Generating Synthetic Time-Series Data on Edge Devices Using Generative Adversarial Networks

Md Faishal Yousuf
Electrical and Computer Engineering
University of New Hampshire
Durham, USA
mdfaishal.yousuf@unh.edu

MD Shaad Mahmud
Electrical and Computer Engineering
University of New Hampshire
Durham, USA
mdshaad.mahmud@unh.edu

***Abstract*—Synthetic data generation has recently garnered substantial attention in the research community. This method crafts data that mirrors authentic datasets and proves invaluable for machine learning endeavors, including classifier training and prediction tasks. Moreover, synthetic data addresses challenges related to data scarcity, preserving privacy, and analyzing specialized domains like healthcare and finance. This study introduces an approach to produce synthetic time series data on edge devices, leveraging the capabilities of Generative Adversarial Networks (GANs). This opens avenues to harness generative machine learning to tackle edge computing challenges. Our approach utilizes a GAN to create time series data that closely resembles datasets found in the real world. Furthermore, this GAN is implemented on an edge device, enabling real-time synthetic data generation. Our findings affirm the GAN's efficacy in producing data closely aligned with actual time series datasets, highlighting the potential of GANs in facilitating real-time synthetic data creation on edge devices for various machine learning applications.**

***Index Terms*—Edge computing, GAN, Synthetic time series, Machine learning**

I. Introduction

Time series synthetic data refers to artificially generated sequences of data points ordered in time. Such data is particularly valuable in domains like finance, healthcare, and energy, where understanding temporal patterns and making future predictions are essential [1]. Creating synthetic time series data can be crucial when historical data is limited and confidential or when there is a need to simulate potential future scenarios that haven't occurred yet. Advanced machine learning techniques are now being employed to produce synthetic time series data that closely mimic the statistical properties of real-world temporal datasets. Researchers and businesses can conduct meaningful analyses, develop predictive models, and test strategies without compromising data privacy or integrity by ensuring high-quality synthetic data that respects the intricacies and dependencies of time series.

Generative machine learning, particularly models like Generative Adversarial Networks (GANs) [2], Variational Autoencoder (VAEs) [3], and more, has revolutionized data synthesis in several significant ways [4], [5]. This has led to the creation of images, sounds, and textual content that are often indistinguishable from genuine content to the unaided eye or ear. However, generating synthetic time series data is a more challenging task [6]. Time series data is often characterized by complex dependencies between data points, and the temporal nature of the data makes it challenging to generate realistic synthetic data. As a result, the generation of synthetic time series data has been a relatively under-explored area of research. However, recent advances in generative machine learning have made it possible to generate synthetic time series data that closely follow the statistical properties of real-world time series data. This has led to the creation of synthetic time series data that can be used for various purposes, including data augmentation, anomaly detection, and data simulation.

Machine learning on edge devices is becoming increasingly popular [7]. Edge devices are small, low-power devices that can perform machine-learning tasks on the edge of the network. This allows for faster processing and lower latency. However, edge devices are often resource-constrained, meaning they have limited memory and processing power. This makes it challenging to run complex machine-learning models on edge devices. Researchers have explored application-specific implementations of machine learning like SVM [8], CNN [9], RNN [10], LSTM [11] in an embedded device setting. But, to our knowledge, there has been no research on generating synthetic time series data on edge devices. This paper presents a novel approach to generating synthetic time series data on edge devices using generative machine learning. The proposed approach uses a generative adversarial network (GAN) to generate synthetic time series data on edge devices. The proposed approach is evaluated on a dataset of stock price prediction. The findings indicate that the suggested method can produce synthetic time series data that accurately reflects the statistical characteristics of genuine time series data. This unveils the opportunity to investigate generative models for application-specific analysis in edge computing environments.

II. Methodology

Our approach unfolds in four stages: data preprocessing, model training, optimization, and deployment to edge devices. Initially, the data is cleansed by removing outliers, normalized, and segmented into training and testing sets. We then employ a Generative Adversarial Network (GAN) to train on this

Identify applicable funding agency here. If none, delete this.

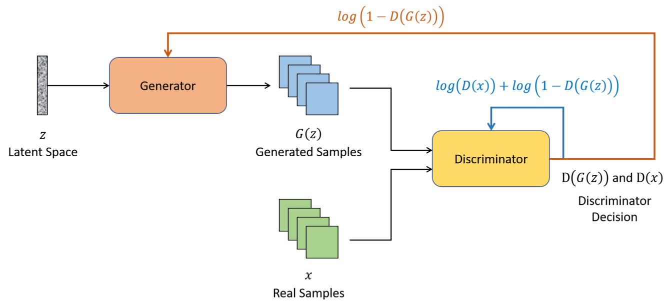

Fig. 1: A standard architecture of a Generative Adversarial Network (GAN) [12]

prepared dataset. Following training, the model is optimized for edge deployment by refining its architecture and reducing parameters. Lastly, the streamlined model is deployed on an edge device, and its performance is validated against the test dataset.

A. Generative Adversarial Networks

Generative modeling, an unsupervised learning approach in machine learning, focuses on identifying and learning patterns within input data. The objective is to allow the model to produce new instances that closely resemble those from the original dataset. Generative Adversarial Networks (GANs) ingeniously handle generative modeling by reframing it as a supervised learning challenge. This involve a pair of networks – a generator G and a discriminator D – that are trained in tandem. The typical GAN architecture is shown in Figure 1.

The generative model in this context is designed to mirror the data's distribution. It is trained with an aim to increase the likelihood of the Discriminator erring in its judgment. Conversely, the Discriminator operates by gauging the probability that a given sample is sourced from the actual training data rather than being produced by the Generator. GANs operate on the principles of a minimax game: the Discriminator works to reduce its reward V(D, G), while the Generator endeavors to increase the Discriminator's loss, effectively maximizing its own. The following mathematical expression can encapsulate this dynamic:

$$\begin{aligned}\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] \\ +\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]\end{aligned} \tag{1}$$

In this equation:

- $\mathbb{E}$ represents the expectation operator.
- $x \sim p_{\text{data}}(x)$ indicates samples drawn from the real data distribution.
- $z \sim p_z(z)$ denotes noise samples taken from a particular distribution, commonly a Gaussian distribution.
- $D(x)$ is the discriminator's estimation of the probability that the real data instance x is genuine.
- $D(G(z))$ is the discriminator's estimation of the probability that a fabricated instance (created by the generator from noise z) is genuine.

The discriminator, within the GAN architecture, produces an output termed as D(x). This output essentially represents the probability that the data point xx is drawn from the real dataset. In simpler terms, D(x) signifies how confident the discriminator is that xx is genuine and not synthetically produced. The primary mission of the discriminator is to enhance its ability to accurately discern and classify the genuine data from its synthetic counterparts. It thrives on its capacity to recognize real data as true and generated data as fabricated.

On the flip side, the generator has a contrasting objective. It endeavors to concoct data that, when passed through the discriminator, garners high D(x) values. In essence, the generator's success is measured by how convincingly it can trick the discriminator into believing that the synthetic data it creates is indistinguishable from authentic data.

This interplay between the generator and discriminator results in a strategic game reminiscent of the classic "minimax" contests in game theory. Here, the generator is on a quest to minimize the value V, which represents its discernibility, while the discriminator, in opposition, is striving to maximize V, indicating its success rate in distinguishing real from fake.

The fine-tuning and balancing of these intertwined objectives are achieved through an iterative process. Specifically, an alternating gradient descent algorithm is employed, ensuring both entities—generator and discriminator—progressively improve and adapt in response to each other, aiming for an equilibrium where the generator produces near-perfect synthetic data and the discriminator becomes adept at its distinguishing task.

B. Model Architecture and Offline Training

To work with time series data, we need a model that takes dependencies between time series data into account and employs an adversarial learning architecture. Long-Short Term Memory(LSTM) is a type of recurrent neural network that can learn long-term dependencies between time series data [13]. This proficiency is derived from their inherent capacity to retain memories of preceding events, thereby giving them a remarkable aptitude for comprehending the influence of historical data points on forthcoming values. Unlike traditional Recurrent Neural Networks (RNNs), which falter with long sequences due to issues like vanishing or exploding gradients, LSTMs use gating mechanisms to efficiently process extended temporal dependencies. This capability ensures they can determine patterns across the varied time interval, which is vital for tasks like forecasting or anomaly detection in sequential data. Moreover, LSTMs can understand and map relationships in sequential data directly, eliminating the need for manual feature extraction and can seamlessly integrate with architectures like Convolutional Neural Networks (CNNs) to capture both temporal and spatial features. Their flexibility also means they can navigate multivariate time series and predict across multiple future time steps. However, while their advantages are considerable, it's crucial to understand that LSTMs aren't a one-size-fits-all solution. Depending on the data's nature and computational constraints, sometimes simpler models might be more appropriate. But for intricate, long-sequence time series challenges, LSTMs stand out as a robust choice.

Consequently, based on its adeptness at handling time-series, we employ LSTM as the foundational architecture for both the generator and discriminator within our setup. This choice stems from LSTM's intrinsic ability to capture long-term dependencies, making it particularly suitable for our requirements. The training process for these two components is conducted in an adversarial setting. Figure 2 shows the high-level design of our proposed model. The main goal of this model is to learn the normal distribution of a given dataset using adversarial training. The adversarial methodology ensures that both entities iteratively improve in tandem, with each striving to outperform the other, ultimately enhancing the model's overall efficacy.

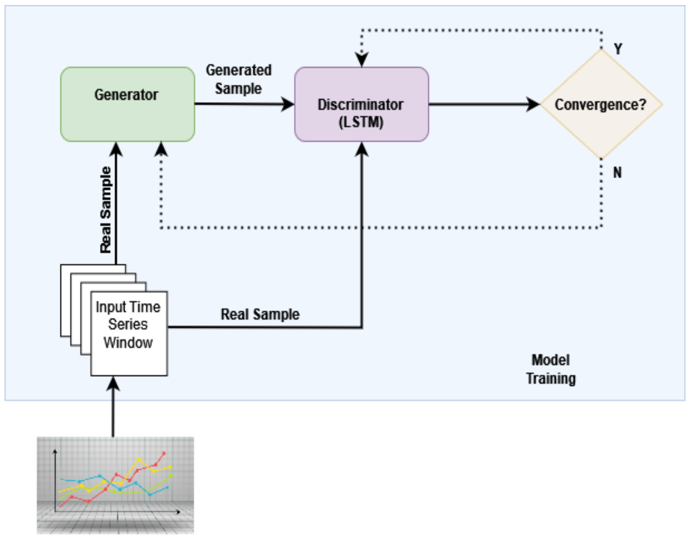

Fig. 2: Flow diagram illustrating the training process of the proposed GAN model

C. Edge Deployment

The GAN model has been designed with the capability to grasp the intricate distribution of temporal parameters, enabling it to efficiently generate synthetic data. Following the completion of offline training, we specifically extracted the generator component, as its primary role is centered around data synthesis. However, a challenge emerges due to GAN's inherent complexity; its training process necessitates extensive computational resources, often limiting it to be executed on robust, high-performance computing systems. While training GANs require high-performance computing, deploying pre-trained GAN models for inference or generating data is more feasible. This approach leverages the strengths of both centralized training resources and decentralized edge processing.

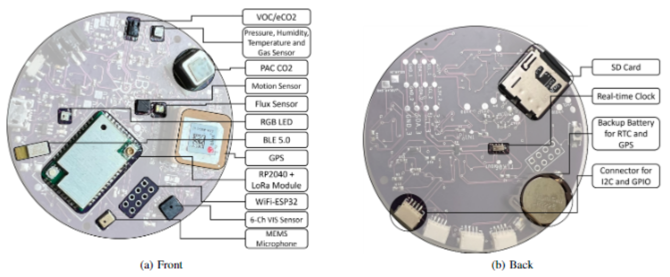

Fig. 3: iBUG: Custom made AI enabled edge device [14]

The model size of GANs, especially in the case of deep architectures, can be substantial. Edge devices have limited storage capacity, and deploying large models is impractical in terms of storage requirements and memory usage. We implemented structural simplifications to enhance the deployability of the model on edge devices, recognizing the inherent resource constraints typically associated with such platforms. In order to optimize the model, we incorporated two primary techniques. **First,** model pruning was employed to effectively reduce the overall size of the model. Identify and prune redundant or less important neurons, connections, or layers from the GAN model. Pruning reduces the model's size by eliminating components that contribute less to its overall performance. **Second,** we utilized model quantization, opting specifically for *float16* quantization. The weights of the GAN's generator and discriminator are converted from float32 to *float16*. This reduces the amount of memory needed to store these parameters. The intermediate activation values during inference are also converted to *float16*. This approach not only reduced the model's size by half but also ensured that the accuracy remained largely unaffected, thereby striking a balance between size and performance. Moreover, the reduced memory requirements also improves the energy efficiency and reduce memory access times, which leads to faster inference. After quantization, the model might experience a drop in performance due to the reduced precision. The quantized model is fine-tuned using a smaller learning rate to help it adapt to the quantized representation.

For edge deployment we used iBUG board, developed by the authors [14], which is a low-cost, low-power, and high-performance edge device. The device boasts a dual-core ARM Cortex-M0+ processor clocked at 133 MHz, coupled with 246 KB of SRAM and 2 MB of onboard flash memory. Additionally, for wireless interactions, it is fitted with a 2.4 GHz IEEE 802.15.4 radio module. A visual representation of the iBUG's physical layout can be found in Figure 3.

D. Dataset

To rigorously assess the efficiency of our introduced model, we utilized a financial time series dataset, specifically curated from Yahoo Finance. This dataset is comprehensive, presenting six integral attributes: Open, High, Low, Close, Adj Close, and Volume. Each of these attributes is encapsulated as integer values, ensuring precision and clarity in data representation. Spanning a noteworthy period from 1 January 2017 to 24 January 2021, the dataset meticulously captures 1,022 distinct financial events. A salient feature of this dataset is its completeness, with no data points missing, ensuring continuity and consistency in the analysis. To prepare the data for our model, it underwent a normalization process using the MinMaxScaler, a technique that transforms features by scaling them to a given range. This ensures that all the features contribute equally to the model's performance. With an intention to set a clear distinction between training and validation, the data was strategically bifurcated into an 80:20 ratio. To ensure clear separation for training and validation, the data was split in an 80:20 ratio. The 80% was used for training and adjusting the model's parameters, while the 20% was utilized for evaluating the model's predictive performance.

III. Results and Discussion

In this paper, we have created an LSTM-based GAN model to generate synthetic time series data. Then we have deployed the model on a resource constraint edge device to evaluate the data generation performance at the edge. For qualitative evaluation, we compare the generated data with the real data.

To comprehensively analyze the distribution and diversity between real and synthetic datasets, two powerful visualization techniques, PCA and t-SNE, are employed. t-SNE is particularly adept at mapping high-dimensional data into 2D or 3D visual spaces, thereby offering an intuitive yet qualitative understanding of data structures and inherent relationships. Its visualization often emphasizes clear cluster formations, and while these visual patterns provide valuable insights, t-SNE does not intrinsically yield quantitative analyses. Thus, for deeper insights or metrics, one might consider supplementing t-SNE with additional quantitative methods.

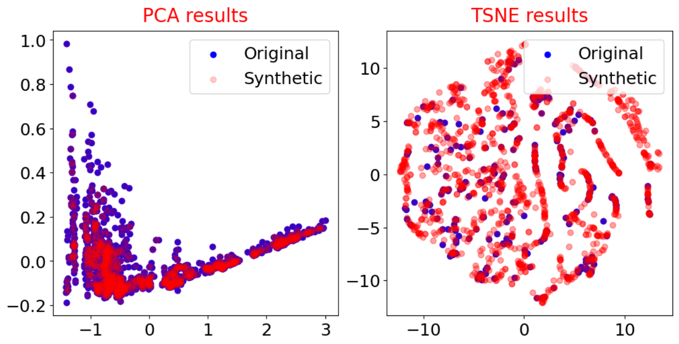

Fig. 4: PCA (left) and t-SNE (right) results of the original data (blue dots) and synthetic data (red dots)

On the other hand, PCA, or Principal Component Analysis, serves as a quantitative approach to dimensionality reduction. It harnesses linear algebraic principles to distill vast datasets into more manageable forms by identifying the directions (or principal components) that capture the most variance. Each of these components contributes a quantifiable amount to the dataset's variance, allowing users to make informed decisions on how many components to retain. For instance, one might opt to preserve components that account for a cumulative 95% of the original data's variance. While PCA doesn't serve as a "metric" in the conventional sense like accuracy or precision in classification scenarios, it undeniably offers valuable numeric insights into the intricate structure and inter-relationships present in the data.

The outcomes derived from the PCA and t-SNE analyses are distinctly illustrated in Figure 4 . In this visual representation, each individual point encapsulates the mean value derived from the six distinct parameters present within the dataset. Distinguishing between the data types, real data points are marked with red dots, while the synthetic ones are highlighted with blue dots. The proximity and overlap between these red and blue dots provide a clear visual metric. By meticulously observing this overlap and the spatial distribution of these points, we can derive insights into the model's performance. Not only does it indicate the accuracy of the synthetic data generation but also underscores how well the generated data emulates the inherent patterns and characteristics of the real data. This visualization thus serves as a compelling tool for a more nuanced evaluation of the model's capabilities in generating synthetic data that closely mirror real-world data points.

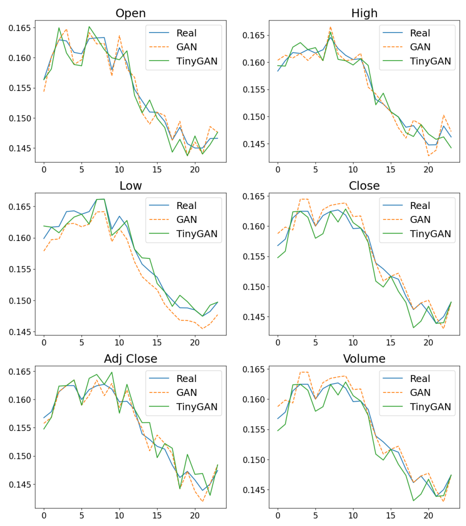

Fig. 5: Real vs. generated comparison of all the parameters in the dataset

In Figure 5, every visual representation provides a comparative analysis, plotting the real data against the generated data for individual parameters within the dataset. This comparative exercise encompasses data that's been generated both by computer and edge device. The blue line represents the real data, the green line presents GAN-generated data and the orange line is the data collected from the edge device. By closely examining these visualizations, we can observe that the generated data aligns with the distribution and patterns observed in the real data. The generated data appears to capture the nuanced trends and inherent characteristics of the real data with commendable accuracy, underscoring the efficacy of the generation methodologies employed.

The results indicate a significant similarity between the synthesized and the original data. Both PCA and t-SNE analyses affirm this similarity between real and synthesized data. This suggests that the GAN model capably produces synthetic time series data mirroring real datasets. The implication is promising, highlighting the potential of employing GANs in edge devices to craft synthetic data that aligns with real-world metrics for various machine-learning applications.

Synthetic data generation through GANs has emerged as a powerful tool, especially for anomaly detection within time series datasets [15]. Current implementations predominantly rely on centralized server-based systems which, while effective, often involve heavy computational overheads. For instance, in industries like manufacturing, real-time detection of anomalies in equipment operation data can prevent costly downtimes. Similarly, in finance, instantaneous identification of irregularities in transaction data can thwart fraudulent activities. However, the lag introduced by sending data to centralized servers for processing can sometimes hinder real-time responses in such scenarios. Thus, transitioning these processes directly to edge devices, closer to the source of data generation, could significantly reduce this lag.

In upcoming research initiatives, the potential of utilizing generative synthetic data techniques on edge devices for immediate anomaly detection is an intriguing direction. Such an approach could provide industries with instant analytical insights, leading to accelerated decision-making and improved operational effectiveness. Particularly in edge contexts, notably within the IoT sphere, the real-time augmentation of data using GANs becomes essential. An example can be observed in the healthcare sector, where synthesizing medical data directly on the device can be instrumental for instantaneous diagnostics. Given the fluidity of edge computing environments, a significant research trajectory could be the development of GANs capable of dynamically adjusting their structures or parameters in response to resource availability or evolving data trends. Furthermore, considering the vulnerability of edge devices to adversarial intrusions, fortifying the resilience of GANs in such scenarios, along with crafting countermeasures against potential security breaches, is of paramount importance.

IV. Conclusion

The results presented in this article are relatively basic compared to the capabilities of GANs to generate synthetic data with real-world values. This data can be used for machine learning tasks, such as training classifiers and predicting outcomes. It can also be shared in a privacy-preserving manner, which means that the original data remains confidential. The results presented in this article are based on a simple GAN architecture. However, more advanced GAN architectures can generate more realistic and complex synthetic data. For example, conditional GANs can be used to generate synthetic data that satisfies certain constraints, such as having a specific distribution or following a particular pattern. The use of GANs to generate synthetic data has a number of advantages. First, it can be used to create large amounts of data, which can be helpful for machine learning tasks that require a lot of data. Second, it can be used to generate data that is similar to real data, which can improve the performance of machine learning models. Third, it can be used to share data in a privacy-preserving manner, which can be essential for protecting sensitive data. Overall, the results presented in this article are promising. However, there is still much potential for GANs to be used to generate synthetic data with real-world applications for machine learning tasks and sharing in a privacy-preserving manner.

Acknowledgment

This material is based upon work supported by the USDA #110378 and NHAES CREATE #11HL14.

References

[1] E. Brophy, Z. Wang, Q. She, and T. Ward, "Generative Adversarial Networks in Time Series: A Systematic Literature Review," *ACM Computing Surveys*, vol. 55, no. 10, pp. 199:1–199:31, Feb. 2023.

[2] I. Goodfellow, "NIPS 2016 Tutorial: Generative Adversarial Networks," Apr. 2017.

[3] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, "Grammar variational autoencoder," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1945–1954. [Online]. Available: https://proceedings.mlr.press/v70/kusner17a.html

[4] Karthika. S and M. Durgadevi, "Generative Adversarial Network (GAN): A general review on different variants of GAN and applications," in *2021 6th International Conference on Communication and Electronics Systems (ICCES)*. Coimbatre, India: IEEE, Jul. 2021, pp. 1–8.

[5] Y. Hong, U. Hwang, J. Yoo, and S. Yoon, "How Generative Adversarial Networks and Their Variants Work: An Overview," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1–43, Jan. 2020.

[6] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar, "Generating High-fidelityy,Synthetic Time Series Datasetss withDoppelGANger," 2019.

[7] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, "Bringing AI to edge: From deep learning's perspective," *Neurocomputing*, vol. 485, pp. 297–320, May 2022.

[8] K. Z. Haigh, A. M. Mackay, M. R. Cook, and L. G. Lin, "Machine Learning for Embedded Systems: A Case Study," 2015.

[9] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 6848–6856.

[10] U. Thakker, J. Beu, D. Gope, G. Dasika, and M. Mattina, "Run-Time Efficient RNN Compression for Inference on Edge Devices," in *2019 2nd Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*, Feb. 2019, pp. 26–30.

[11] H. Jinhai, W. L. Goh, and Y. Gao, "Dynamically-biased Fixed-point LSTM for Time Series Processing in AIoT Edge Device," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Jun. 2021, pp. 1–4.

[12] D. Vint, M. Anderson, Y. Yang, C. Ilioudis, G. Di Caterina, and C. Clemente, "Automatic Target Recognition for Low Resolution Foliage Penetrating SAR Images Using CNNs and GANs," *Remote Sensing*, vol. 13, no. 4, p. 596, Jan. 2021.

[13] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks," Sep. 2019.

[14] M. F. Yousuf, T. Siddique, and M. S. Mahmud, "iBUG: AI Enabled IoT Sensing Platform for Real-time Environmental Monitoring," in *2022 IEEE 31st Microelectronics Design & Test Symposium (MDTS)*, May 2022, pp. 1–7.

[15] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly Detection for IoT Time-Series Data: A Survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, Jul. 2020.