Supervised Time Series Segmentation as Enabler of Multi-Phased Time Series Classification: A Study on Hydraulic End-of-Line Testing

1st Stefan Gaugel
Dptm. Factory of the Future
Bosch Rexroth AG
Ulm, Germany
stefan.gaugel@boschrexroth.de

2nd Binlan Wu
Dptm. Factory of the Future
Bosch Rexroth AG)
Ulm, Germany

3nd Adarsh Anand
Dptm. Factory of the Future
Bosch Rexroth AG)
Ulm, Germany

4nd Manfred Reichert
Institute of Databases and Information Systems
Ulm University
Ulm, Germany
manfred.reichert@uni-ulm.de

***Abstract*—Multi-phased time series are found in many industrial processes. Their classification still poses a big challenge for algorithms compared to single-phased time series forms. To overcome this issue, this paper suggests using deep learning to generate timestamp-wise state labels that serve as semantic annotations for all measured data points. We investigate whether the availability of state labels can boost the performance of machine learning classifiers by enabling state-wise feature extraction in multi-phased time series. The study is performed on a real-world industrial classification problem in a hydraulic pump factory. Various state label predictions with different accuracy scores are created via deep learning-based time series segmentation. We evaluate how the accuracies of the state label predictions affect the results of the binary classification. Our results show that in settings where accurate state labels are present the classification F1-scores were significantly higher compared to baseline approaches. Therefore, we emphasized the need to find well performing time series segmentation methods.**

***Index Terms*—Time Series Classification, Time Series Segmentation, Deep Learning, Machine Learning, End-of-Line Testing**

I. Introduction

A. Problem Statement

Time-series based Artificial Intelligence (AI) gained a lot of popularity in recent years. Historically, image recognition and natural language processing have been the frontrunners of AI research. However, a lot of the successes achieved in these fields could be transferred to the time-series domain as well. Especially in the context of Industry 4.0, the growing need for connectivity and sensor data collection increased the pressure to find intelligent methods to mine time series data. Time Series Classification (TSC) poses a special challenge compared to many other use cases: Usually, time series data are collected continuously, which defines their nature as one long series instead of a number of clearly distinct samples (compared to e.g. images). Therefore, windowing or segmenting the time series into different subsets usually becomes necessary when performing a classification task on them. After the segmentation, each time series subset (called segment) is considered as separate sample. Moreover, a label is assigned to each segment, before using the samples to train a Machine Learning (ML) model. This procedure is observable in the majority of TSC use cases. It is characterized by the fact that segmentation and classification of the time series represent the same task, i.e., finding the correct label for a subset of the series. This fact leads to some confusing overlapping of terminologies in the research body, especially concerning the terms (supervised) Time Series Segmentation (TSS), Time Series Classification and Time Series Labeling.

However, in this paper we discuss a multi-phased time series classification problem in the field of discrete manufacturing where TSS and TSC represent two fundamentally different tasks. Compared to other settings where usually one long, continuously measured time series is discussed (e.g., economics), the sensor-measured production data in discrete manufacturing can be allocated to a specific product. In turn, one sample represents the process data of one product, with the start and end point of a time series sample being automatically defined. The nature of each sample is usually multi-phased: a product goes through a number of highly-varying process phases in a predetermined order. This multi-phased nature of industrial process data poses big challenges for ML-algorithms, especially in the feature extraction stage. We want to discuss this issue exemplarily on the real-word industrial use case of End-of-Line-Testing (EoLT) in a hydraulic pump factory.

EoLT is a functional quality examination performed for a manufactured product before it is delivered to the customer.

Each manufactured pump is mounted on a testbench and examined by following a predefined testing cycle with different test steps. The attached sensors measure the pump function to provide information deciding whether a tested pump is healthy or faulty. The usual decision procedure includes comparing measured values with defined tolerance limits and a visual inspection of the characteristic curves by an human operator. To increase the accuracy and fully automate the fault detection process, it is planned to use a trained ML-model for the quality evaluation (TSC) in the future. However, as most pump faults can only be identified in one or very few test steps of the cycle, global (sample-wide) feature extraction alone is not sufficient for accurate classification. To additionally enable test step-specific local feature extraction, an accurate segmentation of the test steps within the testing cycle becomes necessary (TSS). However, compared to other time series settings, there is close to none research about how to handle such a multi-phased time series classification problem where TSC and TSS represent fundamentally different tasks.

B. Own contribution

We propose using an approach including both time series segmentation and time series classification to solve multi-phased time series classification problems. The approach includes two different kind of labels: TSC aims to assign one specific class label to a sample, while TSS aims to assign a (operational) state label to each timestamp within a sample (dense labeling). A schematic visualization of a time series with both state and class labels is shown in Figure 1. The target of our method is to achieve a high TSC accuracy for predicting the class label, where the state labels act as auxiliary semantic information enabling a precise local feature extraction that is performed isolatedly for each state. For our use class, the class label assigned to each entire sample is defined to be a quality label stating if the product is healthy or faulty. Moreover, each test step ("operational state") is represented by a specific state label value.

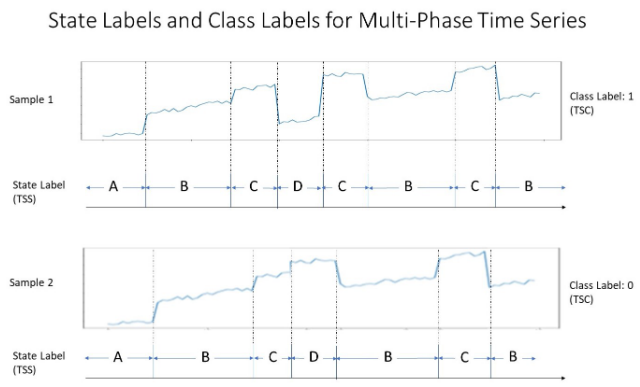

Fig. 1: TSS as timestamp-wise state labeling and TSC as sample-wise class labeling

Performing supervised deep learning-based time series segmentation to achieve a complete state labeling constitutes step one of our suggested two-step approach. With accurate state labels within each cycle defined, state-specific feature extraction can be used to add test step-specific feature to the total feature pool. Using the extracted features, a ML-model can be trained and utilized for TSC in step two. The main research objective of this paper is to examine the relation between the supervised TSS accuracy (state labeling) and its impact on the ML-based TSC results (quality prediction). To the best of our knowledge, no prior research systematically analysed a multi-phased time series classification setting with an interaction between TSS and TSC. Our claimed contributions are threefold:

- The paper presents the first approach and use case that requires a combination of both ML-based TSS and TSC to solve the task effectively. The use case is based on an real-world industrial application and uses a novel recently published dataset [1].
- The benefits of using state-specific local feature extraction are examined compared to baselines for which no state labels are available (e.g., TSC based on global feature extraction). The comparison is done over a range of ML classifiers to draw robust conclusion.
- Three different state label predictions of varying accuracy are created via a TSS deep learning model. It is analyzed how the accuracy of the TSS predictions affect the F1-score of the ML classifier for the class label predictions (TSC).

The paper is structured as follows: Section II presents the results of a literature research. Section III provides an overview of the experimental setup and the used data. Chapter IV summarizes and discusses the results, and Section V concludes the paper and provides an outlook on future research.

II. Literature Research

A. Time Series Classification

TSC is a long-existing problem in the academic world found in many different fields like economics, medicine and manufacturing. A good overview of recently proposed methods is found in [2]. As a particular challenge of TSC, there is no one-size-fits all solution for all types of time series, but instead a strong domain influence leading to a high fragmentation of existing TSC methods. Three distinct main groups of TSC methods exist: distance-based approaches [3], feature extraction-based approaches [4], and deep learning-based approaches [5]. The distance-based method of Dynamic Time Warping combined with 1-Nearest-Neighbour classification is widely considered as the state-of-the-art for general TSC and hard to beat in many use cases [6]. As drawback, Dynamic Time Warping it is a global approach only showing good results when the total signal-to-noise ratio of the time series samples is significantly high. In many industrial use cases, however, the relevant discriminate information is only found within a short subsequence of each time series sample, especially in multi-phased time series. In global approaches, there is the risk that the signal information of single subsequences are obscured by the existing noise and high amount of

indiscriminate information found across the whole sequence [7]. Thus, it was discovered that for many use cases some sort of locally limited feature extraction is necessary, e.g., in manufacturing [8] or medicine [9]. The EoLT fault detection setting discussed in this paper is one example of this, as the discriminate information of many pump faults can only be identified within specific segments of the data. Therefore, this review focuses on existing TSC approaches which include a form of segmentation to extract local information before using a ML-classifier.

B. Segmentation for local feature extraction

Most segmentation methods found in TSC approaches use either a form of unsupervised segmentation [10] or random segmentation [11]. Used classifiers include Support Vector Machines [10] and Ensemble Methods like Random Forests [11]. Most methods which use segmentation before classification belong to the group of time series adapted Bag-of-Words (BoW) / Bag-of-Features (BoF) approaches. These methods originate in the natural language processing field and usually follow a three step procedure: (1) Segmentation, (2) Feature extraction, (3) Encoding. The goal is to achieve a lower-dimension time series encoding helping a classifier to find both the globally present and locally-limited discriminate information within the time series. Other non-BoW / BoF methods use a form of segmentation, extract statistical or hand-crafted feature of selected segments, and train a classifier based on the extracted features [9].

C. End-of-Line-Testing and Fault Detection

Fault detection in EoLT is a complex application of data-driven approaches and ML methods. The multi-phase and multivariate nature of the data often make the signal-to-noise ratio too low for anomaly detection methods and the low prevalence of fault samples leads to high class imbalance aggravating supervised approaches [12] [13]. Nonetheless, a number of contributions that use ML to improve EoLT were published in recent years, tackling problems like sensor redundancy detection [14]. Depending on the available data, some authors interpret fault detection in EoLT as a anomaly detection task solvable by Isolation Forest algorithms [15], while others define it as binary TSC problem and use supervised learning approaches [16].

D. Literature Gap

No previous works tried to solve the challenge of multi-phased TSC by introducing timestamp-specific state labels and relying on supervised TSS as preceding task to TSC. Therefore, we are the first who have the chance to analyse a setting where we can let both supervised TSS and TSC interact with each other directly. To the best of our knowledge, no prior work evaluated the influence of the TSS accuracy on the TSC F1-score in a single setting before. We try to collect evidence that accurate TSS can boost TSC in a practical industrial application (multi-phased time series) and therefore underline the importance to find accurate TSS algorithms.

III. Methodology, Experimental Desgin and Data

A. Problem Formalization

The problem addressed in this paper can be formalized as a supervised multivariate TSS and TSC problem. Let $\mathcal{I}$ be the set of unique samples in the given time series dataset, and V be the set of all possible state labels. All samples $i \in \mathcal{I}$ are time series of length $\mathcal{T}$. For univariate time series, each $i \in \mathcal{I}$ and timestep $t \leq \mathcal{T}$ entry $i \in \mathcal{I}$ is associated with input $(\mathcal{X}_{i,t})_{t\in[0,\mathcal{T}]}$ and groundtruth class label y_i that denotes the class of a sample. For sensor-based multivariate time series, we further formalise $\mathcal{X}_{i,t}$ as: $\mathcal{X}_{i,t} = X_{i,t}^{\mathcal{S}} = (x_{i,t}^{s})_{s\in\mathcal{S}}$ where $\mathcal{S}$ denotes the set of sensors. We then train a deep learning model for TSS based on $(\mathcal{X}_{i,t})_{i\in\mathcal{I},t\leq\mathcal{T}}$ and the ground truth state labels

$$Z = (Z_i)_{i\in\mathcal{I}} = (z_{i,t})_{i\in\mathcal{I},t\leq\mathcal{T}} \quad (z_{i,t} \in V)$$

to learn the set of predicted state labels

$$\hat{Z} = (\hat{Z}_i)_{i\in\mathcal{I}} = (\hat{z}_{i,t})_{i\in\mathcal{I},t\leq\mathcal{T}} \quad (\hat{z}_{i,t} \in V)$$

which minimizes the cross entropy loss:

$$L(Z, \hat{Z}) = -\sum_{v\in V}\sum_{i\in\mathcal{I}}\sum_{t\leq\mathcal{T}} z_{i,t}^{v} \log(\hat{z}_{i,t}^{v})$$

where $z_{i,t}^{v} = \mathbb{1}_{\{z_{i,t}=v\}}$ and $\hat{z}_{i,t}^{v}$ denotes the probability of predicted state label at time $t \leq \mathcal{T}$ of entity $i \in \mathcal{I}$ being v. The variable of interest in the TSC problem is then denoted as $X_{i,t}^{\mathcal{S}\bigcup\hat{\mathcal{Z}}}$. Thus, for each $i \in \mathcal{I}$, the TSC problem can be formulated as

$$\hat{y_i} = f((X_{i,t}^{s})_{t\in\mathcal{T}}^{s\in\mathcal{S}\bigcup\hat{\mathcal{Z}}}) = f(X_{i,1}^{\mathcal{S}\bigcup\hat{\mathcal{Z}}}, X_{i,2}^{\mathcal{S}\bigcup\hat{\mathcal{Z}}}, \cdots, X_{i,\mathcal{T}}^{\mathcal{S}\bigcup\hat{\mathcal{Z}}})$$

The goal is to find a function (classifier) f such that the $F_1 - score$ (1) becomes minimized when comparing the predicted class label $\hat{y_i}$ with the groundtruth class label y_i.

B. Proposed Approach

Our proposed two-step approach combining segmentation and classification is illustrated in Figure 2 and explained in detail in Section III-C. Two types of labels are distinguished. State labels are time-stamp specific and include semantic information about the operational state of the measured data point. In our pump testing use case, a operational state is equivalent to a test step within a testing cycle. Class labels are sample specific and provide information about the quality of the pump (healthy or faulty). In general, the approach requires that groundtruth state labels are available for enough samples to train a deep learning-based TSS model. The groundtruth state labeling is usually performed manually by an process expert. The trained TSS model is then used to add state labels to the remaining unlabeled samples. Next, the predicted state labels are used to extract state-based features that serve as input (feature pool) for the TSC. After splitting the samples into train and test set, a ML-classifier is trained and and evaluated using the assigned class labels. The approach is assumed to be generally applicable to all multi-phased time series classification problems.

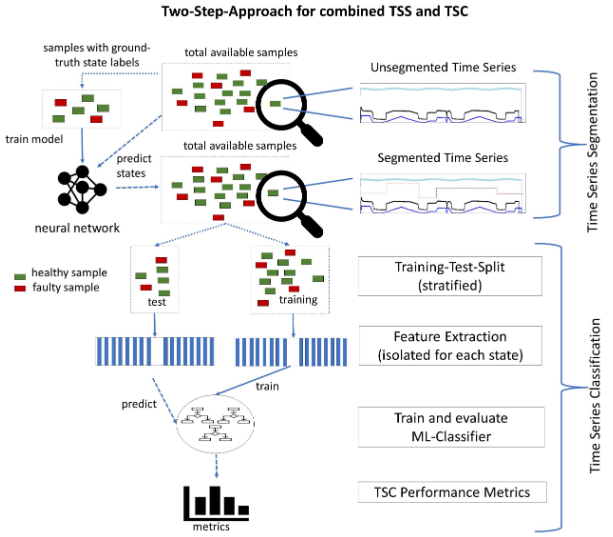

Fig. 2: Visualization of 2-step approach for classifying multi-phased time series

C. Hydraulic Pump End-of-Line Dataset

The dataset used in this paper is a subset of the Hydraulic Pump EoLT dataset. The data are based on hydraulic pumps which were tested on a testbench in a factory (see Figure 3). The dataset is accessible on git (https://github.com/boschresearch/Hydraulic-EoL-Testing/) and was already used for prior research [1]. In our experiments, we focus on the total of 202 Generation B pump samples. One time series sample represents the entire EoLT cycle data of a distinct pump with a unique identifier. All pump samples are multivariate time series and share the identical pump type, size, and version. A sample includes nine sensor channels whose values were normalized (z-score standardized) for confidentiality reasons. Furthermore, the name of the sensors were anonymized and therefore just labeled as 'Sensor 1' to 'Sensor 9'. The lengths of the samples vary, ranging from around 280 to 410 seconds. The timestamp frequency is 100Hz for each sensor and sample. In total, there are 136 healthy and 66 faulty pump samples. The 66 faulty samples can further be distinguished into 11 different fault types, each of them showing a unique and different error signal in the EoLT cycle data (e.g. mean-shift, higher variance, point-wise anomaly). The data were generated in a way that six samples for each fault type exist. There is a class label assigned to each time series sample to indicate the pump health (0 for healthy, 1-11 for the different faulty types). In addition to the class label, each sample contains groundtruth state labels for all time stamps indicating the different testing phases of the cycle. The state labels serve as auxiliary semantic information to help extracting local discriminative information from the time series. In total there are 44 different integer-encoded state labels. For some fault types, the discriminative information is only found within one test step, therefore accurate segmentation is expected to be necessary for high classification accuracy. A description of the different test steps from an engineering point-of-view is not done here for confidentiality reasons. Is should be noted that in the dataset enough fault data are existing, enabling to tackle the fault detection problem via a supervised TSC approach.

Fig. 3: Hydraulic testbench (Source: Bosch Rexroth AG)

D. Experimental Setup and Used Model Architectures

Our experimental design follows the approach presented in Section III-B. The main goal is the comparison of six different feature extraction mode:

1) Using groundtruth state labels to extract features for TSC
2) Using state labels predicted by deep learning to extract features for TSC (three state label predictions with varying accuracies)
3) Two baselines not making use of state-labels (sliding window and global feature extraction)

Feature extraction based on the ground truth state labels serves as a sort of upper baseline for our approach, representing the (unrealistic) case where the supervised TSS would reach 100% accuracy. The ground truth state labeling was performed manually by an expert. The core of our research is, however, the combination of supervised TSS and TSC and the question how the segmentation accuracy influences the classification accuracy. In the following, details about the experimental design and chosen model architectures are provided.

Time Series Segmentation

The goal of supervised TSS (also called segment labeling) is to predict the state label of each timestamp within a time series by a trained model. For multivariate time series, previous works demonstrated that deep learning models show excellent results [17]. In our study, we use a neural network called PrecTime, which is a sequence-to-sequence network based on an LSTM-CNN-architecture [1]. The advantage of sequence-to-sequence TSS architectures is their end-to-end nature, meaning that they take an entire time

series as input and provide state label predictions for all time stamps simultaneously in a single model run. PrecTime consists of three modules (CNN-based feature extraction, LSTM-based context detection, CNN-based prediction refinement), and combines the advantages of sliding window approaches (prediction stability) and dense labeling methods (high labeling granularity)[1]. Figure 4 depicts the network backbone of PrecTime; details about its architecture and hyperparameters can be found in [1]. PrecTime has already been successfully used for TSS the Hydraulic Pump EoLT dataset, for which it outperformed other networks in terms of prediction accuracy [1]. In our experimental design, 20 samples are randomly chosen and assigned to the training set. The model is trained on the groundtruth state labels of the training data and then used to predict state labels for all the remaining 182 samples. Usually, with progressing model training, the validation accuracy is steadily improving, reaching a peak after around 100 epochs. We define three scenarios with the goal to achieve three state label predictions of different accuracy: One scenario for which the training is stopped after 100 epochs (max segmentation accuracy), one for which it is stopped after 50 epochs (high segmentation accuracy), and one for which it is stopped after 30 epochs (medium segmentation accuracy). Therefore, we can assess how the state label accuracy of the TSS step affects the TSC accuracy in the quality prediction step.

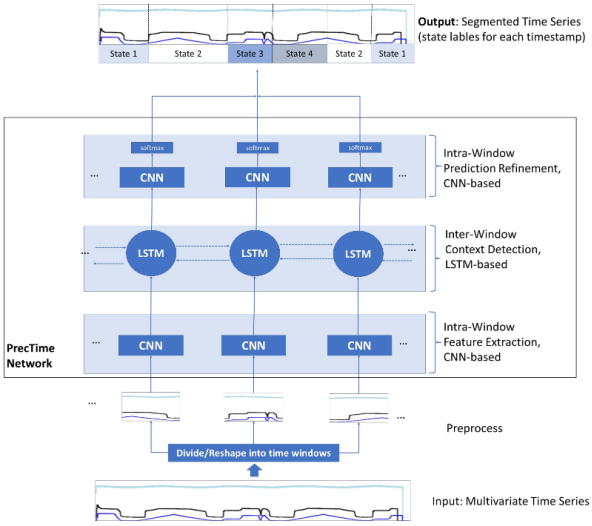

Fig. 4: Architectural Backbone of PrecTime

Feature Extraction

Feature Extraction is kept simple in this paper: For each state and sensor, the two main descriptive statistics, i.e., the mean and the standard deviation, are calculated in isolation and added to the feature pool. An inaccurate segmentation would, in turn, lead to the extracted statistics not accurately representing the actual measured test step results of a cycle.

Time Series Classification

For each feature extraction mode, we trained three different ML classifiers and distinguished two label spaces, namely binary and multiclass. The three selected classifiers were Random Forest (RF), Gradient Boosting (GB), and Support Vector Machine (SVM). While both RF and GB are decision-tree based ensemble methods using a majority vote principle, SVM is a hyperplane-based classifier using a polynomial kernel. Each classifier was once trained in a multiclass label space, where the eleven different fault types got a distinct class label y from 1 to 11, and once in a binary label space, where the class labels y of all fault types were set to 1. Healthy pump samples were labeled as 0 in both cases. The training-test split was done in a 2:1 ratio. The split was accomplished in a stratified way to ensure that the relative occurrence of each faulty type was identical in the training and test set. To account for stochasticity in the models, each model run was repeated 10 times, before the mean and the standard deviation of the measured metrics across the 10 runs were calculated. The goal of the classification is to find a maximal distinction of healthy and faulty pumps. Distinguishing different fault types itself is considered as not being relevant for this task. Therefore, using a multiclass label space is only a mean to help the classifier to better distinguish the healthy and faulty pumps by emphasizing the existence of different fault origins. The resulting multiclass confusion matrix is transformed to a binary one by setting all real and predicted fault labels to 1 (see Figure 5).

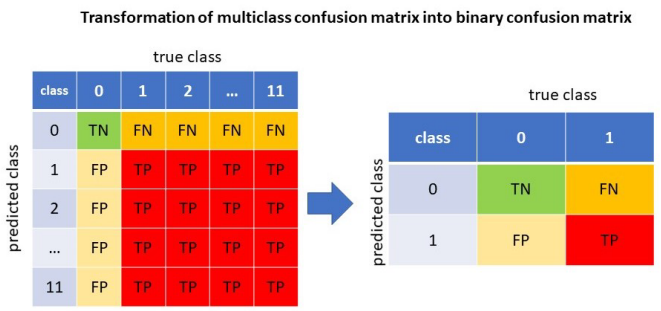

Fig. 5: Confusion matrix transformation in multiclass settings

Metrics

As primary metric to evaluate the performance of a classifier we used the binary F1-score of the fault class y=1 (1):

$$F1_{bin(y=1)} = \frac{2 * TP}{2 * TP + FP + FN} \tag{1}$$

We also calculated the accuracy (2). However, in a setting with high class imbalance the F1-score is more meaningful.

$$Accuracy_{bin} = \frac{TP + TN}{TP + FP + FN + TN} \tag{2}$$

In the multiclass label space, we first calculate the macro F1-score (3) and accuracy (4) over all 12 classes, before transforming the 12-class confusion matrix into a binary one

and calculating the binary F1-score and accuracy. This allows comparing the classifier performances over both label spaces.

$$Accuracy_{mc} = \frac{\text{number of correct predictions}}{\text{number of samples}} \quad (3)$$

$$MacroF1_{mc} = \frac{1}{12}\sum_{c=0}^{11} F1_{bin(y=c)} \quad (4)$$

Baselines

We define two baseline feature extraction modes that are not using any form of state labels. First, we use a sliding window approach with a fixed length of 10 seconds to provide window labels for the time series. The integer encoded labels are identical within a window and incrementally increase with each window (first 10 seconds get label 1, second 10 seconds get label 2, ...). Afterwards, the mean and standard deviation are extracted for each time window like in the other modes. As a second baseline, we do not use any state labels for local feature extraction, but instead perform global feature extraction for each sample via the *TsFresh* python package. *TsFresh* extracts a high-number of features from the temporal and frequency domain of the whole time series, giving the classifier the option to choose the most relevant one.

Implementation Details

The experiments were run locally on an HP Zbook with a 4 core Intel Core i7-10510U processor, 15 GB RAM and 512 GB disk storage based on a Windows 10 operating system. The implementation was based on the Python language, mainly using *Skicit-learn*, *Tensorflow*, *Keras*, and *TsFresh* libraries. The detailed model configuration can be found in Table I.

Model	Category	Hyperparameters
PrecTime	TSS	batch size =24, loss function = cross entropy, optimizer = Adam with steprate 0.01, epochs = [30, 50, 100]
Random Forest	TSC	estimators: 200, max depth = 200, max number features = 200
Gradient Boosting	TSC	n_estimators=200, learning rate=0.1, max depth=1
SVM	TSC	kernel='poly', degree=7, coef0=8

TABLE I: Implementation Details about the used ML-Models

IV. Results and Discussion

The metrics measured for the different feature extraction modes and models can be found in Table II. We focus on comparing the highest measured binary F1-score of any model for each feature extraction mode, as this score is the most relevant metric in EoLT-based fault detection settings. The best binary F1-score means over 10 runs (plus the associated classifier and label space) for the different feature extraction modes are displayed in Figure 6. For multiclass label space models, the binary F1-score of the transformed confusion matrix is used for the comparison. In terms of supervised TSS, the network trained for 100 epochs achieved a state label accuracy of 94.9%, the 50 epoch network 92.1%, and the 30 epoch network 82.2%. Therefore, the respective modes were named 95-seg, 92-seg, and 82-seg in the following to emphasize the approximate segmentation accuracy. As some of the used classifiers contain stochasticity, we added the measured standard deviation in each bar as red lines. Stochasticity was especially present in the Random Forest Classifier (see Table II). When looking at Figure 6, we were able to distinguish two main findings:

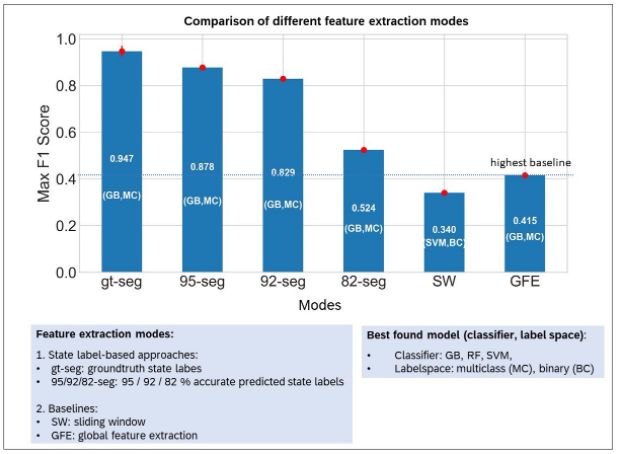

Fig. 6: Maximum achieved F1-Score means (incl. used classifier and label space) for the different feature extraction modes including standard deviation error bars in red

Finding 1: State-based label approaches outperform baseline approaches

All feature extraction modes using state labels show better results than the baseline approaches. Global feature extraction is not sufficient to capture the small discriminative information of the majority of pump faults. A sliding window approach does not work, because the temporal occurrence of the states is not identical over the different samples. The state-label based approaches are successful in extracting discriminative local features that the classifier can use to build a comparably high performing model. Thereby, even faulty pumps just showing an anomaly in one state (test step) within the whole cycle can be discovered. In most cases, the best performing classification approach was to use the Gradient Boosting classifier with a multiclass label space.

Finding 2: A higher accuracy in state labeling tends to increase the classification F1-score

When only focusing on state label-based approaches, we can see how the F1-scores increase with growing segmentation accuracy. While the groundtruth labels (manual labeling) can be seen as 100% accurate, the achieved segmentation accuracies by training and using the PrecTime network were 94.9% (100 epochs), 92.1% (50 epochs), and 82.2% (30 epochs). The results provide an indication that segmentation accuracy positively correlates with the classification performance. The groundtruth label-based feature extraction mode led to the best F1-score of 0.947, setting an upper baseline for the predicted state label approaches. While the two feature extraction modes with segmentation accuracies over 90%, led to satisfying (only

Setting		Binary Labels		Multiclass Labels			
Classifier	Mode	F1	Acc	MacroF1(mc)	Acc(mc)	F1(bin)	Acc(bin)
RF	gt-seg	0.233 (+/-0.08)	0.687 (+/-0.02)	0.489 (+/-0.04)	0.829 (+/-0.01)	0.632 (+/-0.04)	0.824 (+/-0.01)
	95-seg	0.249 (+/-0.08)	0.716 (+/-0.01)	0.262 (+/-0.04)	0.756 (+/-0.01)	0.306 (+/-0.05)	0.757 (+/-0.01)
	92-seg	0.380 (+/-0.05)	0.733 (+/-0.01)	0.270 (+/-0.04)	0.748 (+/-0.01)	0.309 (+/-0.05)	0.748 (+/-0.01)
	82-seg	0.240 (+/-0.07)	0.697 (+/-0.02)	0.187 (+/-0.03)	0.735 (+/-0.01)	0.195 (+/- 0.05)	0.735 (+/-0.01)
	Sliding Window	0.092 (+/-0.03)	0.656 (+/-0.02)	0.069 (+/-0)	0.703 (+/-0)	0.0 (+/-0)	0.702 (+/-0)
	Global Feature Extr.	0.170 (+/-0.06)	0.665 (+/-0.01)	0.086 (+/-0.01)	0.706 (+/-0)	0.026 (+/- 0.04)	0.706 (+/-0.01)
GB	gt-seg	0.851 (+/-0)	0.905 (+/-0)	0.838 (+/-0.02)	0.940 (+/-0.02)	0.946 (+/- 0.02)	0.967(+/-0.01)
	95-seg	0.524 (+/-0)	0.729 (+/-0)	0.840 (+/-0.03)	0.931 (+/-0.01)	0.878 (+/-0.01)	0.931(+/-0.01)
	92-seg	0.500 (+/-0)	0.757 (+/-0)	0.695 (+/-0)	0878 (+/-0)	0829 (+/-0)	0.905(+/-0)
	82-seg	0.465 (+/-0)	0.689 (+/-0)	0.524 (+/-0)	0.667 (+/-0.01)	0.524 (+/-0.01)	0.681(+/-0.01)
	Sliding Window	0.333 (+/-0)	0.621 (+/-0)	0.122 (+/-0)	0.636 (+/-0.01)	0.205 (+/-0.02)	0.666 (+/-0)
	Global Feature Extr.	0.296 (+/-0)	0.614 (+/-0)	0.277 (+/-0.02)	0.747 (+/-0.01)	0.415 (+/-0.01)	0.72 (+/-0.01)
SVM	gt-seg	0.407 (+/-0)	0.567 (+/-0)	0.096 (+/-0)	0.554 (+/-0)	0.381 (+/-0)	0.648 (+/-0)
	95-seg	0.339 (+/-0)	0.527 (+/-0)	0.085 (+/-0)	0.513 (+/-0)	0.195 (+/-0)	0.554 (+/-0)
	92-seg	0.333 (+/-0)	0.459 (+/-0)	0.118 (+/-0)	0.567 (+/- 0)	0.307 (+/-0)	0.635 (+/-0)
	82-seg	0.296 (+/-0)	0.486 (+/-0)	0.108 (+/-0)	0.608 (+/-0)	0.421 (+/-0)	0.703 (+/-0)
	Sliding Window	0.340 (+/-0)	0.581 (+/-0)	0.098 (+/-0)	0.621 (+/-0)	0.242 (+/-0)	0.662 (+/-0)
	Global Feature Extr.	0.292 (+/-0)	0.608 (+/-0)	0.056 (+/-0)	0.473 (+/-0)	0.266 (+/-0)	0.554 (+/-0)

TABLE II: Experimental Results (means and standard deviations over 10 runs) [mc=multiclass, bin=binary]

slightly differing) F1-scores of 0.878 and 0.829, the classifier performance in the 82% segmentation accuracy setting considerably suffered by the inaccuracies found in the state labels. Therefore, accurate state labels can be seen as prerequisite for local-feature extraction-based methods to be successful in time series classification.

It should be noted that, when looking at the total results in Table II, the three classifiers and two label spaces we tested have shown a different behaviour. While GB was successful in selecting the discriminative locally-extracted features, RF and SVM struggled to find a clear distinction between healthy and faulty pumps. The general observation that accurate state labels lead to better classification performance compared to baseline modes can be seen for both GB and RF across both labels spaces. For the SVM however, this difference almost disappears, bringing all settings to a similar (comparably) low level below 0.4. This can be explained by the different model nature of the decision-tree based models (RF, GB) and the hyperplane-based SVM. The main characteristic of EoLT fault detection use cases is the concept of sequentially checking different testing points where one deviation at one point is enough to classify the whole pump as faulty. This fits better to rule-based decision-tree based models compared to SVM. Therefore, even an improved feature space does not lead to a performance boost for the unfit SVM classifier. The most likely explanation for GB outperforming RF is that GB is suited better to deal with the class imbalance found in the data.

Altogether, we were able to show the benefit and necessity of TSS in a real-world industrial use case. Even in settings in which the state labels were created by a deep learning prediction, the baseline approaches could be clearly outperformed. This is highly relevant, because generating groundtruth labels for all samples might not be possible or consume too many resources in many use cases. While we focused on a specific pump testing use case, we assume that our approach and findings are domain-independent. As multi-phased process time series are especially prevalent in discrete manufacturing, our work might provide interesting findings for similar industrial problem settings.

V. Summary and Outlook

In this study, we worked on an real-world industrial EoLT use case that serves as example of a challenging multi-phased classification problem. We proposed an ML-based approach that combines using supervised TSS and TSC. The availability of two types of labels allowed us to analyse the interdependence of supervised TSS and TSC. We made a distinction between the class labels defining if a pump sample is healthy or faulty and timestamp-specific state Labels indicating the testing phase of a cycle. State labels are meant to serve as auxiliary semantic information helping to effectively extract local features for the binary classification problem (fault detection). A sequence-to-sequence neural network was trained three times and used for the TSS step, leading to three state label predictions of varying accuracy. Based on the predicted state label, various test step specific features were extracted isolatedly and fed into a ML-model trying to discriminate the samples into healthy and faulty ones. Additionally, two baselines (sliding window based feature extraction, global feature extraction) and a setting using the groundtruth state labels were implemented to provide upper and lower baselines to our results. We could show that approaches using state label-based extracted features clearly outperform the baseline approaches when using Gradient Boosting as classifier. Additionally, we observed that a high accuracy of the segmentation results increased the F1-scores of the classification steps. However, it also has to be mentioned that the results varied depending on the used classifier and label space. In our case, decision-tree based Gradient Boosting performed best from the chosen classifiers.