Automated Behavior Labeling for IIoT Data

Erik Johannes Husom, Arda Goknil, Simeon Tverdal, Sagar Sen, Phu Nguyen
SINTEF
Oslo, Norway
firstname.lastname@sintef.no

ABSTRACT

We present an automated data analysis tool for IIoT applications that discovers process behavior patterns in sensor data. It takes time-varying sensor data from reference production cycles and performs clustering on summary statistic feature vectors derived from raw sensor data over configurable window sizes. It automatically labels the raw sensor data based on distinct behavior modes represented by the clusters. The tool wraps, as a web service deployed in a Docker container, the AI model represented by clusters/behavior modes discovered in the reference sensor data. We have successfully evaluated the tool over four industrial datasets. Demo video: https://www.youtube.com/watch?v=MhSnwPDnAh0.

KEYWORDS

Machine Learning, Industrial Internet of Things

ACM Reference Format:
Erik Johannes Husom, Arda Goknil, Simeon Tverdal, Sagar Sen, Phu Nguyen. 2023. Automated Behavior Labeling for IIoT Data. In *Proceedings of IoT Conference (IoT'23).* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/xx.xxx/xxx_x

1 INTRODUCTION

The Industrial Internet of Things (IIoT) facilitates the integration of sensors, edge computing, cloud computing devices, and industrial machinery in production networks, providing real-time access to diverse and dynamic sensor data sources. These data sources can be utilized in machine learning (ML) applications, including predictive maintenance, remote quality monitoring, and energy optimization for continuous and repetitive manufacturing processes. ML applications in IIoT are trained with sensor data to enable real-time decision-making and post-mortem analysis of product defects and production failures [5, 7, 12, 13, 15]. They are sensitive to transitions in process behavior, including normal operation, process shifts, and drifts, which are time-ordered trends deviating from the intended target value of measured process parameters.

Process shifts in manufacturing lines can be attributed to the initial manual setup tasks (e.g., sensor calibration) done when producing new lots of goods/parts. Process drifts refer to gradual shifts occurring in one direction over time. Sensor faults, tool wear, and workpiece surface quality contribute to process drifts. Detecting process drifts in the presence of high-volume and high-velocity multivariate sensor data poses a significant challenge due to potential obscuration by other patterns. Thus, sensor data validation and process pattern identification are the keys to revealing process shifts and drifts.

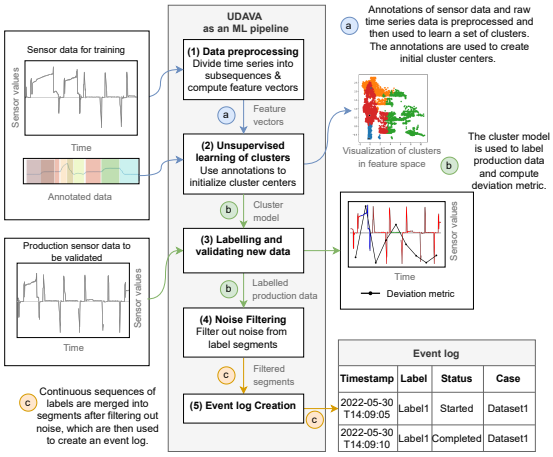

Figure 1: UDAVA Tool Overview.

This paper introduces UDAVA (Unsupervised machine learning pipeline for sensor DAta VAlidation), an automatic process behavior pattern discovery tool designed for IIoT applications. It leverages a reference production cycle to validate subsequent cycles by detecting recurring patterns. Through dimensionality reduction and clustering techniques, UDAVA efficiently analyzes dense time-series data. The resulting cluster model is encapsulated in a docker container, enabling batch data verification, identification of behavior patterns, and quantification of deviations from the reference. It supports easy model updates and offers a semi-supervised mode through manually annotating reference time series data.

2 RELATED WORK

Traditional time series clustering techniques rely on raw data similarity and use distance metrics to divide time series into subsequences [1]. However, Euclidean distance [4] has limitations, such as fixed time series sizes and sensitivity to noise and distortion. Dynamic time warping (DTW) [10] improves upon these limitations but struggles with dissimilar motifs across sensors. UDAVA clusters reduced dimensionality feature vectors, allowing the discovery of multiple similar patterns with diverse motifs (without repetition).

While deep learning techniques such as LSTM, CNN, and autoencoders have shown effectiveness in detecting abnormal behavior [9], their interpretability is limited and they rely heavily on large, high-quality training datasets. These methods primarily focus on residual error-based detection and lack comprehensive support for general process behavior detection in IIoT. Unsupervised outlier

IoT'23, November 7 –10, 2023, Nagoya, Japan

ACM ISBN 978-1-4503-9517-5/23/03…$15.00
https://doi.org/xx.xxx/xxx_x

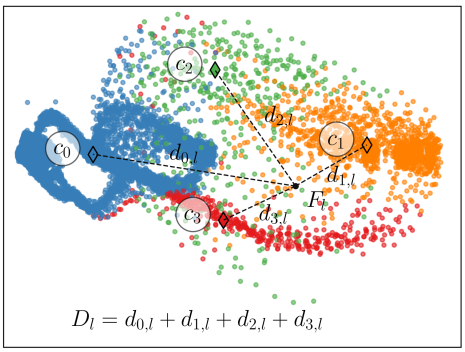

Figure 2: Example clusters in 2D feature space detected by UDAVA. Each colored dot represents an observation (feature vector) of the data set; the four colors correspond to four clusters in the model. Diamond-shaped markers are the cluster centers (c_i) defined by the mean of the features of all feature vectors in each cluster. The annotations illustrate the calculation of the deviation metric D_l for a feature vector F_l (l is the feature vector's index in the data set) shown as a black dot, where $d_{i,l}$ is the Euclidean distance of F_l to c_i.

detection methods [2, 8, 11] are classification-like approaches used for abnormal behavior detection, but they only consider binary classifications (inliers and outliers) and do not address process behavior. In contrast, UDAVA uncovers behavior patterns in IIoT data, enabling the detection of abnormal behavior (anomalies).

3 TOOL OVERVIEW

UDAVA is the tool supporting our approach recently described in our research paper [6]. It is designed and implemented as an ML pipeline. Figure 1 presents the tool overview.

3.1 Data Preprocessing (Step 1)

UDAVA preprocesses reference and production time series (training data and data to be validated). Feature vectors derived from production data are utilized in Step 3. Training data are obtained from optimal production cycles and are the baseline for comparison with data from subsequent production cycles. Raw time series contains temporal relationships between consecutive data points, making clustering computationally intensive, especially with high volumes of data. To address this, UDAVA extracts features from data subsequences, eliminating the temporal dimension. For more granularity, the sliding window technique [3] is applied, albeit at the cost of increased computational time. Feature extraction is performed on each subsequence, generating a corresponding feature vector.

3.2 Unsupervised Learning of Clusters (Step 2)

UDAVA employs an automated clustering algorithm to assign each feature vector to clusters. This process generates a cluster model, represented by cluster centers, which define distinct process behavior patterns for reference data (Figure 2). The time series is divided into subsequences of a window size determined by the sampling frequency. Feature vectors are computed from each subsequence, comprising statistical measures (e.g., mean, median, standard deviation, variance, and frequency) and reducing data dimensionality.

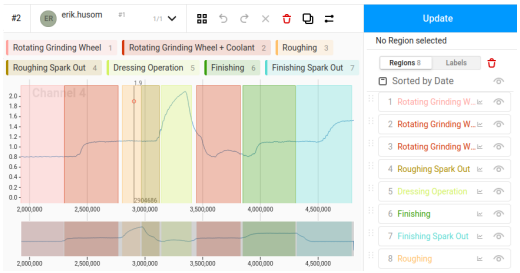

Figure 3: Example manual labels created in Label Studio. The blue line graph shows grinding power data from a roughing process performed by a grinding machine. The x-axis represents time steps; the y-axis indicates grinding power in watts. The text labels above the graph represent seven subprocesses in the production cycle. The colored boxes overlaying the line graph are the labels. They are referred to as "regions"; each region color corresponds to a label.

The clustering algorithm operates on these feature vectors (the cluster centers exist in the feature space rather than the time domain). UDAVA supports several clustering algorithms in Scikit-learn.

UDAVA offers a semi-supervised mode that allows users to provide manual labels by annotating a small set of reference time series data. The annotation involves selecting periods/regions of the time series and assigning them specific subprocess labels, such as *Grinding*, *Roughing*, *Finishing* in Figure 3. Label Studio (https://labelstud.io/) is used to create these labels exported as JSON files for UDAVA (see Figure 1).

When manual labels are provided, the tool derives feature vectors from the annotated data and computes cluster centers for each label. These cluster centers serve as the initial centers for the clustering algorithm. In the absence of annotated data, the algorithm initializes cluster centers randomly. Users can configure UDAVA to keep the initial cluster centers from the annotated data, skipping the clustering process (unless additional cluster centers unrelated to annotated labels need to be identified). Otherwise, the initial cluster centers are adjusted by running the clustering algorithm.

3.3 Labeling and Validating New Data (Step 3)

UDAVA labels and validates new production data using the cluster model. Production data is preprocessed like reference data. Cluster labels are assigned to its feature vectors (see Figure 4) and plotted onto the sensor data, which changes color according to the assigned cluster label. Finally, the feature vectors' labels are assigned to their corresponding subsequences and plotted back into temporal space.

UDAVA calculates a *deviation metric* to measure the dissimilarity between a feature vector (subsequence) and the cluster centers. This metric provides an approximate assessment of the conformity of production data to the expected behavior. It is computed for each feature vector in the production time series data. The reference data, collected during (near-)optimal production cycles, serves as the baseline for comparing against the production data. The deviation metric determines the proximity of a new observation (feature vector) to the existing cluster centers. By comparing the deviation metric values within the production data set or against the reference data set, we can identify spikes or higher averages as indications of deviations or significant changes in the production data.

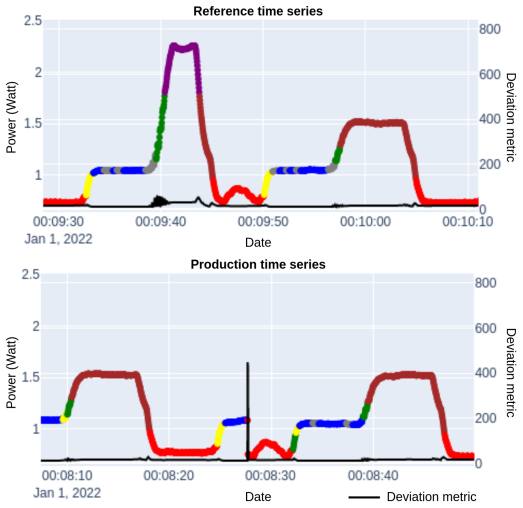

Figure 4: Raw data from an example data set (multicolored line) with the deviation metric D (black line), for the reference (top) and production (bottom) data. The subsequences are colored based on their cluster.

3.4 Noise Filtering (Step 4)

UDAVA post-processes the cluster model output to extract additional data insights. It identifies continuous sequences of the same label to identify meaningful data segments. Segments shorter than a user-defined minimum duration threshold are considered noise and can be disregarded. In such cases, the short segments are merged with the neighboring segments closest in the future space, effectively "swallowing" the noise and ensuring continuity in the data.

3.5 Event Log Creation (Step 5)

UDAVA generates an event log recording the start and end of each segment in sequential order, along with their timestamps and labels. The event log supports visualizing, inspecting, and automatically validating production cycles, ensuring they occur as expected. Each log entry corresponds to a labeled occurrence in the cluster model, enabling verification of the expected duration and order of subprocesses (e.g., *Rotating*, *Roughing*) within the production process.

4 IMPLEMENTATION & AVAILABILITY

UDAVA is deployed as a Docker container, encompassing the training and inference pipelines in Figure 5. The ML pipeline retrieves the required assets from a Docker volume mounted in the host file system and saves the generated models in the cache. The inference pipeline can be accessed via a browser-based GUI or programmatically through the REST API. The REST API of UDAVA, built on Flask, allows seamless integration with other applications and provides a user-friendly interface. It leverages Flask-RESTful, i.e., an extension simplifying endpoint creation and management. It is implemented as a single Python script (api.py), encompassing the necessary code for API setup, route definition, and request handling. By importing various Python modules for data processing, ML, and visualization, the API offers a comprehensive range of features to cater to the needs of end users. UDAVA utilizes several Python libraries (NumPy, Pandas, Plotly, and Scikit-learn). NumPy and Pandas are used to

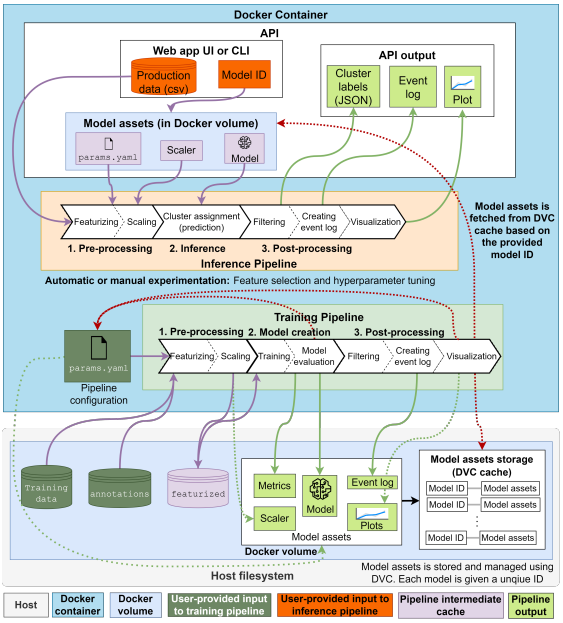

Figure 5: Deployment of UDAVA as a Docker Container.

handle and manipulate large datasets. Plotly enables interactive data visualization. Scikit-learn is used for implementing clustering algorithms and various data analysis techniques. The API makes use of the yaml library for configuration file operations and the UUID library for generating unique model identifiers.

The API provides multiple endpoints. The "create" model endpoint allows users to create a new model by specifying parameters, training data, and optional annotations. It supports GET and POST requests, where GET retrieves the list of existing models, and POST initiates the model creation process. The "infer GUI" enables users to perform dataset inference using a GUI, while the "infer" serves REST API clients for programmatic inference. Both endpoints support GET and POST requests, with GET returning a status code and POST initiating the inference process. UDAVA incorporates file-based storage and DVC to manage data and model assets. Dedicated directories in the application store the raw data, models, and metadata, facilitating convenient access and retrieval. DVC tracks asset changes, allowing users to maintain a version-controlled history of their work. Thus, the API efficiently handles the model and data storage, retrieval, and versioning. DVC ensures a separation between the API's core functionality and the storage mechanism.

The UDAVA REST API is containerized using Docker to simplify deployment and ensure a consistent runtime environment. Docker packages the API and its dependencies into a portable and self-contained container, enabling deployment on diverse platforms. Persistent storage of assets and data files is ensured using Docker volumes during container execution. Thus, data is stored on the host machine, providing accessibility and modification even after the container is closed. Docker facilitates efficient and consistent API deployment across various environments, ensuring a seamless user experience. Additional details about UDAVA, including executable files and a screencast covering motivations, are available at:

https://sintef-9012.github.io/Udava/

5 EVALUATION

This section presents key findings from the evaluation conducted to address the following research questions.

- ***RQ1.*** *To what extent can UDAVA discover behavior patterns in sensor data for a reference production cycle?*
- ***RQ2.*** *Does comparing process behavior patterns reveal process shifts and drifts in subsequent production cycles?*
- ***RQ3.*** *To what extent can UDAVA detect anomalies in IIoT data?*
- ***RQ4.*** *How can UDAVA be deployed, tested and maintained in industrial production environments?*

We assessed UDAVA using four datasets from (a) aerospace industry turbine disc broaching, (b) automotive industry cylinder head milling for car engines, (c) aluminum workpiece processing in an operation sequence for aluminum parts, and (d) grinding power measurements in metal bearing manufacturing.

To address *RQ1*, we compared UDAVA's event log with the expected outcome by defining the anticipated order and duration range for each label. We tallied the occurrences of events followed by the expected label and verified if each event fell within the expected duration range for that label. *Hits* were recorded for correct appearances and durations, while *misses* were tallied for inconsistencies. To quantify the degree of alignment between UDAVA's output and the expected outcome, we computed an event log score (S_e) using the formula (number of hits / (number of hits + number of misses)), providing a scale from 0 to 1. Table 1 displays UDAVA's event log scores (S_e) for various window sizes (w) in the grinding dataset. These scores enable fine-tuning of UDAVA's hyperparameters to enhance its capability in detecting behavior patterns. UDAVA achieves an S_e of 0.82 with a window size of 20.

Table 1: S_e for grinding dataset.

w	S_e
5	0.58
10	0.75
15	0.76
20	0.82
25	0.77
30	0.58

To answer *RQ2*, we compared behavior patterns in the reference and production data in the four datasets. The observed drifts were minimal as the production cycles showed limited deviations. Nonetheless, the deviation metric gave valuable data validation across multiple cycles and facilitated root cause analysis by comparing with other parameters such as tool wear.

To respond to *RQ3*, we utilized UDAVA to detect anomalies in the aluminum processing dataset. This dataset was chosen due to its explicit labeling of anomalies for evaluation. UDAVA was configured with a window size of 1100 and an overlap of 330, employing DBScan with parameters $eps = 0.53$, $min\_samples = 6$, and $metric = Euclidean$. It achieved a recall of 0.76, a precision of 0.72, and an F1-score of 0.74. The test dataset contains anomalous data with higher amplitudes compared to the normal data in the training dataset, facilitating the differentiation between the two types of behavior. However, certain portions of both the anomalous and normal data exhibit similar statistical properties, e.g., mean and value range, which might explain the relatively modest performance.

To address *RQ4*, we analyzed UDAVA's deployment in industrial environments. UDAVA utilizes DVC for versioning reference data, ensuring that changes trigger data preprocessing, unsupervised learning, and the creation of a new web service model. Data versioning and dependency management challenges are crucial in designing UDAVA's architecture. The deployment infrastructure poses scientific challenges in designing architectures [14] along the *edge-cloud continuum* while addressing data privacy and security concerns.

6 CONCLUSION

We introduced a tool for labeling behavior patterns in IIoT data of industrial processes through unsupervised learning of summary statistics. The tool's main features include: (1) comparing production data's feature vectors with cluster centers derived from reference data, (2) utilizing a deviation metric to quantify the divergence of production data from the original cluster centers, and (3) employing the deviation metric to identify potential drifts in production data, enabling the implementation of measures to mitigate drift effects.

ACKNOWLEDGMENTS

The work has been conducted as part of the DAT4.ZERO project (958363) and the ENFIELD project (101120657) funded by the European Commission within the H2020 Programme and the HEU Programme, and the Research Council of Norway's BIA-IPN programme under grant agreement No. 309700 (FLEET).

REFERENCES

[1] Ali Alqahtani, Mohammed Ali, Xianghua Xie, and Mark W Jones. 2021. Deep Time-Series Clustering: A Review. *Electronics* 10, 23 (2021), 3001.
[2] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *MOD'00*. 93–104.
[3] Chia-Shang James Chu. 1995. Time series segmentation: A sliding window approach. *Information Sciences* 85, 1-3 (1995), 147–173.
[4] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *VLDB Endowment* 1, 2 (2008), 1542–1552.
[5] Arda Goknil, Phu Nguyen, Sagar Sen, Dimitra Politaki, Harris Niavis, Karl John Pedersen, Abdillah Suyuthi, Abhilash Anand, and Amina Ziegenbein. 2023. A Systematic Review of Data Quality in CPS and IoT for Industry 4.0. *Comput. Surveys* 55, 14s, Article 327 (2023), 38 pages.
[6] Erik Johannes Husom, Simeon Tverdal, Arda Goknil, and Sagar Sen. 2022. Udava: An unsupervised learning pipeline for sensor data validation in manufacturing. In *CAIN'22*. 159–169.
[7] Mauro Isaja, Phu Nguyen, Arda Goknil, Sagar Sen, Erik Johannes Husom, Simeon Tverdal, Abhilash Anand, Yunman Jiang, Karl John Pedersen, Per Myrseth, et al. 2023. A blockchain-based framework for trusted quality data sharing towards zero-defect manufacturing. *Computers in Industry* 146 (2023), 103853.
[8] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *ICDM'08*. 413–422.
[9] Yuan Luo, Ya Xiao, Long Cheng, Guojun Peng, and Danfeng Yao. 2021. Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–36.
[10] Meinard Müller. 2007. Dynamic time warping. *Information retrieval for music and motion* (2007), 69–84.
[11] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13, 7 (2001), 1443–1471.
[12] Sagar Sen, Erik Johannes Husom, Arda Goknil, Dimitra Politaki, Simeon Tverdal, Phu Nguyen, and Nicolas Jourdan. 2023. Virtual sensors for erroneous data repair in manufacturing a machine learning pipeline. *Computers in Industry* 149 (2023), 103917.
[13] Sagar Sen, Erik Johannes Husom, Arda Goknil, Simeon Tverdal, and Phu Nguyen. 2023. Uncertainty-aware Virtual Sensors for Cyber-Physical Systems. *IEEE Software* (2023).
[14] Sagar Sen, Erik Johannes Husom, Arda Goknil, Simeon Tverdal, Phu Hong Nguyen, and Iker Mancisidor. 2022. Taming Data Quality in AI-Enabled Industrial Internet of Things. *IEEE Software* 39, 6 (2022), 35–42.
[15] Sagar Sen, Simon Myklebust Nielsen, Erik Johannes Husom, Arda Goknil, Simeon Tverdal, and Leonardo Sastoque Pinilla. 2023. Replay-driven continual learning for the industrial internet of things. In *CAIN'23*. 43–55.