Anomaly Detection in Internet of Things (IoT) Time Series Data: A Comparative Study of Various Techniques

Jonathan Rhoads
Research associate, Botfed Research Society

Abstract

Anomaly detection plays an integral role in a broad range of applications within the Internet of Things (IoT), such as preventive maintenance, health monitoring, fraud detection, and fault prediction. This study undertakes a comprehensive exploration of the methods commonly used for anomaly detection in IoT time series data. These methods encompass Statistical Techniques, Isolation Forest, Autoencoder Neural Networks, and Long Short-Term Memory Units (LSTMs), each with their unique strengths and challenges. Statistical techniques, such as ARIMA, ETS, and STL, model the regular pattern of a time series via a stochastic model, highlighting anomalies as instances that deviate from this model. The Isolation Forest algorithm, on the other hand, isolates anomalies based on their shorter average path lengths in an ensemble of Isolation Trees. Autoencoders and LSTMs, as types of artificial neural networks, detect anomalies via high reconstruction error and significant deviation from predicted values, respectively. The research also acknowledges the applicability of other methods such as K-means clustering, DBSCAN, and XGBoost according to the specific requirements of IoT data. Selection of an appropriate model depends largely on the data characteristics and the particular use case, with data properties including multivariate or univariate nature, presence of trends or seasonality, and type of anomalies playing a crucial role.

Keywords: Anomaly Detection, IoT, Time Series, Machine Learning, Neural Networks

Introduction

The Internet of Things, often abbreviated as IoT, marks a pivotal advancement in the field of technology, rapidly reshaping numerous sectors of society. It is a concept that refers to the network of interconnected devices, objects, and systems, which are capable of collecting and sharing data without requiring human-to-human or human-to-computer interaction [1], [2]. Each device or 'thing' in this connected network has a unique identity and the ability to automatically transfer data over a network. The emergence of IoT is driven by an expansion of the Internet, shrinking hardware, progress in data analytics, and the rise of cloud computing. These technological advancements have made it feasible and cost-effective to transform everyday objects into smart devices with added functionality.

The Internet of Things (IoT) is a complex system composed of three major components: sensors, networks, and applications. Sensors, or "things" in the IoT, are the physical devices that collect data from the environment. These could be temperature sensors, humidity sensors, light sensors, motion sensors, etc. They are equipped with unique identifiers and have the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. Different sensors have different capabilities; for example, some sensors are only able to collect data, while others are able to act based on the data they receive [3], [4].

Networks play a crucial role in connecting the "things" in the IoT to the internet. They enable data transfer from sensors to servers, typically via the internet. Networks in IoT might be made up of wired connections (Ethernet, for instance) or wireless connections (such as Wi-Fi, cellular networks, Bluetooth, Zigbee, or LoRaWAN). The choice of network often depends on factors such as data transmission speed requirements, power consumption, cost, and geographical coverage.

Applications are the final component of the IoT [5], [6]. They collect, process, and analyze the data gathered by sensors and transmitted via networks. Applications in IoT can range from simple mobile apps that display data from a personal fitness tracker to complex, machine learning-driven software that can predict equipment failures in a factory before they happen based on sensor data. The application not only makes sense of the raw data but also provides actionable insights that can be used to improve efficiency, safety, comfort, or other important factors in a given context [7].

IoT connectivity refers to the communication technologies used to connect the "things" in the IoT with each other and with the internet. These technologies can be broadly categorized into short-range (or local) and long-range (or wide area) networks. Short-range networks include technologies like Bluetooth, Zigbee, and Wi-Fi. They are typically used for applications where the devices are located close to each other, such as in a smart home or a manufacturing plant.

On the other hand, long-range networks include technologies like cellular (4G, 5G), LoRaWAN, and NB-IoT. These are often used for applications where the devices are spread over a larger geographical area, like a smart city or a supply chain. Each of these connectivity options has its own pros and cons. For example, while Wi-Fi provides high data rates, it consumes more power and has a limited range compared to technologies like LoRaWAN.

The operation of an IoT system involves several steps and each of the key components plays a vital role in this process. The first step in the IoT process chain is data collection, which is performed by the sensors. Sensors continuously monitor their environment and collect data. For example, a temperature sensor in a smart home would continuously monitor the temperature and generate data representing the temperature levels [8].

Once the data is collected, it needs to be sent to a server or cloud platform where it can be processed and analyzed. This is where the network comes in. The sensors send the data they've collected over the network to the server [9], [10]. This can be done in real-time or at intervals, depending on the requirements of the IoT application. For instance, a health monitoring application would require real-time data transfer, while a soil moisture monitoring system in a farm might only require data updates once a day [11].

Once the data arrives at the server, it is processed and analyzed by the application. The application might apply complex algorithms or machine learning models to the data to derive insights [12]–[14]. For example, a factory monitoring application might analyze sensor data to predict when a machine is likely to fail, allowing for preventative maintenance [15].

The final step in the IoT process chain is action. Based on the insights derived from the data, the application might trigger certain actions. These could be as simple as sending a notification to a user's phone, or as complex as automatically adjusting the settings of a machine in a factory to prevent a predicted failure. This closes the loop in the IoT process chain, with the "things" in the IoT collecting data, the data being transferred and processed, and the insights from the data being used to take action.

Different techniques

Statistical techniques for anomaly detection have been in use for a considerable amount of time and are among the most tried-and-true methods of discerning irregularities in datasets. These techniques operate under the assumption that the normal pattern of a time series dataset can be modeled using a stochastic or random process model, and anomalies are defined as those instances that do not conform to this model. There are numerous statistical methods that can be employed for this purpose, and among the most well-known are Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing (ETS), and Seasonal and Trend decomposition using Loess (STL).

The Autoregressive Integrated Moving Average (ARIMA) is a popular statistical method used for forecasting time series data. ARIMA models are based on the idea that the information in the past values of the time series can alone be used to predict the future points [16], [17]. An ARIMA model is characterized by three parameters: (1) the order of the autoregressive part (p),

(2) the degree of differencing (d), and (3) the order of the moving-average part (q). The identification of these parameters is usually done by inspecting the autocorrelation and partial autocorrelation plots. Once the ARIMA model is fitted, the residuals are calculated and tested for randomness. Any non-random patterns could indicate an anomaly.

Exponential Smoothing (ETS) is another classic time series forecasting method. ETS models are suitable for non-stationary data and can handle trends and seasonality. The core idea behind ETS is to give more weight to recent observations and less weight to older ones. Depending on the presence of trend and seasonality, different variations of ETS are used such as Simple Exponential Smoothing (no trend or seasonality), Double Exponential Smoothing (trend but no seasonality), and Triple Exponential Smoothing (both trend and seasonality). After fitting the ETS model, residuals are extracted and checked for anomalies, which are usually indicated by significant deviations.

The Seasonal and Trend decomposition using Loess (STL) is a method that decomposes a time series into three components: trend, seasonal, and remainder (or irregular) components [18]. The 'Loess' in STL stands for locally estimated scatterplot smoothing, which is a non-parametric regression method that combines multiple regression models in a k-nearest-neighbor-based meta-model. STL has the advantage of being able to handle any type of seasonality, not only annual but also daily, weekly, etc. After the time series is decomposed, anomalies can be easily detected in the remainder component as they are the instances that could not be explained by the trend and the seasonal components.

However, it is important to note that while these statistical methods are powerful, they are not without limitations [19]–[21]. For example, ARIMA models assume linearity and Gaussian errors, which might not hold in all situations. ETS methods, although capable of handling trend and seasonality, might struggle with non-linear patterns or when the trend and seasonal patterns change over time. STL, on the other hand, assumes that the seasonality is of known and constant period, which might not be true in some cases. Furthermore, all these methods work best on long time series with at least a few seasons of data.

Despite their limitations, statistical methods are a fundamental part of any toolbox for time series analysis and anomaly detection. They provide a solid statistical foundation, upon which more modern and complex methods, like machine learning, have been built. But their importance does not diminish with the advent of these newer techniques. On the contrary, statistical methods often serve as a first line of defense in the process of anomaly detection, providing initial insights and helping to inform more complex analyses [22].

Isolation Forest is a unique and innovative anomaly detection method. Unlike traditional methods which operate by learning and establishing a profile of what's normal and then identifying anything that doesn't fit this profile as an anomaly, Isolation Forest takes a different approach by isolating anomalies. It's an unsupervised learning algorithm that builds an ensemble of "Isolation Trees" or "iTrees" for a given dataset, with anomalies characterized as instances that have shorter average path lengths on these iTrees [23], [24].

The core idea behind Isolation Forest is that anomalies are data points that are few and different, which should make them easier to 'isolate' from the rest of the data. The algorithm operates by randomly selecting a feature from the dataset and then randomly selecting a split value between the maximum and minimum values of that feature [25], [26]. This process recursively continues, and it results in a tree structure, the iTree, where the path length from the root node to the terminating node is indicative of how anomalous the data point is [27].

The reasoning here is that if a data point is an anomaly, it should not conform to the general pattern of the data and should be easier to isolate, leading to a shorter path length in the iTree. On the other hand, 'normal' data points, which conform to the general pattern, are harder to isolate and thus result in longer path lengths.

A few features make Isolation Forest particularly appealing. For one, it doesn't require a normal profile to be established first, as is the case with most traditional methods. This makes it more

adaptive to changes in what's considered 'normal'. Also, since it's not distance-based, Isolation Forest is more immune to the curse of dimensionality and can handle high-dimensional data more effectively than many traditional anomaly detection methods.

Isolation Forest constructs multiple iTrees on different sub-samples of the dataset, creating an 'ensemble' of iTrees. The anomaly score is then calculated as the average of the path lengths for a data point across all the iTrees. A shorter average path length indicates a higher likelihood of a point being an anomaly.

Like any method, Isolation Forest has its limitations. For instance, it might struggle with detecting global anomalies if the dataset has too many dimensions. Also, if the anomalies are 'swamped' or masked by normal points and hence are not easily 'isolatable', Isolation Forest may struggle to detect them. Despite these limitations, Isolation Forest is a highly effective and efficient method for detecting anomalies, especially for high-dimensional datasets or when the definition of 'normal' changes over time[28]–[30] .

In summary, Isolation Forest is an innovative approach that shifts the focus from profiling 'normal' data points to isolating anomalies. It's a powerful and efficient method for anomaly detection, particularly suited to situations where the 'normal' is evolving, or where the data is high-dimensional. Its unique approach and robust performance make it a vital tool in the data scientist's toolbox for anomaly detection.

Autoencoder Neural Networks are a type of artificial neural network primarily used for learning efficient representations of input data, also known as codings. Autoencoders are a self-supervised learning technique that leverages the principles of data encoding and decoding for anomaly detection. Their architecture is symmetric, consisting of an encoder that compresses the input into a latent-space representation, and a decoder that reconstructs the input from this representation [31], [32].

The primary idea behind using autoencoders for anomaly detection is that they are trained to minimize reconstruction error — the difference between the original input and the reconstructed output [33]. If the input data are normal instances, autoencoders can learn their structure effectively, leading to a low reconstruction error. On the other hand, if the input data are anomalies (which should be rare and significantly different from normal instances), autoencoders will struggle to reconstruct these instances accurately, leading to a high reconstruction error.

During the training process, the autoencoder learns to extract meaningful features from the input data in the encoding stage and uses these features to reconstruct the input in the decoding stage. The goal is to create a compact, efficient representation of the data that captures its most important features, which can then be used to reproduce the original input with a high degree of fidelity [32], [34].

In anomaly detection, autoencoders are typically trained on normal instances only, allowing them to learn the pattern of normal data thoroughly. Once trained, the autoencoder can then be used to reconstruct new instances. If these new instances are similar to the normal instances the model was trained on, they can be accurately reconstructed, and the reconstruction error will be low. However, if these new instances are significantly different or anomalous, the autoencoder will have difficulty accurately reconstructing them, and the reconstruction error will be high. Therefore, a high reconstruction error can be a signal of an anomaly[35] .

The power of autoencoders comes from their ability to handle complex and high-dimensional data, their flexibility and adaptability, and their capacity to capture non-linear relationships in the data. However, they also have limitations [36], [37]. The performance of autoencoders can significantly depend on the choice of architecture (number of layers, number of nodes per layer, etc.), and the selection of a suitable architecture requires domain knowledge and experience. Moreover, autoencoders require large amounts of data and computational resources for training. Despite these challenges, autoencoders have proven to be an effective tool for anomaly detection in various domains, including fraud detection, industrial damage detection, and medical

anomaly detection, to name a few. By focusing on the reconstruction of input data, autoencoders provide an intuitive and powerful approach for identifying instances that do not conform to the norm, making them a valuable tool in the field of anomaly detection [38], [39].

Long Short-Term Memory Units (LSTMs) are a specialized type of Recurrent Neural Network (RNN) that have the ability to learn and remember information over long sequences of data, making them particularly well-suited for handling time series data and hence, for time series anomaly detection [40].

In time series anomaly detection, an LSTM network can be trained on normal sequences of data, thereby enabling it to learn the inherent temporal dependencies and patterns within these sequences. Essentially, the LSTM learns to predict the next value in a sequence based on its understanding of the pattern in the prior data points. This pattern learning and prediction capability forms the basis for anomaly detection [41], [42].

For normal data, the LSTM's prediction for the next data point should align well with the actual value, resulting in a low prediction error. However, for anomalous data, which by definition deviates from the normal pattern, the LSTM's prediction is likely to be significantly off from the actual value, resulting in a high prediction error. This high prediction error is therefore an indicator that an anomaly has occurred.

LSTM's advantage lies in its ability to handle long sequences of data and its memory capability. Traditional RNNs suffer from the vanishing gradient problem, where the contribution of information decays geometrically over time, making them unable to handle long sequences and causing them to forget the earlier data points. LSTMs solve this problem with a unique design of memory cell which includes a 'forget gate', an 'input gate', and an 'output gate'. These elements work together to regulate the addition and removal of information to and from the memory cell, allowing LSTMs to maintain and access information over a longer period of time [43].

Despite the strengths of LSTMs, there are some challenges. Like other deep learning methods, LSTMs require a significant amount of data and computational resources for training. They can also be sensitive to the choice of hyperparameters and might require careful tuning to achieve optimal performance. Moreover, as a black-box model, LSTMs lack interpretability which can be a hindrance in scenarios where understanding the reasoning behind a prediction is important. Nevertheless, due to their ability to understand and learn from long sequences of data, LSTMs have proven to be a powerful tool for anomaly detection in time series data. They have been successfully applied in various domains such as fraud detection, health monitoring, and network intrusion detection. Through their unique design, LSTMs are able to identify anomalies in time-dependent data where traditional methods might fail, thereby making them an indispensable tool in the field of time series anomaly detection [44], [45].

Conclusion

This research has presented an in-depth analysis of the commonly used methods for anomaly detection in the context of the Internet of Things (IoT), a crucial area with wide-ranging applications such as preventive maintenance, health monitoring, fraud detection, and fault prediction. Through our comprehensive exploration, we have established that the choice of method is significantly influenced by the nature and requirements of the time series data, its properties, and the specific application scenario.

We found that statistical techniques like ARIMA, ETS, and STL can effectively model regular patterns in time series data and identify anomalies as deviations from these models. These methods are particularly useful for data with strong trends or seasonality, although they may fall short when faced with highly complex or non-linear data structures. On the other hand, the Isolation Forest algorithm isolates anomalies based on their shorter average path lengths, making it a powerful tool for datasets with large feature spaces, yet its performance can be influenced by the choice of hyperparameters.

Moreover, this research highlighted the potential of deep learning-based methods such as Autoencoders and Long Short-Term Memory Units (LSTMs) in anomaly detection. Autoencoders exploit high reconstruction errors to detect anomalies, while LSTMs identify anomalies based on significant deviations from predicted values. These techniques have shown considerable promise in handling complex, multivariate time series data, but they require ample data for training and may be computationally intensive.

Our research suggests that the selection of the optimal method for anomaly detection in IoT is largely a contextual decision. It is critical to consider the data's characteristics, such as its multivariate or univariate nature, the presence of trends or seasonality, and the type of anomalies. Other methods like K-means clustering, DBSCAN, and XGBoost may also be applicable depending on the specific requirements of the IoT data.

References

[1] E. F. I. Raj, M. Appadurai, S. Darwin, and E. F. I. Rani, "Internet of things (IoT) for sustainable smart cities," *Internet Things (IoT) Eng. Appl.*, 2022.

[2] A. Gopinath, S. Sivakumar, D. Ranjani, S. Kumari, V. Perumal, and R. B. R. Prakash, "A Communication System Built on the Internet of Things for Fully Autonomous Electric Cars," in *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2023, pp. 1515–1520.

[3] S. Gadde, E. Karthika, R. Mehta, S. Selvaraju, W. B. Shirsath, and J. Thilagavathi, "Onion growth monitoring system using internet of things and cloud," *Agricultural and Biological Research*, vol. 38, no. 3, pp. 291–293, 2022.

[4] S. S. I. Samuel, "A review of connectivity challenges in IoT-smart home," in *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, 2016, pp. 1–4.

[5] M. Bilal, "A review of internet of things architecture, technologies and analysis smartphone-based attacks against 3D printers," *arXiv preprint arXiv:1708.04560*, 2017.

[6] S. Heitlinger, N. Bryan-Kinns, and R. Comber, "The Right to the Sustainable Smart City," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, Glasgow, Scotland Uk, 2019, pp. 1–13.

[7] S. Shashi Devi, S. Gadde, K. Harish, C. Manoharan, R. Mehta, and S. Renukadevi, "IoT and image processing Techniques-Based Smart Sericulture Nature System," *Indian J. Applied & Pure Bio*, vol. 37, no. 3, pp. 678–683, 2022.

[8] S. Umamaheswar, L. G. Kathawate, W. B. Shirsath, S. Gadde, and P. Saradha, "Recent turmeric plants agronomy analysis and methodology using Artificial intelligence," *International Journal of Botany Studies*, vol. 7, no. 2, pp. 233–236, 2022.

[9] J. Xu, J. Yao, L. Wang, Z. Ming, K. Wu, and L. Chen, "Narrowband Internet of Things: Evolutions, Technologies, and Open Issues," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1449–1462, Jun. 2018.

[10] T. S. Gunawan *et al.*, "Prototype design of smart home system using internet of things," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 7, no. 1, pp. 107–115, 2017.

[11] S. M *et al.*, "Analysis of Hydroponic System Crop Yield Prediction and Crop IoT-based monitoring system for precision agriculture," in *2022 International Conference on Edge Computing and Applications (ICECAA)*, 2022, pp. 575–578.

[12] S. Durga, R. Nag, and E. Daniel, "Survey on Machine Learning and Deep Learning Algorithms used in Internet of Things (IoT) Healthcare," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 2019, pp. 1018–1022.

[13] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security?," *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 41–49, Sep. 2018.
[14] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: a survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, Aug. 2018.
[15] K. Thiagarajan, C. K. Dixit, M. Panneerselvam, C. A. Madhuvappan, S. Gadde, and J. N. Shrote, "Analysis on the Growth of Artificial Intelligence for Application Security in Internet of Things," in *2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, 2022, pp. 6–12.
[16] M. Valipour, "Long-term runoff study using SARIMA and ARIMA models in the United States," *Meteorol. Appl.*, vol. 22, no. 3, pp. 592–598, Jul. 2015.
[17] S. Wang, C. Li, and A. Lim, "Why Are the ARIMA and SARIMA not Sufficient," *arXiv [stat.AP]*, 16-Apr-2019.
[18] S. Jahandari, A. Kalhor, and B. N. Araabi, "Online forecasting of synchronous time series based on evolving linear models," *IEEE Transactions on*, 2018.
[19] M. Farsi *et al.*, "Parallel genetic algorithms for optimizing the SARIMA model for better forecasting of the NCDC weather data," *Alex. Eng. J.*, vol. 60, no. 1, pp. 1299–1316, Feb. 2021.
[20] A. Kumar Dubey, A. Kumar, V. García-Díaz, A. Kumar Sharma, and K. Kanhaiya, "Study and analysis of SARIMA and LSTM in forecasting time series data," *Sustainable Energy Technologies and Assessments*, vol. 47, p. 101474, Oct. 2021.
[21] H. Wang, L. Liu, S. Dong, and Z. Qian, "A novel work zone short-term vehicle-type specific traffic speed prediction model through the hybrid EMD–ARIMA framework," *B: Transport Dynamics*, 2016.
[22] K. Nova, A, Umaamaheshvari, S. S. Jacob, G. Banu, M. S. P. Balaji, and S, Srithar, "Floyd–Warshalls algorithm and modified advanced encryption standard for secured communication in VANET," *Measurement: Sensors*, vol. 27, p. 100796, Jun. 2023.
[23] J. Lesouple, C. Baudoin, M. Spigai, and J.-Y. Tourneret, "Generalized isolation forest for anomaly detection," *Pattern Recognit. Lett.*, vol. 149, pp. 109–119, Sep. 2021.
[24] Z. Cheng, C. Zou, and J. Dong, "Outlier detection using isolation forest and local outlier factor," in *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, Chongqing, China, 2019, pp. 161–168.
[25] Y. Chabchoub, M. U. Togbe, A. Boly, and R. Chiky, "An In-Depth Study and Improvement of Isolation Forest," *IEEE Access*, vol. 10, pp. 10219–10237, 2022.
[26] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and E. Al, "K-Means-based isolation forest," *Knowledge-based systems*, 2020.
[27] S. Jahandari and D. Materassi, "Analysis and compensation of asynchronous stock time series," *Proc. Am. Control Conf.*, 2017.
[28] K. Nova, "The Art of Elasticity and Scalability of Modern Cloud Computing World for Automation," *American Journal of Computer Architecture*, vol. 6, no. 1, pp. 1–6, 2019.
[29] M. Tokovarov and P. Karczmarek, "A probabilistic generalization of isolation forest," *Inf. Sci.* , vol. 584, pp. 433–449, Jan. 2022.
[30] G. Staerman and P. Mozharovskyi, "Functional isolation forest," *Asian Conference*, 2019.
[31] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," *2008 eighth ieee international*, 2008.
[32] D. Xu, Y. Wang, Y. Meng, and Z. Zhang, "An Improved Data Anomaly Detection Method Based on Isolation Forest," in *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, 2017, vol. 2, pp. 287–291.
[33] A. Bodepudi and M. Reddy, "Spoofing Attacks and Mitigation Strategies in Biometrics-as-a-Service Systems," *ERST*, vol. 4, no. 1, pp. 1–14, Feb. 2020.

[34] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended Isolation Forest," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1479–1489, Apr. 2021.
[35] S. Jahandari, A. Kalhor, and B. N. Araabi, "A self tuning regulator design for nonlinear time varying systems based on evolving linear models," *Evolving Systems*, 2016.
[36] Y. Burda, R. Grosse, and R. Salakhutdinov, "Importance Weighted Autoencoders," *arXiv [cs.LG]*, 01-Sep-2015.
[37] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial Autoencoders," *arXiv [cs.LG]*, 18-Nov-2015.
[38] A. Makhzani and B. J. Frey, "Pixelgan autoencoders," *Adv. Neural Inf. Process. Syst.*, 2017.
[39] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, Helsinki, Finland, 2008, pp. 1096–1103.
[40] S. Jahandari and A. Srivastava, "Detection of Delays and Feedthroughs in Dynamic Networked Systems," *IEEE Control Systems Letters*, vol. 7, pp. 1201–1206, 2023.
[41] A. Makhzani and B. Frey, "k-Sparse Autoencoders," *arXiv [cs.LG]*, 19-Dec-2013.
[42] P. Baldi, "Autoencoders, Unsupervised Learning, and Deep Architectures," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, vol. 27, pp. 37–49.
[43] A. Bodepudi and M. Reddy, "Cloud-Based Gait Biometric Identification in Smart Home Ecosystem," *International Journal of Intelligent Automation and Computing*, vol. 4, no. 1, pp. 49–59, 2021.
[44] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.
[45] W. H. Lopez Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, "Autoencoders," in *Machine Learning*, A. Mechelli and S. Vieira, Eds. San Diego, CA: Elsevier, 2020, pp. 193–208.