Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

SLAS Technology 27 (2022) 319–326

Contents lists available at ScienceDirect

SLAS Technology

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

Full Length Article

Data driven health monitoring of Peltier modules using machine-learning-methods

B.S. Paul Figueroa Cotorogea[a,*], Giuseppe Marino[a], Prof. Dr. Stefanie Vogl[b]

[a] INHECO Industrial Heating & Cooling GmbH, Fraunhoferstrasse 11, 82152, Martinsried, Germany
[b] Faculty for Informatics and Mathematics, University of Applied Sciences, Lothstr. 34, 80335, Munich, Germany

ARTICLE INFO

Keywords:
Supervised machine learning
Predictive maintenance
Condition monitoring
Polymerase-chain-reaction runs
Peltier modules

ABSTRACT

Thermal cyclers are used to perform polymerase chain reaction runs (PCR runs) and Peltier modules are the key components in these instruments. The demand for thermal cyclers has strongly increased during the COVID-19 pandemic due to the fact that they are important tools used in the research, identification, and diagnosis of the virus. Even though Peltier modules are quite durable, their failure poses a serious threat to the integrity of the instrument, which can lead to plant shutdowns and sample loss. Therefore, it is highly desirable to be able to predict the state of health of Peltier modules and thus reduce downtime. In this paper methods from three sub-categories of supervised machine learning, namely classical methods, ensemble methods and convolutional neural networks, were compared with respect to their ability to detect the state of health of Peltier modules integrated in thermal cyclers. Device-specific data from on-deck thermal cyclers (ODTC®) supplied by INHECO Industrial Heating & Cooling GmbH (Fig 1), Martinsried, Germany were used as a database for training the models. The purpose of this study was to investigate methods for data-driven condition monitoring with the aim of integrating predictive analytics into future product platforms. The results show that information about the state of health can be extracted from operational data - most importantly current readings - and that convolutional neural networks were the best at producing a generalized model for fault classification.

Introduction

The ongoing coronavirus pandemic has led to a high demand for laboratory equipment needed for diagnosing and studying the virus. Even though business is booming for companies in the life sciences and laboratory automation industry, and laboratory devices are being produced in high numbers, their reliability remains the highest priority. Any small deviation from the specified functionality could adversely affect sample quality and lead to inconsistencies and subsequent rejection of test results. In the case of unforeseen technical defects, the whole workstation and test process would need to be put on hold. To prevent financial losses caused by sample loss and downtime, predictive analytics can be used to monitor the device's health and warn the operator in case a high failure probability is detected. Predictive maintenance (PdM) is one cost-effective option for that task, which commonly makes use of machine-learning techniques for analyzing the machinery's overall health or the life expectancy of certain machine parts. Carvalho et al. give a systematic review on classical machine-learning methods for PdM and discuss in detail the advantages and drawbacks of random forests, artificial neural networks, support vector machines or k-Means Classification [1]. They point out that the success of PdM methods for specific applications is dependent on the choice of the appropriate machine-learning technique. Recently, methods from the field of deep learning such as deep convolutional neural networks have also been considered for PdM applications as they are supposed to reduce the amount of feature engineering and preprocessing. However, these methods can only be trained effectively whenever a substantial amount of training data is available [2]. Many PdM applications are specifically condition-based maintenance (CBM) systems. CBM was reviewed e.g., by Jardine et al. [3] and described by Ahmad and Kamaruddin as a system that "recommends maintenance actions (decisions) based on the information collected through condition monitoring process" [4]. In that context, condition monitoring can be understood as a continuous assessment of a machine's health by monitoring various parameters, such as vibration, temperature, lubricating oil, contaminants, and noise levels [4]. In the case of rotary machines, condition monitoring using machine-learning models trained on features extracted from vibration data has become a popular technique (e.g. Ahmad et al. [5]).

Pinheiro et al. developed an automatic predictive maintenance model for the diagnosis of incipient failures in rotary machines, based on support vector machines (SVM) [6]. Dhanraj and Sugumaran employed a machine-learning approach using vibration signals through statistical

* Corresponding author.
E-mail address: paul.figueroa.cotorogea@gmail.com (B.S.P.F. Cotorogea).

https://doi.org/10.1016/j.slast.2022.07.002
Received 14 December 2021; Received in revised form 21 June 2022; Accepted 24 July 2022

features for condition monitoring of wind turbine blades [7]. The study involved the classical three steps: feature extraction, feature selection and classification by machine learning. They investigated algorithms such as K-star (KS), locally weighted learning (LWL), nearest neighbor (NN), k-nearest neighbor (kNN), instance-based K-nearest neighbor using log and Gaussian weight kernels (IBKLG) as well as lazy Bayesian rules classifiers (LBRC). The results were compared with respect to the classification accuracy and the computational time of the classifier. In a comprehensive overview of recent research trends and development of lubrication condition monitoring (LCM)-based approaches for maintenance decision support, Wakiru et al. discussed approaches applicable for analyzing data derived from lubricant analysis [8]. The study included several machine-learning algorithms: traditional ones like Logistic Regression, Decision Trees and Support Vector Machines as well as more recent approaches such as Genetic Neural Networks, Fuzzy Neural Networks and General Regression Neural Networks. With increasing amounts of data, collected during manufacturing or production processes, the use of machine-learning methods for PdM gets more and more attractive. This is highlighted by a significantly increasing number of publications in the field of machine-learning-based PdM over the last years (e.g. Cioffi et al. [9], Zonta et al. [10], Sajid et al. [11], Çınar et al. [12] or Leukel et al. [13]). However, for real world applications in industry it is often hard to decide how the results from scientific benchmarks can be transferred to existing technical systems.

The aim of this work was therefore to implement a PdM system in the evolution of laboratory equipment produced by the company INHECO GmbH. We explored the possibilities of incorporating machine-learning algorithms into the design of future generation products, with the ultimate goal of developing a PdM system that facilitates servicing of laboratory instruments and enhances customer experience. We investigated the ability of condition monitoring techniques to make accurate predictions of the state of health of a subsystem of one selected product of INHECO GmbH, namely Peltier modules integrated in On-Deck Thermal Cyclers (ODTC) [INHECO Industrial Heating & Cooling GmbH, Martinsried, Germany]. Various machine-learning (ML) algorithms were set up to classify and predict three different health states of said Peltier modules. For this specific use case, the conventional inspection of the Peltier modules is done by means of external measurement instruments, which requires the casing to be removed. This procedure can be somewhat cumbersome, considering that the ODTC in many cases is installed on a deck together with several other instruments. With data-driven monitoring, these circumstances can be avoided. To study the feasibility of CBM for this use case, the first thing was to investigate whether operational data generated by the ODTCs, and subsequently derived features contain enough information for ML models to reliably differentiate between three predefined health states of the Peltier modules. The insights won from this investigation could then be used to develop a scaled-up PdM system, which covers other ODTC components and even different products.

Materials and methods

Technical background

The On-Deck Thermal Cyclers are primarily used for Polymerase Chain Reaction (PCR), which is a method to replicate DNA sequences. The PCR procedure follows a specific temperature profile, where one cycle consists of steps for template denaturation, primer annealing and primer extension [14]. Such a PCR Profile can be seen in Fig 2, where the three steps appear in the same order as mentioned before on the uppermost graph of the figure (healthy state represented by the blue line). Depending on the case, the exact temperature and length of each step can vary. The over- and undershoots before each temperature plateau in the top of Fig 2, roughly at times 7, 43 and 65 seconds, have been implemented by INHECO GmbH, so that the probe samples reach the target temperature as rapidly as possible.

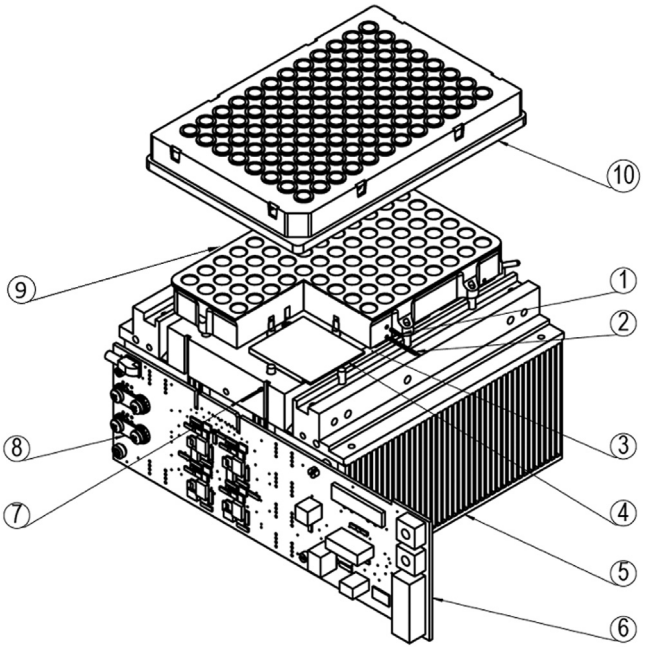

Fig. 1. Thermal unit of the ODTC without casing. 1 & 2: VCM sensor 1 & 2, 3 & 7: Heatsink sensor 1 & 2, 4: Peltier element, 5: Heatsink, 6 Block-PCB (Printed Circuit Board), 8: Power bridge (HBridge); Contact points for ACR measurement, 9: VCM, 10: Disposable which holds the samples.

Fig 1 shows the thermal unit of the ODTC, and the numbers named in the further course of this chapter refer to the labels in said figure. For heating and cooling the samples, an array of six Peltier modules (Nr. 4) is used, which is located between the Vapor Chamber Mount (VCM®, Nr.9) and the heatsink (Nr. 5). The VCM is the thermal interface that holds the disposable labware (Nr. 10) with the samples and has been developed by INHECO GmbH to achieve a quick and uniform heat transfer. The VCM can be understood as a three-dimensional heat pipe, which works with extremely fast internal heat transfer (at the speed of sound) in every direction. The Peltier modules form the core of the thermal system of the ODTC. These components are durable, with a lifespan of 500.000 cycles guaranteed by the manufacturer (II-VI Inc. [15]) but represent one of the few wearing parts of the ODTC. An analysis completed by the INHECO GmbH R&D department, shows the internal resistance of Peltier modules to be the main indicator of their state of health (data not shown). Measuring the internal resistance of Peltier modules by a four-point alternating current resistance measurement (four-point ACR measurement) throughout their lifespans, showed a general increase of the ACR readings with time. To perform an ACR measurement, the casing of the ODTC must be removed to expose the printed circuit board (PCB, Nr. 6), so that the measurement instrument can be connected to the power bridge (Nr. 8). Further investigations revealed that if the ACR value of new Peltier modules exceeded a certain threshold, referred to as *Rth*, within a certain number of thermal cycles, the modules will break shortly after (data not shown). During operation of the ODTC, several parameters are continuously recorded by the control unit of the ODTC. From these parameters a specific subset has been selected which was considered to be either directly or indirectly related to the thermal unit of the ODTC. These variables were temperature measurements provided by five sensors, two of which were fitted inside the VCM (Nr. 1 & 2), two inside the heatsink (Nr. 3 & 7) and one, for recording the ambient temperature, on the inner side of the casing (not shown in Fig 1). Furthermore, current and voltage readings from the Peltier modules and temperature control parameters were taken into account. This collection of features is referred to as raw data in the next chapters.

Data acquisition and labeling

The ML models, discussed below (see section 3.4), were trained to distinguish between the following three health states:

State 0 (Label 0): Healthy state, all Peltier modules are fully operational.

State 1 (Label 1): Degrading state, one or more Peltier modules are beyond a certain functionality specification.

State 2 (Label 2): Defective state, one or more Peltier modules are broken. Heating and cooling function is compromised.

For the training of neural networks, the labels were one-hot-encoded. This coding process transforms integer labels (starting at zero) into a vector with the length of the number of classes used in the classification. The vector contains only zeros except at the position which corresponds with the integer value of the label. These types of labels are called categorical labels:

Label 0: [1,0,0]

Label 1: [0,1,0]

Label 2: [0,0,1]

This trinary classification can be seen as a simplification of the task of constructing a continuous health indicator which is necessary to make predictions on the remaining lifetime of a machine component. A reduced model of the temporal development of the component's health over its lifetime can be seen on Fig 3.

States 0 through 2 would be snapshots in time, first at the start of operation, then after degradation has started and lastly at its end of life (EOL). From previous experience it is known that even under extreme conditions applied during an accelerated life testing, the devices can run for several months before a Peltier module would reach EOL. Data from three of the cyclers, which were subjected to this test, were retrieved and the cyclers were marked with the serial numbers (SN) 1-3. ACR measurements which were taken at regular intervals during the endurance test, were used as reference values for labeling. To supplement data from SN 1-3, three different devices (SN 4-6) were used to generate data in all three states. The ODTCs started data generation in State 0, but due to time restraints, States 1 and 2 were simulated using the physical property that the internal resistance of Peltier modules increases as they start to age. State 1 was simulated by putting a resistor with the value of *Rth* in series with one of the Peltier modules. State 2, which refers to an electrical interruption within one of the Peltier modules, could be easily reproduced by disconnecting one of the modules from the power supply. This electrical interruption translates to an infinitely large ACR value. In short, the internal resistance values of SN 4-6 were artificially induced to replicate States 1 and 2.

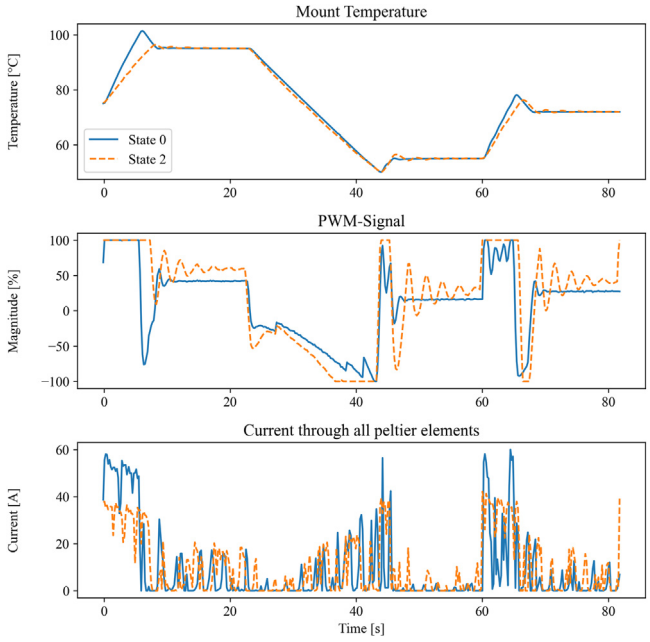

Fig. 2. Comparison of three signals recorded from a device in states 0 and 2. The Plots show the signal for one PCR cycle. Top: Temperature at the VCM. Middle: PWM signal of the temperature control. Bottom: electric current through all Peltier modules.

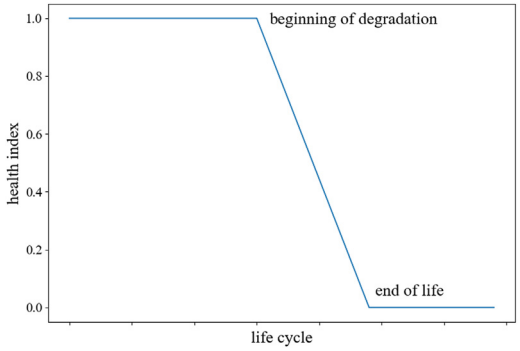

Fig. 3. Reduced model of the temporal development of the health index of Peltier Modules. State 0 at the upper plateau, State 1 at the slope from the beginning of degradation until end of life and State 2 at bottom plateau.

Feature engineering

Most of the work was put into extracting features from the raw data, which show a dependency to the actual state of health of the device. This was a process needed solely for the classical ML methods and not necessary for convolutional neural networks (CNNs) since the architecture of the CNN would be designed to take the raw data as input and extract information about the state of health during the training process by itself. The approach of selecting features was based on physical properties as well as the experience and intuition of experts.

When one of the Peltier modules in an ODTC breaks down, it can be easily recognized by looking at the temperature signature of a PCR cycle (Fig 2, top graph). When a device enters State 2 (orange, dashed line) due to a Peltier module breaking, the remaining modules struggle to keep up the heating rate. This can be seen in the first few seconds of the graph. However, a flattening of the heating slope may indicate other reasons, such as a high filling level of the mount combined with extreme ambient temperatures. To clear up this discrepancy, the ambient temperature, recorded by a sensor on the side of the casing (not shown in Fig 1), was used as a feature.

Another challenge was to find indications within the data, that the modules are in the degrading state without yet being broken (State 1). For that, the temperature readings weren't enough since the temperature control was able to compensate for a considerable declination in the performance of one or more modules. Thus, temperature control parameters, like the PWM signal (middle graph in Fig 2), were taken into consideration. Lastly and even though it was known that they were related to the PWM signal, voltage, and current readings (bottom graph in Fig 2) of the Peltier Modules were also regarded.

Feature engineering was done by analyzing all the signals considered and comparing those recorded from a cycler in State 0 to another one in State 2 (see Fig 2).

The final features were formed by extracting local minima and maxima (green dots, Fig 4) from selected regions and averaging them over a period of four PCR cycles (red, dashed line, Fig 4). The regions where local minima and maxima were extracted from, were chosen based on where the largest deviation was visible when comparing the signals of a device in a healthy state and one in a defective state (State 0 vs State 2 in Fig 2). The process of extracting features from locations within the data where the largest changes occur throughout the lifetime of a de-

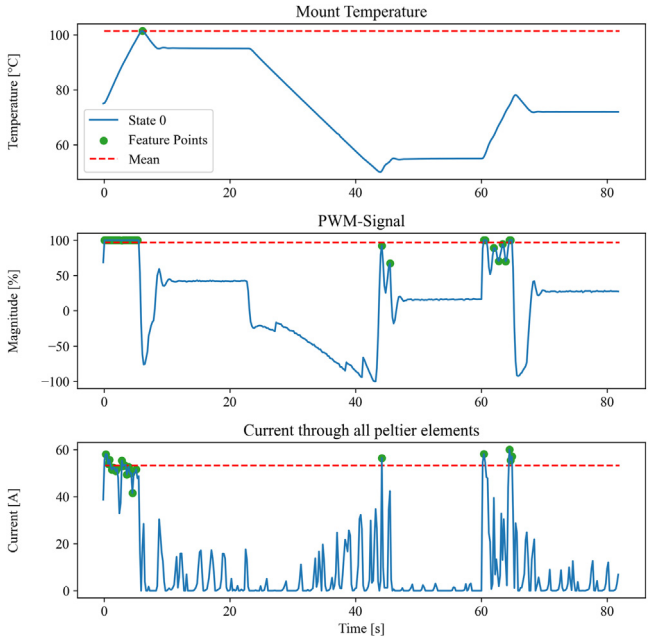

Fig. 4. Example of the feature extraction process. Features were formed by extracting local minima and maxima (green dots) from selected regions, and averaging over four PCR cycles (red, dashed line). The Figure shows three signals for the time period of one PCR cycle. Top: Temperature at the VCM. Middle: PWM signal of the temperature control. Bottom: electric current through all Peltier elements.

vice, was inspired by the work of Mazaev et al. [16]. In a similar way, nine more features were extracted from the raw data to form one feature vector, or sample. This process was repeated for every four PCR cycles in the raw data, so that 529 samples were created from the generated data and 381 from the collected data from past examinations, making a total of 910 samples in the dataset. It was ensured that the number of samples was balanced for the three states.

The data preparation for the CNNs was quite different and much less laborious. All the signals for each cycle in the raw data were concatenated to form a 2D matrix with the dimensions 500×10 (10 equals the number of signals and 500 the number of measurements in each signal). The matrix was then scaled between 0 and 1 along the time dimension using the maximum and minimum values of all signals in the dataset as upper and lower limits for the scaling. The result was a long gray scale picture, as can be seen in Fig 5 (displayed in viridis color-scale for greater clarity).

For the CNN classifier one PCR cycle was used to form a sample. Therefore, the length of the dataset was quadrupled compared to the dataset used for the classical ML methods. The procedure resulted in 3656 samples in total.

Model selection and assessment

Listed below are the ML methods that were investigated in this work. The comparison includes methods from classical machine learning, ensemble methods (for the sake of simplification also referred to as classical methods) and neural networks. For all implementations, the open-sourced Python libraries Scikit-Learn and Keras (with TensorFlow as backend) were used. Table 1 contains the links to the official documentation for each method.

KNN: K Nearest Neighbor Classifier
LSVC: Linear Support Vector Classifier
SVC: Support Vector Classifier - Kernel Method
DT: Decision Trees Classifier
RF: Random Forest Classifier

Table 1
Links to the official documentation of machine-learning methods

Scikit-Learn documentation	https://scikit-learn.org/stable/user_guide.html
Grid Search with Cross Validation	https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
Cross Validation	https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation
K-NearestNeighbors Classifier	https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
Linear Support Vector Classifier	https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
C-Support Vector Classifier	https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html?highlight=svc#sklearn.svm.SVC
Decision Tree Classifier	https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decisiontreeclassifier#sklearn.tree.DecisionTreeClassifier
Random Forest Classifier	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random%20forest#sklearn.ensemble.RandomForestClassifier
Gradient Boosting Classifier	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html?highlight=gradient%20boosting#sklearn.ensemble.GradientBoostingClassifier
Convolutional Neural Network	https://www.tensorflow.org/tutorials/images/cnn

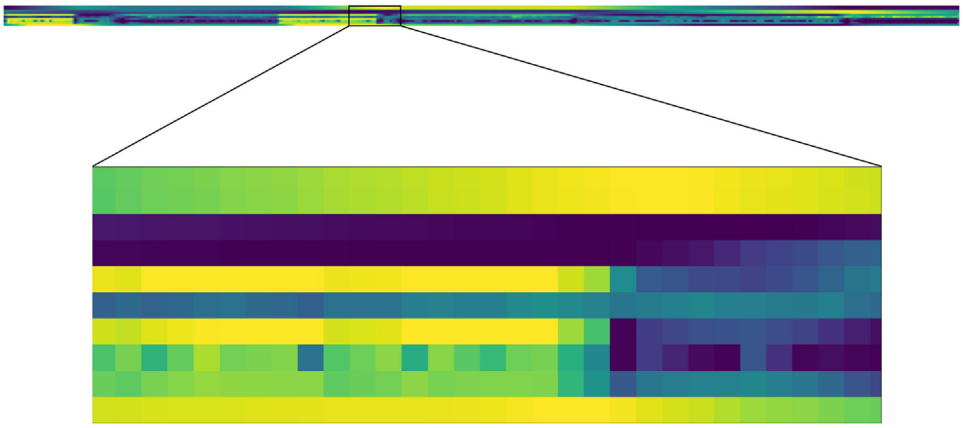

Fig. 5. Top: single sample for the CNN. Bottom: zoomed-in section of the sample. X-axis: Time, Y-axis: 10 signals containing temperature, current and voltage readings, as well as temperature control parameters. Original dimensions: [500 × 10]. Image rotated for better visualization.

GDB: Gradient Boosting Classifier

CNN: Convolutional Neural Network

As the models need to differentiate between three health states, multi-label classification accuracy was used as a metric for estimating the model's performance, which is defined as:

$$accuracy(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N-1} 1\left(\hat{y}_i = y_i\right) \quad (1)$$

Accuracy(y,yˆ(hat)) = 1/n_(samples)*SUMˆ(n_samples-1)_(i=0)(1(yˆ(hat)_i=y_i)) with $1(\hat{y}_i = y_i)$ being the indicator function (returns 1 if the predicted label $\hat{y}_i$ equals the true label y_i, and 0 otherwise) and N the number of samples [17]. In the case of the classical methods, the adjustment of the hyperparameters was done using the "GridSearchCV" function from the Scikit-Learn library [18]. This function carries out an exhaustive search of all the combinations of hyperparameters within a given range and calculates a cross-validation score for each combination. The training set was given as an input to the GridSearchCV function and a cross validation with three folds was specified. The function yields the hyperparameters which achieved the highest cross validation score. With these optimized hyperparameters, the models were again trained using the entire training set and were evaluated afterwards using the test set. The accuracies which resulted from the final testing were used as performance measure for the models.

The optimization of the hyperparameters of the neural network was done using a self-programmed randomized grid search. Similar to the grid-search function, cross-validation scores for every combination of parameters within a given range were calculated, with the difference that only a random subset of the parameter grid was evaluated because of the long training times of the neural networks. With the results of the grid search, small manual adjustments were made to further improve the network's performance. In addition to the multi-label accuracy, the categorical cross-entropy loss was used to evaluate the performance of the CNN. This loss function gives a measure of the confidence that a correct prediction was made and is defined as:

$$loss(y, \hat{y}) = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} y_j log\left(\hat{y}_j\right) \quad (2)$$

loss(y,yˆ(hat)) = 1/N*SUMˆ(N-1)_(i=0)(SUMˆ(M-1)_(j=0)(y_j*log(yˆ(hat)_j))) with N being the number of samples, M the number of classes, y_j the true label and $\hat{y}_j$ the predicted label [19]. Since the CNN takes categorical labels (see section 3.2) as inputs, the predictions made are also three-dimensional and represent the probability of each class being the correct one. Due to the "softmax" output layer, the prediction vector sums up to one.

The cyclers SN 1-3 and SN 4-6 were operated in different modes, meaning slightly different cycle periods caused by varied plateau times. To test the model's behavior on different training data, two partitions of the dataset were made. In the first one (Split 1), the whole dataset including the generated samples, as well as the samples from the historical data, was randomly shuffled and split with a ratio of 58/42 leaving 58% of the data for hyperparameter optimization and training and 42% for testing. For the second partition (Split 2) only the generated samples (from serial numbers 1, 2 and 3) were used to find the best hyperparameters and train the models, while exclusively samples from the historical data (from serial numbers 4, 5 and 6) were used for testing. The relation of the number of the generated samples and the historical samples gives the same ratio of 58/42. This splitting ratio was applied overall to keep a methodological consistency between Split 1 and Split 2. The purpose of Split 1 was to analyze whether the raw data and the extracted features contained enough information for the models to successfully separate the three classes, whereas the second Split served to test the model's ability to generalize between different operation modes.

The architecture of the CNN remained unchanged for both Splits, as different settings of architecture-specific parameters did not influence the classification results during preliminary test runs. The result was a shallow convolutional neural network consisting of two convolution and max-pooling steps, followed by a dense layer and an output-layer. The first convolution had 16 filters, a kernel size of 2 × 2, a step size of 1 and "same" for padding. The second convolution was identical with the difference of having 32 filters. Both max-pooling layers had a kernel size of 2 × 2, step size of 2 and no padding. The dense layer contained 90 neurons with a "ReLU"-activation and the output-layer had three neurons with a "softmax"-activation. For model training the Adam-optimizer was used.

Results and discussion

Results

The hyperparameters for each model, which were optimized performing a grid search as described in section 3.4., can be seen in Table 2. Hyperparameters not listed in Table 2 are meant to have default values, which can be looked up in the respective documentation (follow links in Table 1). For the sake of reproducibility, the random state of the methods that contain a random process in the fitting of the models (Decision Tree, Random Forest, Gradient Boosting and Support Vector Classifier), was set to 0. Due to that setting the training and testing of those models can be repeated arbitrarily often without changing the results. In contrast, due to the random initialization of the weights of the CNN at the beginning of training [20],[21],[22], the optimization of the error function can converge at different local minima on subsequent trials, yielding different results [23]. The results in Table 3 were achieved by running the training procedure inside a loop which was stopped if the final test accuracy and loss reached a certain threshold value. For Split 2 signs of overfitting could be seen after a few epochs, meaning that the train loss decreased, while the validation loss increased steadily [24].

Table 2
Best hyperparameters for investigated models.

Hyperparameter	Split 1	Split 2
K Nearest Neighbor Classifier		
n_neighbors	1	2
Weights	uniform	uniform
Decision Tree Classifier		
criterion	gini	gini
max_depth	10	None
Random Forest Classifier		
criterion	entropy	gini
max_depth	None	None
max_features	auto	auto
n_estimators	20	10
Gradient Boosting Classifier		
criterion	mse	mae
max_depth	20	18
n_estimators	21	13
loss	deviance	Deviance
Linear Support Vector Classifier		
C	126	2
Support Vector Classifier – Kernel Method		
C	4	5
gamma	1	1
kernel	poly	rbf
Convolutional Neural Network		
batch_size	300	305
epochs	70	14
learning_rate	0.016667	0.0175
beta_1	0.89999	0.79999
beta_2	0.96666	0.99999

Table 3
Test accuracies of all models

Split 1		Split 2	
Method	Test accuracy	Method	Test accuracy
Random Forest Classifier	0.9684	Convolutional Neural Network	0.8136
K-Nearest Neighbors Classifier	0.9620	Random Forest Classifier	0.7585
C-Support Vector Classifier	0.9578	K-Nearest Neighbors Classifier	0.7139
Convolutional Neural Network	0.9513	C-Support Vector Classifier	0.6640
Gradients Boosting Classifier	0.9241	Decision Tree Classifier	0.3937
Decision Tree Classifier	0.9219	Linear Support Vector Classifier	0.3281
Linear Support Vector Classifier	0.8945	Gradients Boosting Classifier	0.3071

To reduce overfitting and ensure a more generalized model, an early stopping call back was implemented. The call back was triggered when the validation accuracy reached a value above 65% and the loss a value below 0.5. For consistency, the early stopping call back was also applied to Split 1, however with the thresholds of 95% for accuracy and 0.2 for loss.

The final models were assessed by estimating their classification accuracy based on the test set of each Split of the dataset. An overview of all test accuracies can be found in Table 3.

Almost all Classifiers which received data of all six cyclers during training (Split 1), managed to place above the 90% accuracy mark after testing. The random forest classifier (RF) achieved the highest performance of 96.84% followed by the k nearest neighbor classifier (KNN) with 96.20% accuracy and the support vector classifier (SVC) in third place performing at 95.78% accuracy. The CNN reached fourth place with an accuracy of 95.13% and a loss of 0.1467. Positioned last were the gradient boosting (GDB), the decision tree (DT) and the linear support vector classifier (LSVC) with 92.41%, 92.19% and 89.45% accuracy, the last one being the only classifier to perform below 90% accuracy.

For the generalization test (Split 2), the accuracies of all models significantly dropped. The CNN could maintain a test accuracy of 81.36%, and thus moved up to first place in ranking. With a new loss of 0.4921, however, the confidence of the predictions significantly deteriorated. The RF moved down to second place with an accuracy of 75.85%, the KNN to third with 71.39% and the SVC to fourth with 66.40% accuracy. Placed last were the DT, LSVC and GDB classifiers, all with accuracies below 40%.

The random forest classifier allows to evaluate the importance of the individual features based on the structure of the decision tree ensemble. This property was exploited to examine which features made the largest contribution to the predictions. It was found that the features with the highest importance were those that were extracted from current readings of the Peltier modules. This was consistent with the findings of the research department that the internal resistance was the most significant indicator for the health state of Peltier modules, because the current can be computed from it. The next most influential features were those derived from the mount temperature measurements. Features deduced from control parameters showed an even lower contribution to the predictions and features from the heat sink and ambient temperature proved as least significant.

Discussion

In this study, all the models investigated were able to separate the three predefined health states (healthy, degrading, or defective) of Peltier modules with an overall accuracy above 89%, provided that data from all the test objects were used for training. The generalization ability of the ML models on operational data not seen before was significantly lower. To analyze misclassifications, the confusion matrices of the CNN and the RF models, each generated from both splits, were examined (Fig 6).

On the left-hand side of Fig 6 the results of the CNN (sub figure a)) and of the RF (sub figure c)) can be compared. In sub figures a) and c) most of the predictions lie in the main diagonal, which indicates a high overall accuracy. In both cases, particularly in a), classification errors for label 1 can be seen. This behavior is amplified in both confusion matrices of the Split 2 (sub figures b) and d)), where the spread of predictions across the upper left corner is more evident. For the ML models from Split 2 which ranked lower for classification accuracy, even more predictions moved from the center of the confusion matrix to the left (not shown here), meaning that more samples with the true label of 1 were recognized as 0. The results suggest that the models had difficulties with distinguishing between State 0 and State 1, or more specifically, they often confused State 1 with State 0. On the other side, State 2 was predicted correctly almost all the times with accuracies above 98% in all the cases shown in Fig 5.

The distribution of the classification errors can be traced back to the definition of the health states. State 1 (Label 1) was simulated in the devices SN 4-6 to imitate a point during its life cycle in which the degradation of the Peltier modules has already begun, by connecting a resistor (*Rth*) to the power supply of one of the modules. An increase in the internal resistance of Peltier modules by the value of *Rth* was proven to mark the beginning of the degradation phase, whereby the boundary is not sharp but rather blurred. Contrarily, the defective state

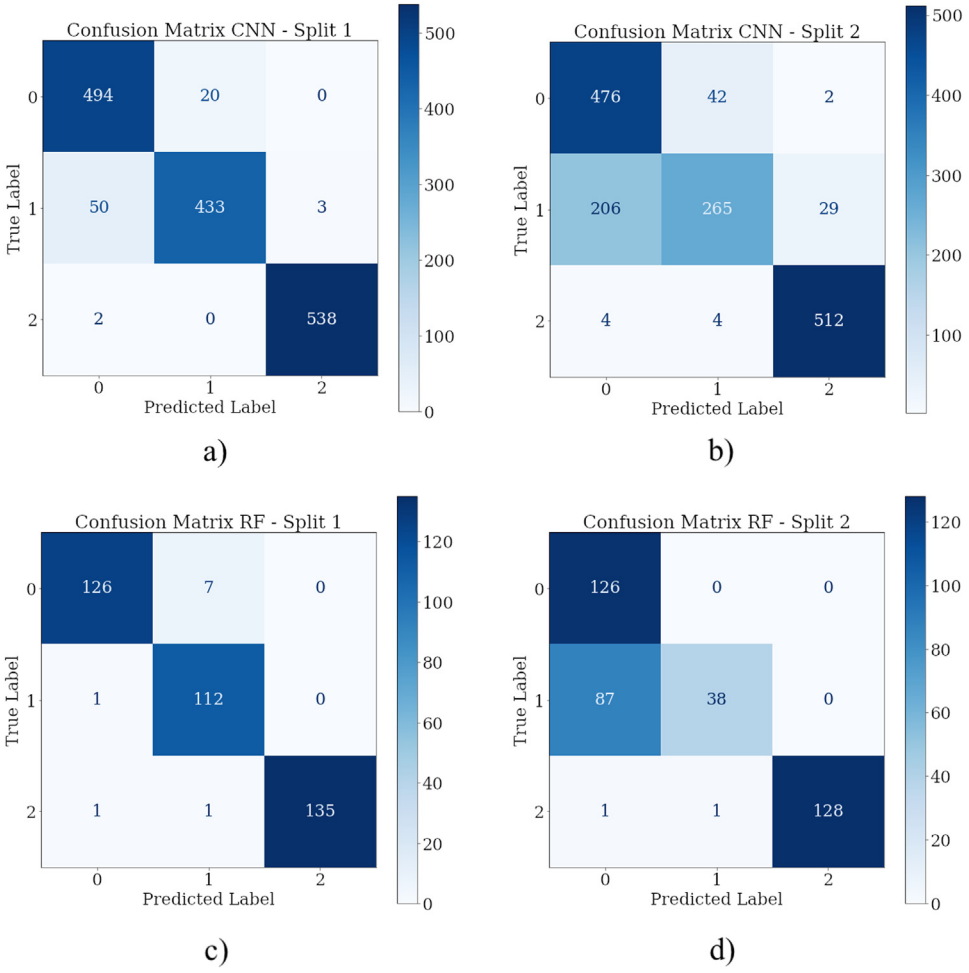

Fig. 6. Confusion matrices. a) CNN Split 1. b) CNN Split 2. c) RF Split 1. d) RF Split 2. X-axis: predicted labels, Y-axis: true labels. Numbers inside cells are the number of samples whose labels were predicted for each true label of the test samples.

can be identified by an infinitely large resistance due to the electrical circuit being interrupted. This consequently results in a clear boundary between State 1 and State 2.

Furthermore, a closer look at the test data revealed another issue, which explains the classification errors. These data were collected from past examinations where the ODTCs were operated under extreme external conditions, which are known to cause a high amount of condensation to accumulate around the Peltier modules. This effect has been proven to lead to a sudden destruction of single Peltier modules contrary to the creeping process of natural aging that is known to go on during correct operation. The samples for a degrading state were extracted from a time interval shortly before the defect of the Peltier module was detected, assuming that a continuous wear had occurred prior to it. With this in mind, it is reasonable to suspect some cases of mislabeling. Because of the rapid transition from State 0 to State 2, data were taken from devices which were de facto still healthy and falsely labeled with 1, leading to a subsequent error in classification.

Conclusion

To conclude, the results of this work show a promising potential for ML models like the random forest classifier or convolutional neural networks to predict the state of health of the Peltier Modules Integrated in ODTCs.

The neural network performed better in generalization and didn't require time-consuming feature engineering. Contrary to the CNN "black-box", decision-tree-based models like the random forest or simple classifiers like k-nearest neighbor can be easily interpreted.

It must be noted that PCR runs can vary substantially in their temperature profile depending on the application, which subsequently leads to overfitted models, when data from only one specific operation mode is used for training. To reach a degree of generalization, which allows an accurate prediction of the health state across a wide range of different operation modes, a lot more data needs to be collected, ideally directly from the field. An alternative approach would be to train specialized models, which are very good at predicting the health state at one specific operation mode. The drawback of this option is that the customer would have to carry out an inspection run while the plant is not being used.

Because of time constraints and limited data, the task investigated in this work was formulated as the classification of three different health states, namely healthy, degrading, and defective. However, in reality the health of Peltier modules is of a continuous developing nature, with the exception of the defective state, which is discrete. To predict the remaining useful life of Peltier modules, training data needs to be sampled during the complete life cycle. Even though its limitations, the study presented in this work was able to provide valuable insights. It was found that the information contained in operational data generated by the current design of the ODTCs is sufficient to identify deviations from a healthy condition of its Peltier Modules. This means that no additional sensors are necessarily needed in the next generation ODTCs for this specific task. Furthermore, it was found that to develop a system of condition-based monitoring, which is able to predict the remaining useful life of Peltier modules integrated in ODTCs in a generalized manner, the biggest challenge will be to collect operational data from customers in the field.

INHECO GmbH is invested to adapt their future product platforms to the standards of modern industry and recognized data-driven applications like condition-based predictive maintenance to be an important part of the development.

Declaration of Competing Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Acknowledgements

We would like to thank Philipp Gritschneder, Luis Huber, Jakob Mannstein, Herwig Angst and Dominik Ellmeier for their help in preparing the test devices and their continuous support throughout the process of data collection. Furthermore, we would like to express our gratitude to Christian George and Aaron Niebergall for supervising the research.

Funding

Funding for the research and publication of this article as well as equipment and laboratories were provided by the company INHECO Industrial Heating & Cooling GmbH.

References

[1] Carvalho TP, Soares FA, Vita R, Francisco RDP, Basto JP, Alcalá SG. A systematic literature review of machine learning methods applied to predictive maintenance. Comput Ind Eng 2019;137:106024.
[2] Silvestrin LP, Hoogendoorn M, Koole G. A comparative study of state-of-the-art machine learning algorithms for predictive maintenance. In: SSCI; 2019. p. 760–7.
[3] Jardine AK, Lin D, Banjevic D. A review on machinery diagnostics and prognostics implementing condition- based maintenance. Mech Syst Sig Process 2006;20(7):1483–510.
[4] Ahmad R, Kamaruddin S. An overview of time-based and condition-based maintenance in industrial application. Comput Ind Eng 2012;63:135–49.
[5] Ahmad B, Mishra BK, Ghufran M, Pervez Z, Ramzan N. Intelligent predictive maintenance model for rolling components of a machine based on speed and vibration. In: 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC); 2021. p. 459–64.
[6] Pinheiro AA, Brandao IM, Da Costa C. Vibration analysis in turbomachines using machine learning techniques. Eur J Eng Technol Res 2019;4(2):12–16.
[7] Dhanraj JA, Sugumaran A. Lazy learning approach for condition monitoring of wind turbine blade using vibration signals and histogram features. Measurement 2020;152:1–13.
[8] Wakiru JM, Pintelon L, Muchiri PN, Chemweno PK. A review on lubricant condition monitoring information analysis for maintenance decision support. Mech Syst Sig Process 2019;118:108–32.
[9] Cioffi R, Travaglioni M, Piscitelli G, Petrillo A, De Felice F. Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions. Sustainability 2020;12(2):492.
[10] Zonta T, da Costa CA, da Rosa Righi R, de Lima MJ, da Trindade ES, Li GP. Predictive maintenance in the Industry 4.0: a systematic literature review. Comput Ind Eng 2020;150:106889.
[11] Sajid S, Haleem A, Bahl S, Javaid M, Goyal T, Mittal M. Data science applications for predictive maintenance and materials science in context to Industry 4.0. Mater Today 2021;45:4898–905.
[12] Çınar ZM, Abdussalam Nuhu A, Zeeshan Q, Korhan O, Asmael M, Safaei B. Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0. Sustainability 2020;12(19):8211.
[13] Leukel J, González J, Riekert M. Adoption of machine learning technology for failure prediction in industrial maintenance: a systematic review. J Manuf Syst 2021;61:87–96.
[14] Promega PCR Amplification; 2019. promega.de/resources/guides/nucleic-acid-analysis/pcr-amplification/ > [accessed march 17].
[15] II-VI Inc Thermocycler; 2021. ii-vi.com/product/thermocycler-series/ > [accessed march 17].
[16] Mazaev T, Ompusunggu AP, Tod G, Crevecoeur G, Van Hoecke S. Data-driven prognostics of alternating current solenoid valves. Prognostics and Health Management Conference (PHM-Besançon); 2020.
[17] Scikit-Learn Accuracy score; 2020. scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score > [accessed march 17].
[18] Scikit-Learn Sklearn model selection GridSearchCV; 2020. scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html > [accessed march 17].
[19] Cybenko G, O'Leary DP, Rissanen J. The mathematics of information coding. In: Extract Distrib; 1999. p. 82.
[20] Keras Dense Layer; 2022. keras.io/api/layers/core_layers/dense/ > [accessed april 22].
[21] Keras Conv2D Layer; 2022. keras.io/api/layers/convolution_layers/convolution2d/ > [accessed april 22].
[22] Keras GlorotUniform Initializer; 2022. keras.io/api/layers/initializers/#glorotuniform-class > [accessed april 22].
[23] Jesus RJ, Antunes ML, da Costa RA, Dorogovtsev SN, Mendes JFF, Aguiar RL. Effect of initial configuration of weights on training and function of artificial neural networks. Mathematics 2021;9:2246 8.
[24] Xue Y. An overview of overfitting and its solutions. J Phys 2019;1168:022022.