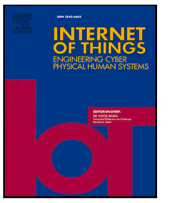
Internet of Things 23 (2023) 100848

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Research article

An unsupervised TinyML approach applied to the detection of urban noise anomalies under the smart cities environment

Sahibzada Saadoon Hammad [a], Ditsuhi Iskandaryan [b,c], Sergio Trilles [a,*]

[a] Institute of New Imaging Technologies, Universitat Jaume I, Avd. Sos Baynat, s/n, 12071, Castelló de la Plana, Spain
[b] Oncology Data Analytics Program, Catalan Institute of Oncology (ICO), Avinguda de la Gran via de l'Hospitalet, 199, 08908, Barcelona, Spain
[c] Colorectal Cancer Group, ONCOBELL Program, Institut de Recerca Biomedica de Bellvitge (IDIBELL), Avinguda de la Gran via de l'Hospitalet, 199, 08908, Barcelona, Spain

ARTICLE INFO

Keywords:
TinyML
Artificial Intelligence of Things
Anomaly detection
Environmental noise
Microcontroller

ABSTRACT

Artificial Intelligence of Things (AIoT) is an emerging area of interest, and this can be used to obtain knowledge and take better decisions in the same Internet of Things (IoT) devices. IoT data are prone to anomalies due to various factors such as malfunctioning of sensors, low-cost devices, etc. Following the AIoT paradigm, this work explores anomaly detection in IoT urban noise sensor networks using a Long Short-Term Memory Autoencoder. Two autoencoder models are trained using normal data from two different sensors in the sensor network and tested for the detection of two different types of anomalies, i.e. point anomalies and collective anomalies. The results in terms of accuracy of the two models are 99.99% and 99.34%. The trained model is quantised, converted to TensorFlow Lite format and deployed on the ESP32 microcontroller (MCU). The inference time on the microcontroller is 4 ms for both models, and the power consumption of the MCU is 0.2693 W ± 0.039 and 0.3268 W ± 0.015. Heap memory consumption during the execution of the program for sensors TA120-T246187 and TA120-T246189 is 528 bytes and 744 bytes respectively.

1. Introduction

According to [1], by 2025 more than 16.44 billion Internet of Things (IoT) devices will be connected and the number of mobile IoT connections is expected to exceed 30.9 billion. IoT devices (also referred to as "connected devices") have been democratised and integrated into everyday objects, such as household appliances, cars and street furniture, among others [2,3]. IoT devices provide a wide variety of applications i.e. from smart homes, healthcare, behavioural studies to industrial systems and assisting in disaster management and mitigation [4,5]. IoT devices are programmed to define behaviour based on their capabilities: to observe and act [2]. These devices generate a continuous pulse (called logs [6]) of the almost endless activities taking place in physical space, and data streams (depending on the particular refresh time) may deliver huge numbers of data observations [7]. However, most of these IoT devices are built using low-cost components [8,9] and are located in hostile locations [10,11]. This situation can facilitate the appearance of errors during the data capture process [12] or while performing actions. In addition, transmission errors can affect communications [13], either by flaws in the technology in which they are deployed or by malicious attacks [14]. In real-time decision-making processes [7] anomalous values or faults (abnormal behaviours) [15] can lead to bad decisions and cause

* Corresponding author.
E-mail addresses: hammad@uji.es (S.S. Hammad), iskandar@uji.es (D. Iskandaryan), strilles@uji.es (S. Trilles).

https://doi.org/10.1016/j.iot.2023.100848
Received 30 April 2023; Received in revised form 28 May 2023; Accepted 14 June 2023
Available online 30 June 2023

unexpected social damage and economic loss, especially in critical contexts (e.g. Internet of Medical Things or Internet of Vehicles [IoV]).

The underlying strategy for most approaches to anomaly detection is to model expected behaviour and exploit this knowledge to identify deviations (anomalies) [16]. Throughout this research, an anomaly is understood as a value that differs from the valid ones since it contains a certain number of erroneous values. These observations are discordant and do not conform to the expected behaviour [17]. Detecting anomalies is not trivial and presents different challenges that must be considered and solved [18]. These challenges can be summarised as follows: (a) the border between normal and abnormal behaviour is not precise, and thus anomalous can be very close to normal; (b) the anomalies originate from malicious actions; (c) normal behaviour can change depending on the context; and (d) data are often interspersed with noise that tends to be quite similar to actual anomalies.

IoT systems need to have an anomaly detection mechanism in place to detect anomalies in the data in real time. There could be different anomalies in different contexts, but it is first essential to define anomalies. A detailed survey study about anomaly detection defines anomaly as something in the data distribution that deviates from normal data patterns. Anomalies can be of different types: (a) point anomalies, which are values in the data that do not adhere to the normal distribution of the data, (b) collective anomalies are a collection of instances in the data which are anomalous in behaviour, and (c) contextual anomalies are more related to the context in which the data is being collected. The data values could be anomalies in a certain context, but in another context, they might be normal values [18]. The detection and classification of anomalies in IoT logs are a popular research topic at present [19]. The first approach was simple, mainly involving the addition of nodes redundantly [20]. Due to their high installation cost, this approach was replaced by mathematical models (mostly statistical), which detected errors when quantifying the degree of relationship between the measured and predicted distributions [21]. However, over time, more complex IoT platforms have been built that include a variety of sensors/actuators that differ in intrinsic features, such as type, manufacturer, grade of quality, accuracy, stability, deterioration over time or external factors [22]. As a result, statistical models became impractical for these heterogeneous systems [23]. They were replaced by approaches based on Machine Learning (ML) methods [24,25], such as Neural Networks (NN) [26], Principal Component Analysis (PCA) [27] or Random Forest [28], among others.

Due to the rapid evolution of IoT devices, they can perform more complex computational operations [29]. This fact has allowed more advanced analysis algorithms to be generated in the edge computing layer [30]. With this, it has become possible to apply ML analysis in the lower layers of an IoT architecture, thereby preventing data from being transferred to higher layers. This helps to perform actions faster since they are not exposed to latencies or high computing loads that are current cloud-level problems [31]. As a result, the concept called Artificial Intelligence of Things (AIoT) [32] has appeared, bringing together the paradigms of Artificial Intelligence (AI) and IoT. Traditionally, ML analytics have been supported by cloud computing. In an IoT architecture, cloud computing sits on the far side of the physical devices. This configuration may not provide the desired Quality of Service (QoS) properties because high latencies are generated [33]. This is due to the long path that large data flows must follow. In many scenarios, these devices need to take action after the result of the analysis. In this case, the connection is not just the path to the cloud but also the path back to the device. This situation is even more compromised when the IoT application requires real-time analytical capabilities [34], especially when working with critical data, such as emergency response, health monitoring or smart assistants. In addition to those mentioned, analysis in the cloud presents other challenges, such as high energy consumption and privacy or reliability problems. Linked to the AIoT paradigm, a new concept has emerged with great force: TinyML [35]. TinyML aims to generate ML models optimised to be executed on the same IoT devices. This work considers Microcontroller Units (MCUs) to be IoT devices. In this way, TinyML allows ML models to be implemented in the same place (at the location of the IoT device) where the information is generated or the decision must be applied.

The main contributions of our proposed study are: (a) Anomaly detection system for urban noise time series data (b) Implementation and deployment of an LSTM model on an embedded device (MCU) with computational resource restrictions (c) Exploring and analysis of performance and computation metrics for TinyML on ESP32 S3 MCU.

The remainder of the paper is organised as follows. Section 2 presents the related works and main concepts. Section 3 introduces a general overview of the methodology implemented in this study for anomaly detection and deployment of the quantised model on the MCU, whereas Section 4 describes the materials and models used to achieve the goal. Experiments and results are shown and discussed in Section 5. Finally, Section 6 summarises the main achievements and indicates future work.

2. Background and state of the art

2.1. Anomaly detection

Anomaly detection approaches based on ML methods can be classified by the type of data required to train the model. The creation of ML models needs two steps, i.e. training and testing. Accordingly, three different strategies are defined depending on whether the data available are labelled: supervised learning (labelled), unsupervised learning (unlabelled) and semi-supervised learning (incompletely labelled). In most use cases, anomalous samples are expected to represent a tiny percentage of the whole data set. Therefore, even when labelled data is available, more normal data samples are available than abnormal ones. Unlike statistical approaches, which focus on understanding the process that generates the data, ML methods are based on building a mechanism that improves its precision by considering previous results [36]. Despite their success, however, ML methods involving manual feature extraction have two significant flaws. First of all, these methods can only work on data with prior processing. They operate with features extracted from the data, which requires a high computational cost and a considerable effort by experts in the node domain. To build them, an expert in the field must consider the peculiarities of the sensors to be analysed in order to control and manually

adjust the input to the models [37]. Second, since these approaches are tailored to specific types of data captured by the sensors, they are sensitive to the slightest changes in the system or data [38].

DL methods are a substantial improvement over ML methods [39]. DL is a subfield of ML that uses several non-linear processing layers to abstract and transform discriminative or generative features for pattern analysis. Currently, DL applications to IoT systems have become an imperative research topic [40]. Compared to traditional ML methods, DL methods have considerably improved more modern applications [41]. The most important advantage of DL over ML is its performance on large data sets. This feature is relevant in a scenario such as an IoT system, which produces a large amount of data [42]. IoT data (logs) are increasingly dimensional (e.g. multivariate data sets), and anomaly detection may require joint modelling of the interactions between each pair of variables. For these situations, DL approaches such as automatic encoders, Variational Automatic Encoders (VAEs), sequence-by-sequence models and Generative Adversarial Networks (GANs) are widely used [43]. In general, DL has clear benefits over ML, although there are also some disadvantages to be considered [44], such as (a) latency: compared to ML linear models, DL models have significant latency associated with inference; (b) data requirements: DL models generally require an extensive data set for effective training. These models are also prone to overfitting and must be carefully evaluated to address this risk; and (c) distribution change management: in many scenarios, the underlying process that generates data can legitimately change such that a previously anomalous data point becomes normal. Changes can be gradual, cyclical or even abrupt in nature. This phenomenon is known as conceptual drift [45].

In the literature, different works have used the aforementioned techniques. For instance, in [46] the authors use Recurrent NN (RNN) to detect anomalies in data collected from environmental sensors. Gated Recurrent Units (GRU) and LSTM models with different configurations, i.e. using different numbers of hidden cells, and dropout rates to evaluate the performance of the anomaly detection system. The study proposed a distributed anomaly detection system where different anomaly detection tasks are distributed over different edge devices (with high performance) instead of computing it on a cloud.

LSTM and Multivariate Linear Regression (MLR) have been used in smart agriculture systems to detect anomalies [47]. In this work, the authors collected time series data from environmental sensors such as atmospheric temperature, humidity, pressure, soil temperature, soil moisture, wind speed and wind direction for time series modelling and used the proposed methods to detect anomalies in the data. The study results conclude that the LSTM model performed better than the MLR technique. An anomaly detection system for smart home sensor network data such as temperature, humidity and light has been deployed using different ML techniques [48]. This study focuses on the detection of point, contextual and combined anomalies using LSTM, One Class Support Vector Machine (OCSVM), Isolation Forest and Convolutional NN (CNN).

Anomalies have been detected in indoor air quality data by using an LSTM autoencoder [49]. The work focuses on the detection of point anomalies in the data by training the LSTM autoencoder on normal instances in the time series. The authors propose the threshold should be the maximum reconstruction error between the original sequence and model output, and values above the threshold are classified as anomalies.

Regarding the detection of anomalies in data from noise sensors, such anomalies are divided into four types [50]: (a) Due to an abrupt fault in the sensor device, e.g. the sensor is affected by water; (b) An incipient fault in the device, which results in slow and unnoticeable malfunctioning of the device system, which is therefore not easily identifiable at the outset; (c) An intermittent fault that occurs due to environmental surroundings. Examples of possible causes include extreme temperatures, rains, etc., and the measurements during these periods can be anomalous; (d) An anomaly due to unexpected or rare events in the surroundings. This study uses a multi-criteria assessment model to detect anomalies in urban noise sensor networks. Different criteria are taken into account, and each criterion has a quality score between 0 and 1. All the criteria are then combined using an Ordered Weighted Average (OWA) to obtain a global quality score, which can then be used to classify sound values into normal and anomalous.

2.2. AIoT: Edge computing and AI

From the literature, edge computing can be defined as the computing and processing of data near the source where it has been created. IoT devices generate an enormous amount of data. Even though cloud computing is faster and more efficient for processing data compared to edge computing, the problem lies in transferring large volumes of data to the cloud due to bandwidth issues causing high data latency; hence the need for edge computing to process data at the edge of the network in order to mitigate these issues [51].

TinyML can be defined as running ML algorithms on ultra-low-power MCUs and getting inferences on data at the source (edge) [52]. TinyML helps IoT devices to become intelligent by executing ML algorithms on them. In a typical IoT framework, the transfer of large amounts of data produced by IoT devices requires higher bandwidth. The computation of data at the edge helps reduce data transmission costs, and thus requires low bandwidth. It also provides low latency, privacy and data security by shifting the data processing to the edge [53]. While TinyML has a series of benefits, it comes with challenges and restraints that it is important to consider while designing and executing ML algorithms on MCUs. Microcontrollers have low processing capacity and low memory issues, i.e. flash and Static Random-Access Memory (SRAM). Other constraints include high cost for massive deployment and energy efficiency because of their battery requirements in order to perform ML tasks [54].

In [55] the authors applied several NN architectures like Deep NN, CNN, LSTM, GRU, Convolutional RNN (CRNN) and Depthwise Separable CNN (DS-CNN) for keyword spotting by deploying them on a Cortex-M7 based STM32F7 46G-DISCO development board using Common Microcontroller Software Interface Standard NN (CMSIS-NN) kernels. The architectures have been optimised for memory and computational power to allow their deployment on microcontroller systems. CNN as TinyML has been able to achieve

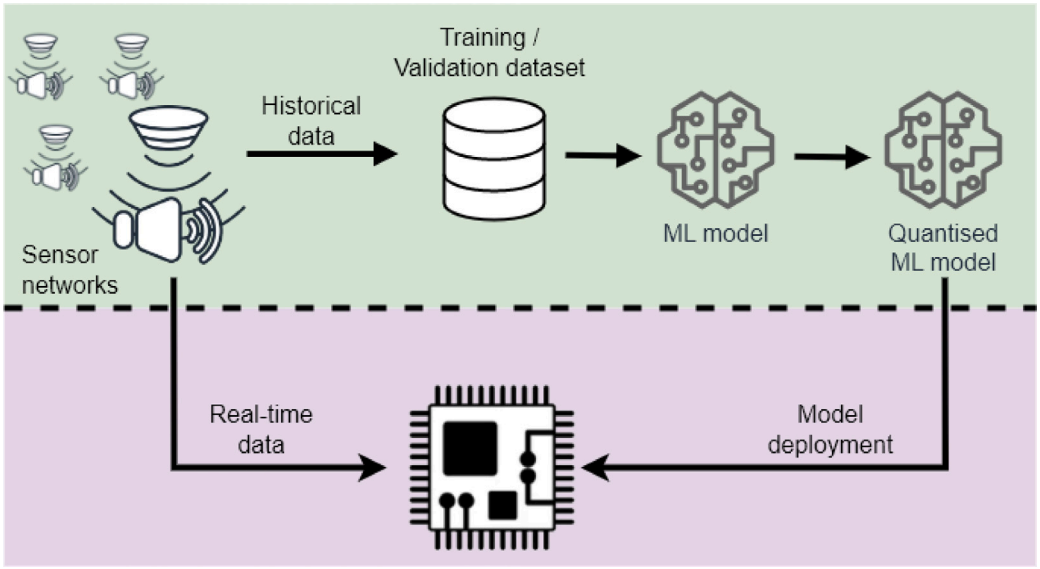


Fig. 1. General overview of the proposed methodology.

an accuracy of 96.4% for hand gesture recognition using an on-board wrist band. The trained model is deployed on Arduino Nano with 1 MB Flash memory and 256 KB RAM [56].

TinyML has been used for the application of anomaly detection in industrial systems. An autoencoder and variational autoencoder model is used to detect anomalies in balanced and unbalanced laundry loads. The experiments use an Arduino Nano 33 BLE for the deployment of the two NN. An autoencoder is able to detect anomalies with an accuracy of 92% compared to a variational autoencoder. The authors also claim the battery time for the embedded application with the autoencoder is 14.9% less than that of a Light Emitting Diode (LED) blink application [57]. After analysing the related literature, it should be noted that research work has yet to be identified that detects urban noise data anomalies and applies appropriate techniques for running them on microcontrollers.

3. Methodology

This section outlines the methodology used in this work. Fig. 1 shows a general overview of the methodology adapted for creating and running ML models on an MCU using TinyML techniques.

Historical data from the urban noise sensor networks are fed into NN for training. The data contain only normal samples so that the NN learns only the pattern and behaviour of normal data. The data are split into training and testing sets. After training and testing the model, it must be converted to TensorFlow Lite format. This conversion is an essential step as it involves the quantisation of the model parameters and helps produce a lightweight version of the model. Quantisation involves the reduction of model size without much loss in the accuracy of the results. There are different types of optimisations on quantisation supported by the main TinyML libraries (TensorFlow Lite), i.e. full integer quantisation, dynamic range quantisation, and float16 quantisation.[1] For the quantised model to be used by the Microcontroller, it needs to be converted to CC code file (C++ Source Code File). Once the CC file has been generated and the microcontroller application code is ready to be flashed on the MCU unit, it can be used to perform inferences on the input. The input data for the inference can be fed directly to the microcontroller from the PC via the Universal Asynchronous Receiver–Transmitter (UART) controller provided in the ESP32 DevKit.

4. Materials and methods

This section provides a detailed description of the implementation of the applied methods.

4.1. Exploratory data analysis

To understand the urban noise phenomenon, this study uses data collected from an urban noise sensor network located in the area of Helsinki, Finland. The data are collected as part of the mySMARTLife EU project [58].

[1] Tensorflow: http://bitly.ws/DDpU [Online; last accessed 10 March 2023].

Fig. 2. Geo-location of urban noise sensors in the network.

Fig. 2 shows the locations of each sensor. Some of the sensors on this list are online, including TA120-T246184 (Mäkelänkatu), TA120-T246187 (Lapinlahden sairaala) and TA120-T246191 Kalevankatu 39.[2] Exploratory data analysis is performed to obtain knowledge about the urban noise data for our use case.[3] This step helps us to understand the context of the observing phenomenon. We used data from the year 2019 and selected two sensors with ids TA120-T246187 (Lapinlahden sairaala) and TA120-T246189 (Korkeasaari). This step is crucial in helping to label the data, as will be explained in Section 4.2.2.

Fig. 3 shows the distribution of varying sound levels for the sensor with id TA120-T246189. The graph below shows the data for the month of March. The time series follows a clear diurnal pattern, i.e. the noise levels are high during the daytime and at peak hours, while noise levels are comparatively low during the night. However, it is important to notice that the noise levels from 10 March to 14 March show an unusual behaviour, i.e. they are different from the normal distribution of the data, which indicates an anomaly in the data. The data during the highlighted period do not follow a normal pattern compared to the rest of the time series.

Fig. 4 shows the visualisation of normal data. The time series indicates a daily pattern with low values during the night and early hours of the morning falling below 50 A-weighted decibels (dBA). Higher sound levels during the daytime hours are recorded, showing the indication of activity in the sensor surroundings. In Fig. 5 the period from 10 to 14 March shows the anomalous samples. It can be seen that the data samples start to deviate from the normal trend around 6 am on 10 March and the values drop to less than 30 dBA. Similarly, for 11 March, there are few peaks in the data while for the rest of the day the values are more or less similar, i.e. below 30 dBA, which is unusual. The values range between 28.5 dBA and 29.5 dBA on 12 March, which clearly shows the deviation from the normal pattern.

Statistics calculated for data in Table 1 on 10–14 March have a mean of 31.50 as compared to the mean of the normal set (51.15). Other statistical values also show a clear difference between normal and anomalous data.

4.2. Data preprocessing

Data preprocessing is an essential stage in ML techniques to prepare the data for model training. The preprocessing techniques used in this work are data normalisation and data labelling.

4.2.1. Data normalisation

The data values in the time series need to be normalised before being fed into the NN. We experimented with different data normalisation techniques and found that MinMax Scaler is the most suitable in the context of our data. MinMax Scaler normalises the original data by scaling it to a range between [0,1] (Eq. (1)).

[2] Noise Sensors: https://bit.ly/3RyqPeb [Online; last accessed 6 February 2023].

[3] Github: http://bitly.ws/DLe2.

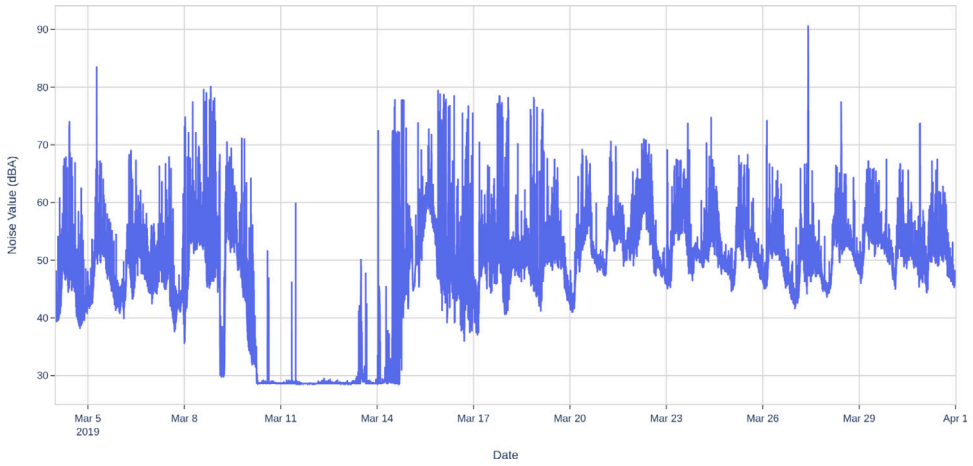


Fig. 3. Lineplot of urban noise level.

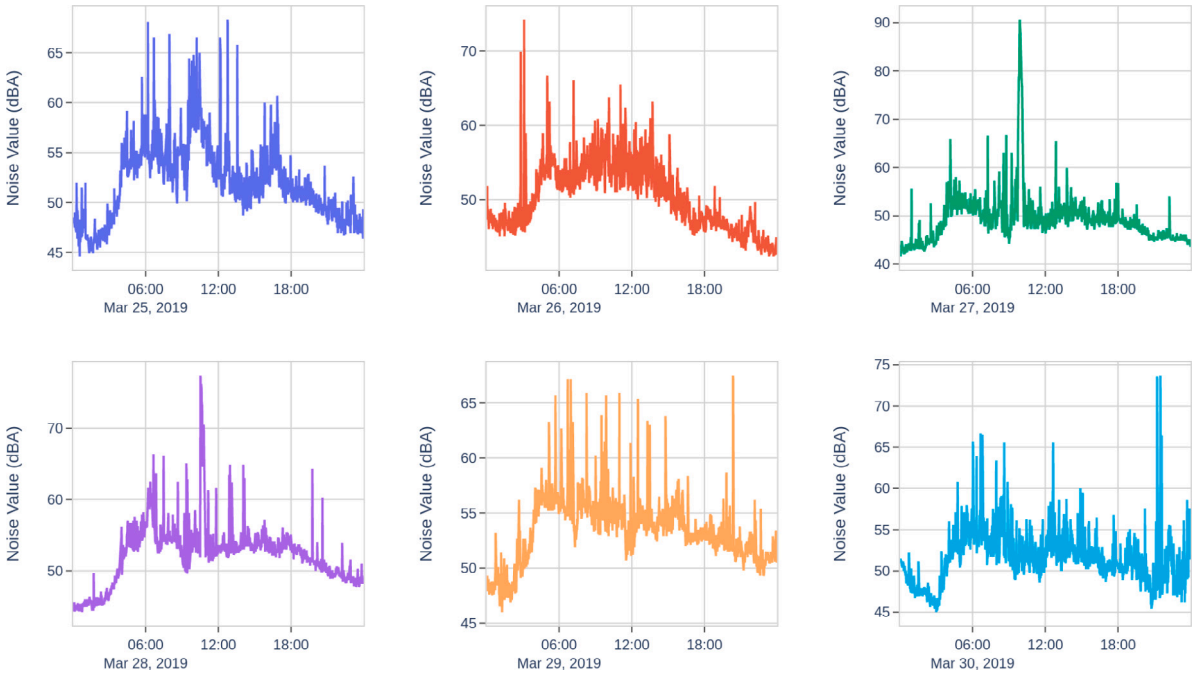


Fig. 4. Daily urban noise trend during normal days.

Table 1
Noise sensor data stats.

Statistical properties	Anomaly set	Normal set
Mean	31.49	51.15
Standard Deviation	7.45	5.81
Minimum	28.50	29.80
25th Percentile	28.60	47.70
50th Percentile	28.70	50.90
75th Percentile	29.0	54.40
Maximum	77.80	90.60

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

X denotes the data point in the time series. X_{min} and X_{max} denote minimum and maximum values in the time series, respectively.

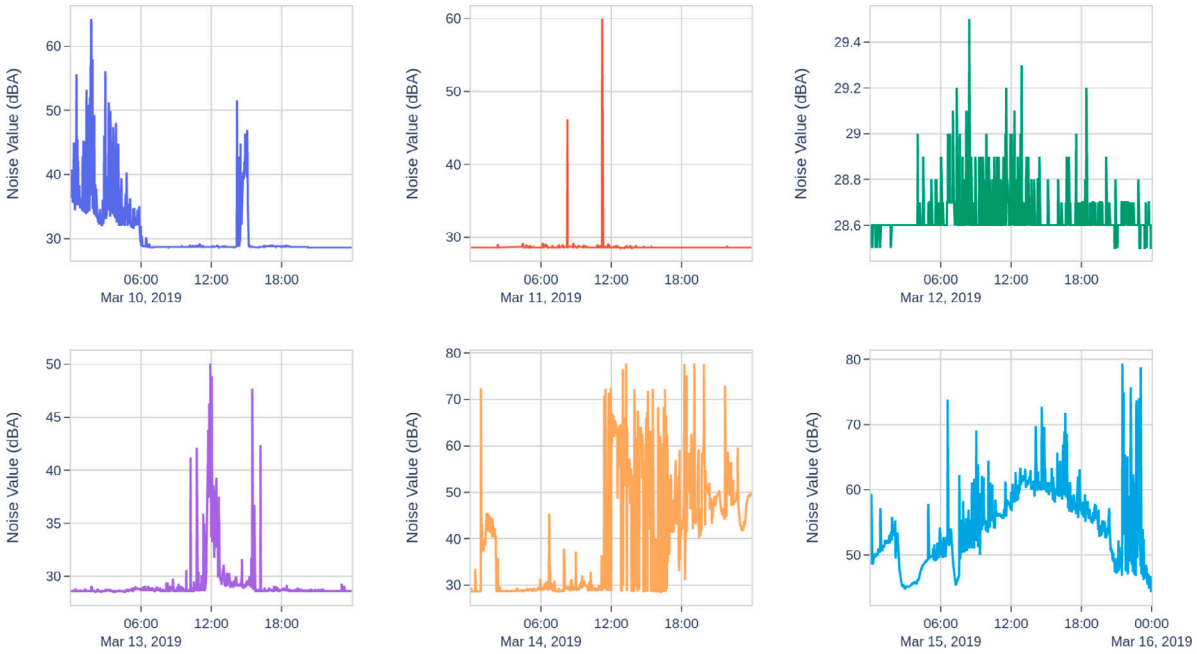


Fig. 5. Daily urban noise trend during anomalous days.

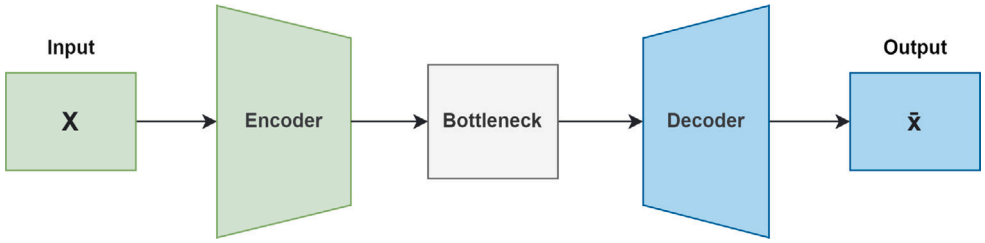


Fig. 6. Autoencoder.

4.2.2. Data labelling

Although we used an unsupervised LSTM to perform anomaly detection, we needed to label our data to evaluate our model and calculate the different performance metrics described in Section 5.3. Due to the lack of available labelled data sets in our use case, anomaly detection in urban noise networks, we labelled the anomalies in the test data set manually based on exploratory data analysis.

4.2.3. Training and testing data

The sensors were selected after exploring the time series data and identifying the type of anomalies that can be helpful in the study. The time series of the sensor with id TA120-T246187 was used to identify point anomalies. Data from the sensor with id TA120-T246189 was used for collective anomaly detection. For the sensor with id TA120-T246187, the total duration of the data is one year. We trained the model with nine months of normal instances of data, and the model was evaluated with three months of data containing both normal and anomalous instances. The sensor with id TA120-T246189 was trained with three months of normal data, and the model was evaluated using one month of the test data set.

4.3. Model

An autoencoder is a type of NN which consists of an encoder, decoder and a bottleneck layer, as shown in Fig. 6. The encoder part takes the input data and produces a low-dimensional, meaningful encoded feature vector at the bottleneck. The decoder takes the encoded feature and reconstructs the input sequence. The reconstruction error measures the error between the original input and the output produced by the decoder.

We used an LSTM Autoencoder to detect anomalies in the urban noise data collected from the sensor network. LSTM is a type of RNN that enables the network to remember inputs over a long period [59]. RNNs have been used for time series and sequential

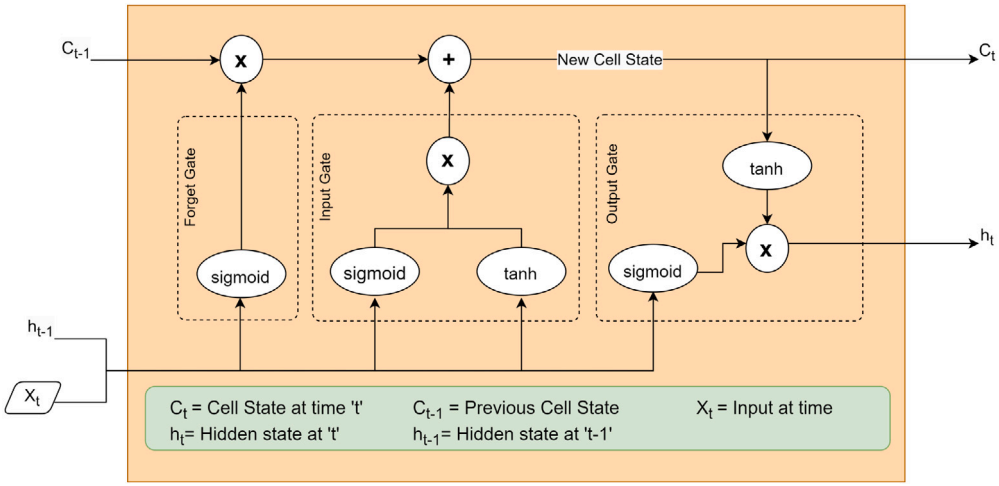


Fig. 7. LSTM architecture.

data as they are good at understanding the sequences in the data. A typical LSTM network consists of three gates: a forget gate, an input gate and an output gate 7.

As its name suggests, a forget gate can decide which information from the long-term memory needs to be forgotten based on the previous hidden state h_{t-1} by passing it through a *sigmoid* function (Eq. (3)). The input gate (Eq. (2)) receives the new input data X_t and previous hidden state h_{t-1} and passes it through a *tanh* function. The second function in the input gate is to process the new input X_t through a *sigmoid* function based on the previous hidden state h_{t-1} and it then decides which information from the new input is to be remembered. Here the results from the *tanh* and *sigmoid* functions are multiplied and then added with the previous cell state from the forget gate, which results in an updated cell state C_t (Eq. (5)).

The output gate determines the new hidden state h_t (Eq. (4)). There are two processes in the output gate. First, the result of the input gate, i.e. the updated cell state C_t is passed through a *tanh* activation function. Second, new input X_t and previous hidden state h_{t-1} are passed through a *sigmoid* function. The result of these two functions is pointwise multiplied, which gives the new hidden state h_t (Eq. (6)). Overall, LSTM can be defined by the following equations [59].

$$i_t = \sigma(W_i^X X_t + W_i^h h_{t-1} + b_i) \tag{2}$$

$$f_t = \sigma(W_f^X X_t + W_{fh} h_{t-1} + b_f) \tag{3}$$

$$o_t = \sigma(W_o^X X_t + W_o^h h_{t-1} + b_o) \tag{4}$$

$$C_t = f_t \circ C_{t-1} + i_t \circ tanh(W_c^X X_t + W_c^h h_{t-1} + b_C) \tag{5}$$

$$h_t = o_t \circ tanh(C_t) \tag{6}$$

where i_t is the input gate, f_t is the forget gate and o_t is the output gate, W is the weight matrix, X_t is the current input data, h_{t-1} is the previous hidden output, C_t is the cell state and $\circ$ is the Hadamard product.

5. Experiments and results

5.1. Model architecture

As described in Section 4.1, data from two different sensors were used to detect different kinds of anomalies. Two different LSTM architectures, i.e. one for each sensor, were chosen to conduct experiments on the data. The architecture and hyperparameters used in the experiment are given in Table 2.

Fig. 8(a) and (b) show the comparison between different model sizes. It can be seen that the size of the TensorFlow lite model is significantly reduced compared to the original TensorFlow model. Full integer quantisation is performed, which involves the quantisation of all operations of the model, including input and output, to make it compatible with the microcontroller.

Table 2
Model architecture and parameters.

Hyperparameters	Sensors	
	TA120-T246187	TA120-T246189
No. of Layers	1	1
No. of Neurons	16	32
Batch Size	32	32
Learning Rate	0.00001	0.0001
Activation Function	tanh	swish

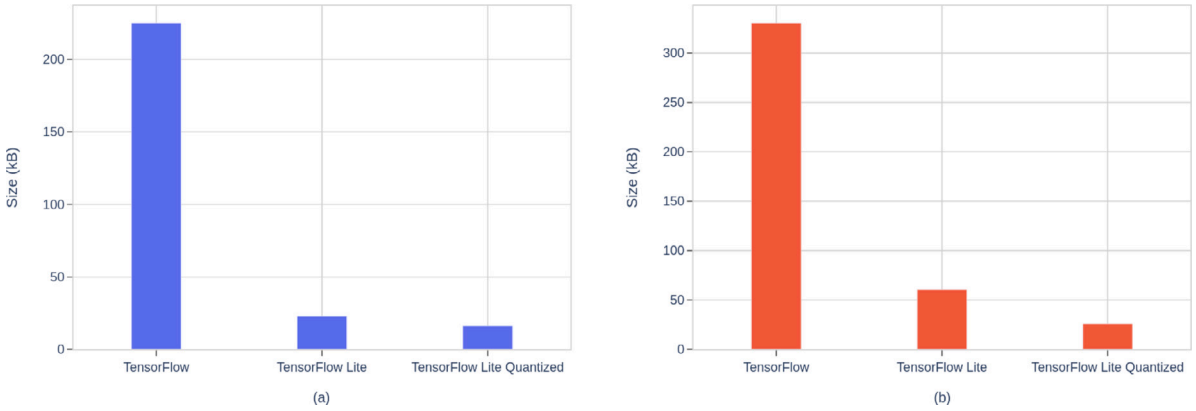


Fig. 8. Size comparison of TensorFlow Models (a) sensor with id TA120-T246187; (b) sensor with id TA120-T246189.

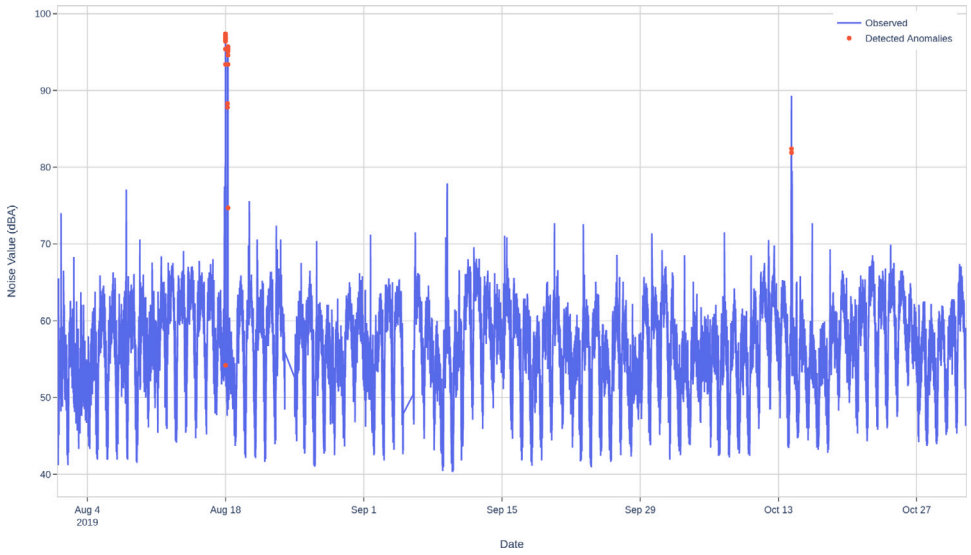


Fig. 9. Anomalies detected in the sensor with id TA120-T246187.

5.2. Detected anomalies

The data used for this study contain two types of anomalies, i.e. point anomalies and collective anomalies. Fig. 9 shows point anomalies detected by the autoencoder model with id TA120-T246187. Very few normal data points are detected as anomalies by the autoencoder model, which is detailed in Section 5.3. Fig. 10 shows collective anomalies detected by the autoencoder model for the sensor with id TA120-T246189. The results of anomaly detection indicated a high number of false positives, leading to a significant number of normal instances being classified as anomalies.

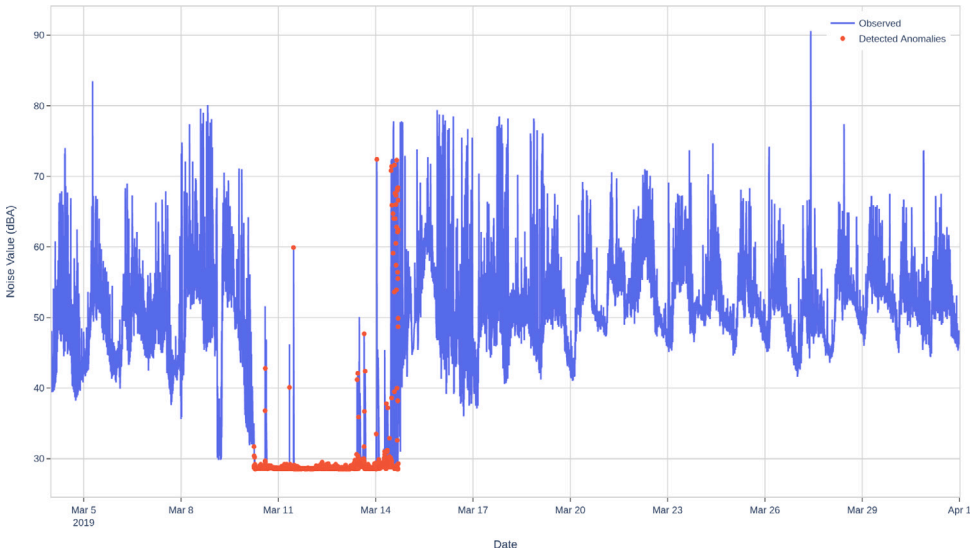


Fig. 10. Anomalies detected in the sensor with id TA120-T246189.

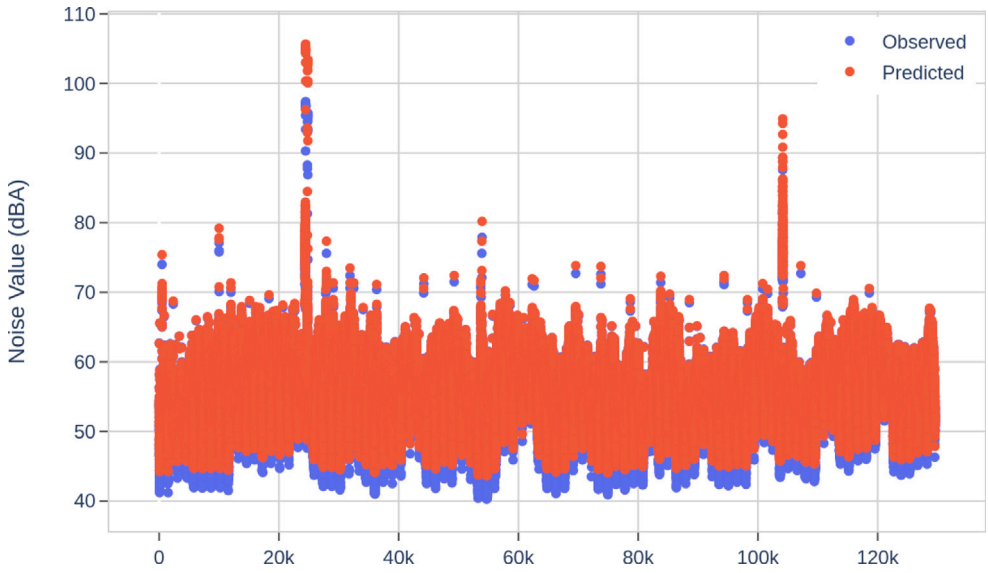


Fig. 11. Graph of values observed and predicted by LSTM for the sensor with id TA120-T246187.

5.3. Evaluation metrics

Model performance is evaluated by the overall accuracy metrics, namely precision, recall, F1-score and Kappa value (k) [60]. Figs. 11 and 12 represent the difference between the values observed and predicted by the two models described above in Table 2. In instances in time series where it deviates from the normal behaviour, the difference between observed and predicted values is higher compared to normal instances. In Fig. 12, where the anomaly is present, the difference between observed and predicted values is significantly higher than the rest, which shows that the model trained on normal data instances is not able to reconstruct the anomalous data points. The maximum reconstruction error from the respective LSTM models has been set as a threshold, and all values above the threshold are classified as anomalies.

A high precision value shows a lower number of false positives (FP), while a high recall value shows a lower number of false negatives (FN). Using precision only or recall only is not a true indication of the performance of the NN and produces a biased understanding of the model's accuracy. For this purpose, F1-scores are a good measure of the model's performance, and it is the harmonic mean between precision and recall values. Higher F1-score values indicate good performance. The equations below explain the calculation of the performance metrics.

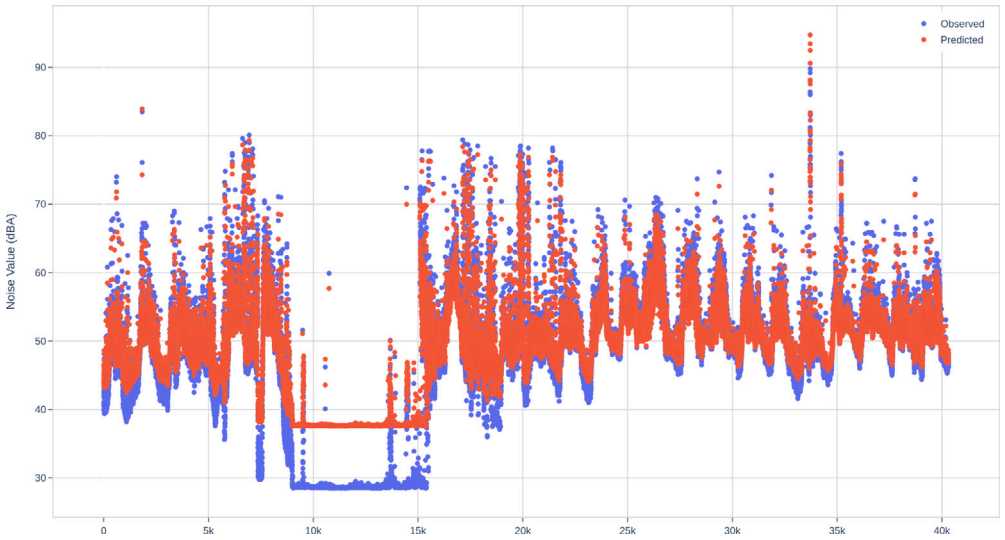


Fig. 12. Graph of values observed and predicted by LSTM for the sensor with id TA120-T246189.

Table 3
Evaluation metrics.

Performance metrics	Sensors	
	TA120-T246187	TA120-T246189
Precision	85.0	97.92
Recall	91.89	97.48
F1-Score	88.31	97.77
Accuracy	99.99	99.34
Kappa Score	0.88	0.97

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$F1\text{-}Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{9}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$Kappa(k) = 2 * \frac{(TP * TN) - (FP * FN)}{(TP + FP) * (FP + TN) + (TP + FN) * (FN + TN)} \tag{11}$$

where *TP, FP, TN* and *FN* refer to True Positive, False Positive, True Negative and False Negative, respectively. We are interested in the detection of anomalies and in this case, anomalies represent the positive class and normal data points are represented by the negative class. TP represents correctly classified anomalous data points, FP shows normal data incorrectly classified as anomalous data. TN shows normal samples correctly classified as normal and FN represents anomalous data incorrectly classified as normal. The evaluation metrics for model performance are given below in Table 3.

The confusion matrices in Figs. 13 and 14 show the performance of both models.

We also computed the Root Mean Square Error (RMSE) given by Eq. (12) between the predicted and observed values. RMSE values of the original TensorFlow model and the TensorFlow Lite model are given in Table 4.

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}} \tag{12}$$

where y_i and $\hat{y}_i$ refer to actual observations and predicted values respectively.

The difference between the value predicted by the TensorFlow original and TensorFlow lite model for the sensors with id TA120-T246187 and TA120-T246189 is 0.91% and 3.87% respectively. Fig. 15 shows a sample from the predicted values.

Fig. 13. Confusion matrix of the LSTM model for the sensor with id TA120-T246187.

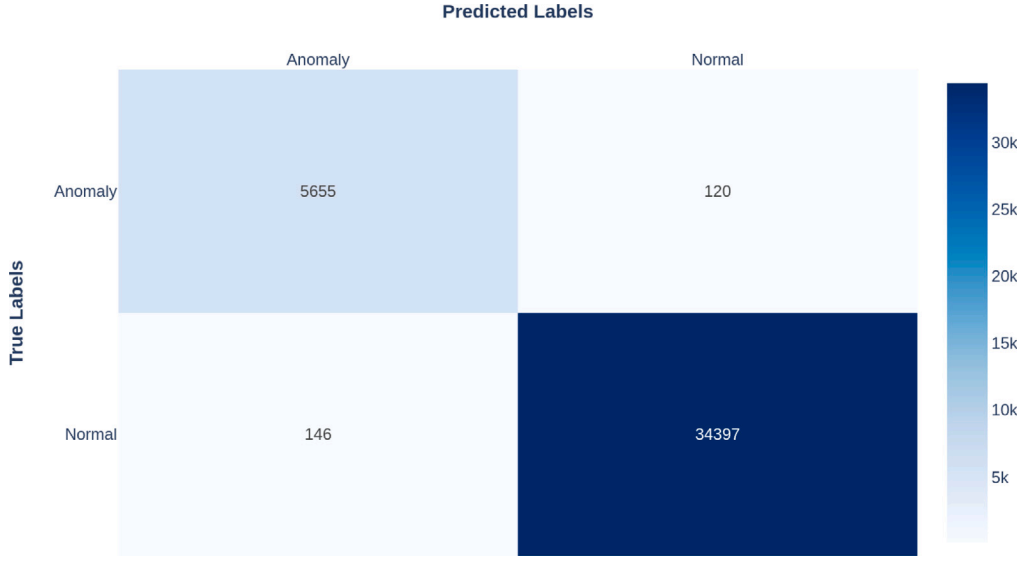


Fig. 14. Confusion matrix of the LSTM model for the sensor with id TA120-T246189.

Table 4
RMSE of TensorFlow original model and TensorFlow lite model.

	Root mean square error	
	TA120-T246187	TA120-T246189
TensorFlow original	1.31	3.18
Tensorflow Lite	1.36	3.21

5.4. Deployment on the microcontroller

The ESP32-S3-DevKitC-1-N32R8V MCU in Fig. 16 is the target device to deploy the quantised TensorFlow model. It has a flash memory of 32 MB and 8 MB PSRAM with Bluetooth and Wi-Fi connectivity support. ESP32 is one of the development boards supported by TensorFlow lite. For software development, the Epressif IoT Development Framework (ESP-IDF) is provided [61]. The TensorFlow lite model is translated into a C++ file and this file is included in the program to be deployed on the MCU.

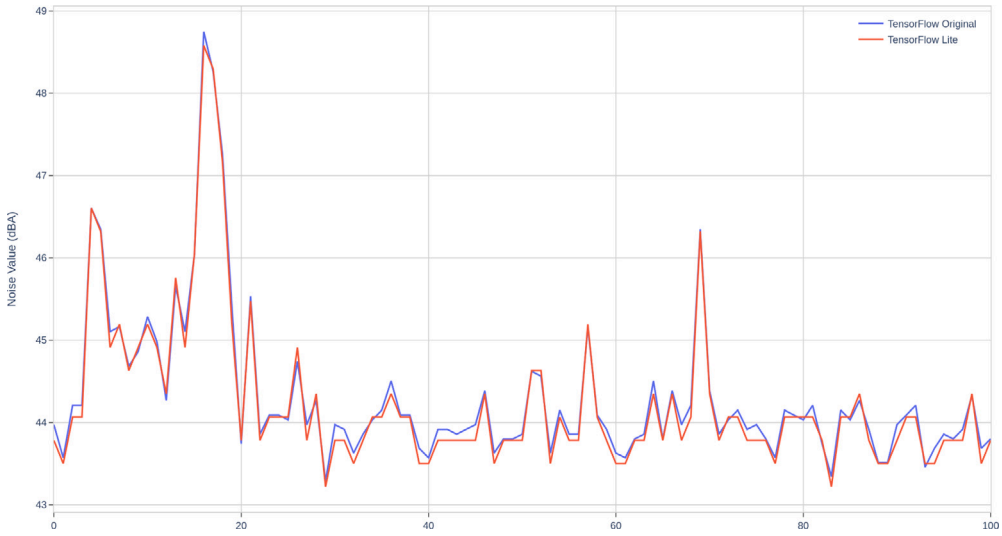


Fig. 15. Graph showing values predicted by the TensorFlow original and TensorFlow Lite model for the sensor with id TA120-T246189.

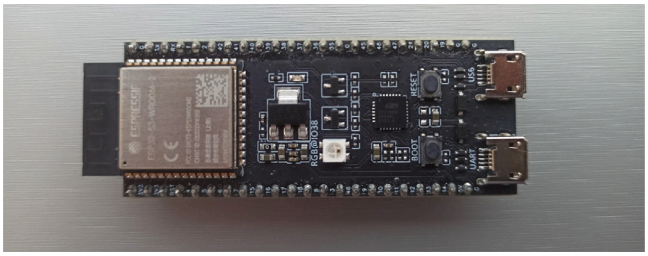


Fig. 16. ESP32-S3-DevKitC-1 MCU.

We provided the following metrics in terms of the performance of the TensorFlow Lite model on MCU: (a) Time, and (b) Memory Consumption.

The TensorFlow Lite model for the microcontroller is run for 300 inferences and repeated 30 times to obtain the power consumption and the average time taken for each inference by the program during the inference cycle. The results suggest that both the models on average take about 4 ms per inference and the power consumed during this time for the two models is 0.2693 W ±0.039 and 0.3268 W ±0.015. The program has a flash occupancy of 800.63 KB, while RAM occupancy is 102.62 KB. Heap memory usage for the sensor with id TA120-T246187 is 528 bytes. For the sensor with id TA120-T246189, the flash and RAM occupancy are 800.63 KB and 112.71 KB and it used 744 bytes of heap memory during program execution.

5.5. Limitations

The proposed study deals with time series data and since the model is once trained with historical data and deployed on MCU. It limits the ability of the anomaly detection system in case of concept drift. Therefore, it is also indicated in future work to make the model learning adaptive to new data streams to address this limitation.

6. Conclusions

This work explores the LSTM Autoencoder model for anomaly detection in an urban noise sensor network and deploys the autoencoder on a resource-constrained MCU. In the context of urban noise sensor data, anomaly detection can be a challenging problem to solve as it involves urban noise from various sources in the urban setting, it is very important to distinguish rare noise events from anomalies. Noise sensor data from two sensors are used to train the LSTM Autoencoder and then evaluated using a test data set. The accuracy achieved with the autoencoder is 99.99% and 99.34%. We also observed that for the sensor with id TA120-T246189, there are more false positives, which could be because the range of values in the anomaly samples is identical to the normal instances of data. In an anomaly detection use case where there is a huge imbalance between the two classes in the data, it is important not to base the findings of the study only on accuracy and other metrics such as precision, recall and F1-score should be taken into consideration. The deployment of the model on the MCU has a power consumption of 0.2693 W ±0.039 and

0.3268 W ±0.015, and the average inference time is 4 ms. For the sensor with id TA120-T246187, the RMSE difference between TensorFlow original and TensorFlow lite is 0.91%. RMSE for the sensor with id TA120-T246189 is 3.87%.

In this study, the model is trained locally, converted and deployed on the MCU. Currently, on-device training for MCUs is not supported by TensorFlow lite. In the future, we aim to explore on-device (MCU) model training so that the model is adaptive to the incoming stream of data. Another part of future work will be to explore the possibility of a trust management system between the different sensor devices in the network based on the reputation index of each device. This reputation index is to be based on the quality of data provided by devices in the network. We also aim to explore the application of other DL models such as CNN, ConvLSTM and Graph Neural Networks for anomaly detection and their deployment on resource constraint devices.

Data availability

The authors do not have permission to share data.

Acknowledgements

Grant PID2019-104065GA-I00 funded by MCIN/AEI/10.13039/501100011033 and, as appropriate, by MCIN/AEI/10.13039/501100011033 and by "ERDF, a way of making Europe", by the European Union. Sergio Trilles has been funded by the Juan de la Cierva - Incorporación postdoctoral programme of the Ministerio de Ciencia e Innovación - Spanish government (IJC2018- 035017-I) funded by MCIN/AEI/10.13039/501100011033 and by "ERDF, a way of making Europe", by the European Union.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] I. Statista, Internet of things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions), 2018, URL https://www.Statista.Com/Statistics/471264/Iot-Number-of-Connecteddevicesworldwide/ (Consulté 17/05/2020).
[2] C. Granell, A. Kamilaris, A. Kotsev, F.O. Ostermann, S. Trilles, Internet of things, in: Manual of Digital Earth, Springer, Singapore, 2020, pp. 387–423.
[3] S. Trilles, A. Calia, Ó. Belmonte, J. Torres-Sospedra, R. Montoliu, J. Huerta, Deployment of an open sensorized platform in a smart city context, Future Gener. Comput. Syst. 76 (2017) 221–233.
[4] M. Irfan, H. Jawad, B.B. Felix, S. Farooq Abbasi, A. Nawaz, S. Akbarzadeh, M. Awais, L. Chen, T. Westerlund, W. Chen, Non-wearable IoT-based smart ambient behavior observation system, IEEE Sens. J. 21 (18) (2021) 20857–20869, http://dx.doi.org/10.1109/JSEN.2021.3097392.
[5] A. Adeel, M. Gogate, S. Farooq, C. Ieracitano, K. Dashtipour, H. Larijani, A. Hussain, A survey on the role of wireless sensor networks and IoT in disaster management, Geol. Disaster Monit. Based Sens. Netw. (2019) 57–66.
[6] M. Caporuscio, F. Flammini, N. Khakpour, P. Singh, J. Thornadtsson, Smart-troubleshooting connected devices: Concept, challenges and opportunities, Future Gener. Comput. Syst. 111 (2020) 681–697.
[7] S. Trilles, Ò. Belmonte, S. Schade, J. Huerta, A domain-independent methodology to analyze IoT data streams in real-time. A proof of concept implementation for anomaly detection from environmental data, Int. J. Digit. Earth 10 (1) (2017) 103–120.
[8] S. Trilles, A. Luján, Ó. Belmonte, R. Montoliu, J. Torres-Sospedra, J. Huerta, SEnviro: A sensorized platform proposal using open hardware and open standards, Sensors 15 (3) (2015) 5555–5582.
[9] S. Trilles, A. González-Pérez, J. Huerta, A comprehensive IoT node proposal using open hardware. A smart farming use case to monitor vineyards, Electronics 7 (12) (2018) 419.
[10] X. Liu, X. Wei, L. Guo, Y. Liu, Q. Song, A. Jamalipour, Turning the signal interference into benefits: Towards indoor self-powered visible light communication for IoT devices in industrial radio-hostile environments, IEEE Access 7 (2019) 24978–24989.
[11] S. Trilles, J. Torres-Sospedra, Ó. Belmonte, F.J. Zarazaga-Soria, A. González-Pérez, J. Huerta, Development of an open sensorized platform in a smart agriculture context: A vineyard support system for monitoring mildew disease, Sustain. Comput.: Inform. Syst. 28 (2020) 100309.
[12] D. Li, L. Deng, W. Liu, Q. Su, Improving communication precision of IoT through behavior-based learning in smart city environment, Future Gener. Comput. Syst. 108 (2020) 512–520.
[13] N. Verma, S. Sangwan, S. Sangwan, D. Parsad, IoT security challenges and counters measures, Int. J. Recent Technol. Eng. (2019) 2277–3878.
[14] M. Shafiq, Z. Tian, A.K. Bashir, X. Du, M. Guizani, CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine-learning techniques, IEEE Internet Things J. 8 (5) (2020) 3242–3254.
[15] M. Wazid, A.K. Das, S. Shetty, J.J. Rodrigues, On the design of secure communication framework for blockchain-based internet of intelligent battlefield things environment, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2020, pp. 888–893.
[16] L. Zhou, H. Guo, Anomaly detection methods for iIoT networks, in: 2018 IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI, IEEE, 2018, pp. 214–219.
[17] M.A. Hayes, M.A. Capretz, Contextual anomaly detection in big sensor data, in: 2014 IEEE International Congress on Big Data, IEEE, 2014, pp. 64–71.
[18] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 1–58.
[19] I. Ullah, Q.H. Mahmoud, Design and development of a deep learning-based model for anomaly detection in IoT networks, IEEE Access 9 (2021) 103906–103926.
[20] D.-I. Curiac, C. Volosencu, D. Pescaru, L. Jurca, A. Doboli, Redundancy and its applications in wireless sensor networks: A survey, WSEAS Trans. Comput. 8 (4) (2009) 705–714.
[21] T. Stibor, J. Timmis, C. Eckert, A comparative study of real-valued negative selection to statistical anomaly detection techniques, in: International Conference on Artificial Immune Systems, Springer, 2005, pp. 262–275.
[22] K. Worden, Sensor validation and correction using auto-associative neural networks and principal component analysis, in: Proceedings of the IMAC-XXI, 2003, pp. 973–982.
[23] D. Sun, G. Chang, L. Sun, X. Wang, Surveying and analyzing security, privacy and trust issues in cloud computing environments, Procedia Eng. 15 (2011) 2852–2856.

[24] N. Elmrabit, F. Zhou, F. Li, H. Zhou, Evaluation of machine learning algorithms for anomaly detection, in: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, 2020, pp. 1–8.
[25] B.M. Wise, N.B. Gallagher, The process chemometrics approach to process monitoring and fault detection, J. Process Control 6 (6) (1996) 329–348.
[26] M. Ahmed, A.N. Mahmood, J. Hu, A survey of network anomaly detection techniques, J. Netw. Comput. Appl. 60 (2016) 19–31.
[27] S. Manimurugan, IoT-Fog-Cloud model for anomaly detection using improved Naive Bayes and principal component analysis, J. Ambient Intell. Humaniz. Comput. (2021) 1–10.
[28] M. Domb, E. Bonchek-Dokow, G. Leshem, Lightweight adaptive Random-Forest for IoT rule generation and execution, J. Inf. Secur. Appl. 34 (2017) 218–224.
[29] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, C.-T. Lin, Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment, IEEE Access 6 (2017) 1706–1717.
[30] W. Shi, S. Dustdar, The promise of edge computing, Computer 49 (5) (2016) 78–81.
[31] K. Bajaj, B. Sharma, R. Singh, Implementation analysis of IoT-based offloading frameworks on cloud/edge computing for sensor generated big data, Complex Intell. Syst. 8 (5) (2022) 3641–3658.
[32] B. Dong, Q. Shi, Y. Yang, F. Wen, Z. Zhang, C. Lee, Technology evolution from self-powered sensors to AIoT enabled smart homes, Nano Energy 79 (2021) 105414.
[33] F.E.F. Samann, S.R. Zeebaree, S. Askar, IoT provisioning QoS based on cloud and fog computing, J. Appl. Sci. Technol. Trends 2 (01) (2021) 29–40.
[34] A. Ometov, O.L. Molua, M. Komarov, J. Nurmi, A survey of security in cloud, edge, and fog computing, Sensors 22 (3) (2022) 927.
[35] P.P. Ray, A review on TinyML: State-of-the-art and prospects, J. King Saud Univ.-Comput. Inform. Sci. (2021).
[36] P. Bhatt, A. Morais, HADS: Hybrid anomaly detection system for IoT environments, in: 2018 International Conference on Internet of Things, Embedded Systems and Communications, IINTEC, IEEE, 2018, pp. 191–196.
[37] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[38] P. Ibarguengoytia, et al., Any time probabilistic sensor validation (Ph.D. thesis), University of Salford, UK, 1997.
[39] Y. LeCun, Y. Bengio, G. Hinton, et al., Deep learning, Nature 521 (7553) (2015) 436–444.
[40] H. Li, K. Ota, M. Dong, Learning IoT in edge: Deep learning for the Internet of Things with edge computing, IEEE Netw. 32 (1) (2018) 96–101.
[41] N. Rusk, Deep learning, Nature Methods 13 (1) (2016) 35.
[42] M.A. Al-Garadi, A. Mohamed, A.K. Al-Ali, X. Du, I. Ali, M. Guizani, A survey of machine and deep learning methods for internet of things (IoT) security, IEEE Commun. Surv. Tutor. 22 (3) (2020) 1646–1685.
[43] D. Zimmerer, F. Isensee, J. Petersen, S. Kohl, K. Maier-Hein, Unsupervised anomaly localization using variational auto-encoders, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2019, pp. 289–297.
[44] M.A. Alsoufi, S. Razak, M.M. Siraj, I. Nafea, F.A. Ghaleb, F. Saeed, M. Nasser, Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review, Appl. Sci. 11 (18) (2021) 8383.
[45] D. Nallaperuma, R. Nawaratne, T. Bandaragoda, A. Adikari, S. Nguyen, T. Kempitiya, D. De Silva, D. Alahakoon, D. Pothuhera, Online incremental machine learning platform for big data-driven smart traffic management, IEEE Trans. Intell. Transp. Syst. 20 (12) (2019) 4679–4690.
[46] K. JaeMyoung, Y.W. Cho, D.-H. Kim, Anomaly detection of environmental sensor data using recurrent neural network at the edge device, in: 2020 International Conference on Information and Communication Technology Convergence, ICTC, IEEE, 2020, pp. 1624–1628.
[47] C. Catalano, L. Paiano, F. Calabrese, M. Cataldo, L. Mancarella, F. Tommasi, Computers in Industry Anomaly detection in smart agriculture systems, Comput. Ind. 143 (June) (2022) 103750, http://dx.doi.org/10.1016/j.compind.2022.103750.
[48] Y. Majib, M. Barhamgi, B.M. Heravi, S. Kariyawasam, C. Perera, Detecting anomalies within smart buildings using do-it-yourself internet of things, J. Ambient Intell. Humaniz. Comput. (2022) 1–17.
[49] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, M. Boulic, LSTM-autoencoder-based anomaly detection for indoor air quality time-series data, IEEE Sens. J. 23 (4) (2023) 3787–3800, http://dx.doi.org/10.1109/JSEN.2022.3230361.
[50] S. Dauwe, D. Oldoni, B. De Baets, T. Van Renterghem, D. Botteldooren, B. Dhoedt, Multi-criteria anomaly detection in urban noise sensor networks, Environ. Sci.: Process. Impacts 16 (10) (2014) 2249–2258.
[51] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646, http://dx.doi.org/10.1109/JIOT.2016.2579198.
[52] B. Sudharsan, S. Salerno, D.-D. Nguyen, M. Yahya, A. Wahid, P. Yadav, J.G. Breslin, M.I. Ali, Tinyml benchmark: Executing fully connected neural networks on commodity microcontrollers, in: 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), IEEE, 2021, pp. 883–884.
[53] L. Dutta, S. Bharali, Internet of things TinyML meets IoT : A comprehensive survey, Internet of Things 16 (September) (2021) 100461, http://dx.doi.org/10.1016/j.iot.2021.100461.
[54] P.P. Ray, A review on TinyML : State-of-the-art and prospects, Journal of King Saud University-Computer and Information Sciences 34 (4) (2022) 1595–1623, http://dx.doi.org/10.1016/j.jksuci.2021.11.019.
[55] Y. Zhang, N. Suda, L. Lai, V. Chandra, Hello edge: Keyword spotting on microcontrollers, 2017, arXiv preprint arXiv:1711.07128.
[56] S. Bian, P. Lukowicz, Capacitive sensing based on-board hand gesture recognition with TinyML, in: UbiComp/ISWC 2021 - Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers, 2021, pp. 54–55, http://dx.doi.org/10.1145/3460418.3479287.
[57] M. Lord, A. Kaplan, Mechanical anomaly detection on an embedded microcontroller, in: 2021 International Conference on Computational Science and Computational Intelligence, CSCI, IEEE, 2021, pp. 562–568.
[58] mySMARTLife, Noise sensor data from Helsinki - Helsinki Region Infoshare hri.fi, 2023, https://hri.fi/data/en_GB/dataset/iot-meludataa-helsingista. The maintainer of the dataset is Forum Virium Helsinki Oy. The dataset has been downloaded from Helsinki Region Infoshare service on 15.10.2022 under the license Creative Commons Attribution 4.0. [Last accessed 18-Apr-2023].
[59] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[60] D. Chicco, M.J. Warrens, G. Jurman, The Matthews Correlation Coefficient (MCC) is more informative than Cohen's Kappa and Brier score in binary classification assessment, IEEE Access 9 (2021) 78368–78381, http://dx.doi.org/10.1109/ACCESS.2021.3084050.
[61] ESP-IDF Programming Guide - ESP32-S3, 2023, https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/. (Last accessed 25-Apr-2023).