Anomaly Detection on Time series Sensor Data Using Deep LSTM-Autoencoder

Stephen Githinji
Centre of Data Science and Artificial Intelligence(DSAIL)
Dedan Kimathi University of Technology
Nyeri, Kenya
stephen.gitau@interns.dkut.ac.ke

Ciira wa Maina
Centre of Data Science and Artificial Intelligence(DSAIL)
Dedan Kimathi University of Technology
Nyeri, Kenya
ciira.maina@dkut.ac.ke

***Abstract*—Anomaly detection is crucial in various applications (e.g., cybersecurity, manufacturing, finance, IoT), and an automatic and reliable anomaly detection tool is necessary for accurate prediction. The proposed method in this paper focuses on using deep LSTM Autoencoder on time-series data from IoT water level sensors deployed on a water catchment. The method uses unsupervised anomaly detection with deviation methods, which involves lower-dimensional embeddings and reconstruction error. The LSTM Autoencoder model includes feature selection by keeping vital features, and learns the time series' encoded representation. The LSTM model is trained for prediction with three hidden layers based on the encoder's latent layer output. Afterwards, given the output from the prediction model if the reconstruction loss of a data point is greater than reconstruction error threshold the value will be labeled anomaly. We also propose and compare an unconventional method of calculating reconstruction error of each sequence with an aim of reducing false positives and false negatives then compare it with frequently used method. The results show that the LSTM Autoencoder performs well on noisy and real-world datasets for detecting anomalies and also the proposed unconventional method of calculating reconstruction loss increases the models accuracy in identifying anomalies.**

***Index Terms*—Anomaly detection, Internet of Things (IoT), LSTM, Autoencoder, Time Series, Environmental Sustainability.**

I. Introduction

Anomalies are an inherent component of practically every system in the modern world, which is surrounded by a vast number of Internet of Things (IoT) that are creating enormous amounts of data. Anomaly detection [1] is the identification of data that does not fit to the distribution of normal data, i.e. does not conform to the normal appearance, semantic content, quality, or expected behavior. IoT devices are physical objects that are embedded with electronics, sensors, and network connectivity or other communication networks that enable them to collect and exchange data across systems and devices. They are increasingly prevalent in various industries and have the potential to revolutionize how we interact with the world around us. In this instance the data is a sequence of data points collected over an interval of time thus referred to as time series [2].

There are different types of anomalies that can occur in data, and they are typically categorized based on their characteristics and underlying causes. Here are some common types of anomalies:

1) **Point anomalies:** These are data points that are significantly different from the rest of the data points in the same dataset. For example, a sudden spike or drop in a time series signal could be a point anomaly [3].
2) **Contextual anomalies:** These observations or sequences that do not follow the expected patterns within a time series. However, if these observations are considered individually, they may fall within the expected range of values for that particular signal [4].
3) **Collective anomalies:** These are groups of data points that are collectively unusual when compared to the rest of the data. The individual data instances in a collective anomaly may not be anomalies by themselves, but their occurrence together as a collection is anomalous For example, a group of servers that are experiencing a sudden increase in traffic could be a collective anomaly. [5]
4) **Temporal anomalies:** These are data points that are unusual over time. For example, a gradual increase in the temperature of a machine over a period of time could be a temporal anomaly.
5) **Spatial anomalies:** These are data points that are unusual in a specific location or region. For example, a sudden increase in air pollution in a specific area could be a spatial anomaly.

Identifying and detecting anomalies in data is important in various fields, such as finance, healthcare, and cybersecurity, as it can help to prevent potential problems and improve decision-making.

The focus of the study is the application of anomaly detection in the IoT sensors deployed physically in the environment to monitor water level height of water catchment, particularly the upper Ewaso Nyiro at Ol-Pejeta conservancy which is also home to a Wildlife Techlab which was setup to develop and test conservation technology. This is because it is one of the major rivers in Kenya and needs to be conserved as a natural resource to protect the global ecosystem and ensure the health and well-being of individuals. Since some of the decisions that impact the environment such as human encroachment into

979-8-3503-3621-4/23/$31.00 ©2023 IEEE

water catchment areas are not felt immediately thus the need to make data driven decisions to protect such vital resources.

Our deployed IoT sensors are advanced river water level data acquisition systems developed by our team [6]. They improve on the effectiveness, size, deployment design and data transmission capabilities of systems being utilized. The main component of the system is a river water level sensor node. The node is based on the MultiTech mDot – an ARM-Mbed programmable, low power RF module – interfaced with an ultrasonic sensor for data acquisition. The data is transmitted via LoRaWAN and stored on servers.

The decision to study and make data driven decisions resulted in the designing and deployment of an IoT sensor system on the river with the aim to capture the water level of the river. This made it possible to share information in a non-intrusive and efficient way. The data collected by the sensors must be sent via communication networks under challenging operating conditions, hence data corruption is a common occurrence. The usefulness of the data for real-time applications might be greatly impacted by undetected inaccuracies during transmission. [7] suggests there is a need for automated quality assurance and control (QA/QC) of data, which can be accomplished via real-time detection of anomalous data.

IoT devices provide massive amounts of real-time data that might be challenging to manually evaluate and analyze thus need for unsupervised anomaly detection techniques. IoT devices are frequently used in remote or inaccessible places, making it challenging to do human inspections or maintenance. These techniques can automatically discover anomalies in this data without the need for physical inspection, saving time and effort. Anomalies in IoT time series data may be signs of possible problems or system failures since unsupervised anomaly detection techniques can identify anomalies in real-time, it enables prompt action to be taken. Early detection of anomalies can assist stop additional harm or failures to the IoT devices, improving the performance and reliability of the IoT system and the data being collected.

There are several types of unsupervised anomaly detection techniques that have been used for time series data include statistical methods [8] which model the time series data using statistical distributions and detect anomalies by identifying data points that fall outside the expected distribution. Clustering-based anomaly detection techniques group similar time series data points into clusters and detect anomalies as data points that do not belong to any cluster or belong to a sparsely populated cluster such as k-means clustering [10] and density-based clustering [11]. [9] proposes the use of deep learning-based methods to overcome challenges when it comes to anomaly detection for time series data such as complexity of data and lack of labels. These methods use deep neural networks, such as Long Short-Term Memory (LSTM) networks, to model the time series data and detect anomalies by comparing the prediction error with a threshold value.

Time series data points are not independent but it is expected that the most recent data points in the sequence have an impact on the timestamps of the data points that follow them, as suggested by [15]. In recent years, deep learning has frequently become popular and has been applied in various anomaly detection algorithms. Deep anomaly detection (DAD) techniques can automatically learn and extract features without developing manual features by domain experts [12].

Unsupervised learning has gained more attention because collecting labels in an imbalance dataset has many difficulties. The dataset is imbalanced if anomaly behavior happens rarely, and most of the records are normal. The purpose of this study is to review a structured and comprehensive state of the art anomaly detection technique, the LSTM Autoencoder is an implementation of an autoencoder for sequential data using an Encoder-Decoder LSTM architecture. By using this model, we can have the benefits of both the LSTM and the autoencoder model. The paper also compares approaches of calculating the reconstruction / test loss using both the traditional and an unconventional way with an aim of detecting anomalies autonomously.

II. Objectives

A. General Objectives

This paper's principal goal is to develop an advanced deep learning model to detect environmental sensor time series data anomalies using LSTM Autoencoder.

B. Specific Objectives

1) To annotate and prepare the Data
2) To build the LSTM Autoencoder Model
3) To reconstruct inputs using the LSTM Autoencoder Model
4) To get the Reconstruction error and threshold to flag anomalies
5) Evaluation Metrics of our model.

III. Methodology

In this paper, the focus will be on the application of anomaly detection in IoT water level sensors particularly using Deep LSTM-Autoencoder. We propose an unconventional method of calculating the reconstruction errors of the model and compare performance with the traditional method with aim of recuing occurence of false positives and false negatives.

A. Time Series and Deep LSTM

The application of Deep Neural Network (DNN) architectures have resulted in significant achievements in many areas [13], especially in time-series modeling. This success is mainly attributed to these networks' stacked architecture that allows a complex task to be partially solved in each layer. One variation of DNNs is the Recurrent Neural Networks (RNNs) when unfolded in time. The primary function of the layers in RNNs is to offer some memory rather than a hierarchical processing setting, which is seen in deep neural networks. An important class of RNNs are LSTM networks that are suitable to represent sequential data. They address the vanishing gradient problem, seen in vanilla RNNs, through multiplicative gated units. LSTM networks have been applied

widely in areas such as time-series analysis [15]. Dense DNNs, a temporal hierarchy of the sequential data is best preserved when hidden layers are stacked to build deeper recurrent networks. The original LSTM model consists of a single hidden LSTM layer followed by a standard feedforward output layer. The Stacked LSTM is an extension to this model that has multiple hidden LSTM layers where each layer contains multiple memory cells. Moreover, stacked LSTM networks can be organized to form an autoencoder that can perform anomaly detection once trained on somewhat "clean" data.

B. Autoencoder

Autoencoder is an unsupervised neural network that aims to learn the best encoding-decoding scheme from data. In general, it consists of an input layer, an output layer, an encoder neural network, a decoder neural network, and a latent space. When the data is fed to the network, the encoder compresses them into the latent space, whereas the decoder decompresses the encoded representation into the output layer. The encoded-decoded output is then compared with the initial data and the error is back propagated through the architecture to update the weights of the network. The model architecture of the autoencoder is as shown in 1. The main purpose of the autoencoder is not simply to copy the input to the output. By constraining the latent space to have a smaller dimension than the input, the autoencoder is forced to learn the most salient features of the training data. In other words, an important feature in the design of an autoencoder is that it reduces data dimensions while keeping the major information of data structure.

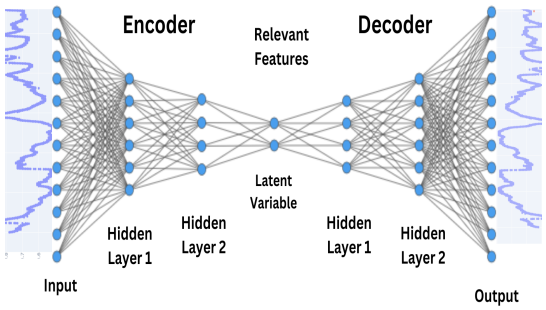

Fig. 1. Conceptual example of an autoencoder

C. LSTM Autoencoder

Several types of autoencoders have been proposed in previous literature, such as vanilla autoencoder, convolutional autoencoder, regularized autoencoder, and LSTM autoencoder. Among these types, LSTM autoencoder refers to the autoencoder that both the encoder and the decoder are in the LSTM network. The ability of LSTM to learn patterns in data over long sequences makes them suitable for time series forecasting or anomaly detection. That is, the use of the LSTM cell is to capture temporal dependencies in multivariate data. The encoder-decoder model learns using only the normal sequences and can be used for detecting anomalies in time-series. The encoder-decoder has only seen normal instances during training and learned to reconstruct them. When it is fed with an anomalous sequence, it may not be reconstructed well, leading to higher errors. This has a practical meaning since anomalous data are not always available or it is impossible to cover all the types of these data. Many advantages of using the autoencoder approach have been discussed in [16]. The use of LSTM autoencoders for anomaly detection on time series data can be seen in recent studies, for example, [17] and [18].

D. LSTM Autoencoder Reconstruction Error

A time series with a total timesteps of T is noted as $X = \{X_t, t = 1, 2, \ldots, T, X_t \in R^n\}$. The reconstruction model takes the original time series data X as input, and outputs a time series data, $\hat{X} = \{\hat{X}_t, t = 1, 2, \ldots, T, \hat{X}_t \in R^n\}$, which is the reconstructed X. The input to the LSTM Autoencoder model is X , the model tries to reconstruct the same input as the output $\hat{X}$ and the result is also referred to as the reconstructed input. The difference between the input and reconstructed input is the reconstructed error. Through many observations of data, LSTM autoencoders learn to minimize reconstruction error between the input and output. When training is complete, any similar data fed to autoencoders produces a reasonably small reconstruction error. The reconstruction error is derived from the equation below:

$$\epsilon^t = \|X_t - \hat{X}_t\| \tag{1}$$

Where $t = 1.....T$. However, if the new data is statistically different from what is seen during the training process, autoencoders fail to reconstruct it properly at the output, resulting in a large error. It is this residual error, ϵ that indicates the presence of an anomaly [14]. We propose finding the mean absolute error loss on the training data, then the max MAE loss value in the training data is used as the reconstruction error threshold. If the reconstruction loss for a data point in the test set is greater than this reconstruction error threshold value then we will label this data point as an anomaly.

1) Timesteps

The number of timesteps / sliding window size used in the input is referred to as the sequence length, and it determines the memory that the model has about the past values in the time series. A larger sequence length will allow the model to capture longer term patterns, while a smaller sequence length will capture more short-term patterns.

We aim to reconstruct the sequence in blocks of size B where $B \ll T$ thus form a matrix of individual segments sliding through the time series data X with a displacement of

one as forming an input matrix 2.

$$X = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_B \\ x_2 & x_3 & x_4 & \vdots & x_{B+1} \\ x_3 & x_4 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ x_{T-B+1} & \cdots & \cdots & x_{T-1} & x_T \end{bmatrix} \tag{2}$$

The reconstructed output matrix 3 as:

$$\hat{X} = \begin{bmatrix} \hat{x}_1 & \hat{x}_2 & \hat{x}_3 & \cdots & \hat{x}_B \\ \hat{x}_2 & \hat{x}_3 & \hat{x}_4 & \vdots & \hat{x}_{B+1} \\ \hat{x}_3 & \hat{x}_4 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \hat{x}_{T-B+1} & \cdots & \cdots & \hat{x}_{T-1} & \hat{x}_T \end{bmatrix} \tag{3}$$

A snippet of what our proposed algorithm does assuming $X = \{x_1, x_2, ...x_6\}$ is our time series collected data. The next step involves reconstruction of the sequence in blocks of size 4 forming a matrix of individual segments sliding through the time series data X with a displacement of one thus forming an input matrix as Equation 4 and reconstructed output as Equation 5.

$$X = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_5 \\ x_3 & x_4 & x_5 & x_6 \end{bmatrix} \tag{4}$$

$$\hat{X} = \begin{bmatrix} \hat{x}_{11} & \hat{x}_{12} & \hat{x}_{13} & \hat{x}_{14} \\ \hat{x}_{21} & \hat{x}_{22} & \hat{x}_{23} & \hat{x}_{24} \\ \hat{x}_{31} & \hat{x}_{32} & \hat{x}_{33} & \hat{x}_{34} \end{bmatrix} \tag{5}$$

When using an LSTM Autoencoder for anomaly detection in time series data, the goal is to reconstruct the original sequence as accurately as possible. However, it is possible for the autoencoder to produce false positives or false negatives when determining if a sample is an anomaly or not. False positives occur when the autoencoder flags a normal sample as an anomaly, while false negatives occur when the autoencoder fails to flag an anomalous sample.

In order to reduce the occurrence of false positives and false negatives, we look into calculating the reconstruction error of each individual time step in the sequence, rather than the reconstruction error of the entire sequence. This is because an anomalous sample at one time step may affect the reconstruction error of the entire sequence, leading to a false positive or negative.

From the matrix 5 individual reconstructed inputs can be calculated by:-

$$\hat{x_1} = \hat{x}_{11} \tag{6}$$

$$\hat{x_2} = \frac{\hat{x}_{12} + \hat{x}_{21}}{2} \tag{7}$$

$$\hat{x_3} = \frac{\hat{x}_{13} + \hat{x}_{23} + \hat{x}_{33}}{3} \tag{8}$$

From the equation 6, 7, 8 we conclude that the of individual reconstructed inputs $\hat{X}_t$ from the model is given by the sum of values x_{ij} where $i + j = t + 1$ divided by p where p is the number of elements as seen in Equation 9.

$$\hat{X}_t = \frac{1}{p} \sum_{ij:i+j=t+1} \hat{x_{ij}} \tag{9}$$

Summing and obtaining the means of the diagonals of the reconstructed output matrix values as shown in 9 reduces False positives and False negatives by obtaining individual reconstructed means instead of reconstructed means of an array containing sequences of data points. This represents the the reconstruction error of each time step. By obtaining the individual reconstructed means instead of the reconstructed mean of the entire sequence, one is able to more accurately detect anomalous samples and reduce the occurrence of false positives and false negatives.

The reconstructed error and test loss are obtained by differencing the original height of the water level input at each individual sample index with the individual reconstructed means calculated above at the same respective sample index as shown by Equation 10 below. Where t is the sample index.

$$|X_t - \hat{X}_t| \tag{10}$$

If the reconstruction loss for a data point in the test set is greater than this reconstruction error threshold value obtained from the mean of training reconstruction error derived from Equation 1 then it is flagged as an anomaly.

IV. Results

A. Data Preparation

The raw data set received from the sensors consists of two columns time and water level height thus just has one feature. The data set is manually annotated with the aim of evaluation metrics for model performance for anomaly detection, An extra column is added which is named as label, where anomalies are labeled as anomaly and normal data points as normal. Anomalies are also manually removed from 70% of the data of which is the training set and the remaining 30% is the test set. Figure 2 shows the visualization of the entire manually labeled dataset where the blue data points are normal water level points received from the sensors and the red data points are the anomalies in the data set received with respect to time.

B. Determining Anomalies

This section deals with results from the built LSTM Autoencoder model which is trained on normal data points and reconstructs the inputs as outputs as shown in Figure 4. The encoder-decoder model learns using only the normal sequences of the training data. The encoder-decoder only sees normal instances during training and learns to reconstruct them. When it is fed with an anomalous sequence of the test data, it may not be reconstructed well, leading to higher errors. In Order to determine the anomalies we find the Mean Absolute Error (MAE) of the training data and make the maximum MAE loss value as the Reconstruction Error Threshold. The model is fed

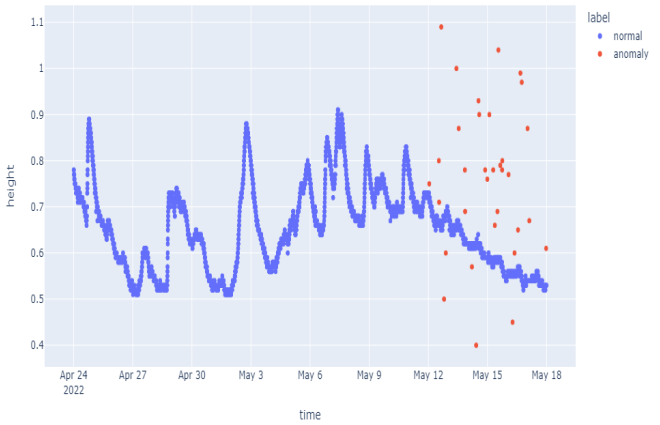

Fig. 2. Water Level Plot

on the test data and since the sequence contains anomalous data points it is expected not to reconstruct well and thus have high reconstruction loss at anomalous data points. Figure 3 shows the reconstruction loss of the test loss and the static reconstruction threshold from the train loss.

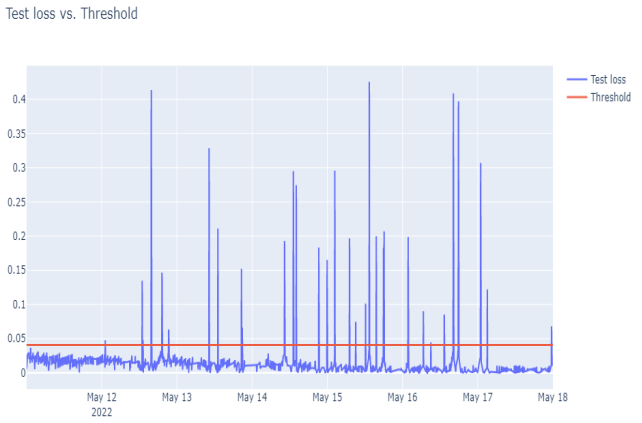

Fig. 3. Test vs Reconstructed Test

Any value of the reconstruction test loss greater than reconstruction error threshold is tagged as anomaly. Figure 5 shows the overview of the detected anomalies.

C. Model Performance Evaluation

To obtain the performance of our model for anomaly detection, we compare ground truth labels of the data set versus the LSTM Autoencoders labels. For an ideal model, the ground truth labels should be the same as the LSTM Autoencoders labels. A confusion matrix as shown in 6 and 7, is used to represent the matrix of the predicted labels and each prediction can have either of the four outcomes showing labels before and after use of proposed methodology to identify anomalies.

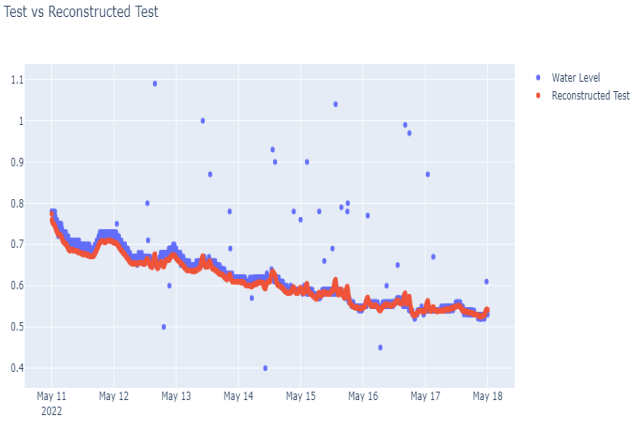

Fig. 4. Test Loss vs Reconstruction Threshold

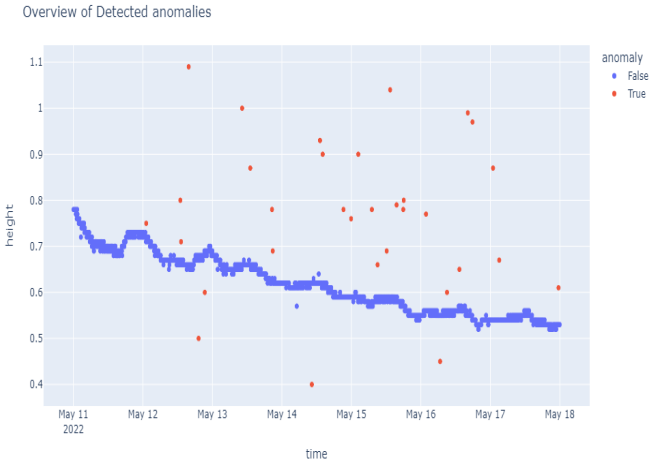

Fig. 5. Detected Anomalies Plot

The results can either be True Positive (TP): Predicted True and True in reality, True Negative (TN): Predicted False and False in reality, False Positive (FP): Predicted True and False in reality and False Negative (FN): Predicted False and True in reality.

The confusion matrix later helps in evaluating accuracy, precision, recall and the F1 Score. Table I displays the results of the evaluation metrics. The F1 score metric in this case is preferred rather than accuracy as data distribution is unbalanced, as the quantity of anomalies class is significantly outnumbered by those found in the normal class.

From the results, we clearly conclude that our proposed unconventional methodology for reducing false positives and false negatives outperformed the traditional methodology as it was able to reduce false positives and negatives.

ACKNOWLEDGMENT

The author of this work would like to thank Jason Kabi of the Centre of Data Science and Artificial Intelligence, Nyeri,

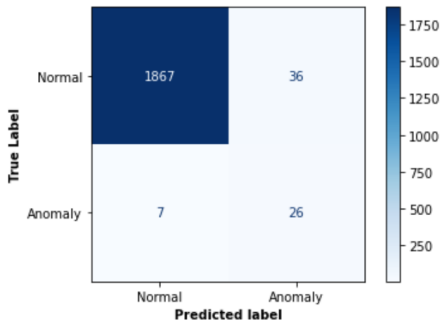

Fig. 6. Confusion Matrix

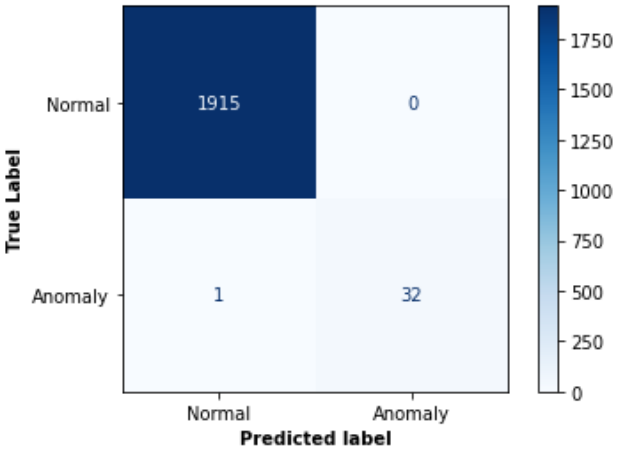

Fig. 7. Confusion Matrix

Kenya, for providing data and knowledge regarding the sensors used in this study. This study was supported by the Centre of Data Science and Artificial Intelligence.

REFERENCES

[1] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, "f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks," Med. Image Anal., vol. 54, pp. 30–44, May 2019, doi: 10.1016/J.MEDIA.2019.01.010.

[2] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly Detection in Time Series: A Comprehensive Evaluation," Proc. VLDB Endow., vol. 15, no. 9, pp. 1779–1797, 2022, doi: 10.14778/3538598.3538602.

[3] S. Dou, K. Yang, and H. V. Poor, "PC2A: Predicting Collective Contextual Anomalies via LSTM With Deep Generative Model," IEEE Internet Things J., vol. 6, no. 6, pp. 9645–9655, Dec. 2019, doi: 10.1109/JIOT.2019.2930202.

[4] V. Chandola, A. Banerjee, V. K.-A. computing surveys (CSUR), and U. 2009, "Anomaly detection: A survey," dl.acm.org, 2009, Accessed: Feb. 22, 2023. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1541880.1541882.

[5] M. H. Hassan, A. Tizghadam, and A. Leon-Garcia, "Spatio-temporal anomaly detection in intelligent transportation systems," Procedia Comput. Sci., vol. 151, pp. 852–857, 2019, doi: 10.1016/J.PROCS.2019.04.117.

[6] J. Kabi, C. wa Maina, E. Mharakurwa, S. M.- HardwareX, and undefined 2023, "Low cost, LoRa based river water level data acquisition system," Elsevier, Accessed: Jun. 13, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468067223000214

	Metric	Scores Before	Scores After
0	Accuracy	0.978	0.999
1	Precision	0.419	1.000
2	Recall	0.788	0.969
3	F1 Score	0.547	0.984

TABLE I
CONFUSION MATRIX

[7] D. Hill, B. Minsker, E. A.-P. of the Congress, and U. 2007, "Real-time Bayesian anomaly detection for environmental sensor data," academia.edu, Accessed: Feb. 22, 2023. [Online]. Available: https://www.academia.edu/download/45363764/Bayesian-anomaly-sensor-IAHR07.pdf.

[8] M. Fahim, A. S.-I. Access, and U. 2019, "Anomaly detection, analysis and prediction techniques in iot environment: A systematic literature review," ieeexplore.ieee.org, Accessed: Feb. 23, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8733806/.

[9] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines," IEEE Access, vol. 9, pp. 120043–120065, 2021, doi: 10.1109/ACCESS.2021.3107975.

[10] J. Zhang, H. Zhang, S. Ding, X. Z.-F. in E. Research, and U. 2021, "Power consumption predicting and anomaly detection based on transformer and K-means," frontiersin.org, Accessed: Feb. 23, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fenrg.2021.779587/full.

[11] A. Abid, A. Kachouri, A. M.-I. W. Sensor, and U. 2017, "Outlier detection for wireless sensor networks using density-based clustering approach," Wiley Online Libr., vol. 7, no. 4, pp. 83–90, Aug. 2017, doi: 10.1049/iet-wss.2016.0044.

[12] R. Chalapathy and S. Chawla, "Deep Learning for Anomaly Detection: A Survey," Jan. 2019, Accessed: Feb. 23, 2023. [Online]. Available: http://arxiv.org/abs/1901.03407.

[13] S. Maleki, S. Maleki, N. J.-A. S. Computing, and U. 2021, "Unsupervised anomaly detection with LSTM autoencoders using statistical data-filtering," Elsevier, Accessed: Feb. 28, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494621003665.

[14] Y. Wang, M. Perry, D. Whitlock, J. S.-J. of Manufacturing, and U. 2022, "Detecting anomalies in time series data from a manufacturing system using recurrent neural networks," Elsevier, 2020, doi: 10.1016/j.jmsy.2020.12.007.

[15] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for time series classification," Neural Networks, vol. 116, pp. 237–245, Aug. 2019, doi: 10.1016/J.NEUNET.2019.04.014.

[16] O. I. Provotar, Y. M. Linder, and M. M. Veres, "Unsupervised Anomaly Detection in Time Series Using LSTM-Based Autoencoders," 2019 IEEE Int. Conf. Adv. Trends Inf. Theory, ATIT 2019 - Proc., pp. 513–517, Dec. 2019, doi: 10.1109/ATIT49449.2019.9030505.

[17] H. Xiao, D. Guan, R. Zhao, W. Yuan, Y. Tu, and A. M. Khattak, "Semi-supervised Time Series Anomaly Detection Model Based on LSTM Autoencoder," Commun. Comput. Inf. Sci., vol. 1415, pp. 41–53, 2021, doi: 10.1007/978-981-16-3150-4_4/COVER.

[18] Z. Que, Y. Liu, C. Guo, X. Niu, Y. Zhu, and W. Luk, "Real-time anomaly detection for flight testing using autoencoder and LSTM," Proc. - 2019 Int. Conf. Field-Programmable Technol. ICFPT 2019, vol. 2019-Decem, pp. 379–382, Dec. 2019, doi: 10.1109/ICFPT47387.2019.00072.