2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
June 28-30, 2023. Seattle, Washington, USA

STAD-FEBTE, a shallow and supervised framework for time series anomaly detection by automatic feature engineering, balancing, and tree-based ensembles: An industrial case study

M.A. Zakeri Harandi[1], Chen Li[1], Casper Schou[1], Sigurd L. Villumsen[2], Simon Bøgh[1], Ole Madsen[1]

***Abstract*— Modern industrial systems are equipped with multi-sensor units, and building anomaly detection modules to monitor their collected data has become a vital task. Missing such abnormal patterns may cause producing faulty products, unwanted shutdowns in the production line, or even catastrophic damages. Sensor measurements of different natures with different sampling frequencies build a multivariate heterogeneous time series data. Conventional machine learning models fail to capture the temporal characteristics of such data. Deep learning models can address this thanks to their internal network architecture, yet training such models requires large datasets with adequate samples from all anomaly classes. This is not the case in real-world problems where class imbalance is a major issue. Tree-based ensembles are reported to have the dominant performance when dealing with structured tabular data. Inspired by this, we propose a supervised framework that combines an automatic feature engineering pipeline converting the time series dataset into its tabular counterpart with tree-based ensembles. The suggested method tackles class imbalance by generating synthetic anomalies using balancing techniques. Moreover, it allows handling heterogeneous multivariate data and augmenting categorical features with sensor measurements. Two real-world industrial datasets of relatively small size from robotized screwing processes are benchmarked, showing better results for the suggested framework compared to commonly used deep learning architectures.**

***Index Terms*— time series anomaly detection; automatic feature engineering; tree-based ensembles; robotic screwing**

I. Introduction and Background

Anomalies refer to observations that differ from the majority of the data to the extent that they raise suspicions. Whether it is the detection of fraudulent transactions in financial systems, intrusions in network systems, or sudden changes in environmental monitoring systems, time series anomaly detection (AD) has been studied extensively [1]. Due to the evolution of Industry 4.0 and thanks to advancements in the Internet of Things, modern industrial manufacturing systems are starting to be equipped with multi-sensor units capturing data from various features. This has enabled the development of monitoring systems built on top of the collected data to find abnormal patterns. Failure to detect such anomalies results in the production of faulty products, unwanted shutdowns in the production line, or even life-threatening accidents.

[1]M.A. Zakeri Harandi, Chen Li, Casper Schou, Simon Bøgh, and Ole Madsen are with the Robotics and Automation Group, Department of Materials and Production Eng., Aalborg University, Aalborg, Denmark. {mzakeri, cl, cs, sb, om}@mp.aau.dk

[2]Sigurd L. Villumsen is with the Global Production Technologies group, VELUX A/S, Østbirk, Denmark. sigurd.villumsen@velux.com

Time series AD is an important task with significant applications, thus a wide range of machine learning (ML) and deep learning (DL) methods are proposed to tackle that [2]. The task becomes more challenging in industrial processes. First, the developed monitoring system should be ***generalizible***, i.e., both process and sensor independent. Second, multi-sensor systems collecting data of different natures with different sampling frequencies build a ***multivariate heterogeneous*** time series dataset. Passing this data to the ML model while preserving internal sensor information and keeping samples from different classes distinguishable is not straightforward [3], [4]. Third, anomalies occur rarely in production, and in-advance replication of them in experimental setups is costly and sometimes ineffective. Therefore, most of the real-world AD datasets suffer from ***class imbalance***, i.e., their number of normal samples outweighs their anomalies by a large margin. Fourth, industrial systems are often subject to changes in their ***sensor configuration***, i.e., introduction of new sensors and the removal of old ones. Creating new datasets and retraining all models after any change in sensor configuration is impractical. Assessing the validity of the model trained on a previous sensor configuration is translated into the identification of ***concept drift*** [5] which should be addressed by the AD model as well. Last, all industrial processes are characterized by a set of ***categorical features*** that are needed to be involved in the anomaly detection process. Combining these features with time series data coming from sensor measurements prior to training is not simple from the viewpoint of data augmentation.

Two major classes of methods have been employed to tackle this problem. The first class involves passing the time series dataset through a data processing pipeline which finds useful features from the dataset. An ML model is then trained on top of the new data representation created by the obtained feature mapping [6]–[8]. We call this ***feature-based time series AD***. The second approach is to borrow DL architectures and apply them directly on the raw time series data [9]–[11]. Here, the feature extraction task is handled by the internal complex architecture of the involved artificial neural network. We call this ***deep time series AD***. Both methods come with challenges. Conventional ML models are tailored for tabular data, i.e., a collection of samples carrying identical features, and they fail to capture temporal and sequential characteristics of the time series data. The data processing step aims to fill this gap by extracting relevant features from the time series data and converting the dataset into a tabular one. However, traditional techniques are domain specific and expert knowledge demanding. Deep learning architectures are developed to remove this dependency and leave the feature engineering task to the internal architecture of the network. Besides, they can address changes in sensor configurations by transfer learning [9]. Training such models, nevertheless, demands large datasets with adequate number of samples representing each class of anomaly, making them prone to class imbalance which is a major issue in almost all real-world anomaly detection datasets. Also, several benchmark studies have revealed that depending on the AD task at hand, deep neural networks (DNNs) might outperform conventional ML models only by a narrow margin or even have a worse performance [12], [13]. Finally, identification of concept drift and augmenting categorical features into the model are difficult to achieve while dealing with DL architectures [2].

Machine learning ensembles have been extensively applied in numerous fields. In an ensemble architecture, instead of training a single model, a collection of models, aka base-learners, are trained on slight alternations of the training data and their estimations are combined to build a meta model known as the ensemble. While each base learner is normally a weak estimator with either high bias or variance, their aggregation builds a model with a drastically better performance that prevents overfitting, is less computationally expensive, and provides a better representation for the hypothesis, in addition to mitigating challenges such as class imbalance, curse of dimensionality, and concept drift [14]. ***Tree-based ensembles*** are the class of algorithms using decision trees as their base-learners. Such models are known to be the go-to model while dealing with *structured* or *tabular datasets* [15], [16]. In comparison to DL models, they have shown to have easier optimization processes, less training and tuning costs, and mostly better prediction performance [13].

Inspired by this, we propose a supervised time series AD framework designed for monitoring multi-sensor systems at the presence of class imbalance. The suggested framework combines an automatic feature engineering pipeline with balancing schemes and tree-based ensembles. Several attempts are reported in the literature following close ideas. In [6], a reconstruction-based AD score is computed using principal component analysis (PCA) applied on sensor measurements after feature extraction. In [7], a hierarchical feature extraction method using minimal sample representations is proposed to train low-cost ML models for detecting anomalies in power consumption systems. In [8], key features extracted from Markov transforms are combined with basic statistical features to form a vector representation of the time series data and detect anomalies in pressure measurements coming from oil pipelines. Though presenting promising results, neither of the frameworks take class imbalance into consideration. Besides, despite building tabular representations of the time series data, the performance of the monitoring system is not assessed using tree-based ensembles. Finally, none of the frameworks is benchmarked versus state-of-the-art DL architectures, failing to provide a versatile solution.

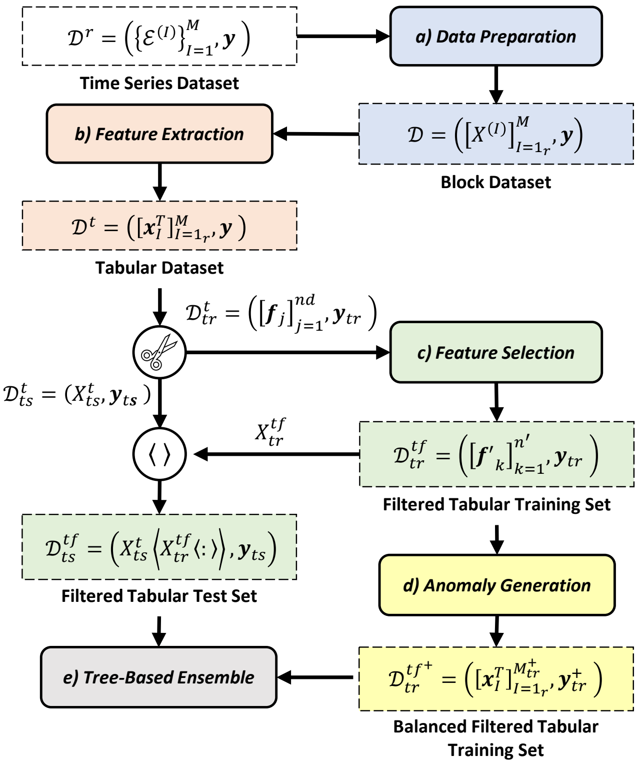

Fig. 1: Various modules of STAD-FEBTE. The input of the framework is a potentially imbalanced time series dataset in its raw format $\mathcal{D}^r$, and the outputs are a balanced filtered tabular training dataset $\mathcal{D}_{tr}^{tf^+}$ together with a filtered tabular test dartaset $\mathcal{D}_{ts}^{tf}$ passed to a tree-based ensemble.

The contributions of this work are as follows. 1) We propose a supervised framework for time series AD that incorporates an automatic feature engineering pipeline with tree-based ensembles; 2) The framework converts the time series dataset into its tabular counterpart allowing it to address class imbalance by synthesizing artificial anomalies using conventional balancing schemes; 3) The framework can handle heterogeneous multivariate sensor measurements, and it can augment categorical features with the time series data; 4) The proposed method is process independent, yet it is applied to two real-world datasets collecting data from robotized screwing processes, benchmarking the outcomes versus state-of-the-art DL models; 5) An open source anomaly detection dataset from a robotized wood screwing process together with the implementation of the proposed framework are available here[1].

II. Method

In this section, we present **S**hallow **T**ime series **A**nomaly **D**etection by **F**eature **E**ngineering, **B**alancing, and **T**ree-based **E**nsembles (STAD-FEBTE), which is a supervised framework for converting a potentially imbalanced time series dataset into

[1]https://github.com/AAU-RoboticsAutomationGroup/STAD-FEBTE

its balanced tabular counterpart and passing that to a tree-based ensemble. The framework is shallow since DNNs are not used in its architecture [12]. Fig. 1 shows an overview of various modules of STAD-FEBTE. It takes a labeled time series dataset as input, and it passes a balanced tabular training set and a test set with original class distribution to a tree-based ensemble. Both datasets are tabular, i.e., they are formed by event vectors having identical features. The framework has five major modules: *Data Preparation* (DP), *Feature Extraction* (FE), *Feature Selection* (FS), *Anomaly Generation* (AG), and *Tree-based Ensemble* (TE). First, the DP module receives the time series dataset in its raw format as a collection of event sets and creates data structures needed for the upcoming modules. Second, the FE module applies a collection of predefined unsupervised feature calculators to convert each event block into an event vector. Third, the resulting tabular dataset is split, and its training portion is passed to the FS module to find its dominant features. Next and to form a balanced training dataset, the AG module receives the filtered output of the FS module and generates synthetic anomalies. To avoid data leakage, samples of the test dataset are not involved in the FS and AG modules. Finally, the created tabular training and test datasets are passed to the TE module.

a) ***Data Preparation Module:*** Assume the time series dataset $\mathcal{D}^r$ consists of M number of events $\{\mathcal{E}^{(I)}\}_{I=1}^{M}$ and their ground truth anomaly labels $\boldsymbol{y} = [y_I]_{I=1}^{M}$. Suppose event $\mathcal{E}^{(I)}$ holds the values of n distinct time series $\boldsymbol{\tau}_j^{(I)}\,(1 \le j \le n)$ in their raw format. An event is a time window of an industrial process with various sensor measurements as its time series. Suppose $\boldsymbol{t}_j^{(I)}$ is the time vector of $\boldsymbol{\tau}_j^{(I)}$ holding the timestamps at which datapoints of the time series are collected. In some cases, all time series of the event $\mathcal{E}^{(I)}$ share the same time vector, i.e., they have an identical start time and a fixed sampling frequency. In industrial systems, this occurs when all measurements are logged on a single data acquisition device with a fixed clock. In some other scenarios, however, various time series of an event might have different time vectors. For instance, in a robotic screwing process, the data coming from the robot controller might have a different sampling frequency than the data collected from the screwdriver controller. The DP module forms two different data structures considering this notion.

Having Identical Time Vectors: Let $\boldsymbol{t}^{(I)} = [t^{(I)} + i]_{i=1}^{m^{(I)}}$ be the time vector shared between all time series of the event $\mathcal{E}^{(I)}$, where $t^{(I)}$ is the start time of the event and $m^{(I)}$ is its length. The *time series sample block* $X^{(I)}|_{m^{(I)}\times(n+2)}$ corresponding to event $\mathcal{E}^{(I)}$ is built by column-wise concatenation of the event's fixed time vector with all of its time series, preceded with an index column.

$$X^{(I)} = \begin{bmatrix} I\mathbf{1}_{m^{(I)}} & \boldsymbol{t}^{(I)} & \begin{bmatrix}\boldsymbol{\tau}_j^{(I)}\end{bmatrix}_{j=1}^{n} \end{bmatrix}$$

where $\mathbf{1}_{m^{(I)}}$ is the ones-vector of size $m^{(I)}$.

Having Varying Time Vectors: Let $\boldsymbol{t}_j^{(I)} = [t_j^{(I)} + i]_{i=1}^{m_j^{(I)}}$ be the time vector of $\boldsymbol{\tau}_j^{(I)}$, where $t_j^{(I)}$ and $m_j^{(I)}$ are its start time and length. In this case, the *time series sample block* $X^{(I)}|_{m^{(I)}\times 4}$ is built by putting all time vectors in the third column and their corresponding time series in the fourth column of a matrix, preceded with two index columns one holding the index of each time series and the other holding the index of the entire event.

$$X^{(I)} = \begin{bmatrix} I\mathbf{1}_{m^{(I)}} & \begin{bmatrix} j\mathbf{1}_{m_j^{(I)}} \end{bmatrix}_{j=1_r}^{n} & \begin{bmatrix}\boldsymbol{t}_j^{(I)}\end{bmatrix}_{j=1_r}^{n} & \begin{bmatrix}\boldsymbol{\tau}_j^{(I)}\end{bmatrix}_{j=1_r}^{n} \end{bmatrix}$$

where $m^{(I)} = \sum_{j=1}^{n} m_j^{(I)}$ is the total length of the event. Whether having identical or varying time vectors, the *time series data matrix* X is built by row-wise concatenation of all sample blocks. This data matrix together with the ground truth anomaly labels form the **block dataset** $\mathcal{D} = (X\,,\boldsymbol{y}) = ([X^{(I)}]_{I=1_r}^{M}\,,\boldsymbol{y})$ which is the input of the next module.

b) ***Feature Extraction Module:*** This module takes the block dataset $\mathcal{D}$ and transforms it into a tabular dataset where each sample block is converted to a sample vector. For this purpose, d number of features $q_{jk}\,(1 \le k \le d)$ are extracted from each time series of each sample block. Assume $\boldsymbol{\phi}_e = [\phi_k]_{k=1}^{d}$ is the feature extraction mapping that converts the times series $\boldsymbol{\tau}_j^{(I)}$ into the feature vector $\boldsymbol{q}_j^{(I)} = [q_{jk}^{(I)}]_{k=1}^{d}$, where each $\phi_k : \mathbb{R}^{m^{(I)}} \to \mathbb{R}$ is a single feature extractor. The computed feature vectors from all time series of the sample block $X^{(I)}$ are put together to build the *time series sample vector* $\boldsymbol{x}_I|_{(nd)\times 1}$. The design matrix $X^t|_{M\times(nd)}$ is constructed by sample vectors of the entire dataset. This data matrix together with the true anomaly labels build the **tabular dataset** $\mathcal{D}^t = (X^t\,,\boldsymbol{y}) = ([\boldsymbol{x}_I^T]_{I=1_r}^{M}\,,\boldsymbol{y})$.

Choosing a set of time series feature extractors $\phi_k\,(1 \le k \le d)$ to build the feature extraction mapping $\boldsymbol{\phi}_e$ in a general manner and with no dependency on domain knowledge has been an active research thread for more than a decade [17], [18]. In development of STAD-FEBTE, we have used the set of feature extractors proposed in [19], where a library of almost 800 features are extracted from each univariate time series of an event, with reported successful applications in a variety of domains [6], [20], [21].

c) ***Feature Selection Module:*** Feature selection refers to the process of identifying the most dominant features of a tabular dataset to remove irrelevant features and reduce the chances of overfitting. Several supervised [22] and unsupervised [17], [23] feature selection frameworks are proposed. In STAD-FEBTE, we employ the FRESH algorithm presented in [24] where for every feature column, an independent hypothesis test is performed to measure its significance in predicting the estimated output of the dataset. Assume the tabular dataset $\mathcal{D}^t$ is decomposed into the training dataset $\mathcal{D}_{tr}^t = (X_{tr}^t\,,\boldsymbol{y}_{tr})$ and the test dataset $\mathcal{D}_{ts}^t = (X_{ts}^t\,,\boldsymbol{y}_{ts})$ after stratified splitting. Suppose the training design matrix $X_{tr}^t = [\boldsymbol{f}_j]_{j=1}^{nd}$ including nd number of feature columns is converted to its filtered version $X_{tr}^{tf} = [\boldsymbol{f}'_k]_{k=1}^{n'}$ via the feature selection mapping $\Phi_s : \mathbb{R}^{M\times(nd)} \to \mathbb{R}^{M\times(n')}$, where $n' < n$ is the total number of selected features.

To avoid data leakage, samples of the test dataset should have been kept hidden from the FS module. Now that the dominant features are obtained using the training dataset, the selected features can be applied to the unfiltered test dataset X^t_{ts} to build $X^{tf}_{ts} = X^t_{ts}\langle X^{tf}_{tr}\langle:\rangle\rangle$, where the operator $X\langle:\rangle$ extracts all column labels of X and $X\langle cols\rangle$ extracts feature labels specified in *cols*. The filtered tabular training dataset $\mathcal{D}^{tf}_{tr} = (X^{tf}_{tr}, \boldsymbol{y}_{tr})$ is the input of the AG module, and the filtered tabular test dataset $\mathcal{D}^{tf}_{ts} = (X^{tf}_{ts}, \boldsymbol{y}_{ts})$ is the dataset passed to TE module for evaluating its performance after training. After feature selection, categorical features can be simply augmented as new columns in the obtained filtered design matrices X^{tf}_{tr} and X^{tf}_{ts}. Though not studied in the paper, drastic changes in the extracted features can also be used as an index to identify concept drift.

*d) **Anomaly Generation Module**:* Anomalies occur on rare occasions, and hence the ML model would have less chance to learn their underlying attributes as minority classes [25]. The AG module aims to generate new anomalies to tackle this. One major advantage of converting a time series dataset into its tabular counterpart is the ability to use conventional data balancing schemes for this purpose. Such methods are classified into three groups: *over-sampling*, *under-sampling*, and *combined*. Over-sampling normally outperforms under-sampling [26], but it comes with the side effect of *cluster overlapping*, i.e., newly generated samples of a minority class entering the underlying cluster of another class. Such noisy samples increase model complexity and chances of overfitting.

To reflect on this, we use one variant of a combined balancing method proposed in [26] and implemented in [27], which integrates SMOTE as a common over-sampling technique with Tomek as a data cleaning approach. The objective of the over-sampling part is to increase the size of the minority classes, and the purpose of data cleaning is to prevent cluster overlapping. For every pair of samples $\boldsymbol{x}_{I_1}$ and $\boldsymbol{x}_{I_2}$ belonging to the *same anomaly class* C_I, SMOTE generates a synthetic anomaly example $\boldsymbol{x}_{new} = \boldsymbol{x}_{I_1} + \lambda(\boldsymbol{x}_{I_2} - \boldsymbol{x}_{I_1})$ using their linear interpolation, where $\lambda \in (0, 1)$ is a randomly generated number. This process is repeated until the desired size of the anomaly class is obtained. To remove overlapping, for every pair of samples $\boldsymbol{x}_I$ and $\boldsymbol{x}_J$ belonging to two *different anomaly classes* C_I and C_J which are distanced by $d(\boldsymbol{x}_I, \boldsymbol{x}_J)$ in the feature space, Tomek aims to find another sample $\boldsymbol{x}_L \in C_I$ which is closer to $\boldsymbol{x}_I$ compared to $\boldsymbol{x}_J$, i.e., $d(\boldsymbol{x}_L, \boldsymbol{x}_I) < d(\boldsymbol{x}_I, \boldsymbol{x}_J)$. If not found, $\boldsymbol{x}_I$ is raised as a potential noise and is removed. The simple underlying idea is that if $\boldsymbol{x}_I$ is not invading the cluster of $\boldsymbol{x}_J$, we should be able to find a closer sample to it from its own class. Similar to the FS module, test dataset should be kept separate from the AG module. Intuitively, samples generated by the ML model itself should not be used to assess its performance. Suppose $\Phi_a : \mathbb{R}^{M_{tr}\times n'} \times \mathbb{R}^{M_{tr}} \to \mathbb{R}^{M^+_{tr}\times n'} \times \mathbb{R}^{M^+_{tr}}$ is the anomaly generation mapping which converts the filtered tabular dataset $\mathcal{D}^{tf}_{tr}$ into its balanced version $\mathcal{D}^{tf^+}_{tr} = ([\boldsymbol{x}^T_I]^{M^+_{tr}}_{I=1_r}, \boldsymbol{y}^+_{tr})$, where $M^+_{tr} - M_{tr}$ is the number of newly generated samples by the AG module. The balanced training dataset $\mathcal{D}^{tf^+}_{tr}$ is the dataset passed to the TE module for training.

*e) **Tree-Based Ensemble**:* As mentioned in section I, tree-based ensembles are among the best estimators when it comes to tabular data. Depending on the notion of sharing information in between base-learners while training, these models are categorized into the two groups of *parallel* and *sequential* methods. In this paper, we have selected an alternation of bagging known as random patches (Bagging) [28], random forest (RF) [29], and extremely randomized trees (ExtraTrees) [30] as parallel tree-based ensembles, and adaptive boosting (AdaBoost) [31] and histogram-based gradient boosting (GradBoost) [32] as sequential tree-based ensembles to benchmark our datasets, all implemented in scikit-learn [33].

III. Experimental Results and Discussions

The main objective of our research is to assess the performance of STAD-FEBTE in anomaly detection of industrial operations. Despite the fact that the framework is process-independent, robotized screwing is studied as a test-bed. Two real-world datasets are benchmarked. The first one is AURSAD' which is a subset of the dataset presented in [34], and the second one is AAUWSD which is the Aalborg University Wood Screwing Dataset presented in this paper. Table I shows the attributes as well various classes of each dataset. We define *process attributes* as the time series data related to the screwing process itself, and *task attributes* as the time series data collected from the robot holding the screwdriver. Both datasets are highly imbalanced and of relatively small size (almost 2000 samples), putting the claims of STAD-FEBTE to the test.

***Datasets**:* To create AURSAD', each event of the original dataset is sliced from the beginning of its engagement phase to the end of its clamping phases [35]. Moreover, the dataset is filtered for its dominant process and task features. The major motivation behind creating AAUWSD was to include additional process attributes which play vital roles in characterizing the screwing process [36], but they were absent in AURSAD. Also, to reflect on the impact of material uncertainties, AAUWSD collects data from self-tapping screwing into wood. Moreover, the dataset covers other commonly observed classes of anomaly. As the names suggest, *under/over-tightening* occur when termination torque

TABLE I: Two screwing datasets studied in the paper

Dataset	Process Attributes	Task Attributes	Classes
AURSAD'	insertion torque $\boldsymbol{\tau}$	TCP pose $\boldsymbol{\rho}_{k=1\ldots6}$ TCP twist $\boldsymbol{\nu}_{k=1\ldots6}$ TCP wrench $\boldsymbol{\mathcal{F}}_{k=1\ldots6}$	normal(70%) damaged screw(11%) extra component(9%) missing screw(10%)
AAUWSD	insertion torque $\boldsymbol{\tau}$ insertion angle $\boldsymbol{\theta}$ insertion angle $\mathbf{d}$ insertion current $\mathbf{i}$	none	normal(33%) under-tightening(34%) over-tightening(28%) pose anomaly(3%) missing screw(3%)

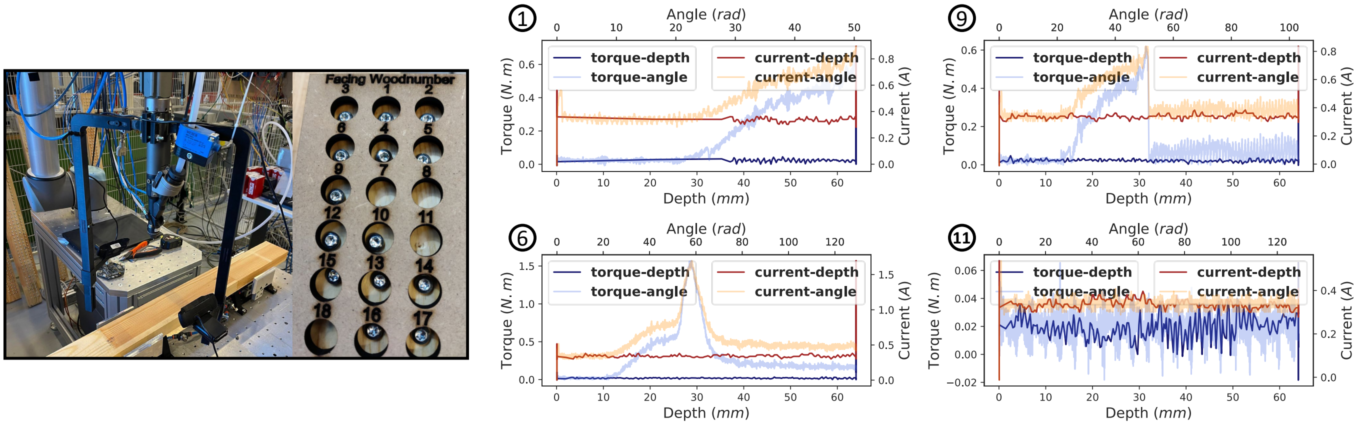

Fig. 2: Process data from 4 samples of AAUWSD, where top-left is *normal screwing*, top-right is *under-tightening*, bottom-left is *over-tightening*, and bottom-right is *pose anomaly*. The *missing screw* anomaly class is not included as it merely depicts noise measurements.

TABLE II: Tree-based ensembles trained by STAD-FEBTE versus DL models trained on the raw time series data

	Model	Accuracy	Balanced Accuracy	Recall	F1	ROC AUC
STAD-FEBTE	AdaBoost	0.835	0.727	0.835	0.8322	0.8915
	Bagging	0.8861	**0.7979**	0.8861	0.8831	0.9135
	ExtraTrees	**0.888**	0.7908	**0.888**	**0.8839**	0.8919
	GradBoost	**0.888**	0.7916	**0.888**	0.8833	0.9076
	RF	0.8762	0.7944	0.8762	0.8753	**0.9186**
DL	Conv1D	0.8771	0.7795	0.7795	0.7945	0.8854
	MHConv1D	0.8796	0.7692	0.7692	0.7818	0.8759
	LSTM	0.6929	0.25	0.25	0.2046	0.5
	ConvLSTM	0.8697	0.7581	0.7581	0.7779	0.8871
	Tansformer	0.8771	0.7752	0.7752	0.7839	0.9178

(a) AURSAD’

	Model	Accuracy	Balanced Accuracy	Recall	F1	ROC AUC
STAD-FEBTE	AdaBoost	0.3267	0.5418	0.3267	0.3073	0.8035
	Bagging	0.982	0.9893	0.982	0.982	0.9994
	ExtraTrees	0.984	0.9774	0.984	0.984	**0.9996**
	GradBoost	**0.986**	**0.9917**	**0.986**	**0.986**	0.9995
	RF	0.984	0.9905	0.984	0.984	**0.9996**
DL	Conv1D	0.9248	0.5772	0.5772	0.5634	0.975
	MHConv1D	0.9323	0.5759	0.5759	0.5619	0.9695
	LSTM	0.6266	0.4429	0.4429	0.3691	0.8048
	ConvLSTM	0.4586	0.2	0.2	0.1082	0.5
	Tansformer	0.9298	0.5795	0.5795	0.5648	0.9748

(b) AAUWSD

of the process is lower/higher than the required fastening torque. On the other hand, *pose anomaly* occurs due to the misalignment of the robot’s end effector with respect to the workpiece resulting in slippage. Fig. 2 illustrates time series data for a normal screwing process together with three of the anomaly classes of AAUWSD.

Evaluation Metrics: In anomaly detection, false negatives are faulty processes which are missed to be detected, and false positives are non-faulty processes which are mistakenly labeled as anomaly. Obviously, the significance of lowering false negatives outweighs the importance of reducing false positives. There is a correlation between false positive and false negative rates, and an ML model cannot minimize both at the same time. In this regard, *accuracy* can easily become a misleading metric, especially while dealing with imbalanced datasets. To address this, we have evaluated each model using 4 extra metrics: *balanced accuracy* which takes class proportion into account, *recall* which measures the true positive rate of each class, *f1-score* which computes the harmonic mean of precision and recall, and *ROC-AUC* score which returns the area under the *true positive-false positive curve* after altering the decision function threshold. Since the task at hand is multi-class classification, all metrics perform weighted averaging taking class proportions into account, and the ROC-AUC score is computed using one-versus-all.

Main Results: Table II summarizes the results of validating tree-based ensembles mentioned in section II trained by STAD-FEBTE on the two datasets studied in the paper. For screwing operation, each event is created by rolling a dynamic window on sensor measurements where its size is adjusted by the length of a complete process. All models are trained with 500 number of estimators. The hyperparameters of the models trained on AURSAD’ are tuned using grid-search combined with cross validation. For AAUWSD, all models are trained using default hyperparametrs. Due to the fact that deep learning models are capable of learning and automatically extracting features and complex patterns in time series data via a hierarchy of non-linear transformations [37], various DL models are also trained and validated on the raw time series data $\mathcal{D}^r$. For this purpose, we have used 1-dimensional convolutional neural nets (Conv1D) [38], multi-head 1-dimensional convolutional neural nets (MHConv1D), long short-term memory networks (LSTM) [39], convolutional LSTM (ConvLSTM) [40], and transformers [41]. All models are implemented in TensorFlow and are trained with learning rate of 0.0001, patience of 30, a batch size of 8, and for 100 epochs. For MHConv1D, three different head sizes, 3, 5, and 11 are selected to aid the model in reading and interpreting time series data at different resolutions. The vanilla transformer is designed with 6 encoder blocks and

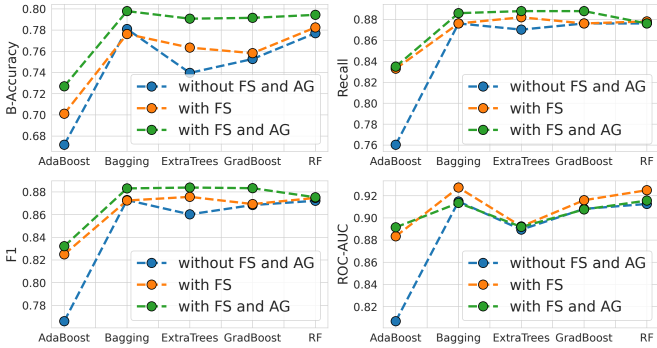

Fig. 3: Impact of FE, FS, and AG modules measured on AURSAD'

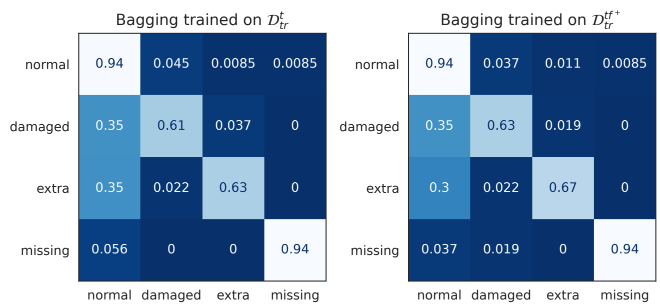

Fig. 4: *Bagging* trained by STAD-FEBTE without and with FS&AG modules for AURSAD'

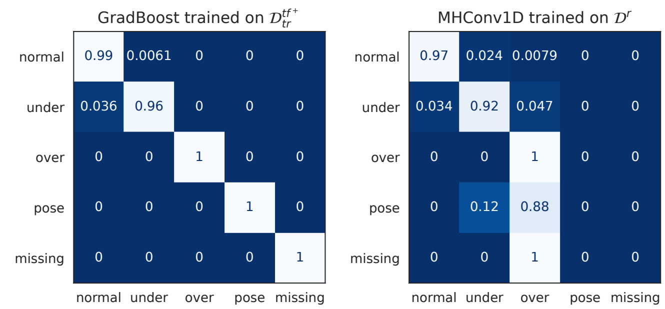

Fig. 5: *GradBoost* trained by STAD-FEBTE versus *MHConv1D* trained on raw data for AAUWSD

4 heads with head size of 256.

We observed that in both datasets, STAD-FEBTE outperforms DL architectures. For AURSAD', while *ExtraTrees* has better results in three metrics, it does not perform well in maximizing the ROC-AUC score. *Bagging*, on the other hand, has the best average performance among all models, and *RF* stands in the second place. To emphasize the impact of various modules of the framework on detecting anomalies, in Fig. 3 each model is trained and validated on three pairs of datasets produced after three major modules of STAD-FEBTE, i.e., FE, FS, and AG. It is observed that the performance of all models with respect to all metrics is improved after passing training data through the FS module. In other words, removing redundant features from the tabular dataset $\mathcal{D}^{t}_{tr}$ and projecting samples into a subspace of the original feature space provides a simpler classification task to solve. The same trend is observed for the AG module with the exception of ROC-AUC score, where negligible drops of almost 1% are observed in three models. The module aims to balance the dataset by generating synthetic anomalies, and at the same time to reduce the negative impact of this process by removing overlapping samples. This improvement is also observed in Fig. 4 where the ability of the *Bagging* classifier to detect damaged and extra component anomalies is improved by 2% and 4% respectively, after inclusion of the FS and AG modules.

The performance difference between STAD-FEBTE and the DL models becomes more significant for AAUWSD. With the exception of *AdaBoost*, evaluation scores of all tree-based ensembles exceed 97% in terms of all metrics. For DL models, even though *Conv1D* and *MHConv1D* result in relatively high accuracies, they are dominated by tree-based ensembles when it comes to balanced accuracy, recall, and f1-score. This is also observed in Fig. 5 where *MHConv1D* mistakenly detects all missing and 88% of pose anomalies as over-tightening samples, indicating why accuracy can become a misleading factor in imbalanced AD. Various reasonses can be addressed to justify this performance. First, these two classes are minorities which occupy only 6% of the dataset population. Where STAD-FEBTE performs balancing to address class-imbalance, the complex architecture of DL models requires more number of samples from each class to learn their underlying features. Second, missing screw and pose anomaly samples have significantly (almost 2500) more number of datapoints in comparison to other classes. Where the FE module of STAD-FEBTE extracts the same number of features from all sensor measurements irrespective of their length, deep learning pipelines tackle this by zero padding or up/down sampling of the time series signals. The resulted misalignment in created sample tensors can also become a source of weak performance.

IV. Conclusion and Future Works

In this paper, we presented STAD-FEBTE as a supervised framework for time series anomaly detection. Benchmarking the pipeline on two industrial datasets, it was observed that converting the time series dataset into its tabular counterpart while dealing with class imbalance, and passing the obtained datasets to tree-based ensembles can outperform state-of-the-art deep learning models. There are various directions to extend the presented outcomes. First, the framework should be applied to other industrial processes to assess how generalizable it is. Second, since factory floor data lacks ground truth labels and generating anomalies is costly and sometimes infeasible depending on the process at hand, the next step is to develop an unsupervised version of the framework. Finally, other sources of process attributes such as images or acoustic data should be integrated into the framework to build even better monitoring systems.

References

[1] K. Shaukat, T. M. Alam, S. Luo, S. Shabbir, I. A. Hameed, J. Li, S. K. Abbas, and U. Javed, "A Review of Time-Series Anomaly Detection Techniques: A Step to Future Perspectives," in *Advances in Information and Communication*, ser. Advances in Intelligent Systems and Computing, K. Arai, Ed. Cham: Springer International Publishing, 2021, pp. 865–877.

[2] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A Review on Outlier/Anomaly Detection in Time Series Data," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–33, Apr. 2022.

[3] M. Dao, N. H. Nguyen, N. M. Nasrabadi, and T. D. Tran, "Collaborative Multi-Sensor Classification Via Sparsity-Based Representation," *IEEE Transactions on Signal Processing*, vol. 64, no. 9, pp. 2400–2415, May 2016.

[4] J. Cao, W. Li, C. Ma, and Z. Tao, "Optimizing multi-sensor deployment via ensemble pruning for wearable activity recognition," *Information Fusion*, vol. 41, pp. 68–79, May 2018.

[5] J. P. Barddal, H. M. Gomes, and F. Enembreck, "A survey on feature drift adaptation," in *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2015, pp. 1053–1060.

[6] H. Y. Teh, K. I.-K. Wang, and A. W. Kempa-Liehr, "Expect the Unexpected: Unsupervised Feature Selection for Automated Sensor Anomaly Detection," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 18 033–18 046, Aug. 2021.

[7] Z. Ouyang, X. Sun, and D. Yue, "Hierarchical Time Series Feature Extraction for Power Consumption Anomaly Detection," in *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration*, ser. Communications in Computer and Information Science, K. Li, Y. Xue, S. Cui, Q. Niu, Z. Yang, and P. Luk, Eds. Singapore: Springer, 2017, pp. 267–275.

[8] D. Zang, J. Liu, and H. Wang, "Markov chain-based feature extraction for anomaly detection in time series and its industrial application," in *2018 Chinese Control And Decision Conference (CCDC)*, Jun. 2018, pp. 1059–1063.

[9] M. Canizo, I. Triguero, A. Conde, and E. Onieva, "Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study," *Neurocomputing*, vol. 363, pp. 246–260, Oct. 2019.

[10] D. Kim, H. Yang, M. Chung, S. Cho, H. Kim, M. Kim, K. Kim, and E. Kim, "Squeezed Convolutional Variational AutoEncoder for unsupervised anomaly detection in edge device industrial Internet of Things," in *2018 International Conference on Information and Computer Technologies (ICICT)*. DeKalb, IL: IEEE, Mar. 2018, pp. 67–71.

[11] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng, "Learning Graph Structures With Transformer for Multivariate Time-Series Anomaly Detection in IoT," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9179–9189, Jun. 2022.

[12] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A Unifying Review of Deep and Shallow Anomaly Detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, May 2021.

[13] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, May 2022.

[14] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, Jul. 2018.

[15] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[16] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[17] B. D. Fulcher, M. A. Little, and N. S. Jones, "Highly comparative time-series analysis: The empirical structure of time series and their methods," *Journal of the Royal Society Interface*, vol. 10, no. 83, p. 20130048, Jun. 2013.

[18] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, "Catch22: CAnonical Time-series CHaracteristics: Selected through highly comparative time-series analysis," *Data Mining and Knowledge Discovery*, vol. 33, no. 6, pp. 1821–1852, Nov. 2019.

[19] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)," *Neurocomputing*, vol. 307, pp. 72–77, Sep. 2018.

[20] A. W. Kempa-Liehr, J. Oram, A. Wong, M. Finch, and T. Besier, "Feature Engineering Workflow for Activity Recognition from Synchronized Inertial Measurement Units," in *Pattern Recognition*, ser. Communications in Computer and Information Science, M. Cree, F. Huang, J. Yuan, and W. Q. Yan, Eds. Singapore: Springer, 2020, pp. 223–231.

[21] D. E. Dempsey, S. J. Cronin, S. Mei, and A. W. Kempa-Liehr, "Automatic precursor recognition and real-time forecasting of sudden explosive volcanic eruptions at Whakaari, New Zealand," *Nature Communications*, vol. 11, no. 1, p. 3562, Dec. 2020.

[22] S. Huang, "Supervised feature selection: A tutorial," *Artificial Intelligence Research*, vol. 4, Mar. 2015.

[23] X. Huang, L. Wu, and Y. Ye, "A Review on Dimensionality Reduction Techniques," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 33, no. 10, p. 1950017, Sep. 2019.

[24] M. Christ, A. W. Kempa-Liehr, and M. Feindt, "Distributed and parallel time series feature extraction for industrial big data applications," Asian Conference on Machine Learning (ACML), May 2017.

[25] R. C. Prati, G. E. Batista, and M. C. Monard, "Class imbalances versus class overlapping: An analysis of a learning system behavior," in *Mexican International Conference on Artificial Intelligence*. Springer, 2004, pp. 312–321.

[26] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, Jun. 2004.

[27] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 559–563, 2017.

[28] G. Louppe and P. Geurts, "Ensembles on random patches," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 346–361.

[29] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[30] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[31] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[32] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] B. Leporowski, D. Tola, C. Hansen, and A. Iosifidis, "Detecting Faults During Automatic Screwdriving: A Dataset and Use Case of Anomaly Detection for Automatic Screwdriving," in *Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems*. Cham: Springer International Publishing, 2022, pp. 224–232.

[35] L. D. Seneviratne, F. A. Ngemoh, S. W. E. Earles, and K. A. Althoefer, "Theoretical modelling of the self-tapping screw fastening process," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 215, no. 2, pp. 135–154, Feb. 2001.

[36] T. Lázár and J. Nagy, "THE COMPARISON OF THE KNOWN MODELS OF SELF-TAPPING SCREW JOINTS," *Machines. Technologies. Materials.*, vol. 11, no. 5, pp. 240–245, 2017.

[37] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, 2017.

[38] W. Tang, G. Long, L. Liu, T. Zhou, J. Jiang, and M. Blumenstein, "Rethinking 1d-cnn for time series classification: A stronger baseline," *arXiv preprint arXiv:2002.10061*, 2020.

[39] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The performance of lstm and bilstm in forecasting time series," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 3285–3292.

[40] Y. Wang, L. Sun, D. Peng, and K. Rajakani, "A multihead convlstm for time series classification in ehealth industry 4.0," *Wirel. Commun. Mob. Comput.*, vol. 2022, jan 2022. [Online]. Available: https://doi.org/10.1155/2022/8773900

[41] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.