

# REAL-TIME ANOMALY DETECTION IN INTERNET OF THINGS DEVICES USING TEMPORAL CONVOLUTIONAL NETWORK

GHAZOUANI MOHAMED<sup>1</sup>, ABDE RAHMANE DAIF<sup>2</sup>, ARDCHIR SOUFIANE<sup>3</sup>, MOHAMED AZZOUAZI<sup>4</sup>

<sup>1,2,3,4</sup>Department of Mathematics and Computer Science  
Hassan II University, Faculty of sciences Ben M'sik, Casablanca, Morocco

E-mail: <sup>1</sup>ghazouani.fsbm@gmail.com, <sup>2</sup>daif.abdou@gmail.com, <sup>3</sup>soufiane79@gmail.com,  
<sup>4</sup>azouazii@gmail.com

## ABSTRACT

Anomaly detection is the examination of specific data points and the detection of rare occurrences that appear suspicious because they are different from the established pattern of behaviors. Anomaly detection is nothing new, but the increase in data volume makes manual tracking impossible. When such an anomaly occurs, it is sometimes difficult to realize it, and the delay between the beginning of the anomaly and its observation can be a day or more, depending on the case. This article proposes a neural network-based model for real-time anomaly detection in Internet of Things sensors. The aim of this study is to detect defective Internet of Things Devices at the Laboratory Information Technology and Modeling, Faculty of Sciences Ben M'sik, Hassan II University Casablanca, Morocco. Different neural network models were compared, namely, Long Short-Term Memory (LSTM), gate recurrent unit (GRU) and a Temporal Convolutional Network (TCN). In conclusion, our experiment has demonstrated that the TCN model surpasses other models in terms of performance. The impressive performance of these models reaffirms the significance of this approach and its potential for enhancing preventive maintenance of Internet of Things devices.

**Keywords:** *TCN, LSM, GRU, Anomaly Detection, Internet of Things*

## 1. INTRODUCTION

The use of connected devices has become increasingly popular in recent years, with many people relying on these devices for a wide range of tasks. However, one concern that has been raised is the potential for these devices to collect incorrect or inaccurate data due to a variety of factors. For example, a malfunction in the device itself or adverse weather conditions could lead to erroneous data being collected. Additionally, security, radio interference or other environmental factors may also impact the accuracy of the data collected. As a result, it is crucial to identify any irregularities at the earliest to prevent or minimize the damage. Researchers have explored data-centric approaches to model and identify anomalies, capitalizing on the emergence of machine and deep learning techniques and the abundance of data from interconnected devices. The causes of anomalous data can be broken down into three points.

- Specific events occurred in the area monitored

by sensor nodes (when a forest fire occurs, the temperature readings of sensors will rise sharply).

- Sensors may not work normally due to their own hardware failures or energy depletion.
- The deviation or measurement error exists in collected data due to the influence of external factors (e.g., illumination intensity readings of sensor nodes in shaded areas are significantly lower than those of nodes directly exposed to sunlight).

In the literature [1], there is a general classification of anomalies that applies in several fields of application and which can be divided into three main types: punctual, contextual and collective.

- **Punctual anomalies** correspond to a data point that is considered an outlier because it is sufficiently different or far from the data set. Figure 1 shows an example of a building energy consumption time series. An observation that has a very high value (overconsumption)

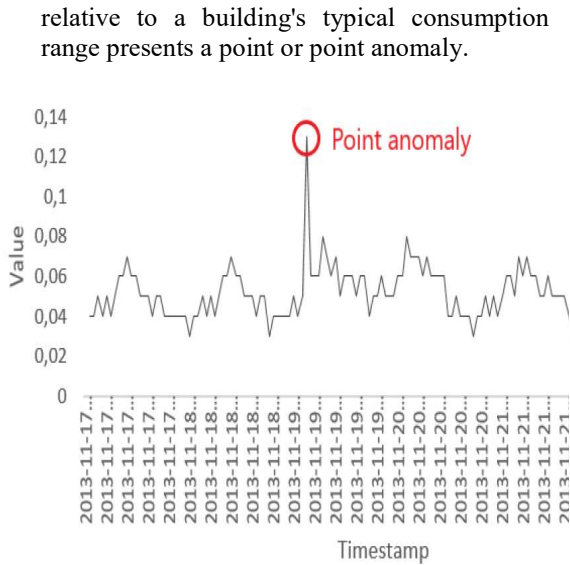


Figure 1: Punctual anomaly in a time series of energy consumption of a building [1].

- **Contextual anomalies** correspond to a data point (or sequence of points) different or distant from other data points but in a specific context (spatial or temporal). For example, Figure 2 shows a contextual anomaly in a monthly temperature time series. A low temperature in winter at time  $t_1$  is considered normal, while the same case might not be normal in high summer at time  $t_2$ .

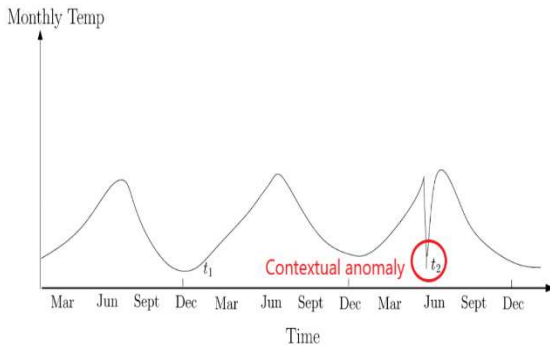


Figure 2: Contextual anomaly in a monthly temperature time series [1].

- **Collective anomalies** correspond to a collection of observations that is different from the data set. For example, Figure 3 shows an example of a time series containing an anomalous subseries because it is different from the set of subsequences in the time series. This may correspond to a stopped counter, which fails to upload data.

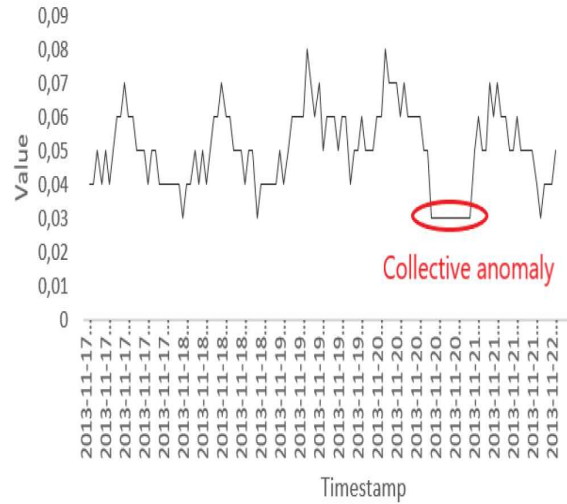


Figure 3: Collective anomaly corresponding to a meter stoppage [1].

Anomaly detection in deep learning is a particular problem that can be dealt with in a supervised or unsupervised way. In a supervised approach, it is possible to approach this as a binary classification problem, the goal then being to classify each observation as an anomaly or not. The peculiarity of this approach is that anomalies are the minority class represented by a very small percentage of the data set. Most classification algorithms are sensitive to this type of imbalance. They are therefore unable to deal effectively with the problem. In addition, great care must be taken in the evaluation of classification models and the choice of metrics. An unsupervised approach is the most suitable for anomaly detection problems. In this approach, the different algorithms try to distinguish outliers by learning on all the data, without having the labels of the observations, there is no set of observations identified a priori as anomalies. It is therefore not necessary to have labeled data and this simplifies the problem of preparing data before learning and all the difficulty in constructing labels. To address the above-mentioned issues, a method of intrusion detection is proposed and implemented using deep learning.

The rest of the paper is organized as follows. Section 2 describes related work. We review the methodology in Section 3. Theoretical explanations and experimental results of the proposed TCN model are denoted in section 4. The conclusion of the proposed model is represented in Section 5.

## 2. RELATED WORKS

There have been numerous research efforts on anomaly detection in Internet of Things Devices Using Deep Learning. Here are a few noteworthy and

recent studies that have been analyzed and discussed. The authors of [2] demonstrate that the LSTM RNN model they create is capable of detecting abnormal and suspicious behavior in IoT systems with high accuracy based on a dataset of IoT sensors readings. The model they have developed offers a remarkably high rate of detection and a correspondingly low rate of false alarms. Where all of the tests performed were assigned different values of volume and iterative periods, randomly-generated values are used for weights and biases as well as the various layers, nodes, and learning rate until the model was able to achieve the best rate of accuracy detection. In article [2], the authors presented a study on the use of deep learning methods in intrusion detection in IoT devices. They used standard dataset Bot-IoT for IoT intrusion detection, they also used different kind of deep learning methods such as Convolutional Neural Network, Gated Recurrent Unit and Long Short Memory Neural Network for intrusion detection in IoT. The authors in [2] investigate the performance of deep learning models, including bidirectional LSTM (BI-LSTM) and Long Short-Term Memory (LSTM), CNN-based Temporal Convolutional (TCN), and CuDNN-LSTM, which is a fast LSTM implementation. They used the SWaT Dataset (December-January 2015-2016, dataset with anomaly) from iTrust, Center for Cybersecurity Research, Singapore University of Technology and Design. The models consist of two hidden layers, each of which has an appropriate number of neurons for the model based on observations made during the model training phase, using the 'tanh' activation function with a dropout rate of 0.3. For the experiments conducted in this study, they set Batch size=150 and epoch=20 for all models to maintain consistency. The algorithm used in the article [5] to detect anomalies called DeepAnT, is an unsupervised learning-based anomaly detection technique for continuous data. This method consists of two modules. The first module, the time series predictor, is responsible for predicting the next time stamp. The predicted value is then passed to the anomaly detection module. This module is responsible for labeling a data instance as normal or abnormal. The prediction module of DeepAnT is based on CNN. CNN is a type of artificial neural network that has been widely used in different fields such as computer vision and natural language processing in a range of different capabilities due to the efficiency of its parameters. As the name suggests, this network uses a mathematical operation called convolution. The article [6] is about IoT network anomaly and attacks detection as the massive rise in

data transmission via various IoT devices and communication protocols has increased security concerns, and for that researchers have focused their attention on more comprehensive artificial intelligence methods of anomaly detection. The CNN models have proven that it's more is more effective than the FFN (FeedForward Neural Network) and RNN (Recurrent Neural Network) models. The main contributions of article [7] are the proposed integrated model of CNN and LSTM-based autoencoder, the use of sliding windows for time series preprocessing, and the evaluation of anomaly detection based on reconstruction error using an autoencoder trained only on normal data. The article discusses the use of deep learning for anomaly detection in time series data. The authors focus on using a combination of convolutional neural networks (CNN) and recurrent neural networks (RNN) to extract spatial and temporal features, respectively. They propose an improved network architecture, which builds on a previous method called C-LSTM that uses a two-stage sliding window in data preprocessing to generate time-dependent subsequences for better feature extraction. The features extracted by the CNN are then input into an autoencoder with two LSTM layers for feature extraction. The authors in article [8] present a new method called Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED) for anomaly detection and diagnosis in multivariate time series data. The authors state that building a system for this purpose is challenging as it requires not only capturing the temporal dependency in each time series, but also encoding the inter-correlations between different pairs of time series. They also note that existing unsupervised anomaly detection algorithms may not address all of these challenges. The proposed MSCRED method first constructs multi-scale signature matrices to characterize multiple levels of the system statuses in different time steps. Then, a convolutional encoder is used to encode the inter-sensor (time series) correlations and an attention-based Convolutional Long-Short Term Memory (ConvLSTM) network is developed to capture the temporal patterns. Finally, a convolutional decoder is used to reconstruct the input signature matrices, and the residual signature matrices are further utilized to detect and diagnose anomalies. Table 1 summarises the literature survey.

Table 1: Summarises literature survey.

Ref.	Objective	Proposal	Drawbacks
[2]	Using deep learning to detecting abnormal behavior in internet of things	LSTM RNN model	LSTM RNN models need enough normal data to effectively train the model, a lack of or insufficient normal data can lead to imperfect anomaly detection results. The result supports the hypothesis that the model's accuracy depends at least in part on the characteristics of the data that it is trained on.
[3]	Intrusion Detection in IoT Using Deep Learning	Convolutional Neural Network, Gated Recurrent Unit and Long Short Memory Neural Network.	The use of deep learning in security enhancement for IoT traffic is limited by the need to balance high accuracy with minimal false alarms during communication, which is particularly challenging when using Convolutional Neural Networks (CNNs).
[4]	Deep Learning-Based Time-Series Analysis for Detecting Anomalies in Internet of Things	LSTM (BI-LSTM) and Long Short-Term Memory (LSTM), CNN-based Temporal Convolutional (TCN), and CuDNN-LSTM, which is a fast LSTM implementation.	This approach needs to remove Trends and seasonality in the time-series dataset before modeling. Also, there are long training time to achieve a better performance in terms of model accuracy.
[5]	A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series	A prediction module based on CNN	Poor data quality can corrupt the data modeling phase. On the other hand, if the level of contamination is too high (more than 5%), the system will try to model these instances, thus considering them as normal at the time of inference. Another limitation is the selection of the network architecture and the corresponding hyperparameters.
[6]	A Deep Learning-Based Model for Anomaly Detection in IoT Networks	Convolutional Neural Networks (CNN)	As the model is used to detect anomalies and attacks, which could occur on IOT devices, several attackers could use evasive techniques to prevent detections by the model. Additionally, the CNN3D model which use 3 convolutional layers is less effective than the CNN1D and CNN2D model.
[7]	Extract spatial and temporal features, respectively for anomaly detection in time series data.	An improved network architecture, which builds on a previous method called C-LSTM.	It is stated that the error vector classification results obtained with this method are insufficient to correctly identify anomalous data. In general, this can be seen as a limitation of this approach for anomaly detection in time series.
[8]	Anomaly detection and diagnosis in multivariate time series data.	Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED)	MSCRED relies on the quality of the data used for training the RNNs. If the data is noisy or incomplete, it may affect the performance of anomaly detection. MSCRED is based on a single dataset, which limits the generalization of the results. The RNNs used in MSCRED are a type of black box model which makes it difficult to interpret the results and understand the causes of the detected anomalies.

Our solution builds upon and addresses the drawbacks of existing related work in this domain. Previous studies on anomaly detection in IoT devices have often struggled with handling the real-time nature of data streams and capturing complex temporal dependencies. Additionally, they have faced challenges in achieving high accuracy while maintaining low computational overhead. In contrast,

our TCN-based approach excels in real-time anomaly detection by effectively modeling and capturing temporal dependencies in IoT data streams. By leveraging the power of TCN, we not only achieve remarkable accuracy in detecting anomalies but also ensure low computational complexity, making it suitable for resource-constrained IoT devices. Our solution's ability to overcome the limitations of

previous approaches marks a significant advancement in real-time anomaly detection for IoT devices, paving the way for improved operational efficiency and enhanced device performance.

### 3. METHODOLOGY

In order to address existing drawbacks and create additional opportunities, we propose a real-time anomaly detection in Internet of Things devices using deep learning. Our proposed system leverages cutting-edge technologies to provide a secure and efficient solution in order to address the nascent challenge of detecting anomalies in IoT. Figure 4 shows the workflow of the proposed model. In this section, we will delve into the specifics of the chosen Recurrent Neural Networks (RNNs) model and expound on the rationale behind our assumptions and decisions. RNNs are a type of neural network that are particularly well-suited for processing sequential data. RNNs have a "memory" that allows them to remember previous inputs, which allows them to understand the context and relationships between elements in a sequence. This makes RNNs useful for tasks such as language modeling, speech recognition, and time series prediction. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are both types of RNNs that have been designed to better handle the problem of vanishing gradients, which can occur when training traditional RNNs.

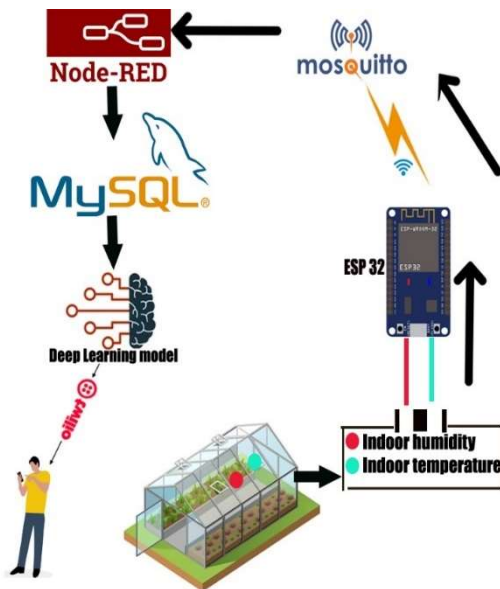


Figure 4: The workflow of the proposed model

#### 3.1 Long Short-Term Memory (LSTM)

Firstly, Long Short-Term Memory (LSTM) is a

type of Recurrent Neural Network (RNN) that is designed to better handle the problem of vanishing gradients, which can occur when training traditional RNNs. LSTMs have a more complex structure than traditional RNNs and use a series of gates to control the flow of information. These gates, called the input gate, forget gate, and output gate, allow LSTMs to selectively choose which information to remember and which to discard, enabling them to preserve information over long periods of time. This makes LSTMs useful for tasks such as language modeling, speech recognition, and time series prediction where the output at a certain time step is dependent on the previous time steps. LSTMs are also used in many other applications such as natural language processing, speech synthesis, and machine translation.

The architecture of a Long Short-Term Memory (LSTM) network is composed of a series of memory cells, input gates, forget gates, and output gates as seen in figure 5. Each memory cell is responsible for maintaining a state that can be used to remember information over long periods of time. The input gate, forget gate, and output gate are used to control the flow of information into and out of the memory cell [9].

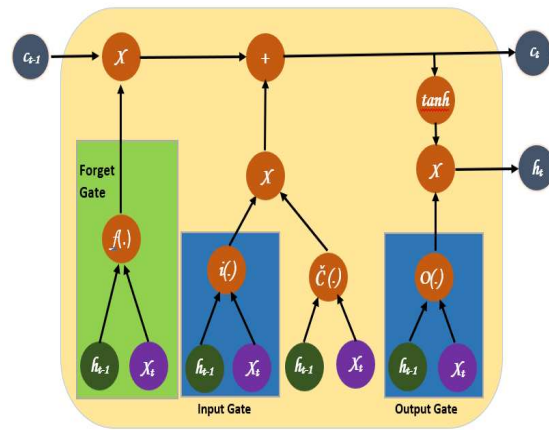


Figure 5: The architecture of a basic LSTM cell

The forget gate  $f(.)$  is used to decide which information from the previous state should be discarded.

$$F(.) = \sigma (W_f x_t + W_f h_{t-1} + b_f) \quad (1)$$

The input gate  $i(.)$  is used to decide which information from the current input should be passed to the memory cell.



$$ht = g ( Whx Xt + Whh ht-1 + bh ) \quad (2)$$

The output gate  $o(\cdot)$  is used to decide which information from the current state should be passed on as the output.

$$o(\cdot) = \sigma ( Wox Xt + Woh ht - 1 + bo ) \quad (3)$$

The LSTM network also has a hidden state, which is passed from one-time step to the next, allowing the network to maintain information over longer periods of time. This hidden state is also passed through a fully connected layer to produce the final output.

LSTMs are often used in deep learning tasks such as language modeling, speech recognition, and time series prediction, where the output at a certain time step is dependent on the previous time steps.

### 3.2 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is a type of Recurrent Neural Network (RNN) that is similar to Long Short-Term Memory (LSTM) in that it is designed to handle the problem of vanishing gradients (Bibi et al., 2020). However, unlike LSTMs, GRUs have a simpler structure and use a single update gate to control the flow of information. This single update gate allows GRUs to decide what information from the past to keep and what to discard, making it more efficient than LSTMs in terms of computation and memory. GRUs are also useful for tasks such as language modeling, speech recognition, and time series prediction where the output at a certain time step is dependent on the previous time steps. They are also used in many other applications such as natural language processing, speech synthesis, and machine translation. GRUs are often considered a good balance between the performance and computational efficiency, making them a popular choice in many deep learning applications.

The architecture of a Gated Recurrent Unit (GRU) network is composed of two gates: the update gate and the reset gate as in figure 6. These gates are used to control the flow of information into and out of the hidden state of the network [10].

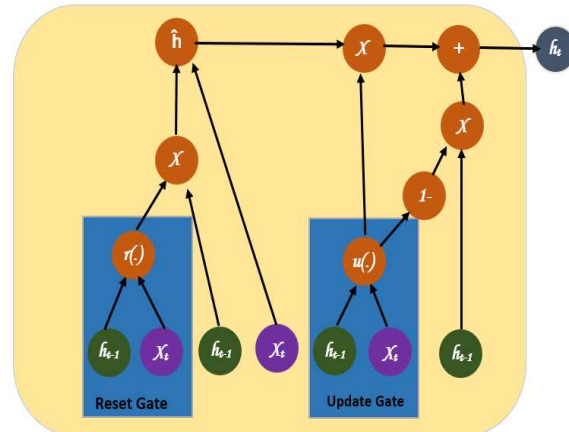


Figure 6: The architecture of a basic GRU cell

The update gate is used to decide how much of the previous hidden state should be passed on to the current hidden state. It ranges from 0 to 1 and the closer to 1, the more information from the previous hidden state is passed on.

The reset gate is used to decide how much of the previous hidden state should be discarded. It also ranges from 0 to 1 and the closer to 1, the more information from the previous hidden state is discarded.

The current hidden state is then computed by a combination of the previous hidden state, the current input and the values of the update and reset gates. This hidden state is then passed through a fully connected layer to produce the final output.

While an LSTM or GRU implementation may seem like the rational choice for the deep learning model, we have implemented two models, the first one with LSTM and the second one with GRU, but we have noticed that they have several weaknesses. Some of the weaknesses of LSTM (Long Short-Term Memory) include the potential for overfitting, difficulty in training with long sequences, and sensitivity to the choice of hyperparameters. Additionally, LSTM models can suffer from vanishing gradients, where the gradients become very small, making it difficult for the model to learn long-term dependencies. Finally, LSTM models can be computationally expensive, making them challenging to train on large datasets. As regards GRU, it includes similar issues as LSTM, such as the potential for overfitting and sensitivity to hyperparameters. Additionally, GRU models may have difficulty learning complex long-term dependencies in some cases. While GRUs were designed to have fewer parameters than LSTMs, they may still be

computationally expensive in certain contexts. Finally, it's worth noting that the effectiveness of GRUs can depend on the specific task and dataset at hand, and they may not always outperform other recurrent neural network architectures.

To address some of the limitations of LSTM and GRU models, the authors have *implemented* Temporal Convolutional Networks (TCN). TCN are a type of deep neural network architecture designed specifically for processing sequential data, such as time series or natural language [11]. The TCN architecture is based on convolutional neural networks (CNNs), but with a few key modifications to better handle the temporal structure of sequential data. A TCN architecture is composed of multiple layers of temporal convolutional blocks, with each block consisting of one or more convolutional layers. The convolutional layers in a TCN use dilated convolutions, which allow for the capture of long-range dependencies in the input sequence by increasing the receptive field of each convolutional kernel. Additionally, the TCN uses causal convolution, meaning that the output of each convolutional layer depends only on the inputs up to that point in the sequence and not on future inputs.

The dilated convolutions and causal convolution in TCN make it well-suited for processing sequential data with long-range dependencies. The network can be trained end-to-end to perform a specific task, such as classification or regression, on sequential input data. The TCN can also be extended with additional layers, such as pooling layers or recurrent layers, to further improve its performance on specific tasks as shown in figure 7.

The main components of a Temporal Convolutional Network (TCN) are: [11]

- **Input layer:** The input layer takes in the sequential data and converts it into a tensor representation that can be processed by the network.
- **Convolutional layers:** The core of the TCN architecture is made up of one or more convolutional layers. These layers use dilated convolutions to capture long-range dependencies in the input data and causal convolution to ensure that the output of each layer depends only on past inputs.
- **Activation functions:** Activation functions are used in each convolutional layer to introduce non-linearity into the network. Common activation functions used in TCN include ReLU, sigmoid, and tanh.
- **Pooling layers:** Pooling layers can be used in

TCN to reduce the spatial dimensions of the output of the convolutional layers, which can help to control overfitting and improve the training time.

- **Output layer:** The output layer takes the processed output of the convolutional layers and performs the final prediction or classification task. The output layer can be a fully connected layer, a softmax layer, or any other type of layer that is appropriate for the task at hand.

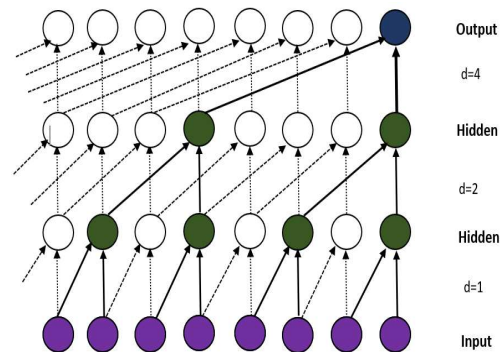


Figure 7: The architecture of a basic TCN cell

#### 4. EXPERIMENTAL RESULTS

Our system was based on research conducted at our university in Casablanca in Morocco. A real mini greenhouse that incorporates numerous sensors, as shown in the figure 8 below, was used to find patterns in our time series data that do not conform to the expected behavior and help us to identify problems with our devices early.



Figure 8: The experimental greenhouse at Hassan II University of Casablanca, Morocco

A stepwise methodology was employed in implementing this research, which involved developing a comprehensive real-time anomaly

detection model using a deep learning approach. This model aims to enhance the accuracy of anomaly detection in the Internet of Things.

### Step 1: Data collection

To collect data, the ESP32 board equipped with a temperature and humidity sensor is utilized. The data is then stored in a MySQL database, with a table that includes the characteristics of time, temperature, and humidity. The data is transferred from the sensor to the MySQL database using Arduino IDE, Mosquitto and Node Red. In order to ensure a consistent and periodic collection of data, the system is set up to capture each list of indoor temperature and indoor humidity data every 10 minutes. The dataset contains 939350 lines.

### Step 2: Data Cleaning

In order to prepare the raw dataset for a DL method, several processing steps are required, which include standardization, normalization, and data cleaning [12]. The entire process is divided into three sub-steps. Firstly, the dataset is standardized to ensure that all data points are on the same scale and have a distribution value between 0 and 1 based on the standard normal distribution. This is crucial to enable accurate comparison and analysis of data. Secondly, data normalization is performed to transform the data and avoid negative values that could be detrimental to neural networks. All data in the dataset are normalized between 0 and 1. Finally, data cleaning is carried out to remove unwanted data, such as NaN and null values. This is necessary to prevent any inaccuracies or errors in the subsequent analysis.

### Step 3: Training, Testing, and Evaluating

During the training, testing, and evaluation of our model, we set aside 20% of the dataset for testing purposes. This helps us to evaluate the accuracy and generalization ability of the model on unseen data. We use the 'RMSE' metric to measure the performance of the model, which is a common evaluation metric used in regression analysis. We also utilize the 'ADAM' optimization function, which is a popular optimization algorithm used to update the model weights during training. In addition, we set the number of epochs to 10, which determines the number of iterations that the training process runs through the entire dataset. Finally, we set the rollback date to 10 minutes, meaning that the model will use the 10 previous time steps as input variables to predict the next time period. Overall, this approach helps us to develop a robust and accurate model for making predictions on future data.

Figure 9 displays the loss value trends during the training and validation stages of constructing a TCN model. The x-axis represents the epoch number, while

the y-axis represents the loss value.

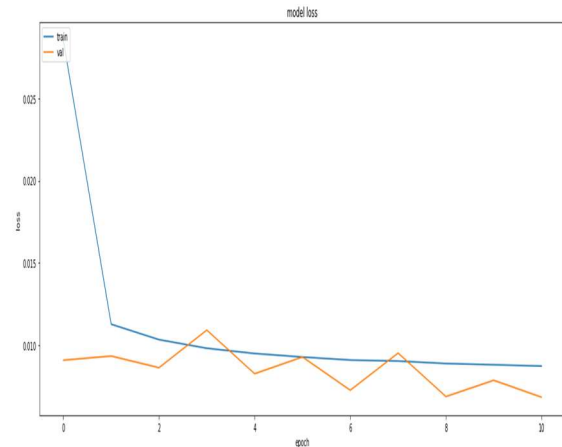


Figure 9: Training and Validation Loss of TCN Model.

The plots given in Figure 10 demonstrate the predicted (colored in blue) and the actual humidity (colored in orange) values for the sensor data studied and obtained by the TCN model.

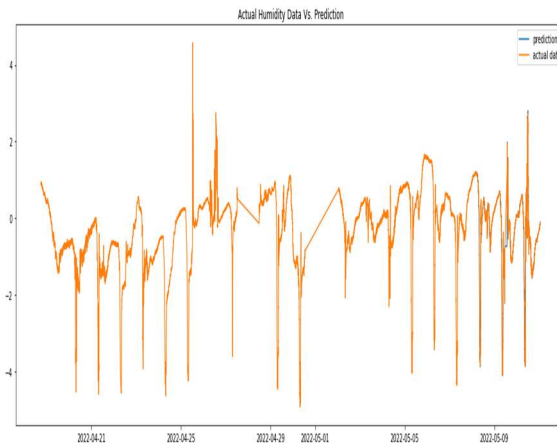


Figure 10: The Original and Predicted humidity Value of TCN Model

The plots given in Figure 11 demonstrate the predicted (colored in blue) and the actual temperature (colored in orange) values for the sensor data studied and obtained by the TCN model.



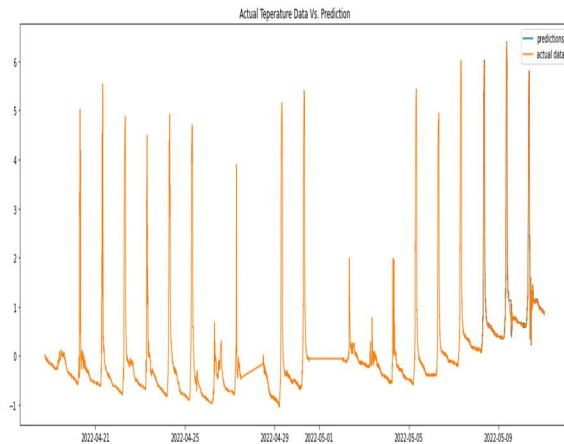


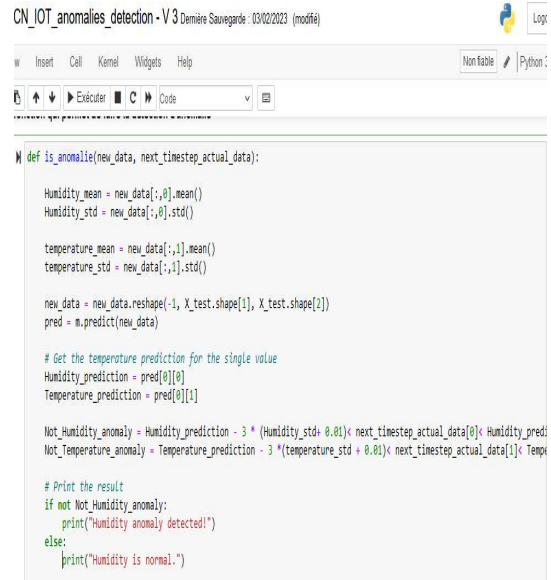
Figure 11: The Original and Predicted temperature Value of TCN Model

#### Step 4: Anomaly detection

In our study, we utilized the Persistence Anomaly Detection (PersistAD) approach for anomaly detection, which involves several steps to detect anomalies in complex systems. First, a time-delay embedding of the data is created to capture the system's dynamics, followed by the application of topological data analysis to identify persistent homology features. To determine the severity of an anomaly, PersistAD employs a threshold based on the persistence level of the homology feature, with higher persistence values indicating more severe anomalies. This allows for the identification of anomalies that are not only statistically significant but also have a significant impact on the system's behavior. The approach also uses a sliding window to account for changes in the system's behavior over time, and it employs a statistical significance test to determine if a detected anomaly is statistically significant or merely a random occurrence. The use of a threshold not only helps to reduce false positives but also provides an interpretable method for detecting anomalies in complex systems. Overall, the PersistAD approach provides a robust and reliable method for detecting anomalies in complex systems with high accuracy.

There are methods to make the anomaly detection threshold adaptive depending on the situation. One of the most common methods is to use data statistics to automatically determine an appropriate threshold. In this paper, the authors used the distribution-based anomaly detection method, which involves using the mean and standard deviation of our data to determine an appropriate threshold. Any value that is greater than a certain number of standard deviations of the mean can be considered an anomaly. The authors used the sliding window to calculate the mean and the

partial and relative standard deviation for the last 3 data entered as shown on figure 12.



```

def is_anomalie(new_data, next_timestep_actual_data):
    Humidity_mean = new_data[:,0].mean()
    Humidity_std = new_data[:,0].std()

    temperature_mean = new_data[:,1].mean()
    temperature_std = new_data[:,1].std()

    new_data = new_data.reshape(-1, X_test.shape[1], X_test.shape[2])
    pred = m.predict(new_data)

    # Get the temperature prediction for the single value
    Humidity_prediction = pred[0][0]
    Temperature_prediction = pred[0][1]

    Not_Humidity_anomaly = Humidity_prediction - 3 * (Humidity_std + 0.01) < next_timestep_actual_data[0] < Humidity_prediction + 3 * (Humidity_std + 0.01)
    Not_Temperature_anomaly = Temperature_prediction - 3 * (temperature_std + 0.01) < next_timestep_actual_data[1] < Temperature_prediction + 3 * (temperature_std + 0.01)

    # Print the result
    if not Not_Humidity_anomaly:
        print("Humidity anomaly detected!")
    else:
        print("Humidity is normal.")
  
```

Figure 12: Detection anomaly function.

#### Step 5: Sending SMS on a mobile phone via Twilio

When anomalies are detected, a real-time SMS notification is sent to the responsible person via Twilio, to promptly alert them and allow them to take necessary measures as shown in figure 13.

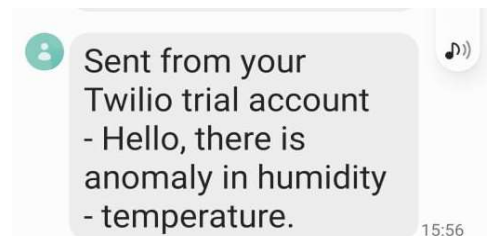


Figure 13: SMS alert from Twilio

## 5. CONCLUSION

The aim of this paper is to evaluate and compare the performance of several machine/deep learning techniques. The study focuses on examining various versions of RNN-based models such as LSTM and GRU, and comparing them with a CNN-based model known as TCN. The primary motivation behind this research is to identify the fundamental dissimilarities in the performance of CNN and RNN models.

Based on our findings, we note that TCN

demonstrates relatively strong anomaly detection capability with lower RMSE values. A noteworthy feature of TCN is its relatively shorter training time when compared to RNN models. In contrast, the fast version of LSTM exhibits the highest level of accuracy, but necessitates a longer training duration. As a potential avenue for future research, it is essential to compare the outcomes of these models with different datasets of varying sizes and assess the trade-off between them.

In a promising outlook, we plan to explore the possibilities of detecting anomalies from multiple sensors simultaneously. By focusing on multi-sensor anomaly detection, we aim to gain a deeper understanding of the complex interactions and correlations between different sensor data streams. By developing techniques that can effectively analyze and identify anomalous patterns across multiple sensor streams, we aim to enhance the overall anomaly detection capabilities and provide a more comprehensive understanding of the IoT system's health.

#### REFERENCES:

- [1] Chandola, V., Banerjee, A. et Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3), pages 1-58.
- [2] Al-Shabi, Mohammed & Abuhamdah, Anmar. (2021). Using deep learning to detecting abnormal behavior in internet of things. *International Journal of Electrical and Computer Engineering*. 12. 2108-2120. 10.11591/ijece.v12i2.pp2108-2120.
- [3] Banaamah, Alaa & Ahmad, Iftikhar. (2022). Intrusion Detection in IoT Using Deep Learning. *Sensors*. 22. 8417. 10.3390/s22218417.
- [4] Gopali, Saroj & Siami Namin, Akbar. (2022). Deep Learning-Based Time-Series Analysis for Detecting Anomalies in Internet of Things. *Electronics*. 11. 3205. 10.3390/electronics11193205.
- [5] Munir, Mohsin & Siddiqui, Shoaib & Dengel, Andreas & Ahmed, Sheraz. (2018). DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access*. PP. 1-1. 10.1109/ACCESS.2018.2886457.
- [6] Ullah, Imtiaz & Mahmoud, Qusay. (2021). A Framework for Anomaly Detection in IoT Networks Using Conditional Generative Adversarial Networks. *IEEE Access*. PP. 1-1. 10.1109/ACCESS.2021.3132127.
- [7] Yin, Chunyong & Zhang, Sun & Wang, Jin & Xiong, Naixue. (2022). Anomaly Detection Based on Convolutional Recurrent Autoencoder for IoT Time Series. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 52. 112-122. 10.1109/TSMC.2020.2968516.
- [8] Zhang, Chuxu & Song, Dongjin & Chen, Yuncong & Feng, Xinyang & Lumezanu, Cristian & Cheng, Wei & Ni, Jingchao & Zong, Bo & Chen, Haifeng & Chawla, Nitesh. (2018). A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data.
- [9] ElMoaqet, Hisham & Eid, Mohammad & Glos, Martin & Ryalat, Mutaz & Penzel, Thomas. (2020). Deep Recurrent Neural Networks for Automatic Detection of Sleep Apnea from Single Channel Respiration Signals. *Sensors (Basel, Switzerland)*. 20. 10.3390/s20185037.
- [10] Bibi, Iram & Akhunzada, Adnan & Malik, Jahanzaib & Iqbal, Javed & Musaddiq, Arslan & Kim, Sung. (2020). A Dynamic DL-Driven Architecture to Combat Sophisticated Android Malware. *IEEE Access*. PP. 1-1. 10.1109/ACCESS.2020.3009819.
- [11] Hewage, Pradeep & Behera, Ardhendu & Trovati, Marcello & Pereira, Ella & Ghahremani, Morteza & Palmieri, Francesco & Liu, Yonghuai. (2020). Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*. 24. 10.1007/s00500-020-04954-0.
- [12] Peterson, J.M., Leevy, J.L., & Khoshgoftaar, T.M. (2021). *A Review and Analysis of the Bot-IoT Dataset*. 2021 *IEEE International Conference on Service-Oriented System Engineering (SOSE)*, 20-27.