analysis. *Int. J. Inf. Manag. Data Insights* **2022**, *2*, 100058. [\[CrossRef\]](#)
5. Čulić Gambiroža, J.; Mastelić, T.; Nižetić Kosović, I.; Čagalj, M. Dynamic monitoring frequency for energy-efficient data collection in Internet of Things. *J. Comput. Sci.* **2022**, *64*, 101842. [\[CrossRef\]](#)
6. Chandra, R.; Goyal, S.; Gupta, R. Evaluation of Deep Learning Models for Multi-Step Ahead Time Series Prediction. *IEEE Access* **2021**, *9*, 83105–83123. [\[CrossRef\]](#)
7. Mahmoud, A.; Mohammed, A. A Survey on Deep Learning for Time-Series Forecasting. In *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges*; Hassanien, A.E., Darwish, A., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 365–392. [\[CrossRef\]](#)
8. Tang, Z.; Fishwick, P.A. Feedforward Neural Nets as Models for Time Series Forecasting. *ORSA J. Comput.* **1993**, *5*, 374–385. [\[CrossRef\]](#)
9. Wu, X.; Wang, Y. Extended and Unscented Kalman filtering based feedforward neural networks for time series prediction. *Appl. Math. Model.* **2012**, *36*, 1123–1131. [\[CrossRef\]](#)
10. Vishwakarma, G.K.; Paul, C.; Elsayah, A. A hybrid feedforward neural network algorithm for detecting outliers in non-stationary multivariate time series. *Expert Syst. Appl.* **2021**, *184*, 115545. [\[CrossRef\]](#)
11. Machado, E.; Pinto, T.; Guedes, V.; Morais, H. Electrical Load Demand Forecasting Using Feed-Forward Neural Networks. *Energies* **2021**, *14*, 7644. [\[CrossRef\]](#)
12. Moustati, I.; Gherabi, N.; Saadi, M. Time-Series Forecasting Models for Smart Meters Data: An Empirical Comparison and Analysis. *J. Eur. Des. Systèmes Autom.* **2024**, *57*, 1419–1427. [\[CrossRef\]](#)
13. Selmy, H.A.; Mohamed, H.K.; Medhat, W. A Predictive Analytics Framework for Sensor Data Using Time Series and Deep Learning Techniques. *Neural Comput. Appl.* **2024**, *36*, 6119–6132. [\[CrossRef\]](#)
14. Kolokythas, K.V.; Argiriou, A.A. An application of a feed-forward neural network model for wind speed predictions. *Int. J. Sustain. Energy* **2022**, *41*, 323–340. [\[CrossRef\]](#)
15. Hewamalage, H.; Bergmeir, C.; Bandara, K. Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *Int. J. Forecast.* **2021**, *37*, 388–427. [\[CrossRef\]](#)
16. Hüsken, M.; Stage, P. Recurrent neural networks for time series classification. *Neurocomputing* **2003**, *50*, 223–235. [\[CrossRef\]](#)
17. Barrera-Animas, A.Y.; Oyedele, L.O.; Bilal, M.; Akinosho, T.D.; Delgado, J.M.D.; Akanbi, L.A. Rainfall prediction: A comparative analysis of modern machine learning algorithms for time-series forecasting. *Mach. Learn. Appl.* **2022**, *7*, 100204. [\[CrossRef\]](#)

18. Tran, T.T.K.; Bateni, S.M.; Ki, S.J.; Vosoughifar, H. A Review of Neural Networks for Air Temperature Forecasting. *Water* **2021**, *13*, 1294. [[CrossRef](#)]
19. Jia, Z.; Lyu, X.; Zhang, W.; Martin, R.P.; Howard, R.E.; Zhang, Y. Continuous Low-Power Ammonia Monitoring Using Long Short-Term Memory Neural Networks. In Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys), Shenzhen, China, 4–7 November 2018; pp. 1–6. [[CrossRef](#)]
20. Lindemann, B.; Müller, T.; Vietz, H.; Jazdi, N.; Weyrich, M. A survey on long short-term memory networks for time series prediction. *Procedia CIRP* **2021**, *99*, 650–655. . [[CrossRef](#)]
21. Liu, C.; Zhang, A.; Xue, J.; Lei, C.; Zeng, X. LSTM-Pearson Gas Concentration Prediction Model Feature Selection and Its Applications. *Energies* **2023**, *16*, 2318. [[CrossRef](#)]
22. Zhang, Y.; Zhou, T.; Huang, X.; Cao, L.; Zhou, Q. Fault diagnosis of rotating machinery based on recurrent neural networks. *Measurement* **2021**, *171*, 108774. [[CrossRef](#)]
23. Kyi, S.; Taparugssanagorn, A. IoT and Machine Learning Models for Multivariate Very Short-Term Time Series Solar Power Forecasting. *IET Wirel. Sens. Syst.* **2024**, *14*, 381–395. [[CrossRef](#)]
24. Wang, H.; Gao, X.; Zhang, Y.; Yang, Y. Natural Gas Consumption Forecasting Model Based on Feature Optimization and Incremental Long Short-Term Memory. *Sensors* **2025**, *25*, 3079. [[CrossRef](#)] [[PubMed](#)]
25. Khatun, M.A.; Bhattacharya, M.; Eising, C.; Dhirani, L.L. Time Series Anomaly Detection with CNN for Environmental Sensors in Healthcare-IoT. In Proceedings of the 2024 IEEE 12th International Conference on Healthcare Informatics (ICHI), Orlando, FL, USA, 3–6 June 2024; pp. 522–524. [[CrossRef](#)]
26. Lateef, R.; Abbas, A. Tuning the Hyperparameters of the 1D CNN Model to Improve the Performance of Human Activity Recognition. *Eng. Technol. J.* **2022**, *40*, 547–554. [[CrossRef](#)]
27. Azizjon, M.; Jumabek, A.; Kim, W. 1D CNN based network intrusion detection with normalization on imbalanced data. In Proceedings of the International Conference on Artificial Intelligence in Information and Communication (ICAIIIC 2020), Fukuoka, Japan, 19–21 February 2020; pp. 218–224. [[CrossRef](#)]
28. Kiranyaz, S.; Avci, O.; Abdeljaber, O.; Ince, T.; Gabbouj, M.; Inman, D.J. 1D convolutional neural networks and applications: A survey. *Mech. Syst. Signal Process.* **2021**, *151*, 107398. [[CrossRef](#)]
29. Tang, W.; Long, G.; Liu, L.; Zhou, T.; Jiang, J.; Blumenstein, M. Rethinking 1D-CNN for Time Series Classification: A Stronger Baseline. *arXiv* **2020**, arXiv:2002.10061.
30. Hall, T.; Rasheed, K. A Survey of Machine Learning Methods for Time Series Prediction. *Appl. Sci.* **2025**, *15*, 5957. [[CrossRef](#)]
31. Kim, J.; Kim, H.; Kim, H.; Lee, D.; Yoon, S. A Comprehensive Survey of Deep Learning for Time Series Forecasting: Architectural Diversity and Open Challenges. *Artif. Intell. Rev.* **2025**, *58*, 216. [[CrossRef](#)]
32. Kong, X.; Chen, Z.; Liu, W.; Ning, K.; Zhang, L.; Marier, S.M.; Liu, Y.; Chen, Y.; Xia, F. Deep Learning for Time Series Forecasting: A Survey. *Int. J. Mach. Learn. Cybern.* **2025**, *16*, 5079–5112. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.