Streaming Data Processing. Use-Case: Anomaly Detection in Sensor Data

1 Purpose

The purpose of this laboratory work is to show a real use-case of streaming data processing, namely detecting anomalies in sensor data. The main focus is on designing a module which detects sharp changes in the data generated by a temperature sensor. To this aim, AXI4-Stream modules will be used to design a circuit capable of identifying unexpected variations based on Two-Sided CUSUM Algorithm.

2 Detecting Anomalies in Sensor Data

In recent years, the emergence and rapid development of Internet of Things (IoT) systems has revolutionized not only the computational needs of a sensor network, but also the importance of the quality of data collected by sensors in such a network. Sensor-based systems now have an increasing range of applicability, such as environmental monitoring, smart cities, factory automation, smart buildings and autonomous vehicles. Current monitoring applications consist of various types of heterogeneous sensors that form systems where neighboring devices can communicate with each other and transmit data to a Cloud environment for further analysis [1].

However, most applications using sensor data assume that they are permanently in the correct operating mode, which is difficult to guarantee, as most sensors are often prone to failure. In addition, many use cases of sensor-based applications involve their exposure to adverse environmental conditions. Therefore, the high probability of measuring errors or corrupted measurements during the transmission process makes data quality really difficult to ensure [2]. Moreover, sensor systems are strongly influenced by anomalies that occur for various reasons, such as the failure of a network node, read or conversion errors, unusual events or malicious attacks.

2.1 Types of Anomalies in Sensor Data

Several types of anomalies can be encountered in sensor data, taking into consideration the real context in which abnormal values can appear and the causing factors. The major types of anomalies in sensor data are briefly described in what follows.

2.1.1 Spike Anomalies

Spike anomalies are strong variations, representing local or even global extrema, deviating significantly from most measurements collected by the same sensor, as shown in Figure 1. They can often indicate a malicious attack threat, especially if such variations are recorded on a regular basis, but they can also be the results of a malfunction of the sensor or an improper power supply.

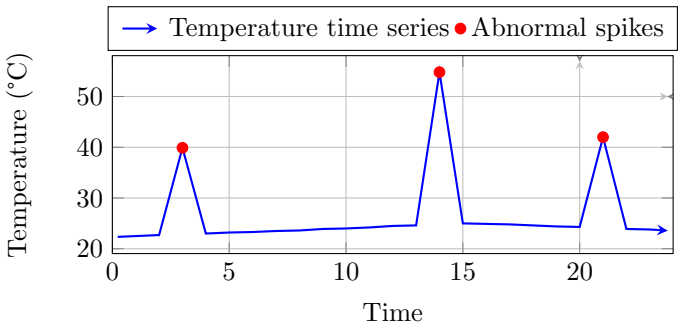

Figure 1: Spike anomalies in temperature time series

2.1.2 Constant Anomalies

Constant anomalies are identified when a sensor reports the same value for the measured phenomenon, regardless of its actual variation, as depicted in Figure 2. They are mainly caused by the failure or improper use of the sensor. The detection of anomalies of this type is strongly dependent on the measuring accuracy. A low measuring accuracy of the sensor makes this type of anomalies difficult to be detected as the changes in the value of the readings are visible after a longer period of time.

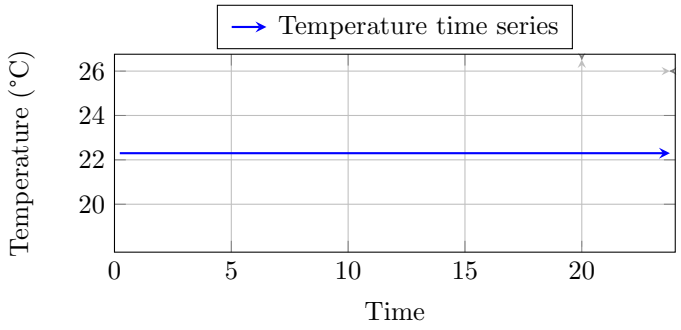

Figure 2: Constant anomalies in temperature time series

2.1.3 Noise Anomalies

Noise anomalies are a subset of observations that show a greater variance than the rest of the measurements. These anomalies are often caused by improper functioning or unstable connection of the sensor within the network. They are characterized by successive variations that are significantly different from the average values, although their final average can approach the normal average value, as shown in Figure 3.

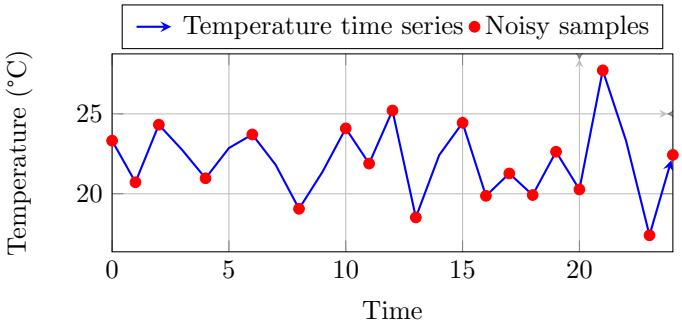

Figure 3: Noisy temperature time series

2.1.4 Drift Anomalies

As Figure 4 depicts, drift anomalies are measurements that show an offset from the true value of the monitored phenomenon at a given time. These are often caused by the lack of proper calibration of the measuring devices. These anomalies are characterized by all measured values deviating by a constant amount from the actual values.

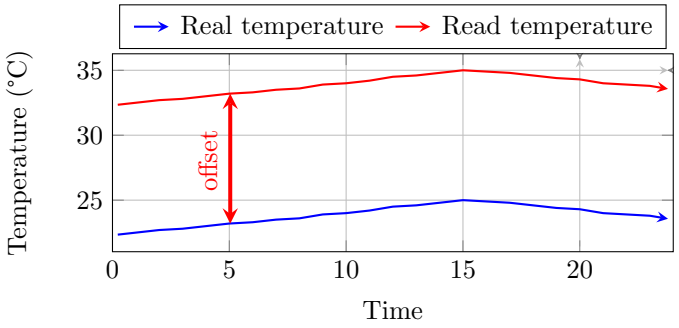

Figure 4: Drift anomaly in temperature time series

2.1.5 Gradual and Continuous Variations

This type of anomaly is specific to real situations where the processes being measured are undergoing gradual but continuous variation over time, indicating unusual real-life scenarios, such as a fire detected by an environmental temperature monitoring system. Since the difference between consecutive measurements is not significant, anomalies specific to these situations are difficult to identify, requiring a detailed analysis of the states of the system at moments which define the normal state or behavior of the monitored environment or phenomenon. An example of such anomaly is shown in Figure 5.

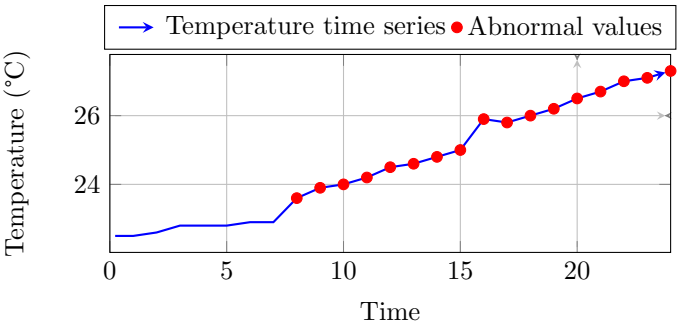

Figure 5: Continuous increase anomaly in temperature time series

2.2 Anomaly Detection Algorithms

Considering a set of N sensors placed in the same area and monitoring the same process or phenomenon, let $\left\{x_t^{(i)}\right\}_{0\leq t\leq n}$ be the set of the n most recent readings recorded by the i^{th} sensor. When labeling a new measurement $x_{n+1}^{(i)}$ recorded by the same sensor i, there are two possible approaches:

- exploiting the correlations between the newest measurement $x_{n+1}^{(i)}$ and all the readings $\left\{x_t^{(i)}\right\}_{0\leq t\leq n}$ previously reported by the same sensor (**temporal correlations**)
- exploiting the correlations between the newest reading $x_{n+1}^{(i)}$ reported by the sensor i and the most recent measurements $\left\{x_{n+1}^{(j)}\right\}_{j\neq i}$ recorded by all the other sensors (**spatial correlations**)

Therefore, two classes of anomaly detection algorithms can be identified:

1. **Temporal Correlations-based Algorithms**: use the measured values from the same sensor as a reference in the detection process to determine whether a current value or set of values aligns with the historical model of the data collected by that particular sensor. These algorithms are mostly used to detect spike anomalies and continuous increases or decreases in the evolution of the monitored process.

2. **Spatial Correlations-based Algorithms**: use the values measured by other sensors placed in the proximity of the considered sensor when determining if an observation reported by the sensor is anomalous or not. This type of algorithms are usually effective in the case of constant, drift and noise anomalies.

3 Two-Sided CUSUM Anomaly Detection Algorithm

This section introduces a temporal, cumulative sums-based anomaly detection algorithm which is a slightly modified version of the *Two-Sided Cumulative Sums* algorithm, described in [3], such to allow considering the differences between consecutive values as reference elements in the detection process. The method identifies anomalies based on the positive and negative differences between the values measured by the same sensor at consecutive points in time. Thus, it proves to be effective for detecting both significant changes such as spikes and continuous and gradual unusual variations in the evolution of the time series.

3.1 Algorithm Description

Supposing that $\{x_n\}_{n\geq 0}$ is a finite sub-sequence of a data stream generated by the same source (sensor), for each x_t, $t \in \{1, 2, ..., n-1\}$, the algorithm computes the difference $S_t = x_t - x_{t-1}$, and then the cumulative sums of positive and negative changes, $g_t^+ = \max(g_{t-1}^+ + S_t - drift, 0)$ and $g_t^- = \max(g_{t-1}^- - S_t - drift, 0)$. If any sum is larger than a given *threshold*, the current value is labeled as an anomaly. To reduce the number of false positive samples, the $drift$ is subtracted from both g_t^+ and g_t^-.

The algorithm is listed below:

procedure CUSUM-ANOMALY-DETECTION($x, threshold, drift$)
    $S_0 \leftarrow 0$
    $g_0^+ \leftarrow 0$
    $g_0^- \leftarrow 0$
    $t_{abnormal} \leftarrow \emptyset$
    for $t = 1$ to $x.length$ **do**
        $S_t \leftarrow x_t - x_{t-1}$
        $g_t^+ \leftarrow \max(g_{t-1}^+ + S_t - drift, 0)$
        $g_t^- \leftarrow \max(g_{t-1}^- - S_t - drift, 0)$
        if $g_t^+ > threshold$ **or** $g_t^- > threshold$ **then**
            $t_{abnormal} \leftarrow t_{abnormal} \cup \{t\}$
            $g_t^+ \leftarrow 0$
            $g_t^- \leftarrow 0$
        end if
    end for
    return $t_{abnormal}$
end procedure

The values of $drift$ and *threshold* are determined depending on the use case, the monitored phenomenon and rate at which the data is sampled. The *threshold* is often chosen as the maximum accepted value of the difference between two or more consecutive readings, according to the time interval between two readings, while the $drift$ parameter is the expected average difference between two consecutive values measured by the same sensor, also taking the sampling interval into account.

3.2 Hardware Design

To design a hardware anomaly detector based on the *Two-Sided CUSUM* algorithm capable of operating on streaming data, AXI4-Stream modules can be used. A block design is depicted in Figure 6.

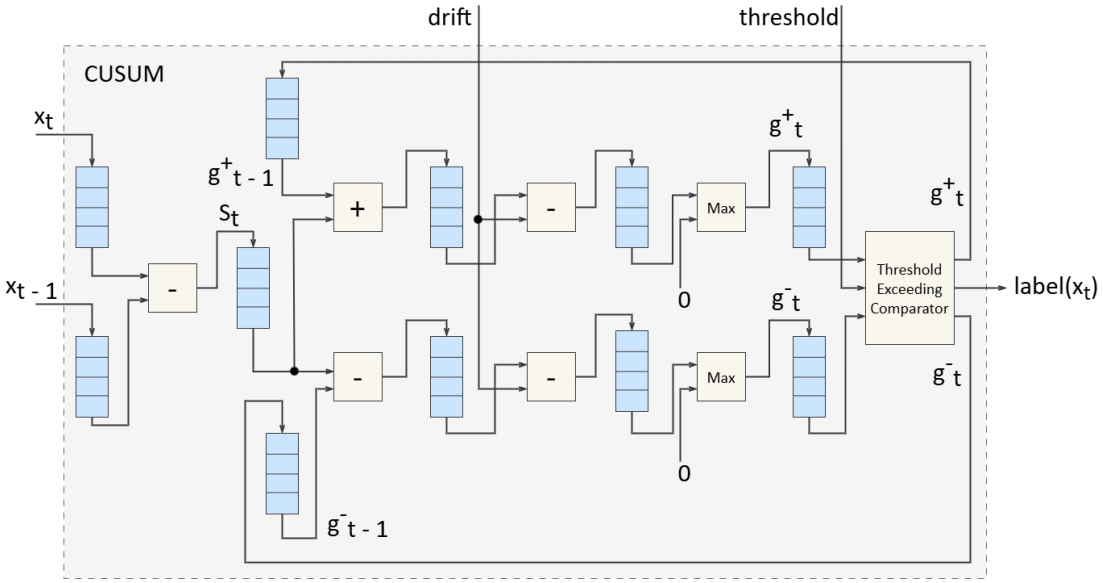

Figure 6: Block design of the CUSUM-based anomaly detector using AXI4-Stream components

The detector receives the most recent measurement (i.e. x_t) and the previously recorded measurement (i.e. x_{t-1}) belonging to the same data stream (recorded by the same sensor) as inputs and outputs the label of the most recent reading. The label can be 0, meaning that x_t is a normal value or 1, reporting an abnormal or unexpected value based on the cumulative sums of the differences between consecutive readings.

The cumulative sums g_t^+ and g_t^- are computed in parallel using adders, subtractors and comparators. Their values are compared with the threshold value using another comparator which also sets their values to 0 each time one of them exceeds the threshold. Thus, the operation performed by the threshold exceeding detector when receiving a valid input triplet $(g_t^+, g_t^-, threshold)$ is:

procedure COMPARE($g_t^+, g_t^-, threshold$)
  if $g_t^+ > threshold$ **or** $g_t^- > threshold$ **then**
    $label \leftarrow True$
    $g_t^+ \leftarrow 0$
    $g_t^- \leftarrow 0$
  else
    $label \leftarrow False$
  end if
  return $(g_t^+, g_t^-, label)$
end procedure

Intermediate results and computations can be stored in AXI4-Stream Data FIFO buffers or AXI4-Stream register slices in order to ease synchronization between component processing frequencies and to achieve better performance.

As the cumulative sums g_t^+ and g_t^- are computed in parallel using the difference between the two most recent measurements $S_t = x_t - x_{t-1}$, the value of S_t has to be broadcast on the two parallel data paths computing the two cumulative sums. Thus, a broadcaster module has to be used in order to ensure that S_t is correctly passed to both the adder which computes the sum $g_{t-1}^+ + S_t$ and the subtractor responsible of computing the difference $g_{t-1}^- - S_t$, as shown in Figure 7. The broadcaster module guarantees that the value is consumed by both modules needing it. Such a broadcaster module is available in the Vivado IP Catalog: *Vivado Repository* → *AXI Infrastructure* → *AXI4-Stream Broadcaster*, allowing multiple output re-mapping options.

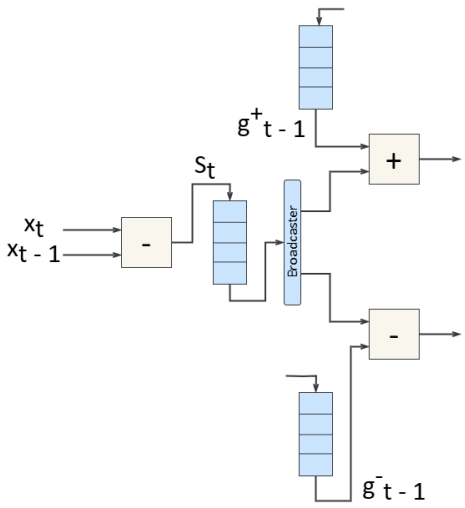

Figure 7: Broadcasting the S_t signal on the two parralel data paths which compute the cumulative sums

4 Exercises

1. `04-12-22_temperature_measurements.csv` **file contains 6 comma-separated temperature time series recorded by 6 temperature sensors placed in the same room. Pre-process the data in** `04-12-22_temperature_measurements.csv` **to get only integer values (i. e. multiply each sample by 100 and round it to the closest integer). The structure of the file is given below:**

```
Timestamp, DS18B20, DHT11, LM35DZ, BMP180, Thermistor, DHT22
12/4/2022 13:00, 23.62, 23, 27.34, 22.19, 23.73, 22.6
12/4/2022 13:00, 23.69, 23, 21, 22.2, 23.83, 22.6
12/4/2022 13:00, 23.75, 23, 46.88, 22.22, 23.93, 22.6
        ...
```

2. **Implement a software prototype of the CUSUM algorithm using a programming language of your own choice. Test the implementation using each temperature time series (column) in the data pre-processed at the previous exercise using a** `threshold` **of** 200 (corresponding to a maximum allowed difference of 2^oC between measurements: $2^oC \cdot 100$) **and a** `drift` **of** 50 (corresponding to an average expected difference between two consecutive values of 0.5^oC: $0.5^oC \cdot 100$). **Plot a line graph for each column, pointing the anomalies out as shown in Figures 1, 3 and 5. Identify types of anomalies in each plot.**

   Note. For convenience you may use `pandas.DataFrame` to store the data and individually process each column and `matplotlib` Python library for plotting the graphs. However, there are no constraints on the programming language to be used.

3. **Using a programming language of your choice, write a program to convert the pre-processed (integer) values in** `04-12-22_temperature_measurements.csv` **to binary. Create 6 individual files, each containing the binary representation of a column in the pre-processed** `.csv` **file.**

4. **Implement each component of the CUSUM hardware design shown in Figure 6 as an AXI4-Stream compliant module. Create a testbench for each component and simulate its behavior. You can use IP Cores for the FIFOs, register slices and broadcasters.**

5. **Implement the CUSUM top level module as depicted in Figure 6. Create a testbench and test the implemented design using each of the 6 stimuli files created at exercise 3. For each stimuli file, save the outputs (labels) in an individual `.csv` file. The structure of an output file should be the following:**

```
index, label
0, 0
1, 0
2, 1
3, 0
4, 1
...
```

Note. In the sample output file listed above, the temperature readings placed in the stimuli file on lines 0, 1 and 3 are labeled as normal values, while values placed on lines 2 and 4 are labeled as anomalies.

6. **Validate the hardware implementation using the implemented software prototype. For each temperature time series (each column in the pre-processed inputs file), compare the results of the hardware implementation with the results obtained using the software implementation of the algorithm. Make sure that they provide the same results on each input time series.**

References

[1] L. Erhan, M. Ndubuaku, M. Di Mauro, W. Song, M. Chen, G. Fortino, O. Bagdasar, and A. Liotta, "Smart anomaly detection in sensor systems: A multi-perspective review," *Information Fusion*, vol. 67, pp. 64–79, 2021.

[2] F. Giannoni, M. Mancini, and F. Marinelli, "Anomaly detection models for iot time series data," *arXiv preprint arXiv:1812.00890*, 2018.

[3] M. Basseville and I. Nikiforov, *Detection of Abrupt Change - Theory and Application*, Prentice Hall, Inc., 1993. [Online]. Available: https://hal.science/hal-00008518.