Real-Time Calibration Model for Low-Cost Sensor in Fine-Grained Time Series

Seokho Ahn[1], Hyungjin Kim[1], Sungbok Shin[2*], Young-Duk Seo[1†]

[1]Department of Electrical and Computer Engineering, Inha University, Incheon 22212, South Korea
[2]Team Aviz, Inria, Université Paris-Saclay, Saclay, France
sokho0514@inha.edu, flslzk@inha.edu, sungbok.shin@inria.fr, mysid88@inha.ac.kr

Abstract

Precise measurements from sensors are crucial, but data is usually collected from low-cost, low-tech systems, which are often inaccurate. Thus, they require further calibrations. To that end, we first identify three requirements for effective calibration under practical low-tech sensor conditions. Based on the requirements, we develop a model called **TESLA**, **T**ransformer for **e**ffective **s**ensor calibration utilizing **l**ogarithmic-binned **a**ttention. TESLA uses a high-performance deep learning model, Transformers, to calibrate and capture non-linear components. At its core, it employs *logarithmic binning* to minimize attention complexity. TESLA achieves consistent real-time calibration, even with longer sequences and finer-grained time series in hardware-constrained systems. Experiments show that TESLA outperforms existing novel deep learning and newly crafted linear models in accuracy, calibration speed, and energy efficiency.

1 Introduction

Externalizing data latent in our world (Elmqvist 2023) can increase our awareness of previously unknown situations. For example, air pollution is a pervasive issue often overlooked due to its invisibility, and increased awareness can help us avoid potential dangers. Sensors from various IoT devices are used to capture real-time information from our surroundings. To take well-informed, suitable actions about our environments, it is integral to obtain accurate data from sensors. But oftentimes, these sensors exist in the form of low-spec computing devices, and these devices are sometimes not accurate (Ray 2022). To address this matter, there have been attempts to improve the quality of data coming from these low-tech devices using naïve linear- and machine learning-based calibration methods (Concas et al. 2021; Aula et al. 2022; Villanueva et al. 2023).

We identify three key requirements that must be addressed to develop an effective, and practical calibration model: (1) handling fine-grained sensor type, (2) ensuring consistency in real-time calibration, and (3) accommodating hardware constraints (Section 3). Recent research has utilized time series prediction for calibrating low-cost sensors, but practical IoT systems face limitations in their effective use. Researchers have adopted time series prediction methods for low-cost sensor calibration (Zhang et al. 2023; Ahn et al. 2024; Narayana et al. 2024) as both rely on similar time series regression models (Narayana et al. 2024; Toner and Darlow 2024). This, together with deep learning and linear-based approaches, has led to improved calibration accuracy, but there is a tradeoff — an improvement in one factor leads to a setback in others (Zhang et al. 2023; Ahn et al. 2024; Narayana et al. 2024). Addressing all three of the factors mentioned above in tandem is not a trivial issue.

To bridge this research gap, we propose TESLA (**T**ransformer for **e**ffective **s**ensor calibration utilizing **l**ogarithmic-binned **a**ttention) to address all of these challenges for IoT systems. TESLA reduces the attention bottleneck in transformers through logarithmic binning (Section 4.3), which couples more past tokens while retaining recent ones in a log scale. TESLA also employs multi-view embedding (Section 4.2) and feature-wise aggregation (Section 4.4) to preserve both local and global time series patterns, using only a single sensor. TESLA operates as quickly as linear-based models and can be integrated into low-tech systems, offering higher accuracy than deep learning-based models to balance the tradeoffs. Experimental results demonstrate that TESLA successfully achieves this balance, efficiently managing calibration speed, energy usage, and accuracy.

To sum up, our contributions are:

- We identify three challenges that have to be addressed to achieve accuracy on par with high-quality sensors within practical IoT systems.
- We propose a model called TESLA to address all of these challenges for low-cost sensor calibration.
- Experiments with real-world benchmarks show that TESLA outperforms the baseline deep learning and newly designed linear models in most cases.

2 Related Works

We present the related work from two perspectives: low-cost sensor calibration and short-term time series prediction.

*The work was done while the author was affiliated with the University of Maryland, College Park.

†Corresponding author.

2.1 Low-Cost Sensor Calibration

Existing works on low-cost sensors tackle specific challenges, but their efficacy is limited as performance improvements come with setbacks. We present these calibration trade-offs from two groups: (1) Machine learning-based and statistical models, and (2) deep learning-based models.

Various machine learning-based and statistical approaches such as linear regression, random forests, and ARIMA-based models have been used to calibrate low-cost sensors (Concas et al. 2021; Aula et al. 2022; Villanueva et al. 2023). In particular, linear regression has been widely used due to its simplicity, but it has often struggled to capture nonlinear trends in time series (Concas et al. 2021; Patton et al. 2022).

Recent advances in deep learning have introduced various data-driven calibration methods, such as long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997)-based (Nath et al. 2021; Ahn et al. 2024; Apostolopoulos, Fouskas, and Pandis 2023), convolutional neural networks (CNNs) (Krizhevsky, Sutskever, and Hinton 2012)-based (Ali et al. 2023), or Transformer (Vaswani et al. 2017)-based methods (Narayana et al. 2024; Ahn et al. 2024). However, training these models usually demands significant computing power, posing challenges for practical use in IoT-controlled systems (Nath et al. 2021). For example, the deployment of Transformers in everyday home devices is limited due to their high resource demands. This highlights a significant gap in the practical application of advanced models in resource-constrained environments.

Our contributions. TESLA meets all three requirements: it achieves high accuracy with deep learning methods and fastens inference speeds by partially implementing linear methods, whereas existing works mainly address one/two of the three issues, at the expense of other requirements.

2.2 Short-Term Time Series Forecasting

Even though they handle similar tasks, time series forecasting models can not be used directly for calibration, as they do not meet the needs of practical calibration models. Here, we describe two types of forecasters: (1) Transformer-based and (2) linear-based forecasters.

The effectiveness of Transformers has been questioned by the advent of linear forecasters (Zeng et al. 2023), but recent models like PatchTST (Nie et al. 2023) and iTransformer (Liu et al. 2024) have demonstrated that Transformers can effectively capture the time series patterns when timely used. Despite these advances, the complexity of Transformer-based models still poses challenges for practical IoT systems. Although iTransformer (Liu et al. 2024) is the most practical option, calibrating it with a single sensor is ineffective as it is specifically designed for multivariate time series.

Linear forecasters are mainly advantageous for their computational efficiency and low energy use. This makes them ideal for IoT systems. However, these models have not yet addressed two critical issues when applied as calibration models: (i) Data collected from various sensors may be inaccurate due to its non-linear characteristics, which are more complex than in time series prediction (Concas et al. 2021; Patton et al. 2022); (ii) Their ability to handle large datasets in practical applications remains unproven due to the small size of the parameters. These limitations must be resolved for successful implementation in practical IoT systems.

Our contributions. In contrast to many Transformer-based models, TESLA is tailored for fast and efficient calibration of a single time series, typical for most low-cost sensors. Furthermore, TESLA merges the strengths of both forecasters to achieve high speed (comparable to linear models) and high accuracy with limited resources.

3 Preliminaries

We define fine-grained time series and sensor calibration model. We then discuss additional requirements when applying sensor calibration models to real-world scenarios.

3.1 Fine-Grained Time Series

We first define the concept of fine-grained time series.

Definition 1 (Fine-grained time series)**.** Consider a sensor $\mathcal{X}$ as a function $\mathcal{X} : T \to \mathbb{R}^+$, where T is a discrete set of measurement times. Then $x_i = \mathcal{X}(t_i)$ represents the i_{th} sensor reading at time $t_i \in T$. Given a window size N, an i_{th} *fine-grained time series* $\mathcal{S}_i$ generated by sensor $\mathcal{X}$ is a sequence $\mathcal{S}_i = (x_{i-N+1}, \cdots, x_i)$ for $i \geq N$, under the condition that the granularity $\mu_{\mathcal{X}} = \min_{i \geq N} \{t_i - t_{i-1}\}$ of the sensor $\mathcal{X}$ is sufficiently small. For notational convenience, we regard the time series $\mathcal{S}_i$ as a fine-grained time series unless otherwise noted. [1]

After observing various time series benchmarks (e.g., ETT, traffic, weather, etc.), we have noted a growing demand for finer time series granularity, down to minute- or second-level intervals. Calibration, in particular, demands finer granularity than most other forecasting tasks. Air quality sensors equipped in home appliances must reflect dynamic, ad hoc environmental changes caused by several activities such as cooking, exhaust emissions, or smoke. For these applications, it is crucial to provide quick and accurate calibration results based on the recent sensor readings. Our research thus focuses on these fine-grained time series.

3.2 Sensor Calibration Model

Here too, we start by defining sensor calibration model.

Definition 2 (Sensor calibration model)**.** Consider two sensors performing the same task $\mathcal{X}$ and $\mathcal{Y}$, where sensor $\mathcal{X}$ is less accurate than $\mathcal{Y}$. Then a *sensor calibration model* $\mathcal{F}_N(*|\Theta_{\text{opt}})$ for sensor $\mathcal{X}$ is a deep learning model, where the learnable parameters Θ_{opt} of the model $\mathcal{F}_N$ is optimized by the following formula with an objective function $\mathcal{L}$:

$$\Theta_{\text{opt}} = \underset{\Theta \in \Omega}{\arg\min} \sum_{i \in I_{\text{train}}} \mathcal{L}\left(\mathcal{F}_N(\mathcal{S}_i|\Theta), y_i\right) \tag{1}$$

[1] Note that sensor sampling period can be delayed by several factors, such as sensor connection errors, invalid readings (*e.g.*, null value), or valid readings that should be truncated (*e.g.*, outliers). Hence, we define granularity as a minimum value of the sampling interval to ensure a consistent understanding of the granularity.

where Ω is the total parameter space of the model $\mathcal{F}_N$ and $I_{\text{train}} \subseteq \mathbb{N}$ denotes the training index set. [2]

Calibration tasks are better served by referencing external time series rather than relying solely on data from low-cost sensors. This is primarily because the data distributions from low-cost sensors and reference sensors can differ significantly. Hence, generating labels from a low-cost sensor itself may restrict their potential accuracy. In such scenarios, employing a high-quality sensor as a reference, such as those used in air quality monitoring stations, provides more reliable calibration results.

However, optimizing learnable parameters is insufficient when designing a sensor calibration model. IoT devices equipped with sensors often have insufficient hardware resources, which limits the application of high-computational calibration models. This necessitates the need to satisfy requirements that match the capabilities of IoT devices, which we describe in the next subsection.

3.3 Model Hardware Requirements

This section describes three key factors that are required to develop an effective consistent, real-time calibration model for practical IoT scenarios:

- **Accuracy.** Accurate data reading from sensors is crucial. Commonly used evaluation metrics include mean absolute error (MAE) and root mean squared error (RMSE) to measure this accuracy. However, in practical IoT systems, real-time performance matters, and constraints on IoT hardware may prevent the deployment of high-performance models.
- **Latency.** For finer-grained time series, calibration must be conducted swiftly, within smaller intervals. We can evaluate this latency using speed-related metrics, such as inference speed, floating point operations (FLOPS), or throughput. It is not necessary to choose the fastest calibration model; instead, the ideal choice would be a model that achieves the optimal performance given the situation.
- **Hardware resources.** Considering hardware resources is crucial when determining the applicability of a calibration model. This involves evaluating model parameters to estimate hardware fixed memory requirements, memory footprints to estimate variable memory for space complexity, and calculating FLOPS to determine time complexity.

In summary, evaluating the superiority of a calibration model requires considering both its accuracy and the additional factors we discussed. The ideal calibration model should exhibit the high accuracy of a deep learning model while maintaining the high speed and low energy consumption of linear models. In the next section, we propose TESLA, which aims to achieve this ultimate goal and will be detailed further. Then, in Section 5, we carefully select evaluation metrics that account for all these factors.

[2]The goal of the sensor calibration model $\mathcal{F}_N$ is to calibrate sensor reading x_i using the sequence $\mathcal{S}_i$, minimizing the difference between the predicted value $\hat{x}_i = \mathcal{F}_N(\mathcal{S}_i|\Theta)$ and the reference reading $y_i = \mathcal{Y}(t_i)$.

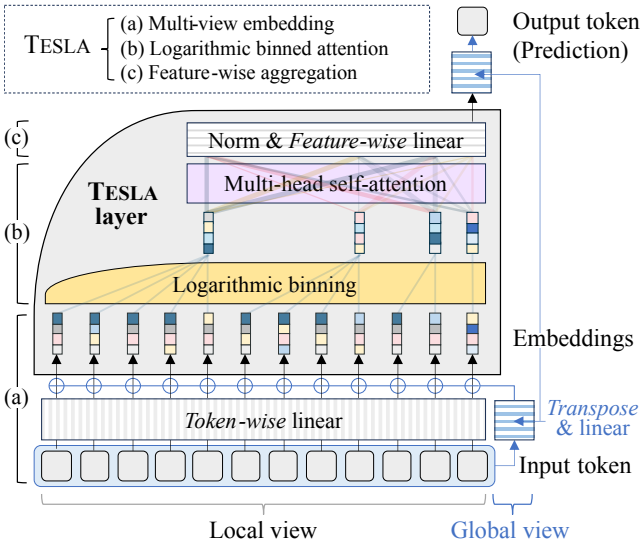


Figure 1: Overview of TESLA, which consists of (a) multi-view embedding, (b) logarithmic binned attention, and (c) feature-wise aggregation in data process order.

4 Sensor Calibration Model: TESLA

This section describes our Transformer architecture for time series calibration for practical IoT systems, named **TESLA** (**T**ransformer for **e**ffective **s**ensor calibration utilizing **l**ogarithmic-binned **a**ttention).

4.1 Model Overview

Figure 1 shows the overall structure of TESLA. From the perspective of a black box model, TESLA takes the time series $\mathcal{S}_i = (x_{i-N+1}, \cdots, x_i)$ as input and finally returns the calibration result $\hat{x}_i$ as defined in the calibration model described in Section 3.2. Our technical contribution lies in designing a novel architecture that meets the three hardware requirements mentioned in Section 3.3, while still maintaining high performance through the use of Transformers.

To begin with, TESLA uses multi-view embedding methods to input tokens to model both global and local features (Section 4.2). This strategy effectively captures changes in fine-grained time series values while reducing the number of tokens. Second, we propose a novel method called logarithmic binning, aggregating a number of past tokens while preserving the detail of recent ones on a logarithmic scale (Section 4.3). Logarithmic binning achieves significant $\mathcal{O}(\log^2 N)$ self-attention complexity. It effectively alleviates the bottleneck problem of attention operations, making them applicable to IoT environments. Finally, we replace the token-wise feedforward network with a linear layer for feature-wise aggregation, which significantly reduces the number of learnable parameters and the computational power required (Section 4.4). Detailed explanations are described in subsequent subsections.

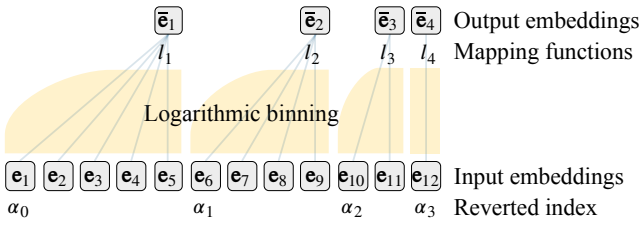


Figure 2: Illustrative example of logarithmic binning in case $N = 12$. Then we have $z = \lceil \log_2 12 \rceil = 4$ with reverted indices $(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4) = (1, 6, 10, 12, 13)$. $l_1, \cdots, l_4$ are mapping functions that satisfy Equation 3.

4.2 Multi-View Embedding

We design an effective embedding method, even in the use of a single low-cost sensor. Our method is designed to consider multi-view data using simpler techniques, despite using only a univariate time series from a single sensor. Relying solely on local token-wise embeddings, which are typical in Transformers, is ineffective for calibration because the receptive field is not large enough to effectively represent the time series. Hence, we add a global representation inspired by Liu et al. to preserve both local and global features. This approach supplements implicit, local positional information within the global representation, removing the need for extra positional or temporal embeddings.

Formally, we first omit the index i and re-index the series as $\mathcal{S} = (x_1, \cdots, x_N) \in \mathbb{R}^{1\times N}$ (*i.e.*, we shift the index by $N - i$) for notational convenience. Then the input representation of TESLA for a time series $\mathcal{S}$ is defined as $\mathbf{E} = (\mathbf{e}_1, \cdots, \mathbf{e}_N)^T \in \mathbb{R}^{N\times d}$, where:

$$\mathbf{e}_i = x_i \mathbf{W}_{\text{local}} + \mathcal{S}\mathbf{W}_{\text{global}} \tag{2}$$

is a d-dimensional column vector for $i \in [1, N]$. $\mathbf{W}_{\text{local}} \in \mathbb{R}^{1\times d}$ and $\mathbf{W}_{\text{global}} \in \mathbb{R}^{N\times d}$ denote local and global learnable parameters, respectively.

4.3 Logarithmic Binned Attention

To alleviate the costly computation of Transformers, we deploy a core method called logarithmic binning. This reduces the number of tokens to approximately $\log_2 N$. Then we apply multi-head attention to these reduced tokens, enabling more efficient and rapid computations.

Logarithmic binning. We first define *logarithmic binning* as a collection of the mapping functions $\{l_1, \cdots, l_z\}$ for $z = \lceil \log_2 N \rceil$, where each function satisfies:

$$l_j : \mathbb{R}^{d\times(\alpha_j - \alpha_{j-1})} \to \mathbb{R}^{d\times 1} \tag{3}$$

where $\alpha_j = \max\left\{1, N - 2^{z-j} + 3\right\}$ for integer $j \in [0, z]$. We provide an illustrative example of logarithmic binning including α_j's and l_j's in Figure 2 for comprehensive understanding. Intuitively, logarithmic binning is a strategy that keeps the latest token while a series of past tokens are grouped in reverse temporal order on a logarithmic scale. Using logarithmic binning, the sequence length is reduced from N to nearly $\log_2 N$, enabling cost-efficient attention complexity $\mathcal{O}(\log_2^2 N) = \mathcal{O}(\log^2 N)$. Furthermore, the recent trend is emphasized, since a large part of recently inputted information is kept in the bin. Emphasizing these trends helps to better understand the recent sensor dynamics in long, finer-grained time series, leading to more accurate calibration results.

Logarithmic binned embedding. To reduce the number of tokens to reduce attention complexity, we then create an embedding by applying logarithmic binning to input embedding. Using the previous collection of the mapping functions $\{l_1, \cdots, l_z\}$, we define the logarithmic binned embedding of $\mathbf{E}$ as $\overline{\mathbf{E}} = (\overline{\mathbf{e}}_1, \cdots, \overline{\mathbf{e}}_z)^T \in \mathbb{R}^{z\times d}$, which is calculated by:

$$\overline{\mathbf{e}}_j = l_j\left(\left(\mathbf{e}_{\alpha_{j-1}}, \cdots, \mathbf{e}_{\alpha_j - 1}\right)\right). \tag{4}$$

The most straightforward mapping function we can think of is the average function, which simply applies an average to each vector for binning. However, this method not only is incapable of providing learning opportunities but also loses information. Hence, we practically select these mapping functions as a set of learnable linear functions (act as learnable weighted average functions) to minimize the information loss while keeping it simple:

$$\overline{\mathbf{e}}_j = \left(\mathbf{e}_{\alpha_{j-1}}, \cdots, \mathbf{e}_{\alpha_j - 1}\right)\mathbf{W}_j \tag{5}$$

where $\mathbf{W}_j \in \mathbb{R}^{(\alpha_j - \alpha_{j-1})\times 1}$ for $j \in [1, z]$ are learnable parameters. This approach is capable of increasing the number of learnable parameters while using only approximately N additional parameters. This parameter efficiency can enable the model to operate effectively in practical IoT systems.

Logarithmic binned attention. Finally, we perform a multi-head attention operation using only the reduced embedding of tokens to capture their interactions. Using the logarithmic binned embedding $\overline{\mathbf{E}}$, we construct the query, key, and value as $\overline{\mathbf{Q}} = \overline{\mathbf{E}}\mathbf{W}_q$, $\overline{\mathbf{K}} = \overline{\mathbf{E}}\mathbf{W}_k$, and $\overline{\mathbf{V}} = \overline{\mathbf{E}}\mathbf{W}_v$ respectively, where $\mathbf{W}_q, \mathbf{W}_k$, and $\mathbf{W}_v \in \mathbb{R}^{d\times d}$ are learnable parameters. Then the logarithmic binned self-attention is defined by:

$$\mathbf{Y} = \text{Softmax}\left(\frac{\overline{\mathbf{QK}}^T}{\sqrt{d}}\right)\overline{\mathbf{V}}. \tag{6}$$

Note that $\overline{\mathbf{QK}}^T$ has a dimension $z \times z$, resulting in the computational complexity of the logarithmic binned attention being $\mathcal{O}(z^2) = \mathcal{O}(\log^2 N)$. We omitted multi-head concepts in the formulas since they are identical to those presented in a vanilla Transformer.

4.4 Feature-Wise Aggregation

Networks that deploy Transformers typically implement token-wise processing with large hidden layers, which can pose computational overload on constrained hardware devices. To address this, we get motivation from previous studies (Zeng et al. 2023; Liu et al. 2024). We replace the token-wise feed-forward network, which incorporates a dual linear layer, with a single feature-wise linear layer. This modification more efficiently captures time series dynamics by repeating calculations only across the number of features.

Models	PM_{10}								$PM_{2.5}$								PM_1							
	RMSE↓				MAE↓				RMSE↓				MAE↓				RMSE↓				MAE↓			
	Ant.	**Oslo**	**Zag.**	**Avg.**	**Ant.**	**Oslo**	**Zag.**	**Avg.**	**Ant.**	**Oslo**	**Zag.**	**Avg.**	**Ant.**	**Oslo**	**Zag.**	**Avg.**	**Ant.**	**Oslo**	**Zag.**	**Avg.**	**Ant.**	**Oslo**	**Zag.**	**Avg.**
Raw	23.05	23.64	15.41	20.70	20.57	10.75	8.04	13.12	16.41	8.46	5.98	10.28	15.15	13.81	5.97	11.64	13.21	6.48	5.58	8.42	9.59	5.13	4.57	6.43
Linear	18.40	20.51	14.71	17.87	10.22	13.23	6.57	10.01	12.33	9.38	3.34	8.35	5.36	5.50	1.78	4.21	6.27	5.19	1.45	4.30	2.98	2.97	1.11	2.36
NLinear	20.78	22.12	15.45	19.45	14.38	13.89	6.81	11.69	14.80	9.91	5.74	10.15	10.14	6.20	3.66	6.67	8.89	6.68	3.62	6.40	5.98	4.36	2.69	4.34
DLinear	18.35	20.45	14.73	17.84	10.10	13.20	6.65	9.98	12.22	9.62	3.35	8.40	5.14	5.79	1.81	4.25	6.26	5.21	1.49	4.32	2.99	2.87	1.17	2.34
Transformer	15.34	17.49	15.11	15.98	9.53	11.20	7.88	9.54	8.49	8.30	3.40	6.73	4.35	4.51	2.01	3.62	**3.62**	5.22	1.43	3.42	**1.83**	2.51	1.14	1.83
Informer	15.47	16.36	14.86	15.56	9.48	10.43	7.16	9.02	**7.52**	8.14	3.26	6.31	4.22	4.68	1.66	3.52	4.43	4.69	1.34	3.49	2.18	2.35	1.00	1.84
PatchTST	14.60	16.90	14.87	15.46	**9.21**	10.80	7.43	9.15	7.66	8.29	3.33	6.43	**4.18**	4.39	1.93	3.50	6.14	4.86	2.57	4.52	3.70	3.13	2.38	3.07
iTransformer	16.54	16.40	14.03	15.65	9.86	10.48	6.38	8.91	8.62	**7.88**	3.23	6.58	4.58	4.58	1.83	3.66	5.10	4.22	1.49	3.60	2.30	2.24	1.32	1.95
TESLA	**14.58**	**15.79**	**13.87**	**14.75**	9.36	**10.10**	**6.26**	**8.57**	7.54	7.90	**3.03**	**6.16**	4.34	**3.96**	**1.38**	**3.23**	4.78	**3.63**	**1.14**	**3.19**	2.34	**2.01**	**0.80**	**1.71**

Table 1: Calibration accuracy in case $N = 360$ on three regions (**Ant.**, **Oslo**, and **Zag.**) and features ($\mathbf{PM_{10}}$, $\mathbf{PM_{2.5}}$, and $\mathbf{PM_1}$). **Avg.** denotes the average performance across three regions. Raw indicates the difference between the low-cost sensor readings and the reference (without any calibration). The highest performance is marked in **bold**, and the second highest in underlined.

Models	Architecture			PM_{10}			$PM_{2.5}$			PM_1			Avg. gain
	Token	**Embedding**	**Aggregator**	**Ant.**	**Oslo**	**Zag.**	**Ant.**	**Oslo**	**Zag.**	**Ant.**	**Oslo**	**Zag.**	
Transformer	Full	Local	Feed-forward	15.34	17.49	15.11	8.49	8.30	3.41	**3.62**	5.22	1.43	-
TESLA	Binned (uniform)	Local	Feed-forward	15.42	18.26	15.09	8.40	8.02	3.36	4.41	4.96	1.35	-1.20%
TESLA	Binned (log-scaled)	Local	Feed-forward	15.46	17.03	15.09	8.62	8.77	3.29	4.23	5.07	1.26	-0.45%
TESLA	Binned (log-scaled)	Local+Global	Feed-forward	15.03	16.29	14.54	7.94	8.16	3.17	5.01	4.70	1.17	+1.90%
TESLA (Ours)	Binned (log-scaled)	Local+Global	Linear	**14.58**	**15.79**	**13.87**	**7.54**	**7.90**	**3.03**	4.78	**3.63**	**1.14**	**+6.88%**

Table 2: Ablation studies in case $N = 360$ for Transformer and TESLA variants in terms of RMSE. **Avg. gain** represents the average increase in performance of the three regions compared to the Transformer.

Given an attention output $\mathbf{Y} \in \mathbb{R}^{z \times d}$, the final prediction $\hat{x}$ is calculated by:

$$\hat{x} = (\text{LayerNorm}(\mathbf{Y})\mathbf{W}_{\text{agg1}})^T \mathbf{W}_{\text{agg2}} \tag{7}$$

where $\mathbf{W}_{\text{agg1}} \in \mathbb{R}^{d \times 1}$ serves as a feature-wise aggregator, and $\mathbf{W}_{\text{agg2}} \in \mathbb{R}^{d \times 1}$ is used to transform the final token embedding into the scala prediction.

5 Experiment Settings

Here, we first explain the datasets and training setup. Then, we describe our baselines, evaluation metrics, and implementation details.

Datasets. Our experiment is based on a large-scale dataset suitable for calibration task (Van Poppel et al. 2023). This dataset includes sensor data collected from numerous sensors across three regions—Antwerp (**Ant.**), Oslo (**Oslo**), and Zagreb (**Zag.**), with three individual features—$\mathbf{PM_{10}}$, $\mathbf{PM_{2.5}}$, and $\mathbf{PM_1}$. Detailed descriptions of the dataset used in our experiment are shown in Appendix A.

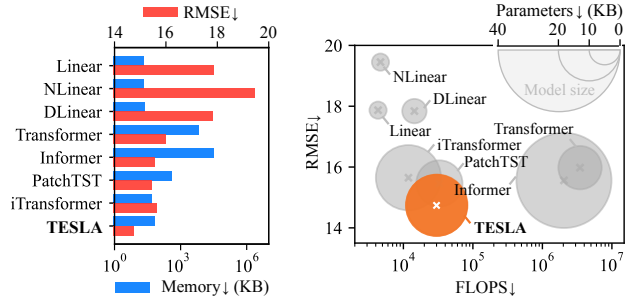
Evaluation setup. Our evaluation process assumes that we use multiple same types of sensors for training in a given space. Each sensors are uniquely identified by the name. We organize the sensors in alphabetical order: the second-to-last sensor is set as the validation set, and the last sensor as the test set. All remaining sensors are used for training. This configuration is repeated across different regions **Ant.**, **Oslo**, and **Zag.** and features $\mathbf{PM_{10}}$, $\mathbf{PM_{2.5}}$, and $\mathbf{PM_1}$. A detailed evaluation setup is provided in Appendix B.

Baselines. We carefully select widely used state-of-the-art techniques for various time series tasks to compare with our proposed method, TESLA. For linear methods, we choose Linear (Freedman 2009), NLinear (Zeng et al. 2023), and DLinear (Zeng et al. 2023), known for their fast inference speeds and outstanding performance in recent time series forecasting. For deep learning methods, we select Transformer (Vaswani et al. 2017), Informer (Zhou et al. 2021), PatchTST (Nie et al. 2023), and iTransformer (Liu et al. 2024), known for their high performance but are less recognized in IoT environments. We give detailed explanations for each baseline in Appendix C.

Evaluation metrics. The factors mentioned in Section 3.3 must be considered when applying the calibration model to real-world scenarios. To comprehensively evaluate the model, we categorize the evaluation metrics into two groups: effectiveness and efficiency metrics.

For effectiveness, we measure accuracy using two widely adopted metrics in time series forecasting and calibration tasks: Root mean square error (RMSE) and mean absolute error (MAE). For efficiency, we assess computational and resource requirements to ensure feasibility in practical IoT applications with low-cost sensors. Key metrics include floating point operations (FLOPS), memory footprint, and the number of model parameters. Also, at the microcontroller level, inference speed and model Flatbuffer size are considered to assess real-time applicability.

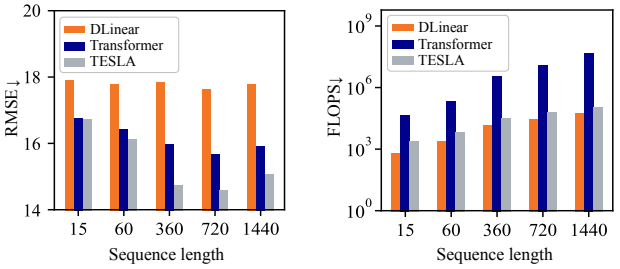
Implementation. All experiments were conducted using a machine equipped with an AMD EPYC 7763 and an NVIDIA RTX A6000 Ada with TensorFlow 2.14, which supports conversion to TensorFlow Lite for Microcon-



(a) TESLA optimizes both RMSE and memory usage.

(b) TESLA balances RMSE, parameters, and FLOPS.

Figure 3: Calibration efficiency in case $N = 360$ for average $\mathbf{PM_{10}}$ performance across three regions. (a) Comparison of RMSE and memory footprint. (b) Scatter map of FLOPS, RMSE, and parameters (diameter of the circles).



(a) Performance rises with scaling but later declines.

(b) TESLA scales efficiently, resembling linear model.

Figure 4: Comparison of RMSE and FLOPS for average $\mathbf{PM_{10}}$ performance across three regions, evaluated at various time spans: 15 minutes, 1 hour, 6 hours, 12 hours, and 24 hours.

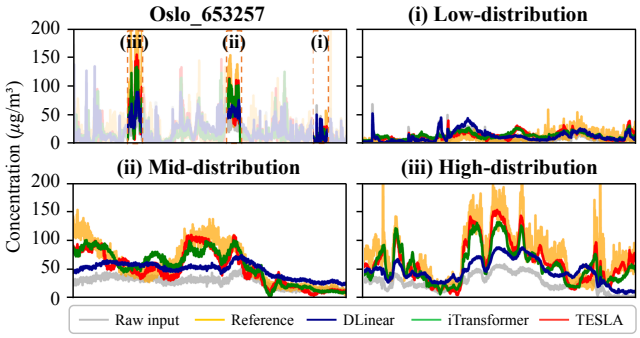


Figure 5: Case study of actual $\mathbf{PM_{10}}$ calibration results with different calibration models for sensor **'Oslo_653257'**, comparing results across three windows with different distributions (**low-**, **mid-**, and **high-distribution**).

Models	Oslo_653257 (PM_{10}) RMSE	MAE	Gain	(i) Low-distribution RMSE	MAE	Gain
Raw	23.64	10.75	-	8.78	6.09	-
DLinear	20.45	13.20	-	9.64	6.59	-
iTransformer	16.40	10.48	-	8.38	6.81	-
TESLA	**15.79**	**10.10**	**+3.81%**	**8.26**	**6.48**	**+3.27%**
Models	**(ii) Mid-distribution RMSE**	**MAE**	**Gain**	**(iii) High-distribution RMSE**	**MAE**	**Gain**
Raw	42.17	30.08	-	57.66	47.67	-
DLinear	30.55	23.60	-	42.22	32.67	-
iTransformer	**16.52**	**11.95**	-	28.67	22.04	-
TESLA	16.58	12.20	-1.21%	**21.78**	**15.96**	**+34.86%**

Table 3: Numerical comparison of the case study results (Figure 5) across each distribution. **Gain** represents the average performance increase in terms of RMSE and MAE compared to iTransformer.

trollers. All models were trained using a batch size of 32 and the Adam optimizer for 10 epochs with mean squared error objective function, following configurations commonly found in existing time series forecasting (Liu et al. 2024). For evaluation on microcontrollers, the trained models were converted to FlatBuffers and deployed on an Arduino Nano 33 BLE Sense for evaluation.

6 Experimental Results

This section summarizes the results of our work.

Accuracy. We first compared calibration accuracy. The results are shown in Table 1. To sum up, overall, TESLA outperformed existing baselines in terms of RMSE and MAE. Transformers outperformed linear models. This contrasts with previous observations in time series forecasting. Among Transformer-based models, PatchTST and iTransformer demonstrated notable effectiveness in token compression for calibration tasks. NLinear demonstrated the least accuracy among all linear-based models, with no significant difference in accuracy between Linear and DLinear.

Ablation study. For ablation study, we made three key modifications and evaluated their impact. We described the changes in Table 2. To begin with, the logarithmic binning showed minimal information loss when converting tokens to binned tokens, with less performance decline compared to uniform interval binning. Second, adding global embedding significantly enhanced accuracy. Last, simplifying the feed-forward network improved performance, boosted calibration speed, and lowered energy consumption.

Calibration tradeoff. We analyzed RMSE, FLOPS, memory usage, and model parameter size to determine whether the model effectively achieves the desired balance (see Figure 3). Linear models are fast and lightweight but lack the accuracy needed for calibration tasks. Transformers generally outperform linear models in accuracy but are less efficient. While Transformer and Informer have high computational demands (*e.g.*, memory footprint and FLOPS), PatchTST, iTransformer, and TESLA balance efficiency and effectiveness, with TESLA achieving the highest accuracy without significant overhead across efficiency metrics.

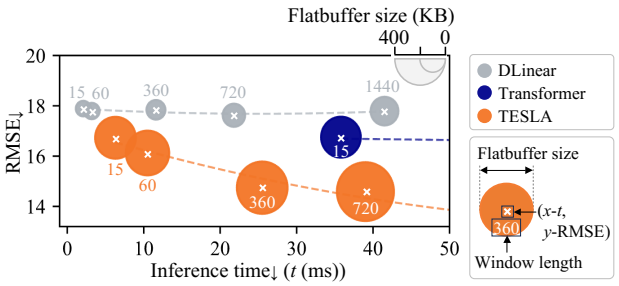


Figure 6: Evaluation of the Arduino Nano 33 BLE Sense microcontroller. The scatter map illustrates RMSE, inference time, and Flatbuffer size (diameter of the circles) with varying windows (15, 60, 360, 720, 1440) for average $\mathbf{PM_{10}}$ performance across three regions. Missing window sizes indicate non-applicability. The dotted line represents the trend.

Sequence length. To evaluate the impact of the sequence length, we compared the most complex linear- and Transformer-based methods with TESLA (see Figure 4). In general, increasing sequence length generally improves accuracy, but excessively long sequences degrade performance (see Figure 4(a)). Note that the improvement rate also differs. DLinear showed slight improvement, while Transformers exhibited significant accuracy gains. TESLA maintains efficiency in FLOPS at less than twice that of DLinear, whereas Transformers are considerably less efficient for extended sequence lengths. For this reason, we set $N = 360$ as our optimal length for all models in our experiment.

Case study. We visually and numerically compared the calibration results of the most practical linear-based model (DLinear), Transformer-based model (iTransformer), and TESLA (see Figure 5 and Table 3). TESLA scored higher accuracy than iTransformer in most cases, with an average improvement of 3.81%. Although this difference does not seem to significantly impact the calibration results, TESLA achieves a remarkable 34.86% improvement over iTransformer under **High-distribution** scenarios. Note that the gap is even more notable when compared to DLinear.

Performance on microcontroller. To evaluate performance in an IoT embedded environment, we measured applicability and inference speed on a microcontroller using representative models (see Figure 6). DLinear is small-sized and supports long sequences but shows low accuracy. In contrast, Transformer handles only short sequences due to high computational demands. TESLA has a model size similar to Transformer but requires far less computation.

7 Discussions and Limitations

We discuss the implications and limitations of our work.

Ineffective linear models. Newly-crafted linear models underperformed compared to Transformer-based models (see Table 1), contrary to trends in forecasting tasks. This discrepancy stems from differences in dataset scale: calibration tasks involve larger datasets, favoring complex architectures with more parameters. Furthermore, the minimal accuracy gap between Linear and DLinear suggests that simple averaging is ineffective for calibration. These results highlight that while effective for forecasting, the linear approach may not be suitable for calibration tasks.

Effectiveness of key modification in TESLA. Table 2 demonstrates the effectiveness of TESLA's three key modifications: (i) Building upon PatchTST's uniform patching approach, TESLA introduces logarithmic binning, which mitigates performance degradation by emphasizing recent values and adapts better to calibration tasks; (ii) Integrating global embedding, motivated by iTransformer, improves accuracy and extends its multivariate design to univariate scenarios; and (iii) Simplifying the feed-forward network to feature-wise linear, inspired by linear forecasters, highlights the importance of relationships between adjacent values and even achieves performance improvements in calibration tasks; In short, TESLA benefits from the strengths of both Transformers and linear modeling techniques.

TESLA leads in effectiveness and efficiency. TESLA is the *desideratum* for addressing both efficiency and effectiveness in calibration tasks, balancing these aspects better than its counterparts (see Figure 3). While PatchTST and iTransformer also achieve this balance, TESLA stands out as the most accurate model with no significant overhead across all efficiency metrics, making it the ideal choice for maximizing performance under resource constraints.

Mean trap phenomenon. The minimal gap between RMSE and MAE arises from the "mean trap," where limited variation in time series data makes these metrics less sensitive to calibration differences. Even in high-distribution scenarios with significant calibration improvements (see Figure 5 and Table 3), these metrics often fail to capture the differences. This underscores the value of using a model like TESLA, where calibration improvements, though seemingly minor in metrics, can have significant real-world impacts.

Static experimental settings. Although we adopt standard settings used in calibration studies (Ahn et al. 2024; Narayana et al. 2024) and TESLA has demonstrated adaptability to IoT environments, the static nature of this approach requires recalibration or drift correction. This limitation can be alleviated as larger datasets become available. We further discuss this in Appendix D.

8 Conclusion

This study introduced TESLA, a Transformer-based calibration model for low-cost sensors in practical IoT systems. We first identified three challenges that must be addressed to adapt these models for calibration tasks. Then, TESLA addresses the challenges in the following manner: (i) Mitigate the attention bottleneck by employing logarithmic binning to reduce the number of tokens, (ii) Integrate multi-view embeddings, and (iii) Ensure fast processing by replacing feed-forward operations with linear layers. Results demonstrated that TESLA excelled in effectiveness and efficiency.

Region	Antwerp (Ant.)	Oslo	Zagreb (Zag.)
#Sensors	24	21	14
Data format	3,113,955×6	810,314×6	939,838×6
Start time	20.04.03 16:12	20.09.17 09:56	20.05.22 16:06
End time	20.06.15 07:07	20.10.14 08:56	20.07.08 12:00
Granularity	1min	1min	1min

Table 4: Metadata for datasets used in our experiment with the data format (#data for all sensors, #features). Each region includes features $\mathbf{PM_{10}}$, $\mathbf{PM_{2.5}}$, and $\mathbf{PM_1}$ from both low-cost and reference sensors.

A Dataset Details

This section outlines the data characteristics and the criteria used to select subsets for the calibration task. As discussed in Section 3, applying benchmarks for time series forecasting to calibration poses several challenges. The SensEURCity dataset (Van Poppel et al. 2023), specifically designed for calibration tasks, provides a robust foundation for addressing these challenges.

To meet task-specific requirements, we select subsets that reflect diverse distributions of measurements for training to capture general measurement patterns. Specifically, we carefully selected: (i) data collected during the first collocation period for each region, as this period provides sensor readings from the same location and time, ensuring consistent calibration under uniform environmental conditions; and (ii) defining Plantower PMS5003 as a low-cost sensor and Alphasense OPC-N3 as a reference reading, categorized by price. The metadata for these subsets is shown in Table 4.

To maintain data quality, we excluded removed outliers or invalid values, and even discarded any sensor for which more than half of its readings were removed. In practical IoT scenarios, sensor data is collected in real-time, so no further refinement is applied beyond removing invalid data.

B Detailed Evaluation Setup

This section describes the evaluation setup for the calibration task and the reasoning behind its configuration. Unlike forecasting tasks, where models predict future values based on past time series, the calibration task focuses on learning general patterns from the temporal behavior of each sensor to ensure robust performance across diverse conditions. To align with the objectives of calibration, our method modifies the train-validation-test split to focus on capturing general patterns from sensor data rather than temporal prediction.

In the SensEURCity dataset, each sensor is identified by its unique ID and organized in alphabetical order within each region. The second-to-last sensor is used as the validation set, the last sensor as the test set, and all remaining sensors are used for training. This configuration ensures that the model learns patterns from diverse training sensors while maintaining consistent evaluation across regions.

C Baselines Details

This section introduces a detailed description of the baselines used in our experiment: (1) Linear-based models, and (2) Transformer-based models. Recent studies suggest that LSTM-based models are less commonly used in time series forecasting, as recent advancements have shifted focus to other architectures. Therefore, our experiment follows commonly used configurations found in recent research (Zeng et al. 2023; Nie et al. 2023; Liu et al. 2024) by adopting linear-based and Transformer-based models. The detailed descriptions of each baseline are as follows:

- **Linear** (Freedman 2009): Linear straightforwardly uses weighted inputs, achieving moderate performance in time series prediction tasks.
- **NLinear** (Zeng et al. 2023): NLinear involves normalizing the input time series data to its last value, which focuses the analysis on detailed dynamics.
- **DLinear** (Zeng et al. 2023): DLinear explicitly separates the time series into trends and residuals and uses individual linear layers.
- **Transformer** (Vaswani et al. 2017): Transformer uses a self-attention mechanism to weigh input data effectively, but it suffers from a quadratic computational bottleneck.
- **Informer** (Zhou et al. 2021): Informer reduces the bottleneck by using selective queries, marking the first attempt to apply a Transformer architecture to a prediction task.
- **PatchTST** (Nie et al. 2023): PatchTST enhances both effectiveness and efficiency by employing uniformly-sized patches as tokens, instead of using single values as tokens.
- **iTransformer** (Liu et al. 2024): iTransformer extends the patch scope of PatchTST to its maximum, treating each sensor as an individual token.

Comparison to TESLA. TESLA combines the advantages of various time series forecasters for calibration tasks. First, unlike PatchTST, it emphasizes recent data more effectively by binning on a logarithmic scale. Additionally, while iTransformer is only effective with multiple sensors or multivariate time series, TESLA adapts to both local and global contexts, making it suitable for univariate applications. Finally, we adapted the token-wise feedforward neural network to a feature-wise linear layer to enhance the calibration speed comparable to that of linear forecasters.

D More Discussions

TESLA achieves fast inference speed and high calibration accuracy, but the limited availability of large-scale calibration datasets constrains its effectiveness. First, additional experiments using varied sensor data, such as temperature and humidity, are required to evaluate TESLA's performance in more diverse scenarios. Second, the reliability of the reference readings remains a concern, as the high-cost sensor used in this study may not provide sufficient precision. Future research should explore using more advanced sensors or alternative references to enhance calibration accuracy. Lastly, using pre-trained weights requires periodic updates through server communication or firmware updates to sustain consistent performance, emphasizing the importance of developing dynamic training methods in future studies.

Acknowledgements

We thank Miro Co., Ltd. for their valuable help. This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1C1C1012408), in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. 2022-0-00448/RS-2022-II220448, Deep Total Recall: Continual Learning for Human-Like Recall of Artificial Neural Networks, and No. RS-2022-00155915, Artificial Intelligence Convergence Innovation Human Resources Development (Inha University)), and in part by INHA UNIVERSITY Research Grant.

References

Ahn, S.; Kim, H.; Lee, E.; and Seo, Y.-D. 2024. SenDaL: An Effective and Efficient Calibration Framework of Low-Cost Sensors for Daily Life. *IEEE Internet of Things Journal*, 11(11): 20619–20630.

Ali, S.; Alam, F.; Arif, K. M.; and Potgieter, J. 2023. Low-Cost CO Sensor Calibration Using One Dimensional Convolutional Neural Network. *Sensors*, 23(2): 854.

Apostolopoulos, I. D.; Fouskas, G.; and Pandis, S. N. 2023. Field Calibration of a Low-Cost Air Quality Monitoring Device in an Urban Background Site Using Machine Learning Models. *Atmosphere*, 14(2).

Aula, K.; Lagerspetz, E.; Nurmi, P.; and Tarkoma, S. 2022. Evaluation of Low-Cost Air Quality Sensor Calibration Models. *ACM Trans. Sen. Netw.*, 18(4).

Concas, F.; Mineraud, J.; Lagerspetz, E.; Varjonen, S.; Liu, X.; Puolamäki, K.; Nurmi, P.; and Tarkoma, S. 2021. Low-Cost Outdoor Air Quality Monitoring and Sensor Calibration: A Survey and Critical Analysis. *ACM Trans. Sen. Netw.*, 17(2).

Elmqvist, N. 2023. Data Analytics Anywhere and Everywhere. *Communications of the ACM*, 66(12): 52–63.

Freedman, D. A. 2009. *Statistical Models: Theory and Practice*. Cambridge University Press, 2 edition.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F.; Burges, C.; Bottou, L.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *International Conference on Learning Representations*.

Narayana, M. V.; Rachavarapu, K. K.; Jalihal, D.; and Nagendra, S. M. S. 2024. Sens-BERT: A BERT-Based Approach for Enabling Transferability and Re-Calibration of Calibration Models for Low-Cost Sensors Under Reference Measurements Scarcity. *IEEE Sensors Journal*, 24(7): 11362–11373.

Nath, P.; Saha, P.; Middya, A. I.; and Roy, S. 2021. Long-term time-series pollution forecast using statistical and deep learning methods. *Neural Computing and Applications*, 33(19): 12551–12570.

Nie, Y.; H. Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.

Patton, A.; Datta, A.; Zamora, M. L.; Buehler, C.; Xiong, F.; Gentner, D. R.; and Koehler, K. 2022. Non-linear probabilistic calibration of low-cost environmental air pollution sensor networks for neighborhood level spatiotemporal exposure assessment. *Journal of Exposure Science & Environmental Epidemiology*, 32(6): 908–916.

Ray, P. P. 2022. A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4): 1595–1623.

Toner, W.; and Darlow, L. 2024. An Analysis of Linear Time Series Forecasting Models. arXiv:2403.14587.

Van Poppel, M.; Schneider, P.; Peters, J.; Yatkin, S.; Gerboles, M.; Matheeussen, C.; Bartonova, A.; Davila, S.; Signorini, M.; Vogt, M.; Dauge, F. R.; Skaar, J. S.; and Haugen, R. 2023. SensEURCity: A multi-city air quality dataset collected for 2020/2021 using open low-cost sensor systems. *Scientific Data*, 10(1): 322.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. arXiv:1706.03762.

Villanueva, E.; Espezua, S.; Castelar, G.; Diaz, K.; and Ingaroca, E. 2023. Smart Multi-Sensor Calibration of Low-Cost Particulate Matter Monitors. *Sensors*, 23(7): 3776.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, 11121–11128.

Zhang, Y.; Ju, C.; Qin, J.; Song, L.; Liu, X.; Sun, W.; and Li, Z. 2023. STCM: A spatio-temporal calibration model for low-cost air monitoring sensors. *Information Sciences*, 644: 119307.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, 11106–11115. AAAI Press.