Environmental Science

https://www.aimspress.com/journal/environmental

AIMS Environmental Science, 12(3): 526–556.
DOI: 10.3934/environsci.2025024
Received: 30 December 2024
Revised: 06 May 2025
Accepted: 20 May 2025
Published: 30 May 2025

*Research*

# Time-generative AI-enabled temporal fusion transformer model for efficient air pollution sensor calibration in IIoT edge environments

**Shagufta Henna**[1,*], **Mohammad Amjath**[2] **and Asif Yar**[3]

Department of Computing, Atlantic Technological University, Donegal, Ireland

* **Correspondence:** Email: shaguftahenna@gmail.com.

**Abstract:** The deployment of real-time sensor calibration models for air pollution monitoring on resource-constrained Industrial Internet of Things (IIoT) edge devices presents significant challenges due to the computational complexity and memory requirements of deep learning models. This paper addressed these challenges by proposing a time-series-generative approach that integrated model quantization, generative artificial intelligence (AI), and temporal deep learning architectures to ensure efficient deployment. Specifically, we introduced a TimeGAN-augmented temporal fusion transformer (TFT) model optimized for edge devices. By leveraging model quantization, the approach reduces the memory footprint and computational demands of the model without compromising calibration accuracy. Furthermore, the integration of generative adversarial networks (GANs) enhances the robustness of the model by generating high-quality synthetic time-series data, compensating for sparse or noisy sensor readings. This ability to generate synthetic data mirrors the real sensor trends, ensuring reliable model performance even in data-limited environments. A comprehensive evaluation of the proposed model, comparing its performance against both float and quantized versions, demonstrates the effectiveness of the TimeGAN-augmented quantized TFT. This model achieves a significant 88% reduction in size (from 800.04 KB to 97.34 KB) while maintaining excellent predictive performance, evidenced by a mean squared error (MSE) of 0.3212 and a mean absolute error (MAE) of 0.4375. Additionally, the TimeGAN-augmented Float TFT model emerges as a strong contender for real-time applications, offering an optimal balance between inference speed and accuracy, with a rapid inference time of 23.4 ms, making it ideal for real-time pollution monitoring.

**Keywords:** sensor calibration; TimeGAN; temporal fusion transformer; edge computing; model quantization; generative adversarial networks; IIoT; real-time monitoring; air pollution

## 1. Introduction

Air quality monitoring has become a critical global concern due to its significant environmental and health implications, with polluted air affecting both communities and industries alike [1]. Traditional air pollution monitoring methods, such as stationary reference stations, have provided accurate measurements but are limited by high costs, restricted coverage, and an inability to capture spatiotemporal variations in air quality, which fluctuate rapidly across different locations and times [2].

To overcome these limitations, low-cost wireless sensor networks (WSNs) have been developed as an affordable and scalable solution for real-time air pollution monitoring. These sensors can be deployed in large numbers across urban and industrial environments, providing improved coverage and richer data. However, the accuracy of data collected by these low-cost sensors can be compromised due to factors such as signal interference, environmental noise, and calibration errors [3, 4].

Sensor calibration has thus become a key strategy to ensure the reliability and accuracy of data from low-cost sensors. Calibration techniques align the raw measurements from these sensors with the true values observed by high-precision reference stations [5]. Typically, calibration involves both pre-deployment adjustments to fine-tune the sensors and post-deployment maintenance to ensure sustained accuracy over time. Given the challenges inherent in manual calibration, automatic techniques such as blind and non-blind calibration methods have been developed to improve the reliability of sensor networks for large-scale deployments [6].

The performance of low-cost sensors in WSNs is strongly influenced by sensor type, environmental variability, and deployment conditions. For instance, metal-oxide sensors, though affordable and compact, are particularly susceptible to environmental cross-sensitivities, such as fluctuations in temperature and humidity. These sensors may also suffer from signal drift and aging effects over time. Deployment in uncontrolled outdoor environments further exacerbates these issues, introducing noise from factors such as ambient particulate matter, precipitation, or inconsistent maintenance. These challenges can lead to measurement biases and reduced sensor reliability [7].

As air pollution monitoring expands, there is growing demand for real-time sensor calibration in Industrial Internet of Things (IIoT) environments, where timely, accurate data is crucial for informed decision-making [8]. Traditional sensor calibration methods, reliant on centralized data processing, are increasingly impractical due to the massive data volume and need for rapid processing. Edge computing offers a solution by processing data closer to the source, allowing real-time calibration on edge devices. This reduces the need for large data transfers, minimizes latency, and enhances privacy and security, thus improving air pollution monitoring [9].

However, deploying calibration algorithms on resource-constrained edge devices like IIoT sensors or smartphones is challenging due to limitations in memory, processing power, and energy. Research is focused on optimizing deep learning models for edge deployment. Idrissi et al. [10] proposed lightweight models for intrusion detection in IoT, while Singh et al. [9] developed a hybrid framework to improve real-time security in mobile edge computing, addressing the need for efficient model deployment in resource-limited environments.

Yandouzi et al. demonstrated the value of edge computing for real-time applications by using drones and lightweight deep learning models for forest fire detection [11]. Similarly, Abusitta et al. showed how deep learning enhances anomaly detection in IoT systems, improving system reliability [12]. Further work by Aversano et al. [13] and Ahmad et al. [14] emphasized optimizing models for edge efficiency. Konaite et al. [15] and Sharma et al. [16] expanded these findings to various domains, confirming the broader potential of deep learning on edge devices. To overcome edge device constraints, techniques such as model quantization, weight pruning, and knowledge distillation have been employed to reduce model size and computational load without significantly sacrificing performance [17–19]. Among these, quantization is especially impactful—reducing weight and activation precision lowers memory usage, speeds up inference, and decreases energy consumption, making it ideal for real-time sensor calibration on edge devices [17]. Although quantization may introduce slight accuracy loss, careful tuning can preserve model performance. Tools like TensorFlow Lite enable the deployment of such optimized models on resource-limited devices, supporting accurate real-time calibration [20]. However, temporal models like recurrent neural networks (RNNs) and long short-term memory (LSTM), while effective at modeling time-series data, remain challenging to deploy on edge devices due to their high computational and memory demands [21].

A further challenge in IIoT applications is the issue of insufficient or poor-quality data, which can undermine the accuracy of sensor calibration. Generative AI, especially generative adversarial networks (GANs), offers a promising solution by generating synthetic time-series data that mimics real-world sensor readings. This enables data augmentation when real-world sensor data is sparse, noisy, or of low quality [22]. By augmenting training datasets with synthetic data, GANs enhance the robustness of calibration models, enabling them to make accurate adjustments even in the presence of unreliable or incomplete data.

This work proposes a TimeGAN-augmented temporal fusion transformer (TFT) model optimized for IIoT edge deployment through quantization. The proposed approach is compared against other temporal models, using both float and quantized versions of the model for calibration tasks. By pushing intelligence to the edge, these temporal models enhance the efficiency of environmental monitoring, making air quality monitoring systems more sustainable and responsive. Notably, the TimeGAN-augmented Float TFT strikes a balance between inference speed and accuracy, achieving an inference time of just 23.4 ms, making it ideal for real-time decision-making in pollution control.

The paper is structured as follows: Section 2 reviews related work on sensor calibration, time-series models, and edge deployment optimization techniques. Section 3 details the materials, datasets, and methods, including the TimeGAN-augmented TFT model and quantization strategies for IIoT edge deployment. Section 4 presents results, evaluating TimeGAN-generated data and comparing the proposed model's performance with other temporal models in terms of accuracy, inference time, and resource efficiency. Finally, Section 5 concludes with a summary of findings and future work directions.

## 2. Related works

Air quality prediction is crucial for addressing environmental and public health concerns, especially in IIoT and edge computing contexts. Various studies have explored machine learning (ML) techniques and edge devices to enhance air quality monitoring systems, addressing challenges like limited resources, sparse data, and real-time prediction needs.

In the study by Sun C et al. [21], the authors propose a hybrid methodology combining multi-factor LSTM, deep reinforcement learning (DRL), and optimal stopping theory (OST) for efficient task offloading in edge computing. Data is acquired from multiple monitoring stations, followed by preprocessing steps like normalization and feature selection using the Boruta algorithm [23]. While the methodology is robust, challenges arise from the variability and sparsity of data across stations, which can impact data quality. Additionally, while OST-based K-best selection aids task distribution, network stability concerns in remote or congested areas limit its efficiency. Furthermore, while the LSTM model with attention mechanisms effectively captures temporal dependencies and addresses missing data, its computational demands pose scalability issues, particularly on resource-constrained edge devices.

Moursi et al. introduced an IoT-enabled system for real-time PM2.5 concentration prediction, combining edge devices and cloud computing [24]. Using a nonlinear auto regression with eXogenous input (NARX) framework, the system integrates past PM2.5 data with meteorological inputs to predict the next hour's air quality. While effective for short-term pollution events, the one-hour forecasting horizon limits its ability to capture long-term trends, which are crucial for sustainable air quality management. Additionally, performance tests on a PC and Raspberry Pi highlight the ongoing challenge of balancing computational demands with real-time constraints on edge devices.

A group of researchers developed a cost-effective AI-IoT system for air quality monitoring aimed at individuals with respiratory problems [25]. The system uses low-cost wireless sensor nodes based on ESP32 microcontrollers and ZPHS01B air quality modules to measure pollutants and environmental factors. While the system is cost-efficient and accessible, the trade-off in sensor accuracy remains a significant challenge. The LSTM model used for 24-hour forecasting performs well with time-series data but faces computational challenges for deployment on resource-constrained IoT devices. The dataset, covering just two months, may limit the model's ability to account for seasonal variations or extreme pollution events. Furthermore, LSTM's computational demands, especially for long-term forecasting or large datasets, impose burdens on edge devices, potentially hindering real-time predictions. Transformer-based models [26], known for their ability to handle long-range dependencies in time-series data, could offer a more promising solution. These models could be optimized for edge deployment using techniques such as model pruning or knowledge distillation, which would reduce computational requirements.

Gong et al. proposed a hybrid predictive maintenance model combining convolutional neural networks for spatial feature extraction and LSTM networks for sequential data analysis, aimed at enhancing predictive maintenance for wind turbines [27]. While effective, the model's high computational demands hinder its deployment in resource-constrained edge environments. Similarly, Aggarwal et al. developed a hybrid P-LSTM model for urban air quality forecasting, integrating LSTM with particle swarm optimization (PSO) to optimize hyperparameters [28]. Despite its improved performance, the combined complexity of LSTM and PSO results in increased computational burden, limiting its feasibility for real-time urban air quality prediction on edge devices.

Wardana et al. optimized a hybrid CNN-LSTM model for edge-based air quality forecasting, using post-training quantization techniques to reduce model size and improve execution on low-power devices like Raspberry Pi boards [29]. While this optimization enabled efficient operation on resource-constrained devices, the trade-off between model size reduction and accuracy became apparent, particularly with full integer quantization, which resulted in some loss of predictive precision. Although quantization improved execution speed and reduced model size, it raised concerns about sacrificing accuracy for real-time air quality predictions.

In a similar vein, Hu et al. introduced FedDeep, a federated deep learning approach for multi-urban PM2.5 forecasting, which utilizes edge computing to process local data and reduce cloud dependency [30]. By incorporating a novel gating fusion layer to adapt weather data for PM2.5 predictions, FedDeep enhances accuracy and alleviates computational pressure on cloud servers. However, while the federated architecture offers advantages in data privacy and reduced cloud reliance, it introduces complexity in coordination and communication between edge servers and the cloud, which can challenge scalability and operational efficiency.

Koziel et al. proposed a machine learning-based calibration method for low-cost nitrogen dioxide sensors using neural network (NN) surrogates, specifically multi-layer perceptrons, to predict correlation coefficients from environmental parameters [31]. Although the approach demonstrated high calibration efficiency with a correlation coefficient over 0.9 and RMSE below 3.2 $\mu g/m^3$, its complexity, requiring multiple sensors and advanced techniques, limits scalability, especially in resource-limited or large-scale deployments.

Yu et al. introduced AirNet, a dual encoder architecture that maps the calibration task into a sequence-to-point format, utilizing data from both mobile and static stations [32]. While it outperformed several baseline approaches in reducing forecasting errors and improving accuracy, the increased complexity, incorporating gated recurrent units (GRU), convolutional layers, and a social-based guidance mechanism, raises concerns about computational intensity. Extended training times and high computational requirements make it unsuitable for real-time applications or resource-constrained environments.

In the study by Yar A et al. [33], the authors explored the use of graph convolutional betworks (GCN) and extreme learning machines (ELM) for self-calibration in large-scale WSNs. GCN achieved 95% accuracy by capturing spatial and temporal features, while ELM, with faster learning times (109 seconds), offered lower accuracy (70%). This illustrates the trade-off between accuracy and computational efficiency in environmental monitoring. Schmitz et al. compared multiple linear regression and random forest (RF) models for calibrating low-cost metal oxide gas-phase sensors [34]. Both models achieved $R^2$ values above 0.8, but RF outperformed linear regression by capturing non-linear relationships. However, RF's increased complexity poses computational challenges, particularly for real-time or resource-constrained applications. Wang et al. applied RF to enhance calibration in low-cost air quality monitoring systems, achieving $R^2$ values between 0.70 and 0.99 for multiple pollutants [35]. While the model demonstrates high accuracy, its computational intensity makes it impractical for large-scale, real-time deployment, especially in resource-limited environments. In the study by Rahardja U et al. [36], the authors proposed AIKU, a transfer learning and meta-learning-based approach for calibrating low-cost PM2.5 sensors. AIKU outperformed traditional methods by enabling rapid adaptation to new sensor locations with minimal training data. However, it remains computationally demanding, with longer training times than simpler models.

A different approach, GenCast, a probabilistic weather forecasting model, significantly outperformed traditional ensemble systems, generating sharper, more realistic weather trajectories in about 8 minutes [37]. However, GenCast's high computational demands raise scalability concerns, particularly for real-time forecasting in resource-limited settings, emphasizing the need for models that balance high predictive accuracy with computational efficiency. Li et al. introduced the scalable ensemble envelope diffusion sampler (SEEDS), a deep generative model for generating large weather forecast ensembles quickly [38]. While SEEDS outperformed operational ensemble forecasts in reliability and skill, its significant computational requirements limit its applicability in environments with limited resources.

The reviewed literature emphasizes advancements in ML for low-cost air quality monitoring and weather forecasting, highlighting the trade-offs between prediction accuracy and computational efficiency. Techniques such as neural networks [31] and transfer learning [36] deliver high accuracy but are computationally demanding, which limits their deployment in real-time, resource-constrained environments. Models like GCN [33] and generative AI systems [37] show promise, but their substantial computational requirements hinder their practical use in edge applications within the IIoT. Furthermore, many generative models struggle with reconstruction loss, affecting their generalization ability in dynamic and noisy environments. These challenges emphasize the need for lightweight, scalable models that strike a balance between accuracy and operational feasibility, particularly for edge-based IIoT air quality systems that demand real-time processing and low power consumption.

## 3. Materials and method

This section presents the proposed approach for calibrating air pollution sensor networks, which includes the TimeGAN TFT model and its deployment on edge devices for the IIoT.

### *3.1. Time generative adversarial network (TimeGAN)*

In this section, we apply the TimeGAN to the task of generating synthetic air pollution data for sensor calibration. The TimeGAN was selected due to the temporal nature of sensor readings, as it uniquely combines adversarial training with explicit temporal supervision. This enables the model to preserve both the time-dependent structure and the statistical properties of multivariate sensor data. Unlike conventional GANs, the TimeGAN captures long-term dependencies through recurrent architectures, ensuring the generation of realistic and temporally coherent sequences, an essential requirement for accurate calibration. As a result, it is particularly effective in replicating the dynamic behavior of environmental data, outperforming other generative models in terms of both fidelity and downstream utility.

Figure 1 illustrates the architecture of the TimeGAN model, which consists of several key components that work together to model and replicate the complex patterns observed in time series data, especially those associated with fluctuations in air quality measurements. The core of the model consists of multiple modules, each designed to handle a specific aspect of the data generation task. The Embedder module, built using an LSTM network, processes the input time series and transforms it into a latent space representation, capturing the temporal dependencies within the data. This embedded representation is then passed to the Generator, which also uses LSTM layers to generate synthetic latent sequences. The Generator's goal is to learn the underlying distribution of the input data and generate new time series sequences that closely resemble the real data. The Recovery module, another LSTM-based component, is responsible for transforming the generated latent representations back into the original feature space, producing synthetic time series data that mirrors the original measurements [39]. Finally, the Discriminator module evaluates the generated sequences, distinguishing between real and synthetic data by learning to identify which sequences belong to the original dataset. During the training process, the model is optimized through a combination of reconstruction loss, adversarial loss, and other specialized losses. The reconstruction loss ensures that the generated time series are as similar as possible to the original data, while the adversarial loss helps the model distinguish real

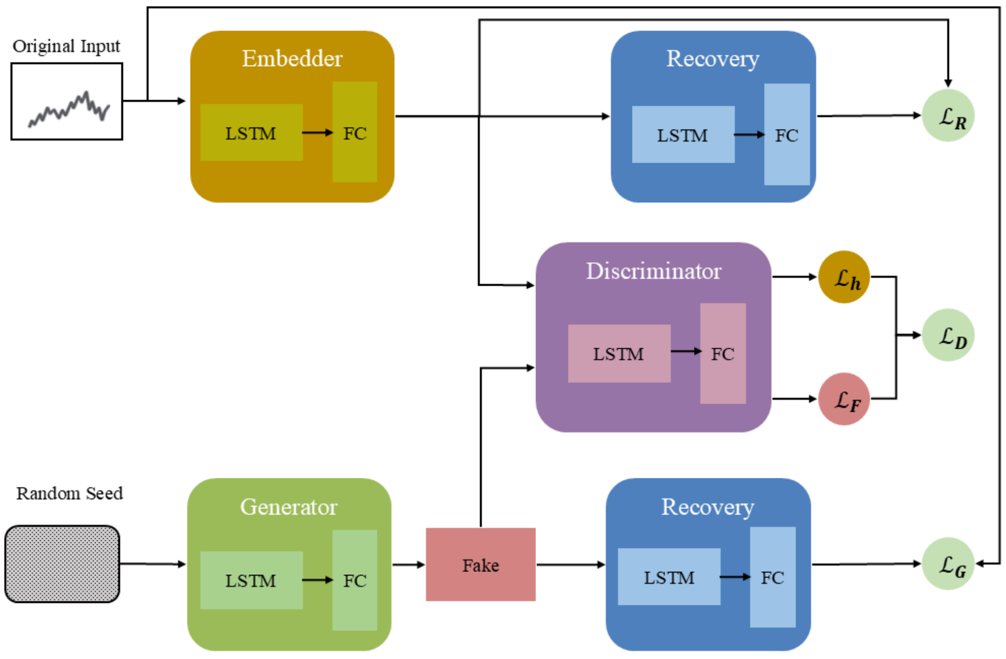

**Figure 1.** The TimeGAN architecture for generating synthetic air pollution data for sensor calibration. The model consists of an Embedder that encodes time series into a latent space, a Generator that creates synthetic sequences, a Recovery module that reconstructs the generated data, and a Discriminator that distinguishes real from synthetic data.

from fake data through the interaction between the Generator and the Discriminator. This interplay encourages the Generator to improve its data generation capabilities and produce sequences that are increasingly difficult for the Discriminator to differentiate from real data. These losses are minimized over several epochs, gradually refining the model's ability to generate synthetic time series that are indistinguishable from real sensor data. The algorithm for training the TimeGAN, detailed in Algorithm 1, outlines the step-by-step process of data preprocessing, model initialization, and training. The algorithm emphasizes the iterative nature of the training process, where the model is optimized over multiple epochs. After training, the Generator and Recovery modules work together to produce synthetic time series that are then denormalized to match the scale of the original data. This approach preserves the crucial temporal and structural characteristics of the original air pollution measurements, ensuring that the generated data can be used for accurate sensor calibration and effective air quality monitoring.

### 3.2. *Temporal fusion transformer model*

The TFT module combines several advanced components to improve the predictive performance of the calibration of air pollution monitoring as given in Figure 2. Its different components are explained below.

**Algorithm 1** TimeGAN for Time Series Data Generation

**Input:** Time series data $D$, Sequence length $L$, Samples $N$, Epochs $\mathcal{E}$, Batch size $B$, Hidden dimension $H$, Latent dimension $Z$, Learning rate $\alpha$

**Output:** Synthetic time series data $S$

```
1:  Preprocessing:
2:  Normalize $D$ and create sequences of length $L$
3:  Model Initialization:
4:  Initialize models: Embedder $E$, Recovery $R$, Generator $G$, Discriminator $D$
5:  Define loss functions $\mathcal{L}_{\text{MSE}}$ and $\mathcal{L}_{\text{BCE}}$
6:  Training:
7:  for $epoch = 1$ to $\mathcal{E}$ do
8:      for each batch of size $B$ from $D$ do
9:          Step 1: Reconstruction Loss
10:         $H \leftarrow E(\text{batch})$                         ▷ Encode batch
11:         $\hat{B} \leftarrow R(H)$                              ▷ Reconstruct batch
12:         Compute $\mathcal{L}_{\text{recon}} = \mathcal{L}_{\text{MSE}}(\hat{B}, \text{batch})$
13:         Update $E$ and $R$ using $\mathcal{L}_{\text{recon}}$
14:         Step 2: Generator Loss
15:         $Z \sim \mathcal{N}(0, 1)$                             ▷ Generate latent noise
16:         $H' \leftarrow G(Z)$                                   ▷ Generate hidden states
17:         $\hat{B}' \leftarrow R(H')$                            ▷ Recover synthetic data
18:         Compute $\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{MSE}}(\hat{B}', \text{batch})$
19:         Update $G$ using $\mathcal{L}_{\text{gen}}$
20:         Step 3: Discriminator Loss
21:         Compute $\mathcal{L}_{\text{disc}}$ using $H$ and $H'$
22:         Update $D$ using $\mathcal{L}_{\text{disc}}$
23:     end for
24: end for
25: Synthetic Data Generation:
26: Sample $Z \sim \mathcal{N}(0, 1)$
27: Compute $H' \leftarrow G(Z)$ and $S \leftarrow R(H')$
28: Denormalize $S$ to the original scale
29: return $S$
```

### 3.2.1. Gated residual networks

The Gated Residual Network (GRN) is a core building block in TFT, which allows the model to perform adaptive non-linear transformations. The GRN enables the model to decide which parts of the input should be transformed, making it flexible for various types of data [40]. The GRN applies a residual connection combined with gating mechanisms. Specifically, for a given feature $x$, and an optional context vector $c$ (static metadata or context), the GRN is computed as:

$$\text{GRN}_\omega(x, c) = \text{LayerNorm}(x + \text{GLU}_\omega(\eta_1)) \tag{1}$$

where the output of the GRN is the result of a gated linear unit (GLU), applied to a transformation of the input as given in two steps:

$$\eta_1 = W_{1,\omega}\eta_2 + b_{1,\omega} \tag{2}$$

where $\eta_2$ is an intermediate result computed as:

$$\eta_2 = \mathrm{ELU}(W_{2,\omega}a + W_{3,\omega}c + b_{2,\omega}) \tag{3}$$

The exponential linear unit (ELU) is the activation function, which introduces non-linearity into the transformation in the first step. $W_{2,\omega}$ and $W_{3,\omega}$ are learned weight matrices for the input data $x$ and the context vector $c$, respectively, while $b_{2,\omega}$ is a bias term.

The GLU function applies a non-linearity to the intermediate result $\eta_1$. The GLU operation is expressed as:

$$\mathrm{GLU}_\omega(\gamma) = \sigma(W_{4,\omega}\gamma + b_{4,\omega}) \odot (W_{5,\omega}\gamma + b_{5,\omega}) \tag{4}$$

where $\sigma$ is the sigmoid activation, and $\odot$ denotes element-wise multiplication. The GLU controls the extent of non-linearity applied to the input data.

### 3.2.2. Variable selection network

The variable selection network (VSN) is responsible for identifying and selecting the most relevant input variables at each time step. By focusing on the most important features, it helps the model minimize the influence of irrelevant or noisy variables, thereby enhancing overall performance.

At each time step $t$, the variable selection weight $v_\chi(t)$ for each variable in $X_{\text{combined}}$, derived from $X_{\text{real}}$ and $X_{\text{synthetic}}$, is computed as:

$$v_\chi(t) = \mathrm{Softmax}(\mathrm{GRN}_v(x_t, c_s)) \tag{5}$$

where $x_t$ denotes the transformed input features for all variables at time $t$, with $x_t^j$ representing the feature corresponding to the $j$-th variable. The static context vector $c_s$ contains additional metadata or fixed information. The Softmax function normalizes the weights across all variables at time $t$, ensuring the model focuses on the most relevant ones. Each feature $x$ is processed independently by its own GRN:

$$\tilde{x}(t) = \mathrm{GRN}((x(t)) \tag{6}$$

This transformation allows each feature to be processed in a way that emphasizes its most important aspects. The transformed features are then weighted by their corresponding weights:

$$\tilde{x}(t) = \sum_{j=1}^{m_\chi} v_{\chi,j}(t)\tilde{x}_j(t) \tag{7}$$

This weighted sum helps the model focus on the most relevant variables while reducing the effect of irrelevant features.

### 3.2.3. Multi-head attention mechanism in TFT

The attention mechanism computes the output as a weighted sum of the values $V$, where the weights are determined by the similarity between the query matrix $Q$ and the key matrix $K$. This mechanism enables the model to selectively focus on the most relevant parts of the time series input sequence by evaluating the degree of similarity between the query and each part of the sequence, thereby allocating attention dynamically. The model learns to emphasize the most important features, which enhances its ability to capture long-range dependencies within the data. The output of the attention mechanism is given by:

$$\text{Attention}(Q, K, V) = A(Q, K)V \tag{8}$$

Here, $A(Q, K)$ represents the attention function, which computes the attention scores based on the relationship between the query and key matrices. This function is implemented through the scaled dot-product attention mechanism, formulated as:

$$A(Q, K) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_{\text{attn}}}}\right) \tag{9}$$

In this context, $Q$ is the query matrix, which encodes the time series features at the current time step that the model is attending to. The key matrix $K$ represents the various parts of the time series input sequence that might be relevant to the current query. The value matrix $V$, in turn, contains the actual information associated with each corresponding key. By calculating the similarity between the query and key, and normalizing the resulting scores, the attention mechanism facilitates a weighted aggregation of values from the most relevant parts of the sequence, thus improving the model's capacity to capture meaningful temporal relationships.

The parameter $d_{\text{attn}}$ denotes the dimension of the attention vectors, and the scaling factor $\frac{QK^T}{\sqrt{d_{\text{attn}}}}$ plays a crucial role in stabilizing the model's training process by normalizing the dot product. This normalization ensures that the attention scores remain within a manageable range and helps prevent the gradients from becoming excessively large. After computing the scaled dot product, the Softmax function normalizes the attention scores into a probability distribution, ensuring they sum to one.

To capture multiple aspects of the time series data, the multi-head attention mechanism splits the attention process into $m_H$ parallel heads. Each attention head captures different subspaces of the input sequence, allowing the model to learn richer representations of the time series features. The multi-head attention is computed as:

$$\text{MultiHead}(Q, K, V) = [H_1, \ldots, H_m]W_H \tag{10}$$

where each head $H_h$ computes attention as:

$$H_h = \text{Attention}(QW_Q^{(h)}, KW_K^{(h)}, VW_V^{(h)}) \tag{11}$$

In this case, $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)}$ are learned weight matrices for each head, which allow the model to capture different aspects of the time series data at each attention head. The final output of the multi-head attention mechanism is obtained by applying a linear transformation to the concatenated heads:

$$\text{MultiHead}(Q, K, V) = \frac{1}{m_H} \sum_{h=1}^{m_H} \text{Attention}(QW_Q^{(h)}, KW_K^{(h)}, VW_V^{(h)}) \tag{12}$$

This approach allows the model to focus on different aspects of the time series data, capturing long-term dependencies across different time steps while preserving interpretability.

### 3.2.4. Static covariate encoders

In contrast with other time-series forecasting architectures, the TFT integrates information from static metadata, using separate GRN encoders to produce four different context vectors, $c_1$, $c_2$, $c_3$, and $c_4$. These context vectors are given as input to the temporal fusion decoder through GRNs. Specifically, this includes contexts for temporal variable section $c_1$, locally processed temporal features $c_2$, $c_3$, and enriched temporal features with static information $c_4$.

### 3.2.5. Temporal fusion decoder

The temporal fusion decoder combines the outputs from previous layers to model temporal dynamics and generate final predictions. It first applies a sequence-to-sequence layer to capture the local context, as shown in the equation:

$$\hat{y}(t, n) = \text{LayerNorm}(x_t + \text{GLU}_\theta(\varphi(t, n))) \tag{13}$$

Here, $\hat{y}(t, n)$ is the output at time step $t$ for the $n$-th feature. $x_t$ is the raw input time series data at time $t$, while $\varphi(t, n)$ is the transformed feature at time $t$ for the $n$-th variable. The GLU, $\text{GLU}_\varphi(\varphi(t, n))$, modulates the importance of different parts of the transformed feature. Layer normalization, LayerNorm$(\cdot)$, is applied to the sum of $x_t$ and the GLU-transformed feature to stabilize training and ensure consistent input distributions. This operation helps the model capture short-term dependencies in the data. The temporal features are then enhanced with static metadata by using a GRN, as shown in the equation:

$$\theta(t, n) = \text{GRN}_\theta(\hat{y}(t, n), c_2) \tag{14}$$

In this expression, $\theta(t, n)$ represents the enriched feature at time step $t$ for the $n$-th variable. $\tilde{\xi}(t, n)$ is the transformed temporal feature, and $c_2$ is the static context vector, which contains fixed, non-time-varying information about the data. The GRN helps integrate these static features with the dynamic temporal data, providing the model with additional context to improve the accuracy and interpretability of the predictions.

### 3.2.6. Self-attention and feed-forward processing

After applying multi-head attention to the enriched features, the decoder applies a feed-forward layer:

$$\psi(t, n) = \text{GRN}_\theta(\delta(t, n)) \tag{15}$$

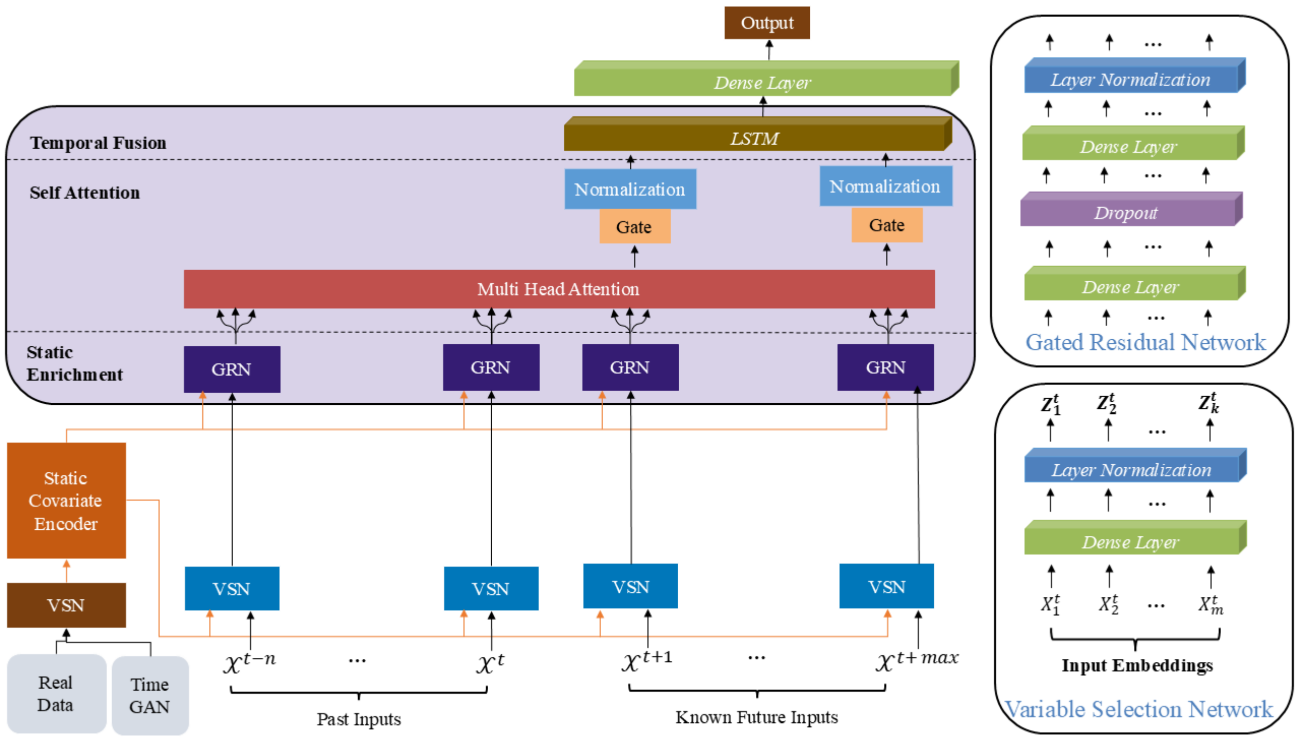

**Figure 2.** The figure illustrates the different components of the TimeGAN-augmented temporal transformer model, including the VSN, GRN, covariate encoder, and temporal fusion.

where $\delta(t, n)$ is the result of the self-attention layer. This final layer further refines the temporal features. The quantile forecast for a given quantile $q$ is computed by a linear transformation of the output $\psi(t, \tau)$ from the decoder:

$$\hat{y}(q, t, \tau) = W_q \psi(t, \tau) + b_q \tag{16}$$

where $W_q$ and $b_q$ are the learned weights and biases for the $q$-th quantile. The prediction is computed for each forecast horizon $\tau \in \{1, \ldots, \tau_{\max}\}$, ensuring that the forecasts are made for future time steps only.

The steps, detailed in Algorithm 2, leverage a multi-layer framework combining both real-time and augmented data to improve model training.

The model architecture consists of several key components: a VSN, GRN, multi-head attention mechanism, LSTM layer, and FFN. The VSN learns feature importance by applying a dense layer followed by layer normalization, selecting relevant features from the input time series. The GRN, which follows the VSN, performs additional transformations through dense layers, incorporating dropout for regularization.

The attention mechanism enables the model to focus on critical temporal dependencies within the data by computing multi-head attention between the transformed sequences. This is followed by the LSTM layer, which captures long-term dependencies in the time series, and an FFN that further refines the output predictions.

By combining real-time time series data with synthetic data generated by the TimeGAN, the model

**Algorithm 2** TimeGAN-Augmented Temporal Fusion Transformer

```
Input: $X_{\text{real}}, Y_{\text{real}}, X_{\text{synthetic}}, Y_{\text{synthetic}}$
Output: Trained model and predictions on test data
 1: Normalize and reshape $X_{\text{real}}, X_{\text{synthetic}}$
 2: $X_{\text{combined}} \leftarrow \text{concat}(X_{\text{real}}, X_{\text{synthetic}})$
 3: $X_{\text{input}} \leftarrow \text{DefineInputLayer}((L, F))$
 4: $vsn \leftarrow \text{LayerNormalization}(\text{Dense}(\text{activation='relu'})(X_{\text{combined}}))$
 5: $grn \leftarrow \text{LayerNormalization}(\text{Dense}(\text{activation='relu'})(vsn))$
 6: $grn \leftarrow \text{Dropout}(0.2)(grn)$
 7: $attn \leftarrow \text{LayerNormalization}(\text{MultiHeadAttention}(grn))$
 8: $attn \leftarrow \text{Dropout}(0.1)(attn)$
 9: $lstm \leftarrow \text{LSTM}(attn)$
10: $lstm \leftarrow \text{Dropout}(0.2)(lstm)$
11: $ffn \leftarrow \text{Dense}(\text{activation='relu'})(lstm)$
12: $ffn \leftarrow \text{Dropout}(0.2)(ffn)$
13: $output \leftarrow \text{Dense}(1)(ffn)$
14: Loss $\leftarrow$ MAE, Optimizer $\leftarrow$ Adam
15: Train for $\mathcal{E}$ epochs on $(X_{\text{train}}, Y_{\text{train}})$, validate on $(X_{\text{val}}, Y_{\text{val}})$
16: Return: Trained model and predictions $Y_{\text{pred}}$
```

is able to learn more robust representations, even in the presence of limited or noisy real data. The synthetic data helps augment the model's training, improving its ability to generalize and predict unseen sequences. During training, the model is optimized using the MAE loss function and Adam optimizer, iterating through multiple epochs to achieve convergence.

Once trained, the TFT model can make accurate predictions on future time series data for calibrating air pollution sensors in an air pollution monitoring sensor network.

### 3.3. *Deployment of the TimeGAN temporal fusion transformer (TimeGAN TFT) model for air pollution monitoring in the IIoT*

The deployment of the TimeGAN TFT model in the IIoT for air pollution monitoring requires substantial optimization to ensure its performance on low-power, resource-limited edge devices. This section describes the process of deploying the TimeGAN TFT model on IIoT edge devices, emphasizing memory and computational efficiency, which are critical for such environments.

#### 3.3.1. Model conversion and optimization for edge devices

The first step in deploying the TimeGAN TFT model on edge devices is to convert the trained model into the TFLite format. TFLite is designed specifically for deployment on mobile devices and embedded systems, making it ideal for edge computing applications such as the IIoT. This conversion transforms the model into a lightweight, optimized format, suitable for running on devices with limited resources such as microcontrollers and low-power processors commonly used in IIoT systems. To achieve this, the trained model is first converted into TFLite format using the function convert_to_tflite(). As shown in Algorithm 3 (Step 1), this step ensures that the TimeGAN TFT model

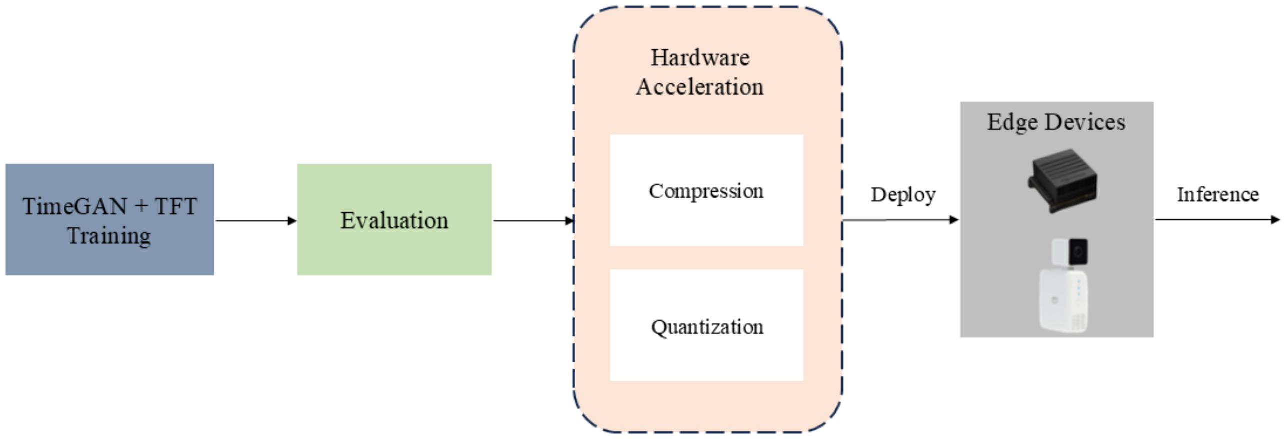

**Figure 3.** The figure depicts the deployment of the TimeGAN-augmented Temporal Fusion Transformer model, utilizing quantization and compression techniques.

**Algorithm 3** Deployment of the TFT Model for IIoT Edge Devices

**Input:** Trained TFT Model, Test Data ($X_{\text{test}}$)

**Output:** Inference from the deployed model

```
1: $model \leftarrow$ convert_to_edge_format($TFT\_model$)            ▷ Convert model for edge deployment
2: $model \leftarrow$ optimize_for_edge($model$)                      ▷ Optimize for resource constraints
3: $interpreter \leftarrow$ initialize_interpreter($model$)           ▷ Initialize model interpreter
4: $input\_details, output\_details \leftarrow$ get_io_details($interpreter$)   ▷ Get input/output details
5: $sample \leftarrow$ select_sample($X_{\text{test}}$)               ▷ Select a sample from test data
6: $reshaped\_data \leftarrow$ prepare_data($sample$)                 ▷ Reshape and type-cast data for input
7: $interpreter \leftarrow$ set_input($interpreter, reshaped\_data$)  ▷ Feed input to model
8: $predictions \leftarrow$ run_inference($interpreter$)              ▷ Run inference to get predictions
9: Return: $predictions$                                              ▷ Return the model's predictions
```

is compatible with edge devices. However, the initial TFLite conversion does not address the model's memory footprint and inference latency, which may still be considerable on resource-constrained devices. To tackle these challenges, additional optimizations are applied. Following the conversion, Algorithm 3 (Step 2) details the additional optimizations performed, including enabling resource variables, setting supported operations, and disabling tensor list lowering. These optimizations preserve the integrity of the model and ensure that it runs efficiently on the target hardware, especially for recurrent operations in the TimeGAN model, which is common in time-series forecasting tasks like air pollution prediction.

### 3.3.2. Post-training quantization for model compression

Post-training quantization is an important technique used to make machine learning models smaller and faster, especially when they need to run on devices with limited memory and processing power, such as edge devices. This is particularly relevant in applications like air pollution monitoring, where predictions must be made in real time with limited hardware resources. As described in Algorithm 4, the quantization process starts by converting the model's weights from standard 32-bit floating-point

**Algorithm 4** Post-Training Quantization of TimeGAN-Augmented TFT

**Input:** Trained TimeGAN Model, Test Data (`X_test`)

**Output:** Quantized model predictions

```
1: Step 1: Model Conversion and Quantization
2: $tflite\_float\_model \leftarrow$ convert_to_tflite($timegan\_model$)   ▷ Convert model to TFLite.
3: $tflite\_quantized\_model \leftarrow$ apply_quantization($tflite\_float\_model$)   ▷ Post-training quantization
4: save_model("timegan_quantized.tflite")   ▷ Save quantized model.
5: $quantized\_model\_size \leftarrow$ get_model_size($tflite\_quantized\_model$)   ▷ Get quantized model.
6: Step 2: Load Model and Allocate Memory
7: $interpreter\_quantized \leftarrow$ load_model(”$timegan\_quantized.tflite$”)   ▷ Load quantized model.
8: $interpreter\_quantized$.allocate_tensors()   ▷ Allocate model tensors.
9: Step 3: Prepare Input Data
10: $reshaped\_data \leftarrow$ reshape_data($X\_test$)   ▷ Reshape input data.
11: Step 4: Inference on Quantized Model
12: $predictions \leftarrow$ run_inference($interpreter\_quantized, reshaped\_data$)   ▷ Run inference.
13: Step 5: Return Results
14: return predictions   ▷ Return model predictions.
```

numbers to lower-precision 8-bit integers. This step significantly reduces the model's size and speeds up how quickly it can make predictions. For example, the model is first converted into a TFLite format in a floating-point version. Then, a simplified post-training quantization step reduces the file size while preserving most of the model's original accuracy. While quantization can slightly reduce prediction accuracy, the trade-off is carefully managed. We evaluated the model's performance after quantization to ensure it still meets the accuracy needs of pollution monitoring applications. Despite the minimal loss, the model remains effective at capturing important patterns in the sensor data, making it suitable for real-world deployment.

### 3.3.3. Inference pipeline on edge devices

Once the model has been converted and quantized, it is ready for deployment on edge devices for real-time inference. In Algorithm 3 (Step 2), the TFLite model is loaded into the interpreter, which then assigns tensors for inference. This step ensures that the model is ready to accept input data and produce predictions. The prepared input data is passed to the model (as shown in Algorithm 4 (Step 3), where it is reshaped and converted to the required format for inference. In Algorithm 3 (Step 4), the input data is processed by the interpreter and predictions are generated. These predictions represent the air pollution levels and can trigger necessary actions, such as activating air filtration systems or sending alerts to monitoring stations.

The final inference results from the quantized model are returned in Algorithm 4 (Step 4). In this step, the quantized model performs inference efficiently on the edge device, leveraging hardware accelerators like tensor processing units. This allows the model to process the data rapidly, making real-time decisions in IIoT systems that monitor environmental parameters for air quality.

## 4. Performance evaluations

This section begins with an overview of the dataset, followed by a detailed analysis of the results from the TimeGAN-generated data. It also includes an evaluation of downstream regression tasks, specifically focusing on sensor calibration for air pollution applications.

### *4.1. Dataset description*

The dataset used in this research was collected from the Captor17013 sensor node deployed in Manlleu, Barcelona, Catalonia, Spain, between June 21, 2017, and August 11, 2017. This data was part of the European H2020 Captor project, which aimed to enhance the calibration of low-cost air pollution sensors through a WSN [41]. The Captor sensor nodes, equipped with IoT devices and processing units, were used to measure various air quality parameters. Specifically, the Captor17013 node features four metal-oxide sensors, including those for ozone (O3), temperature (Temp), and relative humidity (RelHum). For this study, data from the first O3 sensor, along with the temperature and relative humidity sensors, were used for training, while O3 pollutant concentration data from a reference station provided by the Government of Catalonia, Spain, served as the ground truth [42].

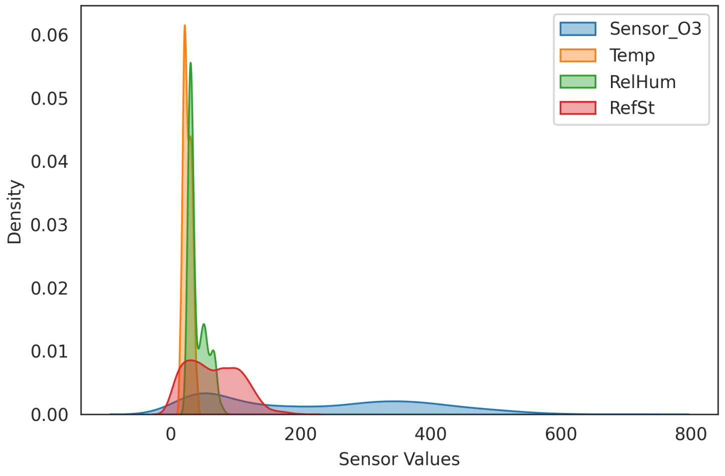

**Figure 4.** KDEs of Sensor_O3, temperature, and relative humidity.

Table 1 presents descriptive statistics for the Captor17013 dataset, highlighting the variability in ozone concentration (Sensor_O3) with values ranging from 20.83 to 682.36 and a high standard deviation, indicating significant fluctuations in air quality. Temperature and relative humidity exhibit moderate variability, while the reference station ozone values (RefSt) show a lower mean and range, reflecting the difference in sensor sensitivity and the need for calibration. As shown in Figure 4,

**Table 1.** Descriptive statistics of the Captor dataset.

| Feature | Count | Mean | Std | Min | 25% | 50% |
|---|---|---|---|---|---|---|
| Sensor_O3 | 1241 | 222.59 | 158.87 | 20.83 | 60.69 | 209.26 |
| Temp | 1241 | 25.74 | 6.34 | 12.33 | 21.00 | 24.97 |
| RelHum | 1241 | 40.02 | 13.78 | 23.43 | 29.63 | 33.20 |

the kernel density estimation (KDE) for Sensor_O3 reveals a unimodal distribution with a gradual increase, followed by a slow decline, suggesting a moderately symmetric and broad spread of ozone concentration values. In contrast, the temperature exhibits a sharper, left-skewed distribution, indicating a more concentrated temperature range. Meanwhile, relative humidity displays a multimodal pattern with a primary peak around 30–31% and secondary fluctuations, suggesting more variability and complexity in humidity levels compared to the other two parameters.

The dataset includes several key features. The date feature represents the timestamp of each data sample, with samples taken every hour. The Sensor_O3 feature provides the measurement of ozone (O3) concentration recorded by the SGX Sensortech MICS 2614 metal-oxide (O3) sensor, with the concentration reported in Kelvin (K). The Temp feature captures the temperature value recorded by the temperature sensor at the time of measurement. The RelHum feature represents the relative humidity percentage, measured by the humidity sensor during the same period. Finally, the RefSt feature contains the ground truth O3 pollutant concentration, as provided by government reference stations, with the concentration measured in micrograms per cubic meter $\mu g/m^3$ [42].

The model uses an LSTM layer to process sequential data, followed by a feed-forward network (FFN) for the regression task. Dropout layers (0.2 and 0.1) are applied throughout to prevent overfitting. Preprocessing involves normalizing input features using Z-scores and standardizing the training and test sets with Scikit-learn's StandardScaler. The model is compiled with the Adam optimizer and trained for 20 epochs with a batch size of 32, using MAE as the loss function.

### *4.2. Results and analysis*

#### 4.2.1. TimeGAN for data synthesis: analysis and evaluation

The training results for the TimeGAN model, shown in Figure 5, demonstrate successful learning for generating synthetic time series data. At epoch 0, the reconstruction (0.0378), generator (0.0437), and discriminator (0.1808) losses indicate initial instability. By epoch 50, losses stabilize, with both discriminator and reconstruction losses nearing zero, indicating effective learning of the sensor data's temporal structure. The consistent decrease in reconstruction loss reflects the embedder and recovery networks' ability to accurately reconstruct input sequences. The generator's loss reduction shows improved synthetic sequence generation, whereas the discriminator's near-zero loss by epoch 50 (0.0001) confirms its ability to distinguish real from generated data. The similar behavior of generator and discriminator losses highlights the stable adversarial dynamics throughout training.

The synthetic sensor readings generated by the TimeGAN model, as shown in Figure 6, reveal the model's capability to generate realistic and coherent data sequences over time. Figure 6 illustrates the hourly sensor measurements for multiple parameters, including Sensor_O3, temperature, and relative humidity. Notably, the Sensor_O3 values exhibit considerable fluctuations, which is expected given the dynamic nature of air pollution levels. This behavior closely mirrors real-world sensor data, where pollutant concentrations often experience sharp spikes and declines, influenced by various environmental factors.

The synthetic data reflect realistic temperature and relative humidity trends, aligning with typical patterns seen in uncontrolled environments, such as daily temperature increases and corresponding humidity fluctuations.The TimeGAN effectively captures these trends, making it a valuable tool for simulating sensor data for calibration. This is crucial for air pollution monitoring networks, where

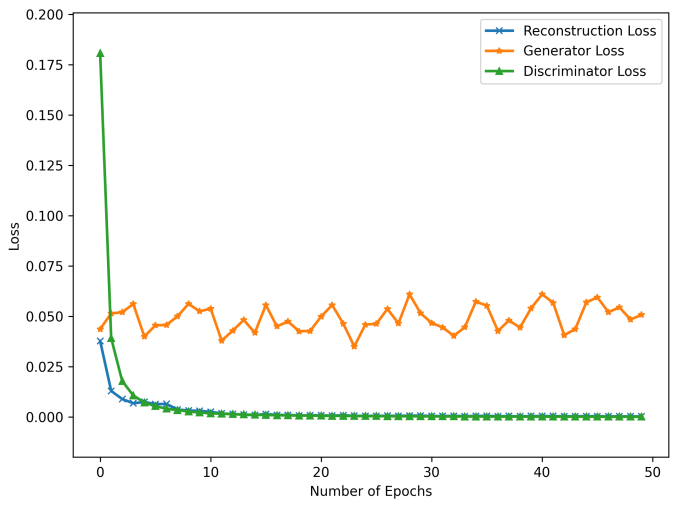

**Figure 5.** The plot illustrates the progression of reconstruction Loss, generator Loss, and discriminator Loss during the training of a TimeGAN model.

accurate calibration ensures sensor reliability across diverse real-world conditions. As shown in Figure 6, the generated data preserve key characteristics of the original time series, including peak timing, general trends, and statistical consistency.

The t-SNE results as given in Figure 7 reveal that the synthetic data generated by the TimeGAN is more concentrated, with most points clustered in the central region of the 2D space, reflecting a more uniform distribution. In contrast, the real data display a wider spread, with points extending further across the space, indicating greater variability and outliers. This suggests that while the synthetic data capture the general structure of the real sensor data, it underrepresents the broader range of variations and extreme values seen in the real data.

To quantitatively assess the similarity between real and TimeGAN-generated time series, we adapted the dynamic time warping (DTW) and a fréchet inception distance (FID)-like score for the time series data (Table 2) [43]. The DTW distances, ranging from 6.16 to 7.41 across the three features, suggest a moderate but consistent temporal alignment between synthetic and real sequences. *RelHum* showed slightly greater deviation (DTW = 7.4060), reflecting its higher natural variability. The FID-like score of 0.1941 indicates a high degree of statistical similarity in the underlying distributions. These results confirm that although the synthetic data is not identical to the real observations, it closely captures both the temporal structure and statistical properties of the original sensor signals.

### 4.2.2. TimeGAN-generated data for downstream calibration regression

In this section, we evaluate the effectiveness of synthetic data generated by the TimeGAN in supporting downstream regression tasks for sensor calibration. Specifically, we employ the TFT model for the regression task using TimeGAN-generated data and compare its performance with a range of baseline models, including LSTM-Attention, CNN-Attention, and variational autoencoder (VAE)-based architectures.

The training loss of the TFT model, shown in Figure 8, illustrates its learning dynamics when trained

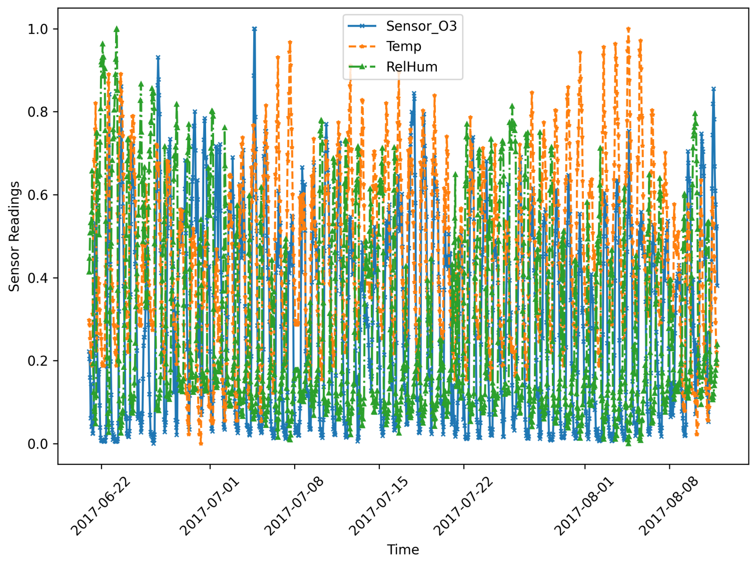

**Figure 6.** The plot depicts the synthetic sensor readings generated over time using the TimeGAN model.

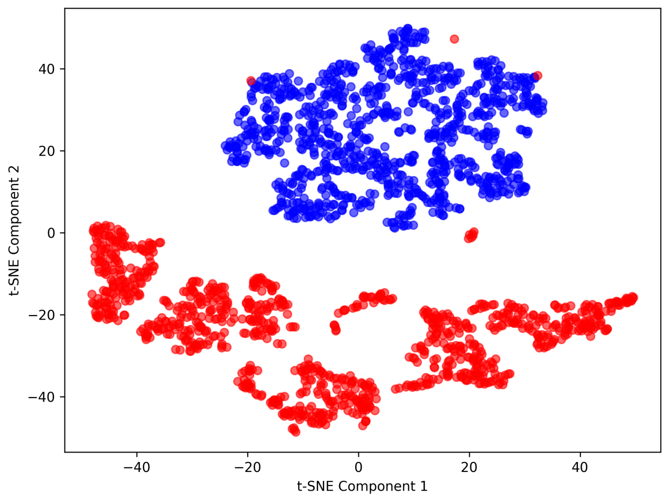

**Figure 7.** t-SNE visualization of the first 1000 samples from synthetic (TimeGAN-generated) and real sensor data.

**Table 2.** Comparison metrics between real and TimeGAN-generated time series data. DTW distances reflect temporal alignment, while the FID-like score quantifies distributional similarity.

| Feature | DTW Distance | Interpretation |
|---|---|---|
| Sensor_O3 | 7.3850 | Moderate temporal deviation |
| Temp | 6.3153 | Relatively well-aligned |
| RelHum | 7.4060 | Most divergent temporal pattern |
| Overall FID-like Score | 0.1941 | High distributional similarity |

on TimeGAN-augmented synthetic time-series data. A sharp loss reduction from 55.11 (epoch 1) to 10.93 (epoch 4) indicates rapid initial convergence, suggesting the synthetic data effectively captures core temporal patterns. After epoch 10, the loss plateaus near 5, reflecting potential limitations in data complexity or the presence of noise. Between epochs 30 and 70, slight fluctuations (3.6–4.2) may indicate overfitting or sensitivity to synthetic artifacts. However, the loss resumes a downward trend post-epoch 70, reaching 3.55 by epoch 100. This steady improvement underscores the model's capacity to learn meaningful features from synthetic data and demonstrates the viability of the TimeGAN to support downstream forecasting tasks.

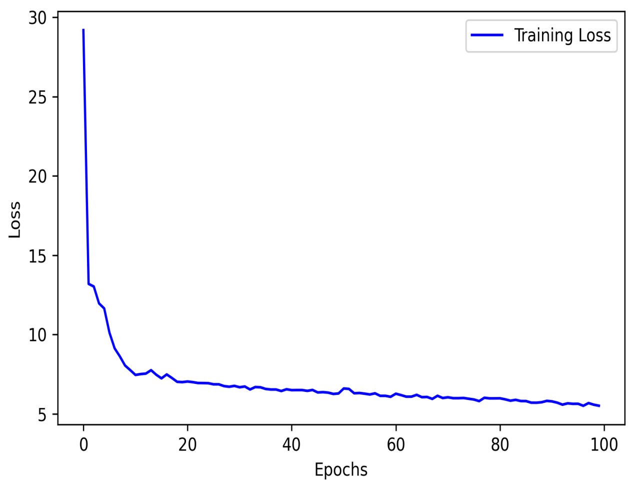

**Figure 8.** Training loss (MAE) of the TFT model trained on TimeGAN-augmented data.

The experimental results in Figure 9 demonstrate that the TimeGAN-augmented TFT model substantially outperforms all other hybrid configurations, achieving the lowest MAE (1.045), MSE (1.277), and RMSE (1.130). These results indicate superior accuracy, generalization, and robustness in handling synthetic time-series data. The TFT's performance advantage stems from its ability to leverage both temporal attention and gating mechanisms, enabling it to effectively capture long-term dependencies and contextual dynamics that simpler models might miss. In contrast, the TimeGAN-augmented LSTMAttention and CNNAttention models exhibit considerably higher error metrics, indicating that their predictions not only deviate from the ground truth but also fail to align meaningfully with the underlying data trends. The identical performance of the TimeGAN-augmented VAE and CNNAttention models suggests inefficiencies or redundancies in extracting temporal representations from the generated data. These results highlight that while the TimeGAN provides high-quality synthetic sequences, the architecture of the downstream model plays a decisive role in determining predictive performance. The better results achieved by the TFT model underscore the importance of using interpretable, attention-based frameworks capable of fully exploiting the richness of the TimeGAN's outputs—making it the most suitable choice for high-stakes time-series forecasting applications.

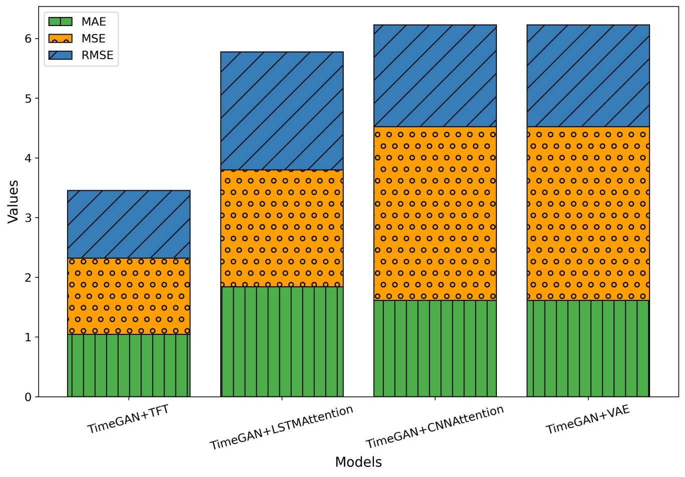

**Figure 9.** Comparison of predictive performance across TimeGAN-augmented hybrid models: TimeGAN+TFT, TimeGAN+LSTMAttention, TimeGAN+CNNAttention, and TimeGAN+VAE.

## *4.3. TimeGAN-augmented models for IIoT edge deployment*

To enable the practical deployment of data-driven models on resource-constrained IIoT edge devices, it is essential to optimize both model accuracy and computational efficiency. In this section, we evaluate the downstream regression performance of TimeGAN-augmented architectures—specifically TFT, Transformer, and GRU-based models—under three optimization strategies: standard (non-quantized), float-precision, and post-training quantization. By analyzing trade-offs in prediction accuracy, model size, and inference latency, we demonstrate the deployment potential of these models for real-time air pollution calibration tasks on low-power edge platforms.

The results depicted in Figure 10 compare the predictive performance of two TFT variants—TimeGAN+TFT (full-precision) and TimeGAN+Quantized TFT—against reference station measurements used as the ground truth for calibration. Both models exhibit a high degree of temporal alignment with the reference data across the evaluation period, capturing dynamic fluctuations and broader trends with remarkable fidelity. What is particularly noteworthy is that the quantized variant, optimized for edge deployment, demonstrates predictive accuracy nearly indistinguishable from its full-sized counterpart. This observation underscores the strength of the quantization strategy: not merely as a means of reducing the memory footprint and latency, but as a precision-preserving optimization technique. The ability of the quantized model to maintain performance parity while being tailored for constrained hardware environments highlights its viability for real-world IIoT applications, where responsiveness, efficiency, and reliability are non-negotiable. This balance of compactness and accuracy makes TimeGAN-augmented, quantized models especially compelling for scalable, real-time environmental monitoring on the edge devices.

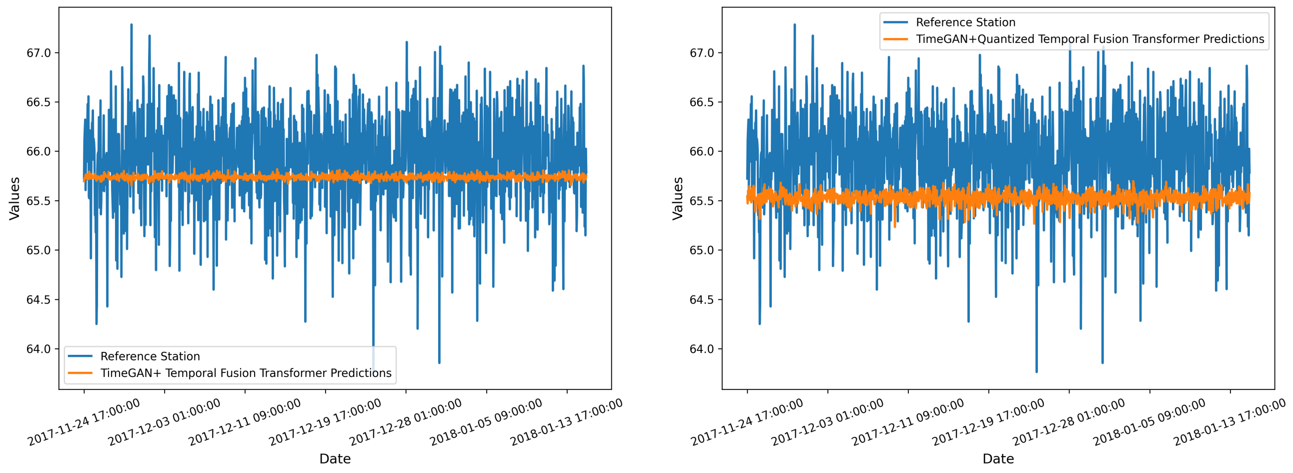

**Figure 10.** Comparison of the calibration values predicted by the TFT model and the reference station values, trained on TimeGAN-augmented data: (left) without quantization and (right) with quantization.

The results presented in Figure 11 present a comparison of the predictive performance of two variants of the GRU-Attention-based model trained on TimeGAN-augmented data: the TimeGAN+GRU Attention (non-quantized) model and its quantized counterpart, TimeGAN+Quantized GRUAttention. The comparison reveals that both models exhibit similar performance, with the TimeGAN+Quantized GRU Attention model showing only slight deviations from the TimeGAN+GRU Attention model. The predicted values from both models align closely with the reference station values across the entire observed period. The minimal difference in performance between the TimeGAN+GRU Attention and TimeGAN+Quantized GRU Attention models can be attributed to efficient weight quantization techniques that effectively reduce model size without significantly compromising predictive power. During quantization, the model's weights are mapped to lower precision, ensuring that the most important parameters are preserved. As a result, even though the quantized model uses fewer bits per weight, it still retains the essential characteristics of the original model, leading to a similar predictive performance as the floating-point model. This makes the quantized model well-suited for deployment in resource-constrained environments where computational resources are limited.

The results presented in Figure 12 demonstrate the performance and efficiency of three variants of the TimeGAN-augmented TFT: the original TimeGAN+TFT, the intermediate-precision TimeGAN+Float TFT, and the highly efficient TimeGAN+Quantized Temporal FT. Notably, the quantized model achieves competitive predictive accuracy, evidenced by the lowest MSE (0.3212), MAE (0.4375), and RMSE (0.5667), outperforming the original model (MSE: 2.4776, MAE: 1.5049, RMSE: 1.5740) and the float variant (MSE: 1.7434, MAE: 1.3204, RMSE: 1.3204). This excellent performance parity, despite significant reductions in model size, underscores the robustness of quantization techniques in preserving the predictive capabilities of complex transformer architectures

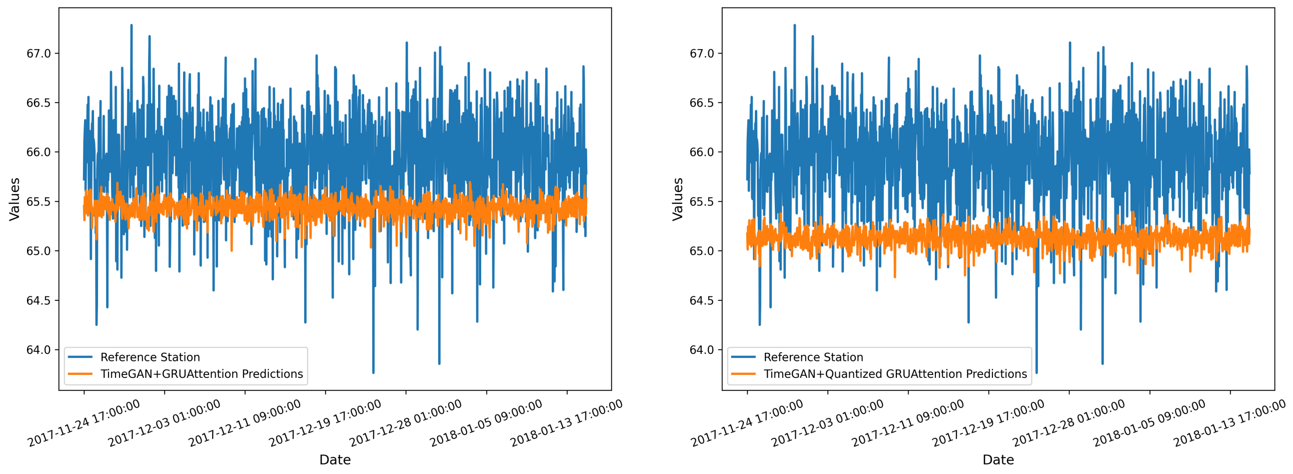

**Figure 11.** Comparison of the calibration values predicted by the GRU model and the reference station values, trained on TimeGAN-augmented data: (left) without quantization and (right) with quantization.

like TFT. Moreover, the quantized model's compact size of 97.34 KB—an 88% reduction compared to the original model (800.04 KB) and 62% smaller than the float model (259.83 KB)—makes it well-suited for deployment on resource-constrained IIoT edge devices. The float model, while larger than the quantized version, serves as a valuable intermediate solution by balancing computational efficiency with predictive accuracy.

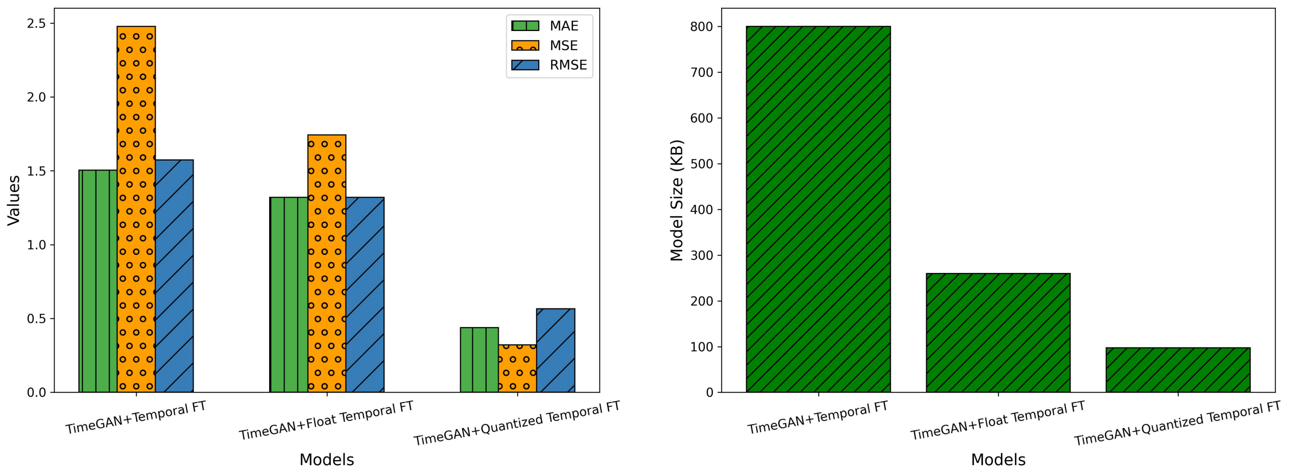

**Figure 12.** Comparison of TimeGAN-augmented TFT models: (left) performance metrics (MSE, MAE, RMSE) highlighting predictive accuracy, and (right) model size analysis showcasing the efficiency of float and quantized models for IIoT deployment.

The analysis of the results illustrated in Figure 13 highlights a trade-off between model accuracy and deployment efficiency, emphasizing the transformative potential of float and quantized models for IIoT edge deployment. The original TimeGAN+Transformer model achieves robust predictive accuracy, as evidenced by its MAE (0.3618), MSE (0.2192), and RMSE (0.4682). However, its larger model size of 56.32 KB poses challenges for deployment in resource-constrained environments typical of IIoT devices. To address these limitations, the TimeGAN+Float Transformer and TimeGAN+Quantized Transformer models offer significantly reduced sizes at 9.80 KB each, which is an approximately 82.6% reduction in model size. This compression makes these models highly suitable for edge deployment. Despite this efficiency, the Float Transformer demonstrates superior predictive performance (MSE: 0.0475, MAE: 0.2089, RMSE: 0.2180), nearly matching the original model in terms of accuracy. These results indicate that the float model retains much of the representational power of the original while being optimized for deployment. Conversely, while the QuantizedTransformer maintains the same compact size as the float model, its performance metrics (MAE: 0.5665, MSE: 0.4537, RMSE: 0.6736) suggest a noticeable decline in predictive accuracy.

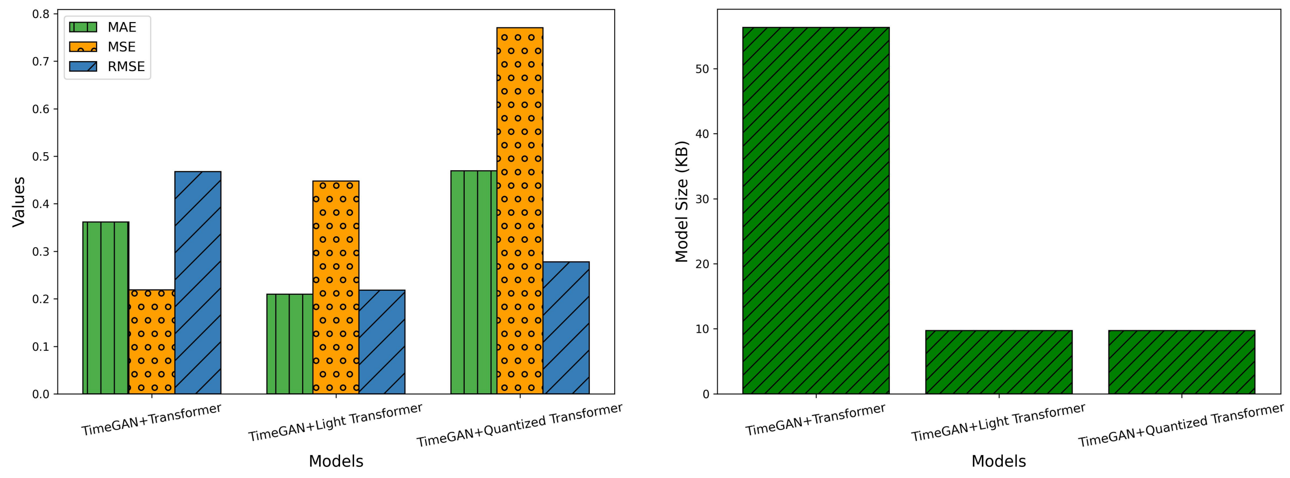

**Figure 13.** Comparison of the Transformer model trained on TimeGAN-augmented data: (left) performance metrics including MAE, MSE, and RMSE, and (right) model size analysis. The visualization highlights the trade-off between predictive accuracy and deployment efficiency, emphasizing the potential of float and quantized models for IIoT edge deployment.

The analysis of results illustrated in Figure 14 shows that the original TimeGAN+GRU model achieves an MAE of 6.8173 and an MSE of 46.6912. However, with a relatively large model size of 205.33 KB, it poses challenges for deployment in resource-constrained IIoT environments. To address this, the TimeGAN+Float GRU model reduces the model size to 106.72 KB, a 48% decrease, while maintaining similar performance (MAE: 6.8173, MSE: 46.6912). This demonstrates that the float model is highly suitable for edge deployment, offering an efficient balance of performance and model size. Further compression is achieved with the TimeGAN+Quantized GRU model, which reduces the model size to 48.21 KB, around 35% of the float model's size and only 23.5% of the original model's size. Despite this reduction in size, the quantized model retains the same MAE (6.8173) and MSE (46.6912) as the original and float models, confirming that quantization does not lead to a loss in predictive accuracy. These results suggest that both float and quantized models offer significant benefits for deployment on IIoT edge devices by reducing model size without sacrificing predictive accuracy.

The results depicted in Figure 15 reveal the performance of the TimeGAN-augmented GRU Attention model in its various forms, showcasing its ability to predict time-series data effectively. In its original form, the model achieves a commendable baseline with an MSE of 0.2952, MAE

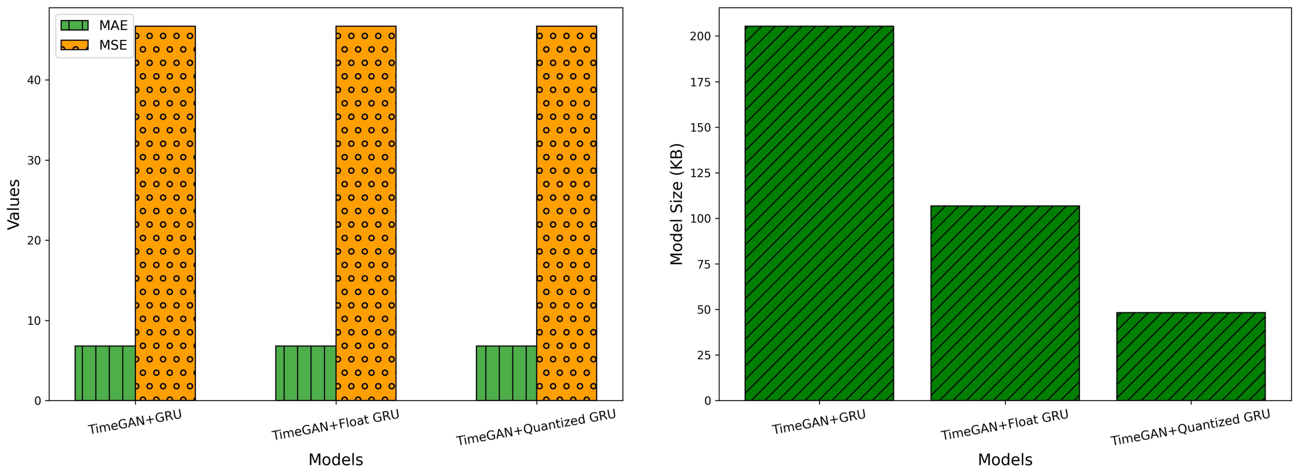

**Figure 14.** Comparison of the TimeGAN-augmented GRU model with float and quantized versions: (left) model performance metrics (MAE and MSE) and (right) model size analysis.

of 0.4171, and RMSE of 0.5433, demonstrating its capacity for accurate predictions. However, the model's substantial size of 3595.93 KB limits its applicability for edge devices with stringent memory and computational constraints, making it unsuitable for deployment in resource-constrained IIoT environments. The TimeGAN-augmented Float GRU Attention model significantly mitigates this issue by reducing the size to 1210.79 KB—approximately 66.3% smaller than the original model—while retaining identical performance metrics (MSE: 0.2952, MAE: 0.4171, RMSE: 0.5430). This reduction in size, without compromising predictive accuracy, positions the float model as a strong candidate for IIoT deployment, offering an optimal balance between computational efficiency and accuracy. On the other hand, the TimeGAN-augmented Quantized GRU Attention model further optimizes the model by reducing its size to just 356.45 KB, approximately 70% smaller than the float model. Despite the dramatic size reduction, the quantized model's performance, with an MSE of 0.3212, MAE of 0.4375, and RMSE of 0.5667, demonstrates only a slight trade-off in accuracy. This minor decrease in performance, however, remains within acceptable limits for many IIoT applications, where minimizing resource consumption is a priority. The quantized model, with its significantly reduced size, offers an ideal solution for environments where computational and memory resources are at a premium, without severely sacrificing predictive reliability.

The inference time comparison, as presented in Table 3, provides a comprehensive analysis of the suitability of various TimeGAN-augmented temporal models for real-time deployment in IIoT edge environments. In time-sensitive applications, particularly in the IIoT, low latency is essential to ensure the timely processing and delivery of predictions. Inference time, measured in milliseconds, is a critical metric in this context as it directly impacts the model's efficiency in producing predictions within the constraints of real-time systems. Among the models evaluated, the TimeGAN-augmented Float Temporal Fusion Transformer stands out with an impressive inference time of 23.4 ms, making it the fastest transformer-based model in the comparison. This performance highlights its ability to deliver nearly instantaneous predictions without sacrificing accuracy, positioning it as an ideal choice for real-time decision-making in IIoT applications, where both low latency and high predictive accuracy are paramount. The TimeGAN-augmented Quantized GRUAttention model also demonstrates a

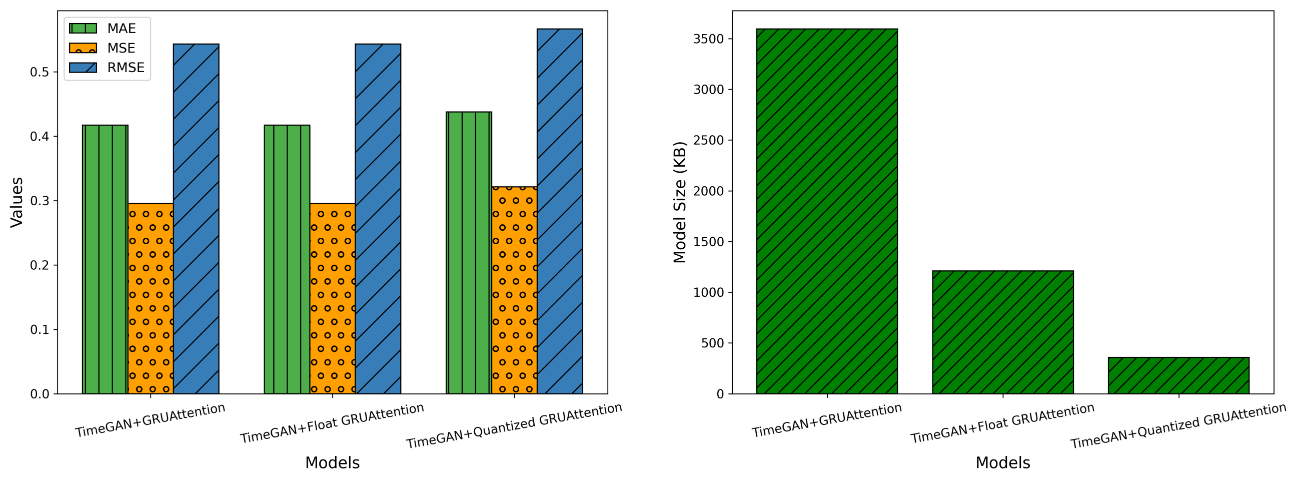

**Figure 15.** Comparison of the training loss (MAE) of the TFT model trained on TimeGAN augmented data: (left) without quantization and (right) with quantization. The graphs depict the training performance of the model over epochs.

competitive inference time of 29.1 ms, which is the fastest across all models tested. This makes it particularly well-suited for deployment in environments with limited computational resources, where speed is the critical factor. However, this comes at the cost of a slight reduction in predictive accuracy, as discussed earlier. The trade-off between speed and accuracy must be carefully considered depending on the specific requirements of the IIoT application, whether prioritizing rapid predictions or high precision. Other models, such as the TimeGAN-augmented Light GRUAttention and TimeGAN-augmented Quantized Transformer, offer moderate improvements in inference time relative to their original versions, with times of 647 ms and 125 ms, respectively. These models balance improved inference times with a reasonable level of accuracy, making them suitable for applications where there is some tolerance for latency and model complexity but still a need for faster responses than the original models could provide.

**Table 3.** Inference time comparison for TimeGAN-augmented temporal models for IIoT edge deployment.

| Temporal Models | Inference Time (ms) |
|---|---|
| TimeGAN+GRUAttention | 1340 |
| TimeGAN+Light GRUAttention | 647 |
| TimeGAN+Quantized GRUAttention | 29.1 |
| TimeGAN+Transformer | 235 |
| TimeGAN+Light Transformer | 24 |
| TimeGAN+Quantized Transformer | 125 |
| TimeGAN+TFT | 384 |
| TimeGAN+Float TFT | 23.4 |
| TimeGAN+Quantized TFT | 178 |

The analysis of TimeGAN-augmented temporal models reveals their suitability for real-time IIoT deployment, especially in applications like pollution monitoring. Key deployment factors include model accuracy, inference speed, and resource efficiency. The TimeGAN-augmented Quantized TFT model excels by reducing its size by 88% (from 800.04 KB to 97.34 KB) while maintaining strong accuracy (MSE: 0.3212, MAE: 0.4375). This makes it ideal for resource-constrained devices in IIoT environments, offering a compact yet highly efficient solution for real-time pollution monitoring. The TimeGAN-augmented Float TFT offers an optimal balance of speed and accuracy, with an inference time of 23.4 ms, the fastest among transformer-based models. Its MSE (1.7434) and MAE (1.3204) remain strong, ensuring both low latency and high predictive accuracy for real-time decision-making. The TimeGAN-augmented Quantized GRUAttention, while the fastest (29.1 ms), sacrifices slightly in accuracy (MSE: 0.3212, MAE: 0.4375). However, its minimal size (48.21 KB) makes it highly efficient for edge devices where speed is critical, such as rapid anomaly detection in pollution levels. These findings highlight that TimeGAN-augmented models offer robust performance, with each variant optimized for specific IIoT deployment needs, balancing accuracy, speed, and resource efficiency for real-time environmental monitoring.

## 5. Conclusion

In this work, we introduce a generative temporal approach for calibrating air pollution sensor networks through the integration of the TimeGAN and TFT models, specifically designed for deployment on resource-constrained edge devices in IIoT environments. By leveraging the TimeGAN for synthetic data generation and TFT for accurate predictions, our approach effectively calibrates sensors, even in settings with limited high-quality data, ensuring reliable real-time operation. The synthetic data produced by the TimeGAN mirrors the temporal characteristics of real sensor data, making it a valuable tool for sensor calibration in challenging real-world scenarios. A detailed evaluation of the TimeGAN-augmented temporal models reveals crucial insights into their suitability for IIoT applications. The key considerations for effective deployment in these environments—accuracy, inference time, and resource efficiency—are addressed through our models. Specifically, the TimeGAN-augmented Quantized TFT model excels in balancing high predictive accuracy and minimal resource consumption, reducing the model size by 88% (from 800.04 KB to 97.34 KB) without compromising performance (MSE: 0.3212, MAE: 0.4375). This drastic reduction in size makes it highly suitable for edge devices with limited computational resources, ensuring accurate, real-time air quality monitoring. Furthermore, the synthetic data generated by the TimeGAN aligns well with the real sensor data, as evidenced by low DTW distances and high distributional similarity, confirming the effectiveness of the TimeGAN in generating realistic data for calibration. On the other hand, the TimeGAN-augmented Float TFT model offers an optimal compromise between speed and accuracy. With a fast inference time of 23.4 ms, it stands out among transformer-based models, making it ideal for applications where low-latency, real-time decision-making is paramount. While slightly less efficient in terms of model size (MSE: 1.7434, MAE: 1.3204), it guarantees timely, precise predictions—crucial for dynamic pollution monitoring systems. The Float TFT model also benefits from synthetic data that closely tracks real-world trends, ensuring that its predictions remain robust despite the potential for data scarcity. These findings underline the potential of TimeGAN-augmented models to provide both high accuracy and operational efficiency in IIoT-based pollution monitoring, especially in edge device deployment within real-time, resource-constrained environments.

## Use of AI tools declaration

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

## Author Contributions

Shagufta Henna (first author) contributed significantly to the research, taking the lead in the overall study conception, methodology development, model implementation, and experimentation. Mohammad Amjath (second author) was responsible for the literature investigation, design aspects of the model, and contributing to the overall structure of the study. Asif Yar (third author) assisted primarily with the review process and helped with the overall design aspects of the research.

## Conflict of interest

The authors declare that they have no competing interests.

## References

1. Yar A, Henna S, McAfee M, et al. (2023) Air Pollution Monitoring Using Online Recurrent Extreme Learning Machine. *31st Irish Conference on Artificial Intelligence and Cognitive Science (AICS)* 2023: 1–6. https://doi.org/10.1109/AICS60730.2023.10470534
2. Alsamrai O, Redel-Macias MD, Pinzi S, et al. (2024) A systematic review for indoor and outdoor air pollution monitoring systems based on Internet of Things. *Sustainability* 16: 4353. https://doi.org/10.3390/su16114353
3. Castell N, Dauge FR, Schneider P, et al. (2017) Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates? *Environ Int* 99: 293–302. https://doi.org/10.1016/j.envint.2016.12.007
4. Wang Z, Zhang Z, Wang Z, et al. (2024) A novel intelligent indoor fire and combustibles detection method based on multi-channel transfer learning strategy with acoustic signals. *Process Safety and Environmental Protection* 189: 1217–1225. https://doi.org/10.1016/j.psep.2024.06.020
5. Henna S, Yar A, Saheed K, et al. (2023) Wireless Sensor Networks Calibration using Attention-based Gated Recurrent Units for Air Pollution Monitoring. *IEEE International Conference on Big Data (BigData)* 2023: 3779–3784. https://doi.org/10.1109/BigData59044.2023.10386318
6. Maag B, Zhou Z, Saukh O, et al. (2017) SCAN: Multi-Hop Calibration for Mobile Sensor Arrays. *ACM Trans Sens Netw* 1. https://doi.org/10.1145/3090084
7. Feng H, Xu C, Jin B, et al. (2024) A Deployment Optimization for Wireless Sensor Networks Based on Stacked Auto Encoder and Probabilistic Neural Network. *Digital Communications and Networks* https://doi.org/10.1016/j.dcan.2024.06.003
8. Yar A, Henna S, McAfee M, et al. (2023) Extreme Learning Machines for Calibration and Prediction in Wireless Sensor Networks: Advancing Environmental Monitoring Efficiency.

*31st Irish Conference on Artificial Intelligence and Cognitive Science (AICS)* 2023: 1–4. https://doi.org/10.1109/AICS60730.2023.10470795

9. Singh A, Chatterjee K, Satapathy SC (2022) An edge based hybrid intrusion detection framework for mobile edge computing. *Complex Intell Syst* 8: 3719–3746. https://doi.org/10.1007/s40747-021-00498-4

10. Idrissi I, Azizi M, Moussaoui O (2022) A Lightweight Optimized Deep Learning-based Host-Intrusion Detection System Deployed on the Edge for IoT. *International Journal of Computing and Digital Systems* 11: 209–216. https://doi.org/10.12785/ijcds/110117

11. Yandouzi M, Grari M, Idrissi I, et al. (2022) Review on forest fires detection and prediction using deep learning and drones. *Journal of Theoretical and Applied Information Technology* 100: 4565–4576.

12. Abusitta A, Carvalho GHS, Wahab OA, et al. (2022) Deep learning-enabled anomaly detection for IoT systems. *Internet Things* 21: 100656. https://api.semanticscholar.org/CorpusID:253431423

13. Aversano L, Bernardi ML, Cimitile M, et al. (2021) Effective anomaly detection using deep learning in IoT systems. *Wireless Communications and Mobile Computing* 2021: 9054336. https://doi.org/10.1155/2021/9054336

14. Ahmad Z, Shahid Khan A, Nisar K, et al. (2021) Anomaly Detection Using Deep Neural Network for IoT Architecture. *Applied Sciences* 11: 7050. https://doi.org/10.3390/app11157050

15. Konaite M, Owolawi PA, Mapayi T, et al. (2021) Smart hat for the blind with real-time object detection using Raspberry Pi and TensorFlow Lite. *Proceedings of the International Conference on Artificial Intelligence and its Applications* ISBN 9781450385756. https://doi.org/10.1145/3487923.3487929

16. Sharma K, Eskicioglu R (2022) Deep learning-based ECG classification on Raspberry Pi using a TensorFlow Lite model based on PTB-XL dataset. *International Journal of Artificial Intelligence & Applications* 13: 55–66. https://doi.org/10.5121/ijaia.2022.1340455

17. Rokh B, Azarpeyvand A, Khanteymoori A (2023) A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Computing Surveys* 14. https://doi.org/10.1145/3623402

18. Phuong M, Lampert C (2019) Towards understanding knowledge distillation. *Proceedings of the 36th International Conference on Machine Learning* 97: 5142–5151. PMLR. https://proceedings.mlr.press/v97/phuong19a.html

19. Yang L, He Z, Fan D (2020) Harmonious coexistence of structured weight pruning and ternarization for deep neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 34: 6623–6630. https://doi.org/10.1609/aaai.v34i04.6138

20. TensorFlow Lite. Ultralytics, 2024. *Available from:* https://developers.googleblog.com/en/tensorflow-lite-is-now-litert

21. Sun C, Li J, Sulaiman R, et al. (2023) Air Quality Prediction and Multi-Task Offloading based on Deep Learning Methods in Edge Computing. *Journal of Grid Computing* 21. https://doi.org/10.1007/s10723-023-09671-0

22. Yoon J, Jarrett D, van der Schaar M (2019) Time-series Generative Adversarial Networks. In: *Neural Information Processing Systems*, 2019. https://proceedings.neurips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf

23. Kursa MB, Jankowski A, Rudnicki WR (2010) Boruta – A System for Feature Selection. *Fundamenta Informaticae* 101: 271–285. https://doi.org/10.3233/FI-2010-288

24. Moursi AS, El-Fishawy N, Djahel S, et al. (2021) An IoT enabled system for enhanced air quality monitoring and prediction on the edge. *Complex and Intelligent Systems* 7: 2923–2947. https://doi.org/10.1007/s40747-021-00476-w

25. Felici-Castell S, Segura-Garcia J, Perez-Solano JJ, et al. (2023) AI-IoT low-cost pollution-monitoring sensor network to assist citizens with respiratory problems. *Sensors* 23. https://doi.org/10.3390/s23239585

26. Li S, Jin X, Xuan Y, et al. (2019) Enhancing the locality and breaking the memory bottleneck of Transformer on time series forecasting. *ArXiv* abs/1907.00235. https://api.semanticscholar.org/CorpusID:195766887

27. Gong L, Chen Y (2024) Machine learning-enhanced IoT and wireless sensor networks for predictive analysis and maintenance in wind turbine systems. *Int J Intell Netw* 5: 133–144. https://doi.org/10.1016/j.ijin.2024.02.002

28. Aggarwal A, Toshniwal D (2021) A hybrid deep learning framework for urban air quality forecasting. *J Clean Prod* 329: Article ID 129660. https://doi.org/10.1016/j.jclepro.2021.129660

29. Wardana INK, Gardner JW, Fahmy SA (2021) Optimising deep learning at the edge for accurate hourly air quality prediction. *Sensors (Switzerland)* 21: 1–28. https://doi.org/10.3390/s21041064

30. Hu Y, Cao N, Guo W, et al. (2024) FedDeep: A federated deep learning network for edge-assisted multi-urban PM2.5 forecasting. *Appl Sci (Switzerland)* 14: Article ID 1979. https://doi.org/10.3390/app14051979

31. Koziel S, Pietrenko-Dabrowska A, Wojcikowski M, et al. (2024) High-performance machine-learning-based calibration of low-cost nitrogen dioxide sensor using environmental parameter differentials and global data scaling. *Sci Rep* 14: 26120. https://doi.org/10.1038/s41598-024-77214-y

32. Yu H, Li Q, Geng YA, et al. (2020) AirNet: A calibration model for low-cost air monitoring sensors using dual sequence encoder networks. *AAAI* 34: Article ID 5464. https://doi.org/10.1609/aaai.v34i01.5464

33. Yar A, Henna S, McAfee M, et al. (2024) Accelerating deep learning for self-calibration in large-scale uncontrolled wireless sensor networks for environmental monitoring. *Proc 35th Irish Syst Signals Conf (ISSC 2024)*, Article ID 10603082. https://doi.org/10.1109/ISSC61953.2024.10603082

34. Schmitz S, Towers S, Villena G, et al. (2021) Unravelling a black box: An open-source methodology for the field calibration of small air quality sensors. *Atmos Meas Tech* 14: 7221–7241. https://doi.org/10.5194/amt-14-7221-2021

35. Wang G, Yu C, Guo K, et al. (2024) Research of low-cost air quality monitoring models with different machine learning algorithms. *Atmos Meas Tech* 17: 181–196. https://doi.org/10.5194/amt-17-181-2024
36. Rahardja U, Aini Q, Manongga D, et al.(2023) Enhancing machine learning with low-cost PM 2.5 air quality sensor calibration using image processing. *APTSI Trans Manag (ATM)* 7: 194–202. https://doi.org/10.33050/atm.v7i3.2062
37. Price I, Sanchez-Gonzalez A, Alet F, et al. (2024) Probabilistic weather forecasting with machine learning. *Nature* https://doi.org/10.1038/s41586-024-08252-9
38. Li L, Carver R, Lopez-Gomez I, et al. (2023) SEEDS: Emulation of weather forecast ensembles with diffusion models. *ArXiv* abs/2306.14066. https://api.semanticscholar.org/CorpusID:259252403
39. Wang Z, Chen C, Zeng Y, et al. (2023) Where did I come from? Origin attribution of AI-generated images. *Advances in Neural Information Processing Systems* 36: 74478–74500. https://proceedings.neurips.cc/paper_files/paper/2023/file/ebb4c188fafe7da089b41a9f615ad84d-Paper-Conference.pdf
40. Tan K, Chen J, Wang D (2019) Gated Residual Networks with Dilated Convolutions for Monaural Speech Enhancement. *IEEE/ACM Trans Audio Speech Lang Process* 27: 189–198. https://doi.org/10.1109/TASLP.2018.2876171
41. Barcelo-Ordinas JM, Ferrer-Cid P, Garcia-Vidal J, et al. (2021) H2020 project CAPTOR dataset: Raw data collected by low-cost MOX ozone sensors in a real air pollution monitoring network. *Data in Brief* 36: 107127. https://doi.org/10.1016/j.dib.2021.107127
42. Barcelo-Ordinas JM, Ferrer-Cid P, Garcia-Vidal J, et al. (2019) Distributed multi-scale calibration of low-cost ozone sensors in wireless sensor networks. *Sensors* 19:2503. https://doi.org/10.3390/s19112503
43. Gonsek A, Jeschke M, Rönnau S, et al. (2021) From Paths to Routes: A Method for Path Classification. *Frontiers in Behavioral Neuroscience* 14:610560. 10.3389/fnbeh.2020.610560
44. Bai S, Kolter JZ, Koltun V (2018) An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *International Conference on Machine Learning (ICML).*
45. Lea C, Flynn MD, Vidal R, et al. (2017) Temporal Convolutional Networks for Action Segmentation and Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*
46. Oord AV, Dieleman S, Zen H,et al. (2016) WaveNet: A Generative Model for Raw Audio. *arXiv preprint arXiv:1609.03499*
47. Chung J, Kastner K, Dinh L, et al. (2015) Recurrent Latent Variable Models for Sequential Data. *Advances in Neural Information Processing Systems (NeurIPS).*
48. Fraccaro M, Sønderby SK, Paquet U, et al. (2016) Sequential Neural Models with Stochastic Layers. *Advances in Neural Information Processing Systems (NeurIPS).*
49. Walker J, Doersch C, Gupta A, et al. (2016) The Uncertainty in Action: Unsupervised Action Prediction with Variational Autoencoders. *European Conference on Computer Vision (ECCV).*

## Appendix

**Table 4.** Summary of limitations across different CNN and VAE variants when applied to time-series data for IIoT applications. The table highlights challenges such as the inability to model long-term temporal dependencies, overfitting, handling complex non-stationary patterns, and preserving fine-grained temporal information in sequential data.

| Model | Limitation |
|---|---|
| CNN [44] | Captures local spatial patterns; lacks temporal modeling across long horizons; prone to overfitting on small datasets |
| Temporal CNN (TCN) [45] | Improves temporal range but struggles with complex long-term dependencies under limited receptive fields |
| Dilated CNN [46] | Expands temporal receptive field but can introduce gridding artifacts, harming fine-grained temporal modeling |
| VAE [47] | Global latent focus; loses temporal granularity in sequential reconstruction; overfits when not regularized properly |
| Sequential VAE (SVAE) [48] | Introduces time dependencies but still underperforms on complex non-stationary temporal patterns |
| Conditional VAE (CVAE) [49] | Captures conditional structure but struggles with modeling long-term temporal correlations; may overfit when conditioned on insufficient data |