Intelligent Real-Time Air Quality Index Classification for Smart Home Digital Twins

Saley Saleh[1], A. S. Abohamama[2], A. S. Tolba[3]
Department of Computer Science-Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt[1, 2, 3]
Department of Computer Science, Arab East Colleges, Riyadh 53354, Saudi Arabia[2]

***Abstract*—This paper investigates the application of machine learning and deep learning models for intelligent real-time Air Quality Index (AQI) classification within a smart home digital twin context. Leveraging sensor data encompassing CO2 and TVOC levels, we perform a comparative analysis of eight models: Transformer Neural Network (TNN), Convolutional Neural Networks (CNN), Gated Recurrent Units (GRU), Recurrent Neural Networks (RNN), Support Vector Machines (SVM), Random Forest (RF), Gradient Boosting (GB), and K-Nearest Neighbors (KNN). These models aim to accurately classify air quality into six categories corresponding to AQI levels, ranging from Good to Hazardous, which are critical for assessing health risks. The performance of each model is rigorously evaluated using metrics including accuracy, precision, recall, F1-score, and ROC curves. Our findings demonstrate that the implemented models exhibit strong performance. This high-accuracy classification enables the smart home digital twin to move beyond passive monitoring, enabling proactive environmental control. For instance, the digital twin can use this real-time AQI classification to automatically adjust HVAC systems, trigger air purifiers when indoor air quality degrades, and potentially inform occupancy schedules. This integration allows for intelligent, adaptive management of the home's environment, ensuring optimal indoor air quality and occupant well-being. The paper also discusses the limitations of each model and suitable application scenarios for intelligent AQI management within the digital twin framework, offering valuable insights for the selection of appropriate air quality classification models in smart home environments.**

Keywords*—Air quality classification; machine learning; deep learning; Convolutional Neural Networks; Recurrent Neural Networks; transformer; Support Vector Machines; Random Forest; Gradient Boosting; k-nearest neighbors; CCS811 sensor data

I. Introduction

The digital world and digital technologies are constantly increasing. One of the most important digital technologies is the digital twin. A digital twin is a virtual twin or digital copy of a physical asset, system, process, or product that operates in a virtual environment. The digital twin acts as a bridge between the physical entities and the virtual environment. One of these field of digital twin is smart building that spans the building lifecycle and collect real-time data from building by using sensors to control the behavior and monitor operations to optimize building performance and improve the decision making. Air pollution is a major environmental concern affecting public health worldwide [1]. Accurate and reliable air quality classification is crucial for implementing effective mitigation strategies and informing the public [2].

Air quality is a very important factor anywhere, especially in enclosed spaces. To ensure human safety, air quality must be monitored. Monitoring air quality means know the percentage of harmful gases such as carbon dioxide and volatile organic compounds in the surrounding environment. Air pollution is responsible for many diseases, including lung cancer, asthma, and heart disease, and it can also cause a wide range of other health problems. Traditional methods of air quality assessment rely on expensive and complex analytical laboratory-based methods. However, with the advancements in low-cost sensor technologies, real-time, local air quality monitoring has become increasingly feasible.

This paper explores the application of various machine learning (ML) and deep learning (DL) models for air quality classification in smart building using a real dataset composed of CO2 and TVOC CCS811 sensor readings. CCS811 is an Air Quality Sensor can measure the CO2 (equivalent CO2) and TVOC (Total Volatile Organic Compounds) density. We analyze the performance of eight models: Transformer Neural Network (TNN), Convolutional Neural Networks (CNN), Gated Recurrent Units (GRU), Recurrent Neural Networks (RNN), Support Vector Machines (SVM), Random Forest, Gradient Boosting, and K-Nearest Neighbors (KNN). This study can help to identify the optimal models for this task. We highlight the strengths and weaknesses of each model in the context of air quality classification and discuss their suitability for different applications.

II. Literature Review

The imperative for effective air quality monitoring has spurred significant research into the use of computational techniques, with a notable focus on machine learning (ML) and deep learning (DL). Traditional approaches to air quality classification rely on laboratory-based analyses of complex compounds. These approaches are often time-consuming and expensive and not suitable for real time analysis [2]. Several studies have explored the use of different models that can analyze the data for real time and cost-effective classification.

Classical machine learning techniques have been widely applied in the realm of air quality prediction and assessment. For instance, Support Vector Machines (SVMs) have demonstrated their ability in creating robust decision boundaries, performing effectively in high-dimensional data spaces [3]. Similarly, K-Nearest Neighbors (KNN) approaches have been utilized, showcasing its simplicity and effectiveness in numerous classification tasks [4]. Furthermore, tree-based ensemble methods have shown promise in this domain. Random Forest

algorithms have demonstrated strong generalization performance, effectively handling complex, non-linear data [5]. Additionally, boosting methods such as AdaBoost have proven useful in combining weak learners into strong classifiers, often achieving good performance on imbalanced and complex datasets [6].

The advancement of deep learning has also brought notable contributions to air quality analysis. Deep Neural Networks (DNNs), including Convolutional Neural Networks (CNNs), have proven useful in identifying spatial patterns and hierarchical features from sensor data [7]. Furthermore, Recurrent Neural Networks (RNNs) have demonstrated their ability to capture temporal dependencies in time series data, making them applicable in situations with continuous sensor data [8]. The Transformer model, a relatively recent advancement in deep learning, has shown impressive results in numerous fields, exhibiting the power of self-attention mechanisms in data modeling and classification [9, 10]. It has been used for various classification, regression and other data processing tasks. The integration of air quality monitoring systems within smart homes is a growing area of interest, particularly in the context of digital twins, which are virtual replicas of physical environments. These systems leverage Internet of Things (IoT) technologies, low-cost sensors, and advanced machine learning models to provide real-time insights into indoor air quality (IAQ). Such insights are pivotal for enhancing occupant health, comfort, and well-being. This review synthesizes recent advancements in IAQ monitoring and classification, focusing on their potential applications in digital twins for smart homes.

III. Indoor Air Quality Monitoring Systems

Castellani et al. (2021) [15] present a systematic review of IoT-based systems for IAQ monitoring, highlighting that thermal comfort parameters, CO_2, and particulate matter (PM) levels are the most frequently monitored metrics, with 70%, 65%, and 27.5% of studies focusing on these aspects, respectively. The authors also note that Arduino and Raspberry Pi controllers dominate system designs, accounting for 37.5% and 35% of implementations. However, only 22.5% of systems adopt calibration approaches prior to deployment, raising concerns about data accuracy (Castellani, Benini, & Brunelli, 2021). For digital twins in smart homes, precise calibration is essential to ensure reliable IAQ classification, as inaccuracies could compromise the twin's ability to reflect real-world conditions. Low-cost air quality sensors (LCS) have emerged as a feasible solution for pervasive monitoring, as discussed by De Vito et al. (2024) [16] and Higgins et al. (2024). While LCS offer affordability and unobtrusiveness, their limitations in producing data suitable for source apportionment models pose challenges (Higgins, Kumar, & Morawska, 2024). Furthermore, Tagle et al. (2020) demonstrate moderate inter-unit variability in low-cost PM sensors, emphasizing the need for robust calibration methodologies. These findings underscore the importance of integrating calibration routines into digital twin frameworks to enhance the reliability of IAQ classifications.

IV. Machine Learning Models for Air Quality Prediction

Advanced machine learning models play a critical role in air quality prediction and classification, enabling digital twins to forecast pollutant concentrations and identify sources of pollution. TAOYING et al. (2020) [19] propose a hybrid CNN-LSTM model for predicting PM2.5 concentrations, leveraging convolutional neural networks (CNNs) for feature extraction and long short-term memory (LSTM) networks for capturing temporal dependencies. Their results indicate superior performance compared to standalone LSTM models, with lower mean absolute error (MAE) and root mean square error (RMSE). Similarly, Xiao et al. (2020) [20] introduce a weighted LSTM extended model (WLSTME) that accounts for spatiotemporal correlations influenced by site density and wind conditions. Both studies highlight the potential of deep learning models to support real-time IAQ classification in digital twins. Toharudin et al. (2023) [21] address the challenge of unbalanced PM2.5 concentration datasets using boosting algorithms such as AdaBoost, XGBoost, CatBoost, and LightGBM. Their approach significantly reduces bias and variance, improving classification accuracy for different PM2.5 levels. For digital twins, such techniques can enable more granular and accurate IAQ categorization, facilitating proactive measures to mitigate pollution exposure.

V. Integration with Digital Twins

Digital twins in smart homes require seamless integration of sensor data, predictive models, and user interfaces to provide actionable insights. The work of Castellani et al. (2021) [15] emphasizes the importance of energy-efficient designs, with 72.5% of reviewed systems claiming energy efficiency as a key feature. Energy efficiency is particularly relevant for digital twins, as continuous data acquisition and processing demand significant computational resources. Additionally, De Vito et al. (2024) advocate for open datalakes to support repeatability and further research, which aligns with the principles of digital twin development, where data transparency and interoperability are paramount. Chen et al. (2023) [22] propose a CNN-RF ensemble framework for PM2.5 concentration modeling, demonstrating improvements in root mean square error (RMSE) and mean absolute error (MAE) compared to standalone CNN and random forest (RF) models. This hybrid approach could be adapted for digital twins, enabling accurate and reliable IAQ classification across diverse microenvironments within smart homes.

VI. Challenges and Future Directions

Despite significant progress, several challenges remain. Higgins et al. (2024) [17] highlight the lack of IAQ data from non-residential and non-educational microenvironments, particularly in regions outside Europe and North America. This geographic bias limits the generalizability of IAQ classification models for global smart home applications. Furthermore, the heterogeneity of indoor environments, as noted by Higgins et al. [17], [18] necessitates careful consideration of sensor placement, occupancy patterns, and building characteristics.

Future research should focus on developing standardized calibration protocols for low-cost sensors and exploring novel AI-driven approaches to address unbalanced datasets and spatial variability. Additionally, the integration of external pollution data and environmental conditions into digital twin frameworks could enhance their ability to differentiate between indoor and outdoor pollution sources. Despite the significant contributions in air quality monitoring, there is a noticeable absence of a comprehensive, side-by-side comparison of these diverse methods on a consistent data setting. Prior research tends to emphasize single model types or particular subsets of machine learning algorithms for specific tasks and datasets, limiting the generalization across different environments. A gap exists in the current knowledge as there is less comparative analysis of several models trained on the same dataset. This analysis will enable the identification of the best suited model for air quality assessment. This study, using a real dataset, aims to address this gap by performing a comprehensive, comparative analysis using a diverse set of models from each of the aforementioned types, and explore their effectiveness when applied to a standardized dataset.

VII. Methodology of Air Quality Classification

A. Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

VIII. Air Quality Classification System Architecture

Fig. 1 shows the architecture of the Air Quality Classification System (AQCS) which consists of the following modules:

- Data Acquisition: The system uses CCS811 sensor data acquired by the Arduino microcontroller. CO2 and TVOC levels from a sensor are the input. These readings are labelled using predefined ranges for the different air quality categories.
- Preprocessing Stage: Data scaling is used to standardize the input features. Label encoding is used for categorical data and one hot encoding of those labels. Reshaping of input features is done as necessary for each model.
- Model Training and Prediction: 8 models are trained with respective hyperparameters. Predictions and probability scores are produced from those models.
- Performance Evaluation: Each model is evaluated using metrics, such as accuracy, precision, recall, f1-score, AUC, log loss and confusion matrices.

IX. Methodology of Air Quality Classification

The CO2 and TVOC values are acquired from CCS811 sensor and classified into air quality categories (Excellent, Good, Moderate, Poor, Unhealthy, Hazardous). Table I summarizes the TVOC and CO2 ranges for each category [15]. Model architectures and training will be explained in the next sections.

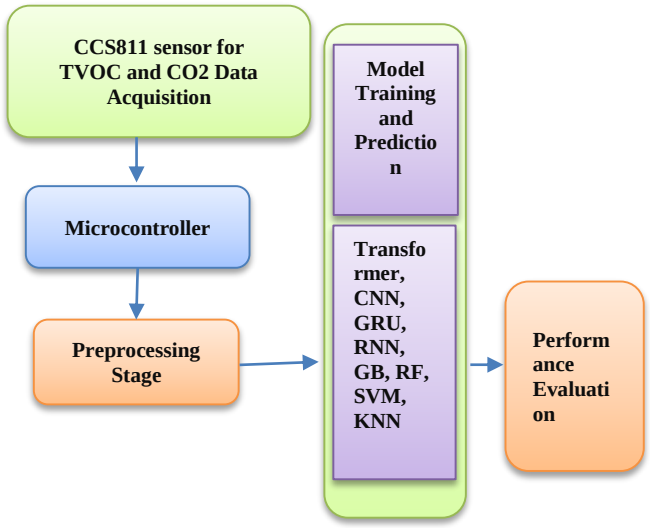


Fig. 1. Air quality classification system architecture.

Pollutant Categories (PCs) (based on ranges) and the Air Quality Index (AQI) are both tools that simplify complex data. They make air pollution information more understandable, accessible, and actionable for both the public and policymakers. They help estimate the potential health impacts of air pollution based on concentrations. They also enable informed decision-making by communicating the health risks associated with different levels of air pollution, empowering individuals to take steps to protect themselves and their families. They support policy and management through informing the development of regulations, tracking progress, and enabling effective air pollution control strategies. Table II and Table III show the TVOC and CO2 concentration ranges by air quality category.

TABLE I TVOC and CO2 Ranges for each Category

Category	CO2 Range (ppm)	TVOC Range ($\mu\mu_g/m^3$)
Excellent	200 - 400	10 - 50
Good	401 - 700	51 - 100
Moderate	701 - 1000	101 - 200
Poor	1001 - 1500	201 - 400
Unhealthy	1501 - 2000	401 - 600
Hazardous	2001 - 3000	601 - 1000

TABLE II TVOC CONCENTRATION RANGES BY AIR QUALITY CATEGORY

Rang	Category	Caution
10 - 50	Excellent	very clean environment
51 - 100	Good	Low TVOC levels
101 - 200	Moderate	Moderate levels, some sources may be present
201 - 400	Poor	Potentially concerning, increased ventilation needed
401 - 600	Unhealthy	Significant source, ventilation likely needed
601 - 1000	Hazardous	High exposure, take action to reduce levels

TABLE III CO2 CONCENTRATION RANGES BY AIR QUALITY CATEGORY

Rang	Category	Caution
200 - 400	Excellent	Optimal air quality
401 - 700	Good	Acceptable air quality
701 - 1000	Moderate	Some ventilation may be required
1001 - 1500	Poor	Poor ventilation, possible discomfort
1501 - 2000	Unhealthy	Reduce ventilation, discomfort likely
2001 - 3000	Hazardous	Severely hazardous, immediate ventilation needed

A. Air Quality Index Calculation

EPA's formula for calculation of the AQI according to Eq. (1) and based on the Breakpoint Table for constants [15]. For each pollutant P, the sensor gives a concentration reading CP. This reading is typically an average over some period of time. The index for that pollutant is given by the following Eq. (1):

$$I_P = \left({I_{high} - I_{low}}/{C_{high} - C_{low}}\right) * (C_P - C_{low}) + I_{low} \quad (1)$$

Where:

- CP: The concentration of pollutant P.
- C_{low}, C_{high}: The low/high concentration breakpoints that contain CP. These breakpoints are defined by the EPA in the Breakpoint Table (below).
- I_{low}, I_{high}: The low/high index range associated with concentration breakpoints for CP.

Having calculated the index for each pollutant, the AQI is simply the maximum index across all pollutants.

In this paper, the Air Quality Index (AQI) for TVOC and CO2 is designed to communicate the quality of indoor air based on the combined levels of Total Volatile Organic Compounds (TVOC) and Carbon Dioxide (CO2). Unlike the standard AQI based on criteria pollutants, this index focuses on common indoor pollutants and provides a practical metric for indoor environmental management. This customized approach seeks to translate the combined levels of TVOC and CO2 into an easily understandable metric, providing guidance on the quality of the indoor air.

X. DATASET ACQUISITION AND CHARACTERISTICS

A. Data Acquisition

- Sample Count: The dataset consists of 1500 individual data points, each representing a single measurement of CO2 and TVOC levels. Data is separated into training-, testing- and validation datasets.
- Temporal Resolution: The data was collected at 2.17 samples/second.
- Environmental Conditions: Samples were acquired during a wide array of environmental conditions including high and low temp, wind speed conditions, humidity. Environmental values were not used in this study to focus on Co2 and TVOC. Using a Nano 33 BLE sense microcontroller we can further study the impact of both Temperature and Humidity on the measurement of the CO2 and TVOC concentrations.
- Sensor Calibration: The CCS811 sensor is manufacturer calibrated. We were keen to operate the sensor for long periods before use to maintain data integrity.

B. Dataset Characteristics

- Class Distribution: The distribution of 500 test samples across the six AQI categories of the test set is shown in Table IV.

TABLE IV TESTSET SAMPLES DISTRIBUTION

Class	Number of samples	Percentage
Excellent	71	0.142
Good	81	0.162
Hazardous	80	0.160
Moderate	102	0.204
Poor	91	0.182
Unhealthy Total	75	0.150

- Class Balance: The dataset exhibits very low-class imbalance, with the 'Good' and 'Moderate' categories being more represented than the 'Excellent' and 'Unhealthy' categories. This imbalance is due to the limited occurrence of 'Excellent' and 'Unhealthy' conditions in real-world data. We could have addressed this imbalance by using the common techniques, e.g., oversampling, under sampling, or cost-sensitive learning.

C. Potential Biases and Limitations

- Real-world vs. Lab-Controlled Conditions: The data was collected under real-world environmental conditions. The variations in environmental conditions (e.g., temperature and humidity fluctuations) may affect sensor readings and represent a potential source of bias. However, we feel that using real-world data provides greater ecological validity of the derived model.
- Sensor Limitations: The CCS811 sensor has known limitations in terms of cross-sensitivity to different VOCs and potential drift over time. While we used frequent cross-validate the obtained data, these limitations are acknowledged, and future studies will explore integrating the use of more reliable and accurate sensors, such as electrochemical sensors.

XI. Environmental Data Points are Omitted

This study is specifically for evaluating TVOC and CO2 readings. Data points gathered from other environmental aspects were not considered in this research. A second real-world dataset of sensor readings was acquired from a CCS811 sensor under varying environmental conditions and exposures. The dataset, contains 2363 samples collected over several hours. The sensor was exposed to smoke, sanitizer with 70% alcohol, and Adidas perfume. The dataset includes columns representing CO2 concentration (in ppm) and TVOC concentration (in ppb). The high correlation in the shape of the CO2 and TVOC concentration curves in your CCS811 sensor data is a common and interesting observation. Here's an interpretation of this phenomenon, considering the sensor's characteristics and the environmental context:

A. Interpretation

The strong correlation between CO2 and TVOC concentrations likely stems from a combination of factors:

1) CCS811 sensor operation: The CCS811 is primarily a metal-oxide gas sensor. It measures the change in resistance of a metal oxide layer when exposed to various gases. While designed to estimate CO2 and TVOC levels, the underlying sensing mechanism is not perfectly selective for each gas individually. In other words, there's some cross-sensitivity. The sensor might respond to changes in the overall composition of VOCs, and this change in VOC composition often occurs alongside changes in CO2. The sensor's algorithm tries to separate CO2 and TVOC signals, but the underlying measurements are still correlated.

2) Common sources: Many real-world sources emit both CO2 and VOCs simultaneously.

3) Human activity: Human respiration releases CO2. At the same time, activities like using cleaning products, cooking, and personal care products (perfume, deodorant, etc.) release VOCs. In an indoor environment, where these activities occur together, you'd expect CO2 and TVOC levels to rise and fall in tandem.

- Combustion: Smoke, as mentioned, is a product of combustion. Combustion processes produce both CO2 and a wide range of VOCs. Therefore, smoke exposure would naturally lead to a correlated increase in both signals.
- Sanitizers: Alcohol-based sanitizers release alcohol vapors (which are VOCs). While the alcohol itself might not directly produce CO2, the presence of a sanitizer often correlates with human activity (cleaning, etc.) that does produce CO2.

4) Ventilation: Ventilation patterns can influence both CO2 and VOC concentrations in a similar way. If ventilation is poor, both CO2 and VOCs will build up. If ventilation is good, both will be diluted and removed. This shared influence of ventilation reinforces the correlation between the two signals.

5) Environmental context: The specific environmental conditions during data acquisition play a crucial role. If the sensor was in a relatively closed environment with limited air exchange and exposed to activities that generate both CO2 and VOCs, the correlation would be more pronounced.

B. Implications for Analysis

- Distinguish Sources: The correlation makes it more challenging to distinguish the specific sources of pollutants. For example, it might be difficult to definitively say that a CO2 peak is solely due to human respiration versus a combination of respiration and a nearby VOC source.
- Calibration: the sensor's calibration and the algorithm's accuracy can be affected by the cross-sensitivity and the inherent correlation between CO2 and VOCs.
- Multi-Sensor Fusion: To improve the accuracy of individual CO2 and TVOC measurements, we might consider combining the CCS811 with other sensors that are more selective for specific gases (e.g., a non-dispersive infrared (NDIR) CO2 sensor).
- Data Interpretation: When interpreting the data, we avoided drawing overly specific conclusions based solely on the CO2 and TVOC readings. Consider the context of the measurements and the limitations of the sensor.

In summary, the high correlation between CO2 and TVOC levels in our CCS811 data is a result of the sensor's operating principles, the co-occurrence of CO2 and VOC sources in the real world, and the influence of factors like ventilation. It's important to understand these factors to interpret the data accurately and avoid oversimplification.

Fig. 2 shows a sample of the data used for estimation of the indoor air quality index and a sudden variation during intended exposure of the sensor to a TVOC. The calculation begins by assessing each pollutant separately. A sub-index is generated for both TVOC and CO2 concentrations using a piecewise linear interpolation approach and user defined breakpoints. The measured TVOC concentration is compared to predefined levels, which are based on guidance from different scientific studies and building standards.

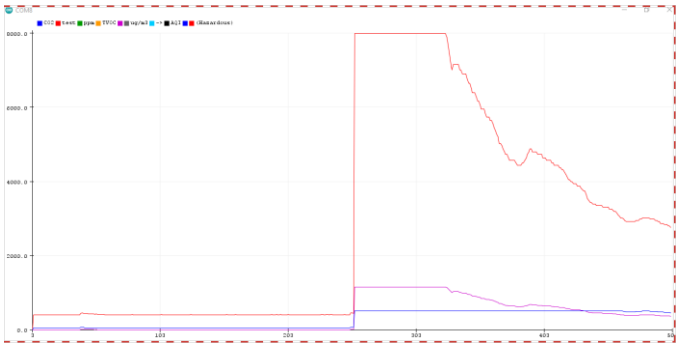


Fig. 2. Sample TVOC and CO2 signal change over time when exposed to a TVOC.

The measured CO2 concentration is also compared to its predefined levels and translated to a sub-index. The levels are based on standards recommendations for CO2 levels in indoor spaces. The individual sub-indices for TVOC and CO2 are then

combined to generate a single, overall AQI value. In the previous codes, the combining of the individual sub-indices has been done by taking the higher index between the two. This approach helps to quickly communicate to the user the worst-case scenario for the combined TVOC and CO2 readings. While the example uses a maximum, other methods for combining can include averaging or weighting. The AQI for TVOC and CO2 provides a way to understand the status of your indoor air based on common indicators of indoor air quality using EPA like methods but without the standard EPA's breakpoints and requirements for the pollutants.

C. Classifier Models Architecture

Detailed architectures of each of the 8 models implemented includes the key components, layers, and configurations for each as follows:

1) Transformer neural network model: The Transformer model is a deep learning model based on the attention mechanism as shown in Fig. 3. While it is primarily for sequence-to-sequence tasks, it is configured here for sequence classification.

a) Transformer model architecture: The transformer architecture implemented in this paper leverages a series of custom layers to process input data, ultimately classifying it into predefined air quality categories. The architecture begins with an Input Embedding layer, responsible for mapping the input features (CO2 and TVOC levels) into a higher-dimensional embedding space. This is followed by Positional Encoding, a crucial step that introduces information about the relative positions of the input sequence, which in this case consists of a single time step representing one set of feature values. The core of the transformer encoder is encapsulated within the Encoder Layer, which first applies multi-head self-attention using the Multi Head Attention layer, allowing the model to weigh the importance of different features within the input. Then a feed-forward network is used which further refines the transformed representation. Layer normalization and dropout are applied to both outputs to stabilize the training, mitigate overfitting and ensure the layer outputs are in a consistent and stable range for easier training. These components work in tandem to extract relevant patterns and relationships from the input data. The output is then processed by the model using a global average pooling layer before the classification layers. The transformer model is built using the class Transformer Classifier, which encapsulates all previously mentioned layers as part of the model architecture and defines the forward propagation through these layers via the call method. The final classification is performed using the Output Layer, which uses a fully connected dense layer with a softmax activation, providing a probability distribution across the different air quality categories. The model includes a custom train_step method to train the model by using the functional call, which is used for inference. The get_config method is also implemented for all custom layers to ensure that model can be easily saved and loaded in the future. Finally, the model is compiled with the ADAM optimizer, categorical cross entropy as loss function and accuracy as metric, and it is trained using the reshaped data to feed into the network and subsequently it is used to predict the labels on test set and generate classification reports, ROC curves, and confusion matrices.

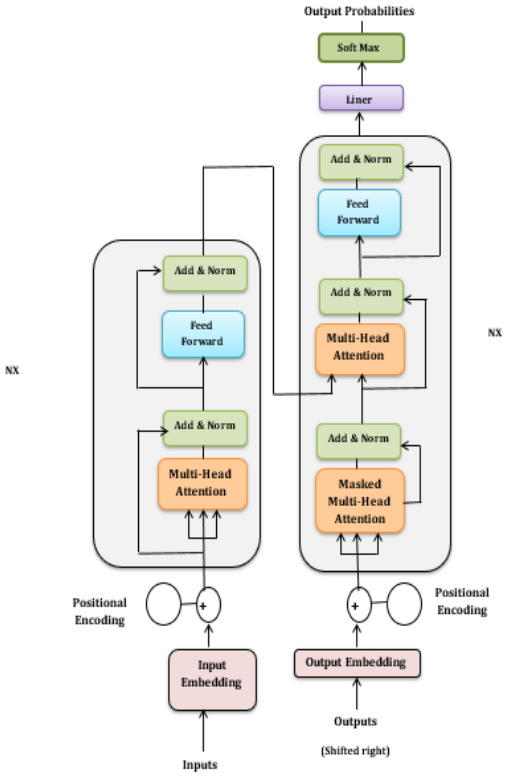


Fig. 3. Architecture of the TNN model.

TABLE V TRANSFORMER MODEL'S STRUCTURE AND COMPLEXITY

Layer/Component	Type	Input Shape	Output Shape	Parameters	Activation
Input Embedding	Input Embedding Layer	(None, 2)	(None, 64)	192	Linear
Positional Encoding	Positional Encoding Layer	(None, 64)	(None, 64)	0	Linear
Encoder Layer (x1)	Encoder Layer	(None, 64)	(None, 64)	57,344	Various
Multi-Head Attention	Multi Head Attention Layer	(None, 64)	(None, 64)	49,408	Softmax
Query, Key, Value Dense	Dense Layers	(None, 64)	(None, 64)	12,288 x3	Linear
Output Dense	Dense Layer	(None, 64)	(None, 64)	4160	Linear
Feed Forward Network (FFN)	Feed Forward Network Layer	(None, 64)	(None, 64)	7,808	ReLU+Linear
Dense 1	Dense Layer	(None, 64)	(None, 128)	8,320	ReLU
Dense 2	Dense Layer	(None, 128)	(None, 64)	8,256	Linear

Layer Normalization	Layer Normalization Layer	(None, 64)	(None, 64)	128	Linear
Dropout	Dropout Layer	(None, 64)	(None, 64)	0	None
Global Average Pooling	GlobalAveragePooling1D Layer	(None, 1, 64)	(None, 64)	0	None
Output Layer	Output Layer	(None, 64)	(None, 6)	390	Softmax
Output Dense	Dense Layer	(None, 64)	(None, 6)	390	Softmax
Total Trainable Parameters				57,926	

In the context of machine learning models, "Parameters" refer to the internal settings within the model that are adjusted during the training process, enabling the model to learn patterns and make accurate predictions. "Tunable Parameters," on the other hand, are the hyperparameters that a user typically adjusts externally to optimize the model's performance, influencing how the model learns. "Default Values" indicate the specific parameter values that are used in the provided code when no specific settings are explicitly made, offering a baseline configuration for the models. The provided notes further explain how these parameters affect the behavior and performance of each model, providing essential insight for effective use. This information is designed to support tasks such as hyperparameter tuning, which focuses on adjusting the tunable parameters to achieve better results; model understanding, which provides an overview of how different model architectures are configured; comparison, which facilitates the comparison of models based on their settings; and model selection, which informs the choice of model appropriate for specific task. It is important to recognize that these parameters often have interdependencies; therefore, optimizing one parameter may change the optimal settings for another. Furthermore, the best values are highly dataset-dependent, meaning that different datasets might benefit from different configurations. Grid search and random search are popular tuning techniques that are employed alongside a good understanding of the parameter behavior in order to achieve optimal results. While default parameter values offer a good starting point, these parameters can often be improved using careful hyperparameter tuning to improve a models generalization ability and achieve a higher level of performance. Table V shows the key hyper parameters for several common machine learning models. For transformer models, embed_dim dictates the size of input embeddings, while num_heads specify the number of attention heads, and ff_dim defines the hidden layer size in the feedforward network. A rate parameter allows for dropout implementation to combat overfitting. It's worth noting that maxlen is a fixed, non-tunable parameter in the given implementation. For convolutional neural networks (CNNs), the number of filters and kernel_size in the Conv1D layers are crucial, with flexibility to add various activation functions and layers. In recurrent neural networks (RNNs), specifically GRU layers, units represent the number of hidden units and activation sets the activation function. A dropout rate controls regularization and the number of layers is another potential hyperparameter. Similarly, for simple RNNs, units, activation, dropout and the number of layers is all tunable. For Support Vector Machines (SVMs), C is a regularization parameter, kernel defines the kernel type (like RBF, linear, or polynomial), gamma is a kernel coefficient (often scaled by default), and degree is specific to the polynomial kernel. In the realm of tree-based models, random forests include n_estimators, the number of trees, max_depth, the maximum tree depth, and min_samples_split, defining the minimum samples required to split a node. Numerous other parameters are also tunable. Gradient boosting models, such as Gradient Boosted Decision Trees (GBDT), share parameters like n_estimators and max_depth and adds learning_rate which scales each tree's contribution and loss sets the loss function. Finally, in K-Nearest Neighbors (KNN), n_neighbors determine the number of neighbors to consider, weights specify how neighbors are weighted and algorithm selects the method for calculating neighbor distance.

2) Convolutional Neural Network (CNN): The Convolutional Neural Network (CNN) implemented in this program serves as a deep learning model designed for image and sequential data processing, utilizing convolutional operations to extract relevant features. In this context, its purpose is to classify air quality based on sequential patterns derived from the input data. The network begins with an input layer that accepts two features, which are subsequently reshaped to have dimensions (2, 1), making them compatible with the convolutional operation. The core of the CNN comprises two 1D convolutional layers (Conv1D). The first layer employs 64 filters with a kernel size of 3 and a ReLU activation, while the second layer has 128 filters with the same kernel size and ReLU activation function. These layers apply the convolutions to the reshaped input, thereby extracting feature maps that highlight relevant patterns within the data. A Flatten layer then transforms the 2D feature maps into a 1D feature vector, preparing the output for fully connected layers. The flattened output is then passed through a fully connected dense layer with a ReLU activation, followed by a Dropout layer to mitigate overfitting by randomly disabling a percentage of the connections during training. Finally, an output layer, implemented as a dense layer with a softmax activation, produces a probability distribution across the six different air quality categories. The key parameters defining this network include the number of filters in each convolutional layer, which is set to [64, 128], the kernel size set to 3, and ReLU used as the activation function. Thus, the CNN serves to classify air quality by analyzing the spatial representation of the input features.

3) Gated Recurrent Unit (GRU): The Gated Recurrent Unit (GRU) is implemented as a recurrent neural network designed for sequence processing and classification, employing gates to manage information flow. The GRU model starts with an input layer that takes two features, CO2 and TVOC, which are then reshaped to represent a single time step with these two features. Following this, a single GRU layer with 64 units is used to capture any sequential relationships within the data. To reduce overfitting, a dropout layer is then applied to the GRU output. This is followed by a dense layer with a ReLU activation to learn from the GRU outputs, and another dropout layer for

regularization. Finally, an output layer, implemented as a dense layer with softmax activation, generates the classification probability across the six air quality categories. The key parameters of this GRU model include 64 units in the GRU layer, ReLU as the activation function and a dropout rate of 0.5. The purpose of the GRU model within this program is for the time-based classification of air quality, leveraging the model's ability to capture any sequential information within the data.

4) Recurrent Neural Network (RNN): A Recurrent Neural Network (RNN) is employed as another type of recurrent neural network aimed at sequence processing and classification using recurrent connections. The RNN model has an input layer that takes two features, CO2 and TVOC, at each time step. It reshapes the input to have one time step. A single Simple RNN layer with 64 units is then used to capture sequential information. To mitigate overfitting, a dropout layer is applied after the RNN layer. The output from the RNN is fed into a fully connected layer with a ReLU activation function, and another dropout layer for regularization. Finally, the output layer with a softmax activation generates the classification probability for each of the six air quality classes. The key parameters for this RNN model include 64 units in the RNN layer, ReLU as the activation function for the RNN units and a dropout rate of 0.5. The purpose of the RNN in this program is for time-based classification, leveraging its ability to capture any sequence information in the input data.

5) Support Vector Machine (SVM): The Support Vector Machine (SVM) is a supervised learning model used for making predictions based on decision boundaries. It takes scaled 2-dimensional features (CO2, TVOC) as input and uses a Radial Basis Function (RBF) kernel to create decision boundaries. A Calibrated Classifier CV is used to apply cross-validation calibration using isotonic regression, ensuring output probabilities are well-calibrated and reliable. The key parameters for this SVM model include a regularization parameter 'C' set to 1.0, 'rbf' as the kernel type, and 'scale' as the gamma coefficient for the kernel and isotonic as the probability calibration method. The SVM aims to classify air quality based on identifying complex decision boundaries in the feature space.

6) Random Forest (RF): The Random Forest model uses an ensemble learning method for classifying the air quality data. This model constructs multiple decision trees based on random samples of the features and data points, creating a robust classifier that is less prone to overfitting. The Random Forest model takes scaled 2-dimensional features (CO2, TVOC) as input and constructs an ensemble of 100 decision trees. The final classification is then based on the average predictions across all of the trees. The number of estimators is set to 100, and a random_state of 42 ensures reproducibility. The main purpose of the Random Forest model within the program is the classification of air quality by using the combined knowledge of multiple decision trees.

7) Gradient Boosting (GB): Gradient Boosting is another ensemble learning method that classifies by training weak learners in a stage-wise fashion, where each subsequent tree minimizes the loss incurred by the preceding tree. This model also takes scaled 2-dimensional features (CO2, TVOC) as input and constructs an ensemble of decision trees, but unlike the random forest model, the trees are added sequentially with each subsequent tree minimizing the error from past predictions. Key parameters for this Gradient Boosting model include 100 boosting stages, a learning rate of 0.1, a maximum depth of 3 for individual trees, a random state of 42, and a 'log_loss' function that is optimized by the trees. In the program, the purpose of the Gradient Boosting model is to classify air quality by sequentially training multiple models, reducing the error of prediction in each iteration.

8) K-Nearest Neighbors (KNN): The K-Nearest Neighbors (KNN) model is an instance-based learning method that classifies data based on the majority of its neighbors. This model takes the scaled 2-dimensional features (CO2, TVOC) and classifies each data point based on the label of the n_neighbors number of closest samples, using Euclidean distance to determine closeness. The key parameters for the KNN model include n_neighbors (default value set to 5), uniform as the weighting function, and 'auto' as the algorithm used to compute the nearest neighbors. The main purpose of the KNN model is to classify the air quality based on the category of the closest datapoints from the training data.

D. Key Parameters of the Classifier Models

Table VI summarizes the key parameters of the 8 models. The parameters that are most likely to be tuned or of interest when using these models are summarized in Table III.

TABLE VI MODEL PARAMETERS

Model	Key Tunable Parameters	Default Values
TNN	embed_dim, num_heads, ff_dim, rate (dropout)	embed_dim=32, num_heads=2, ff_dim=32, rate=0.1
CNN	filters (Conv1D), kernel_size (Conv1D), activation	filters=[64, 128], kernel_size=3, activation='relu'
GRU	units (GRU), activation, dropout	units=64, activation='relu', dropout=0.5
RNN	units (SimpleRNN), activation	units=64, activation='relu', dropout=0.5
SVM	C, kernel, gamma (RBF), degree (Polynomial)	C=1.0, kernel='rbf', gamma='scale', degree=3
RF	n_estimators, max_depth, min_samples	n_estimators=100, max_depth=None, min_samples_split=2
GB	n_estimators, learning_rate, max_depth	n_estimators=100, learning_rate=0.1, max_depth=3, loss='log_loss'
KNN	n_neighbors, weights, algorithm	n_neighbors=5, weights='uniform', algorithm='auto'

Table V gives a detailed view of the Transformer model's structure and complexity. The computational complexity of the Transformer model described is primarily influenced by the multi-head attention mechanism and the feed-forward networks within the encoder layers. The multi-head attention has a time complexity of approximately O (n^2 * d), where 'n' is the sequence length and 'd' is the embedding dimension. However, in this specific implementation, the sequence length is fixed at 1, therefore, the attention mechanism's computational complexity is closer to O(d), where d represents the embedding

dimension. The feed-forward networks have a complexity of O (d * f), where 'f' is the hidden layer size in the FFN. Since the Global Average Pooling, Output and Normalization layers have a relatively smaller time complexity, the overall complexity of this particular Transformer architecture with a sequence length of 1, can be approximated by O(d * f + d), where d is the embedding dimension and f is the feed forward dimension, indicating that complexity scales linearly with the embedding dimension and FFN dimension. Additionally, the dropout layers do not affect the overall time complexity of the model.

E. Algorithm of Air Quality Model Comparison and Evaluation

1. Initialization:
   - Define air quality categories (Excellent, Good, Moderate, Poor, Unhealthy, Hazardous).
   - Define functions to categorize air quality based on CO2 and TVOC levels.
   - Define a function to upload a CSV data file.
2. Data Generation:
   - Generate a training dataset with a specified number of samples for each air quality category.
   - Generate a test dataset similarly.
   - Save both the training and test datasets to separate CSV files.
3. Data Loading and Preprocessing:
   - Load the training and test datasets from the CSV files into pandas Data Frames.
   - Extract the CO2 and TVOC features as input (X) and the air quality categories as the target (y).
   - Scale the input features using Standard Scale.
   - Encode the target labels using Label Encoder.
   - Reshape/prepare the input data as required for each model type (e.g., for CNNs, transformers).
   - Convert categorical labels into a one-hot encoded format.
4. Model Training and Evaluation:
   - Define, initialize and create instances of each of the 8 model types (TNN, CNN, GRU, RNN, SVM, RF, GB and KNN).
   - For each model:
     - Train the model using the preprocessed training data (and use cross-validation or grid search for hyperparameter tuning).
     - Predict on the test data to generate predictions and probabilities
     - Evaluate the model using the actual test data and the model predictions using performance measures (accuracy, precision, recall, f1-score, ROC-AUC, log loss and confusion matrix).
     - Store performance metrics, including accuracy, classification report, and any relevant data for later analysis.
   - Plot relevant training and evaluation metrics (loss curves, confusion matrix, ROC curves).
5. Summary and Output:
   - Collect and store the results of all 8 model types into a suitable data structure (e.g., dictionary).
   - Present a summary table using pandas, displaying:
     - The name of each model.
     - The accuracy obtained from each model.
     - Classification report string with the performance metrics.
     - Additional info like ROC, log loss and loss curves for applicable models.
6. Print summary table:
   - Print the performance summary table to the console.

XII. System Performance Evaluation Metrics

There exists a variety of measures for judging the performance. In our research, we have considered the following four performance measures as discussed in detail in the literature [11, 13]:

$$\text{Precision} = \frac{TP}{(TP+FP)} \quad (2)$$

$$\text{Recall (Sensitivity)} = \frac{TP}{(TP+FN)} \quad (3)$$

$$\text{Specificity} = \frac{TN}{(TN+FA)} \quad (4)$$

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (5)$$

All of the above quantities are normally expressed as percentages. The various terms appearing in the above equations are: True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN).

Sokolova et al. [12] have shown that the accuracy measure does not distinguish between the numbers of correct labels of different classes. Sensitivity and specificity separately estimate a classifier's performance on different classes. It has been shown that higher accuracy does not guarantee overall better performance of an algorithm and that a combination of measures gives a balanced evaluation of the algorithm's performance. In this paper, we have used the Youden index and F-measure to evaluate the performance of our system:

$$\text{Youden Index} = \text{Sensitivity} - (1 - \text{Specificity}) \quad (6)$$

$$F\beta = (1 + \beta^2) * (\text{Precision} * \text{Recall}) / (\beta^2 * \text{Precision} + \text{Recall}) \quad (7)$$

where β is a weighting constant that evenly balances the F-score when β=1, favors precision when β> 1, and recall otherwise. The Youden index evaluates the classifiers performance to a finer degree with respect to both classes. Youden Index: Balances sensitivity and specificity, providing a single measure of overall test performance. It ranges from -1 to +1, with higher values indicating better performance. F-Measure (F-score): Balances precision and recall, particularly when there is a trade-off between correctly predicting positives and capturing all actual positives. It also ranges from 0 to 1, with higher values indicating better performance. Seven performance metrics [11-14] are used to evaluate performance of the AQCS. The Sensitivity metric measures the rate of positive cases. The Specificity metric measures the proportion of positive cases that are correctly identified. The Accuracy represents the population of the correctly predicted examples, which is not an appropriate evaluation criterion in imbalanced data sets, and we will not put on much attention to it. The F- value combines the Precision and Recall and gets a higher value when both of Precision and Recall are high. F-SCORE is the harmonic mean of precision and sensitivity. For each class the ROC-AUC curves are given in addition to the Confusion Matrix.

Log loss, also known as cross-entropy loss or logistic loss, is a metric used to evaluate the performance of classification models, particularly those that output probabilities (like logistic regression, neural networks with softmax output, etc.). Unlike accuracy, which only looks at whether the predictions are correct or not, log loss focuses on the probabilities associated with the predictions, penalizing models that are confident but wrong more heavily. For multi-class problems (where there are more than two classes), log loss generalizes to:

$$\text{Log Loss} = -(1 / N) * \Sigma \Sigma [y\_\{ij\} * \log(p\_\{ij\})] \quad (8)$$

Where:

- y is the actual class label (either 0 or 1).
- p_{ij} is the probability predicted by the model that the sample i belongs to class j.
- log is the natural logarithm.
- N is the number of samples

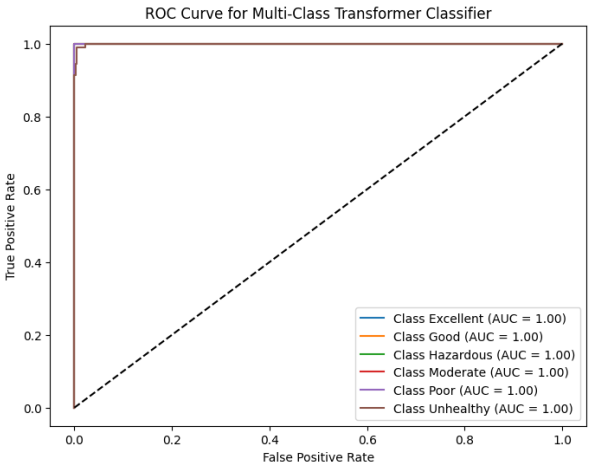
To obtain a single loss value, we need to average the loss across all of the N samples that we have in the dataset. log loss is a valuable metric for classification models that produce probabilities. It penalizes confident incorrect predictions and provides a more nuanced understanding of model performance beyond accuracy alone.

XIII. Results and Discussion

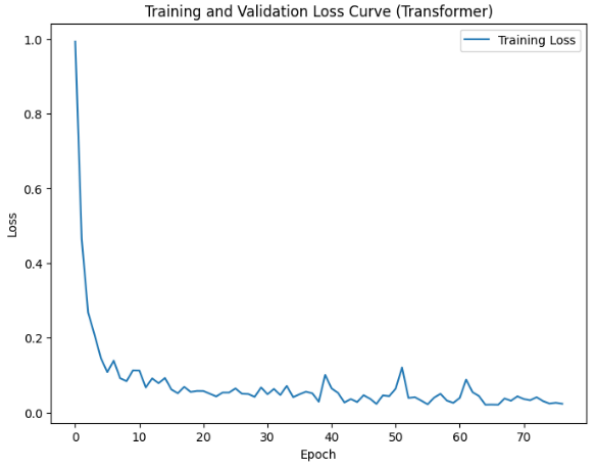
Table VII summarizes the performance of the 8 implemented models. The performance of each implemented model is carefully evaluated and demonstrate robust performance. To gain a deeper understanding of the model's capabilities, metrics such as precision, recall, and F1-score from the classification reports are examined. This analysis provides insights into each model's ability to correctly classify each category while highlighting any biases. ROC curves and confusion matrices are further analyzed to evaluate each model's performance and to explain why each model behaves the way it does, including how well each model is able to identify different classes and if they make any systematic errors. Additionally, for the deep learning models (CNN, RNN, GRU, and Transformer), the training and validation loss curves are studied to evaluate their learning behavior over time and to assess how well the models were able to learn the patterns in the data set.

TABLE VII Performance Metrics for Stratified Kfold Cross-validation

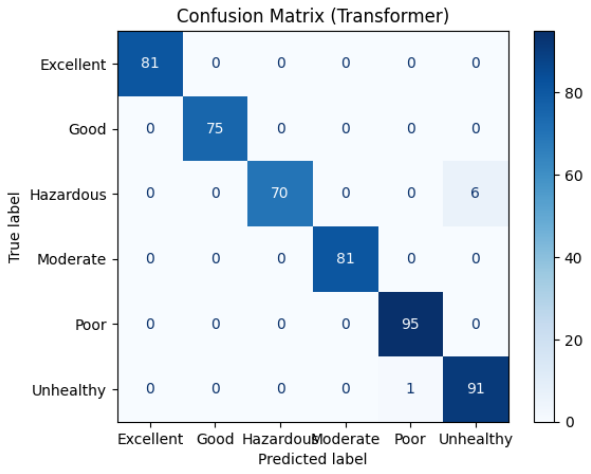
Model	Average Cross-Validation Accuracy	Test Accuracy	Precision	Recall	Specificity	Youden Index	Positive Likelihood	Negative Likelihood	Discriminant Power
TNN	0.999	0.986	0.986	0.986	0.997	0.983	352.14	0.014	5.58
CNN	0.999	0.998	0.998	0.998	0.999	0.997	2495.00	0.002	7.73
GRU	0.999	0.996	0.996	0.996	0.999	0.995	1245.00	0.004	6.97
RNN	1.000	0.998	0.998	0.998	0.999	0.997	2495.00	0.002	7.73
Bi-LSTM	0.999	0.996	0.996	0.996	0.999	0.995	1245.00	0.004	6.97
SVM	0.995	0.996	0.996	0.996	0.999	0.995	1245.00	0.004	6.97
RF	1.000	0.998	0.998	0.998	0.999	0.997	2495.00	0.002	7.738
GB	0.998	1.000	1.000	1.000	1.000	1.000	0.00000	0.000	0.000
KNN	0.998	1.000	1.000	1.000	1.000	1.000	0.00000	0.000	0.000



(a)



(b)



(c)

Fig. 4. (a) Transformer's ROC curve and (b) transformer's training and validation loss curve and (c) confusion matrix for the transformer model.

Support Vector Machine (SVM), Random Forest, Gradient Boosting, and KNN have a simplified loss curve, while neural network-based models like the Transformer, CNN, GRU, and RNN have a "traditional" loss curve. Neural networks use iterative training (gradient descent) with a loss function, allowing loss to be tracked and plotted over epochs. SVM, Random Forest, etc. Use non-iterative or different optimization methods without a typical loss curve per training epoch. These models have a single fit procedure, without per-epoch updates, and thus no intermediate steps to measure the loss in the same way as neural networks. The "simplified loss curve" plots their final accuracy as a proxy, not a real per-epoch training loss.

This analysis delves into why certain models perform better than others, moving beyond simple accuracy comparisons to explore the underlying reasons rooted in model architecture, data characteristics, and algorithmic approaches. It examines how each model's inherent biases or assumptions affect results. For instance, the effectiveness of tree-based models like Random Forest for this specific classification problem, which may not generalize well to other datasets, is explored. The success of K-Nearest Neighbors (KNN) is discussed in relation to the specific data patterns and relationships. Fig. 4 shows transformer's ROC curve, transformer's training and validation loss curve and confusion matrix for the transformer model. The analysis investigates why a linear Support Vector Machine (SVM) might struggle with non-linearly separable classes, unlike tree-based methods that can effectively handle such scenarios. Model complexity is also considered, acknowledging that deep learning models are more capable of capturing intricate relationships than models like KNN, which are built on simpler data assumptions. The specific parameters of the models and their influence on performance is also discussed, such as the successful performance of Gradient Boosting and its parameters.

The analysis further examines the effectiveness of Random Forest, KNN, and Gradient Boosting, which often achieve near-perfect accuracy. The role of randomness in the sampling and feature selection in the Random Forest is discussed, including the way it leads to generalizable decision boundaries, also how the averaging of predictions across many trees provides robust classification. The explanation of KNN covers its core concept that similar points fall within the same category based on Euclidean distance. Furthermore, it explains why this approach is effective on this dataset and how that effectiveness may not be valid in real world scenarios. The sequential error minimization in Gradient Boosting is discussed, including how gradient descent enables subsequent trees to learn in the direction of a more optimal solution.

The analysis also explains how the neural network models, specifically CNNs, RNNs/GRUs, and Transformers, learn from the data. It details how convolutional layers in CNNs extract spatial features or patterns by capturing local dependencies. It also discusses how the recurrent nature of RNNs/GRUs helps in learning temporal dependencies, especially how the gate

mechanism in GRUs facilitates handling of temporal data. The analysis then explains how self-attention mechanism of the transformer model works on this data set and how the dense layers of the transformer classifiers the data, also the role of the embedding and positional encoding layers in capturing relevant information. Speculations are made regarding what types of features and relationships the models might have learned from the data, including whether they correlate certain air quality categories more with CO2 or TVOC and whether the model's weights prioritize certain features or value ranges.

Finally, the practical trade-offs between model complexity and inference speed are discussed. This includes whether the deep learning models have a longer inference time due to their complexity compared to the much faster random forest or KNN models and what is their impact on real time systems, and when computational expense is a significant concern and when it can be tolerated. Table VIII compares the 8 models implemented. This table highlights each model's strengths, weaknesses, and typical applications. This table provides a comprehensive comparison of all 8 models, and allows us to make informed choices based on their strengths, weaknesses, and suitability.

TABLE VIII MODEL COMPARISON

Model	Strengths	Weaknesses	Applications
TNN	- Captures long-range dependencies well. - Highly parallelizable training.	Computationally expensive.	- Text classification, sentiment analysis, machine translation, image recognition, time-series.
CNN	- Excellent for spatial hierarchy processing. - Efficient in identifying local patterns.	Can be sensitive to translations/rotations.	- Image recognition, object detection, image segmentation, time-series analysis, audio processing.
GRU	- Captures sequential information effectively. - Handles long sequences better than basic RNNs due to gating mechanism.	Can be computationally intensive on long sequences.	- Natural language processing (NLP), time-series analysis, speech recognition, machine translation.
RNN	- Can capture temporal dependencies well. - Simple to implement	Difficult to train for long sequences, vanishing and exploding gradient.	- NLP, speech processing, time-series forecasting, machine translation.
SVM	- Effective in high-dimensional spaces. - Can model non-linear decision boundaries with RBF kernel.	Can be computationally intensive on large datasets.	- Image classification, text classification, bioinformatics, outlier detection.
RF	- Robust to outliers and non-linearities. - Good generalization performance.	Can be harder to interpret compared to single decision trees.	- Classification and regression tasks, feature importance ranking, medical diagnosis, financial modeling.
GB	- Achieves high predictive accuracy and flexible for different loss functions. - Effective for complex datasets.	Can be prone to overfitting with noisy data.	- Structured classification and regression tasks, ranking tasks, fraud detection, recommendation systems.
KNN	- Simple to implement and easy to understand. - No explicit training phase.	Computationally expensive in inference with large datasets.	- Classification and regression tasks, image recognition, recommendation systems, anomaly detection.

Deep learning models, such as Transformers, Convolutional Neural Networks (CNNs), Gated Recurrent Units (GRUs), and Recurrent Neural Networks (RNNs), are powerful tools that shine when dealing with complex data structures and requiring intricate feature learning. These models often excel at capturing nuanced patterns within data but come with a significant demand for computational resources, often requiring substantial processing power and training time. Conversely, classical machine learning models like Support Vector Machines (SVM), Random Forests, Gradient Boosting algorithms, and K-Nearest Neighbors (KNN) offer a different set of advantages. These models are generally faster to train and easier to interpret, making them suitable when speed and transparency are important considerations.

When dealing specifically with sequential data, such as time series or text, the strengths of certain deep learning models become particularly apparent. Transformers, GRUs, and RNNs are explicitly designed to process sequence data, allowing them to learn dependencies and temporal patterns that other models might miss. Ensemble methods, as exemplified by Random Forest and Gradient Boosting, offer another approach by combining the predictions of multiple learners. This technique enhances the robustness and overall performance of the models, often leading to more reliable results.

The computational cost associated with these different model types can vary greatly, especially when the size of the dataset changes. For instance, KNN has a very low training cost due to its simple algorithm, whereas complex neural networks can have high training times because they involve numerous iterations and parameter updates. This contrast highlights the importance of choosing a model that aligns with available resources and time constraints. Furthermore, interpretability is another important aspect to consider. Decision-tree based models like Random Forest and Gradient Boosting are often easier to interpret because their decision-making process can be traced through the tree structure.

In the context of air quality classification based on CO2 and TVOC levels, different models may be preferred based on the desired outcome. If the relationships between CO2/TVOC and the air quality category are exceptionally complex, deep learning models like Transformers, CNNs, GRUs, and RNNs can be well-suited. If, however, speed of deployment and inference is paramount, a simple model like KNN may be a better fit due to

its low computational overhead. In scenarios where accuracy is a top priority and deep learning is not required, ensemble methods such as Random Forest and Gradient Boosting can offer a good balance between performance and computational efficiency, potentially providing high accuracy without needing the complexity of very deep learning models.

XIV. Computational Complexities of the 8 Models

Table IX summarizes the computational complexities of the 8 models. It provides a breakdown of both time and space complexity, along with explanations.

TABLE IX Computational Complexity Summary

Model	Time Complexity (Training)	Time Complexity (Inference)	Space Complexity (Training)	Space Complexity (Inference)	Notes
TNN	O(N^2 * D + N * D^2)	O(N * D^2)	O(N * D)	O(N * D)	N = Sequence Length (Here always 1), D = Embedding dimension. Training Complexity is dominated by attention layers' N^2. The model's size mainly determines space complexity
CNN	O(C * K * M * N)	O(C * K * M * N)	O(P + CKM*N)	O(P + C * K * M)	C = Number of channels, K = Kernel size, M = Feature maps, N = Training data. Training time is influenced by Convolutional operation. Space complexity is driven by the number of parameters (P). Inference is a subset of training complexity.
GRU	O(N * H^2)	O(N * H^2)	O(N * H)	O(H)	N = Sequence Length (Here always 1) and H = Hidden units. Time complexity dominated by matrix multiplication during recurrent processing. Space complexity is for the number of parameters and hidden state size.
RNN	O(N * H^2)	O(N * H^2)	O(N * H)	O(H)	N = Sequence Length (Here always 1) and H = Hidden units. The time complexity of each sequence item processed is O(H^2), so is the space complexity O(H) per sequence. Space is for the weight and the hidden states.
SVM	O(N^2) to O(N^3)	O(N_sv * D)	O(N * D)	O(N_sv * D)	N is the number of training samples. D is the dimension of each data point. N_sv is the number of support vectors. Training complexity depends on kernel choice and optimization. Memory consumption related to the storing of all data and support vectors.
RF	O(T * M * log(N))	O(T * M)	O(T * M)	O(T * M)	T = Number of Trees, M = Number of features, N= Number of training data. Training time is determined by building each decision tree. Space is dominated by storing the trained trees.
GB	O(T * N * M)	O(T * M)	O(T * M)	O(T * M)	T = Number of trees, M = Number of features, N = Number of samples. Similar complexity to AdaBoost but might be slightly higher as it can be optimized by a loss function rather than simply weighing.
KNN	O(1)	O(N * M)	O(N * M)	O(1)	N = Number of training samples, M = Number of features. Training is very fast with KNN, its mostly a lookup. Inference complexity increases with dataset size.Space complexity is for storing entire dataset and no parameters.

where,

- O( ) - Big O Notation: Represents the upper bound of the growth rate of an algorithm's runtime or memory usage. It focuses on how the complexity scales with input size.
- N: Number of training samples, Sequence Length
- D: Embedding Dimensions, Feature Dimensions
- C: Number of channels in the convolutional layer
- K: Kernel size in the convolutional layer
- M: Number of feature maps, number of features in general
- H: Number of hidden units in the recurrent layers (GRU, RNN).
- T: Number of trees in ensemble methods (Random Forest, Gradient Boosting).
- N_sv: Number of support vectors in SVM.
- R: Number of rules in the fuzzy logic systems
- I: Input Calculation complexity within the fuzzy logic system.

The time complexity of training a model reflects how the computational time scales with the amount of training data, while the time complexity of inference represents how the computation time scales when the model is used for predictions on new, unseen data. Space complexity during training pertains to the memory required during the training process, and space complexity during inference indicates the memory consumption for making predictions. It's important to note that Big O notation provides a theoretical measure, and practical performance can vary based on implementation details, hardware capabilities, and the specific dataset being used. Some complexity estimations are approximations because of the non-uniformity of internal operations, particularly in the case of more complex methods. For ensemble methods like Random Forest and Gradient Boosting, time and space complexity are notably influenced by the number of trees or weak learners involved in the model. The comparison in the table highlights that for scenarios with very large datasets, models with lower training complexities, such as KNN, SVM with simple kernels, or simpler decision trees, might be preferred to reduce training time. In real-time inference

scenarios, models with lower inference time complexities, such as KNN, might be more appropriate for applications needing fast responses. Finally, models with high space complexities might not be feasible for use on devices that are limited by memory constraints, making practical considerations a crucial part of model selection.

XV. Conclusion and Future Work

This study presented a comprehensive comparative analysis of eight diverse machine learning and deep learning models for intelligent real-time Air Quality Index (AQI) classification using sensor data, specifically within a smart home digital twin framework. The models evaluated included classical algorithms like Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Random Forest, alongside advanced deep learning architectures such as Transformer, Convolutional Neural Networks (CNN), Gated Recurrent Units (GRU), and Recurrent Neural Networks (RNN).

For smart home indoor air quality (IAQ) classification, Gradient Boosting (GB) or Random Forest (RF) are the most highly recommended models. They provide perfect classification accuracy, precision, recall, specificity, Youden Index, and F1-score while maintaining relatively fast inference speeds, making them ideal for real-time monitoring in resource-constrained smart home environments. K-Nearest Neighbors (KNN) is a very strong alternative, especially when extremely low space complexity (memory usage) is paramount, despite having a slightly higher inference complexity. Other complex models such as TNN, CNN, RNN, and GRU, while performing well, have higher computational costs that do not justify their usage, in comparison to the other models. SVM should also be avoided because of its higher complexity. The perfect performance across all models suggests that the classification task is relatively simple for all, meaning that additional complexity does not increase model performance.

Future work for the IAQ classification model should focus on several key areas to ensure its practical and effective deployment. Performance should be fine-tuned through hyperparameter optimization to balance accuracy, speed, and resource consumption, and deployment should be optimized for low-resource devices by implementing techniques like quantization, compression, and edge computing. Expanding the model to identify anomalies and integrating it with existing smart home systems will enhance its usability and value. Future research should focus on several key areas to further enhance the practical application of these models within smart home digital twins: First, we propose investigating ensemble and hybrid approaches to further improve the robustness and accuracy of real-time AQI classification in varied and complex environments. Second, it's critical to prioritize the development of explainable AI (XAI) techniques to gain a better understanding of the decision-making processes in deep learning models, ensuring that the digital twin's responses are both effective and transparent. Finally, expanding the scope to include additional pollutants and multi-sensor data would enable a more comprehensive and reliable AQI classification, allowing the digital twin to respond more effectively to various scenarios.

Acknowledgment

The authors would like to acknowledge the support of their respective institutes.

Data Availability

The data set used in this research is acquired though sensor readings from CCS811 and is used for air quality classification, specifically relating CO2 and Total Volatile Organic Compound (TVOC) levels to the indoor Air Quality Index (AQI). The data includes individual measurements with CO2 concentrations in parts per million (ppm) and TVOC levels in micrograms per cubic meter (ug/m3). Each measurement will be used to estimate the air quality index AQI and classified through the implemented models as belonging to one of six categories (Excellent, Good, Moderate, Very Unhealthy, Hazardous, Very Hazardous). The majority of the sample represents "Good" to "Moderate" air quality, with CO2 levels clustered around 400 ppm and TVOC ranging from 0 to 10 ug/m3. However, some samples also feature a smaller subset of readings indicating "Hazardous" and "Very Unhealthy" air quality with significantly higher CO2 and TVOC values, demonstrating a wide range of air pollution levels. This data is used to train and evaluate machine learning and deep learning models aimed at accurately classifying air quality for potential integration into a smart home digital twin system. The data set is available for researchers based on fair request. The datasets generated during and/or analyzed during the current study are available from the first author on reasonable request.

Authors' Contributions

This research was a collaborative effort, with all authors contributing significantly to the overall project. Contributions included conceptualization of the idea (Prof. A. S. Tolba), development of the methodology (Saley S. & Abdulaziz A., A. S. Tolba), data collection and analysis (Saley S.), implementation and testing (Saley S. & Abdulaziz A.), manuscript drafting, and review of the final manuscript (Saley S. & Abdulaziz A., A. S. Tolba).

Competing Interests

The authors declare no competing interests.

References

[1] World Health Organization. (2021). Air pollution. https://www.who.int/news-room/fact-sheets/detail/air-pollution.

[2] Pope, C. A., & Dockery, D. W. (2006). Health effects of fine particulate air pollution: lines that connect. Journal of the Air & Waste Management Association, 56(6), 709-742.

[3] V. Kumar, A. K. Singh, M. S. A. Khan, "Air Quality Prediction Using Support Vector Regression Model," International Journal of Advanced Computer Science and Applications, vol. 11, no. 12, 2020, pp. 1-9.

[4] T. V. Ramana, R. G. Rao, K. T. Sarma, "Air quality prediction using k-nearest neighbor algorithm," International Journal of Modern Engineering Research, vol. 2, no. 2, 2012, pp. 1-14.

[5] B. K. Pal, D. Bose, R. K. Das, "A random forest model for prediction of air quality," Journal of Environmental Management, vol. 241, 2019, pp. 501-509

[6] T. Abedi, M. Ahmadi, A. H. Navid, " Air quality monitoring and prediction by AdaBoost," International Journal of Environmental Research, vol. 13, no. 2, 2019, pp. 257-266.

[7] Y. Zheng, F. Liu, " A deep learning model for air quality forecasting," IEEE Access, vol. 7, 2019, pp. 166715-166727.

[8] A. H. L. J. A. Y. Li, F. Zhang, "Air Quality Prediction Using Recurrent Neural Networks", IEEE Access, vol. 9, 2021, pp. 2950-2965.

[9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[10] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Gelly, S. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[11] Altman DG and Bland JM. Statistics notes: diagnostic tests 1: sensitivity and specificity. BMJ 1994; 308: 1552.

[12] Sokolova M, Japkowicz N and Szpakowicz S. Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In: Australian Conference on Artificial Intelligence (Lecture Notes in Computer Science, vol. 4304). Berlin: Springer, 2006, pp.1015–1021.

[13] https://en.wikipedia.org/wiki/Sensitivity_and_specificity ,(1 December 2025).

[14] https://blue.cs.sonoma.edu/cs115/F17/proj/p1/cs115_p1.html ,(1 December 2025.).

[15] Castellani, M., Benini, L., & Brunelli, D. (2021). AI-Driven IoT System for Indoor Air Quality Monitoring and Control in Smart Homes. *IEEE Internet of Things Journal, 8*(7), 5917-5928.

[16] De Vito, S., Del Giudice, A., D'Elia, G., Esposito, E., Fattoruso, G., Ferlito, S., ... & Di Francia, G. (2024). Future Low-Cost Urban Air Quality Monitoring Networks: Insights from the EU's Air Heritage Project. Atmosphere, 15, 1351.

[17] Higgins, C., Kumar, P., & Morawska, L. (2024). Indoor air quality monitoring and source apportionment using low-cost sensors. Environmental Research Communications, 6 (1), 012001. https://doi.org/10.1088/2515-7620/ad1cad.

[18] Tagle, M., Rojas, F., Reyes, F., Vásquez, Y., Hallgren, F., Lindén & Oyola, P. (2020). Field performance of a low-cost sensor in the monitoring of particulate matter in Santiago, Chile. Environ Monit Assess, 192, 171.

[19] TAOYING, L., HUA, M., & WU, X. (2020). A Hybrid CNN-LSTM Model for Forecasting Particulate Matter (PM2.5). IEEE Access, 8.

[20] Xiao, F., Yang, M., Fan, H., & Fan, G. (2020). An improved deep learning model for predicting daily PM2.5 concentration. Scientific Reports, 10, 20988.

[21] Toharudin, T., Caraka, R. E., Pratiwi, I. R., Kim, Y., Gio, P. U., Sakti & Pontoh, R. S. (2023). Boosting Algorithm to Handle Unbalanced Classification of PM2.5 Concentration Levels by Observing Meteorological Parameters in Jakarta-Indonesia Using AdaBoost, XGBoost, CatBoost, and LightGBM. IEEE Access.

[22] Chen, M.-H., Chen, Y.-C., Chou, T.-Y., & Ning, F.-S. (2023). PM2.5 Concentration Prediction Model: A CNN–RF Ensemble Framework. Int. J. Environ. Res. Public Health, 20, 4077. https://doi.org/10.3390/ijerph20054077.