Accelerometer-Based Bearing Condition Indicator Estimation Using Supervised Adaptive DSVDD

Maarten Meire[1,2,3], Robert Brijder[4], Gert Dekkers[1,5], Peter Karsmakers[1,2,3]

[1] *KU Leuven, Dept. of Computer Science, ADVISE-DTAI, Kleinhoefstraat 4, B-2440 Geel, Belgium.*

[2] *Leuven.AI - KU Leuven institute for AI.*

[3] *Flanders Make - DTAI-FET.*
maarten.meire,peter.karsmakers@kuleuven.be

[4] *Flanders Make - DecisionS.*

[5] *Magics Technologies, Cipalstraat 3, B-2440 Geel, Belgium.*

Abstract

To prevent unexpected and costly asset downtime, an accurate estimate of the assets condition is needed. In such prognostics setting typically a Condition Indicator (CI) score is calculated based on measurement data. When a future CI surpasses some predefined threshold an alarm can be automatically triggered. Recently, in the literature deep learning (data-driven) methods were proposed to estimate the CI. To the best of our knowledge this paper is the first to evaluate Deep Support Vector Data Description (DSVDD) to estimate a CI for a rolling element bearing setup that is measured by an acceleration sensor. DSVDD typically is used for a task of anomaly detection and hence evaluated in terms of anomaly detection performance. In this paper the DSVDD model is learned as is done for anomaly detection but is evaluated differently. Based on the DSVDD model a CI score can be estimated. In this paper the DSVDD model is evaluated in terms of its ability to estimate appropriate CI scores. Due to a distributional shift between data from the training and test set the model performance can degrade. Therefore, it is investigated which model adaptations strategies can be used to compensate for this effect. All strategies are compared in terms of both CI estimation performance and computational complexity. Only adapting the final layer of the model gave a performance comparable to that when the full model is adapted to the target domain while requiring less calculations. The paper also proposes a simple and easy to calculate center adaptation strategy. This procedure gave a slightly reduced CI estimation performance compared to the alternatives but does not require any model training.

1. Introduction

To prevent unexpected and costly asset downtime an accurate estimate of the assets condition is needed. Prior to breakdown, maintenance is needed when the condition of the asset does not meet the required norm anymore. When looking at rotating machinery, the cause of most system failures are Rolling Element Bearings (REB) (Nabhan, Ghazaly, Samy, & M.O, 2015). A common method to monitor REB in an asset is to measure the vibrations produced by REB (Hoang & Kang, 2019) and to use this data to perform condition monitoring. In the last few years, various data-driven algorithms have been studied for this purpose, both in terms of their accuracy and reliability. There are two main approaches to condition monitoring in bearings, either it is considered as a diagnostics problem where a signal “simply” is classified as healthy or faulty and possibly categorized by a type of fault (Jiang, Chang, & Sheng, 2019), or it can be viewed as a prognostics problem where a condition or Condition Indicator (CI) score is calculated, which forms a basis to predict the Remaining Useful Life (RUL) such as e.g. is done in (She, Jia, & Pecht, 2020). Using the latter prediction of the RUL enables to trigger an alarm when the future RUL is surpassing some predefined threshold. This work considers methods that estimate a CI score.

In the literature a variety of CI estimation methods are described. The most basic CI methods use engineered features in the time, frequency and/or time–frequency domains. These parameters can reflect the state characteristics of the bearing, and usually have a steady trend until the failure becomes more serious (Zhao, Tang, & Tan, 2016). When combined with traditional machine learning techniques such as support vector machines the CI properties (to enable a better RUL prediction) can be improved (Benkedjouh, Medjaher, Zerhouni, & Rechak, 2013). Traditional machine learning requires manual engineering to preprocess the measured signals, which is a tedious and time-consuming process. Deep Learning (DL) can (partly) automate the signal preprocessing and thereby reduce the amount of manual labour. DL was already successfully applied to estimate CI scores. In (Wu, Feng, Wu, Jiang, & Wang, 2019) a multi-scale Convolutional Neural Network (CNN) was used to estimate the CI that was assumed to have a specific curvature over time. In (Su, Li, & Wen, 2020) a variational Auto-Encoder (AE) was learned to extract features from time-series that are suited as an input to a time-window sequence neural network to perform RUL prediction. The combination of a sparse AE and a self-organizing map was used to create a CI in (She et al., 2020), a state space model was then used to perform RUL prediction. A two-step combination of DL and traditional machine learning was used in (Mao, Ding, Tian, & Liang, 2020), where a VGG-16 model was first adapted as deep feature extractor, and then a SVDD model used these features to estimate a CI.

This work also adopts a deep learning approach to automatically generate features using an AE but different to the previously mentioned papers. The latent features are learned by optimizing a one-class Support Vector Data Description (SVDD) objective. A one-class method aims to differentiate between normal and all abnormal behavior. In this formulation of the problem, a single CI can be computed from the deviation from normal behavior described in the latent space defined by the AE. Therefore, it has the potential to detect early faults and track performance degradation. The combination of AE and SVDD, termed Deep SVDD (DSVDD), is described in (Ruff et al., 2018). In this work DSVDD is evaluated for the purpose of CI estimation. Furthermore, in (Ruff et al., 2019) a modified DSVDD objective enables to include examples of faulty behavior, which can boost performance.

A distributional shift between the training and test data might lead to a degradation of the model performance (Wen et al., 2021). In the literature this problem is handled using domain adaptation, or more generally Transfer Learning (TL) (Kouw, 2018; Pan & Yang, 2010). The simplest method to perform deep TL is by fine-tuning a previously trained network (Razavian, Azizpour, Sullivan, & Carlsson, 2014; Yosinski, Clune, Bengio, & Lipson, 2014). Another method commonly found in the literature is a Domain Adversarial Neural Network (DANN), which employs both a domain classifier to discriminate between a source and target domain, and a label predictor to predict the correct labels for data from the source domain. When the domain classifier can no longer discriminate between the source and target, the features are considered joint for both target and source domain (Ganin et al., 2016; Liu & Gryllias, 2020).

A comparison between three classical TL methods and two CNN based TL methods for the construction of a CI was made in (Jiaxian, Wentao, & Yuejian, 2020). It was concluded that a CNN might not benefit from data from different working conditions when constructing a CI and that the use of TL showed a clear benefit with regards to the CI. As mentioned earlier, in (Mao et al., 2020) the authors fine-tuned a VGG-16 model using normal and degradation state data from a set of bearings to adapt the model objective from image classification to CI estimation. In this study we will evaluate different strategies to fine-tune a given DSVDD model based on data acquired during normal operation in the target domain. This situation links a practical scenario where multiple assets are already present and a new asset is acquired, for which a model should then be made.

To evaluate CI estimation methods a novel and unique run-to-failure bearing dataset consisting of multiple test rig setups and multiple runs per setup is used. Compared to publicly available life time test datasets, IMS (Qiu, Lee, Lin, & Yu, 2006) and PRONOSTIA (Nectoux et al., 2012) it is much larger, which is beneficial when using DL methods that typically require a relatively large amount of data. The considered data set contains a total of 70 run-to-failure tests measured across 7 different setups, compared to 12 tests across 3 setups for IMS and 17 tests across 3 setups for PRONOSTIA. Next to this private data set the same experiments were also carried out on the publicly available IMS dataset[1].

In order to compare DL methods an appropriate metric is required. When considering the classification of faults for the purpose of diagnostics, a fault classification is calculated based on an isolated signal pattern. For prognostics a sequence of consecutive realizations should be considered because degradation is a continuous stochastic process and a different metric is needed that describes the behavior of the CI that slowly moves from a normal to faulty state. For the latter, (Kim et al., 2016) used the Spearman's ρ to evaluate a CI score, which was adopted in this work with regards to the linearity between the CI score and the time (i.e. RUL). As results obtained on the IMS dataset in (Kim et al., 2016) will be used in this work for the creation of a ground truth, it was opted to use the same evaluation metric.

The rest of the paper is organised as follows. In Section 2 a more in-depth explanation of the methods that are employed is given. More details about the different fine-tuning strategies for DSVDD are given in Section 3. Section 4 discusses the employed datasets. The experimental setup that includes preprocessing, evaluation metrics and the architecture of the models is explained in Section 5. Section 6 presents and dis-

[1]https://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/

cusses the results of this work. Finally, conclusions and future work are given in Section 7.

2. METHODS

Before briefly reviewing the DSVDD method, first the AE modelling technique is concisely revisited as this method is typically used to initialise the DSVDD learning cycle.

2.1. Convolutional Auto-encoder

A CAE is used to find a compact latent representation $\mathbf{z} \in \mathbb{R}^{D'}$ of input signals $\mathbf{X} \in \mathbb{R}^{D \times T}$ with $D' << D$ using an encoder function $\mathcal{E}$ without loss of a significant amount of information. Thus, given $\mathbf{z}$, a decoder $\mathcal{D}$ calculates a reconstruction $\hat{\mathbf{X}}$, which should resemble $\mathbf{X}$ as closely as possible. When defining the latent representation specifically for data that relates to normal asset operation it is expected that in case data is measured during abnormal asset operation the reconstruction process is less accurate. This creates an opportunity to detect anomalous behavior based on the reconstruction error, termed residual, or by observing differences in the latent representation. For example, when assuming (which might be oversimplifying the problem) that all data that is measured during the first use of an asset can be considered as normal data, the encoder and decoder functions can be learned in an one-class manner. Formally, this is defined as:

$$\mathbf{z} = \mathcal{E}(\mathbf{X}|\theta_E), \tag{1}$$

$$\hat{\mathbf{X}} = \mathcal{D}(\mathbf{z}|\theta_D), \tag{2}$$

where θ_E and θ_D respectively are the parameters of the encoder $\mathcal{E}$ and decoder $\mathcal{D}$ models. During learning these parameters are determined using the following objective:

$$\min_{\theta_E, \theta_D} \frac{1}{N} \Sigma_{i=1}^{N} ||\mathbf{X}_i - \hat{\mathbf{X}}_i||^2, \tag{3}$$

where N is the number of healthy training samples and the subscripts of $\mathbf{X}$ denote a specific sample. This objective minimizes the mean of the squared reconstruction errors. The latter being the differences between the inputs $\mathbf{X}$ and reconstructions $\hat{\mathbf{X}}$. To calculate the CI associated with this algorithm the following equation is used:

$$\text{CI}_{\text{AE}}(\mathbf{X}) = ||\mathbf{X} - \hat{\mathbf{X}}||^2. \tag{4}$$

2.2. Deep support vector data description

In AE abnormal situations are detected using the reconstruction error (which involves evaluating $\mathcal{E}$ and $\mathcal{D}$). However, it is expected that deviant behavior already can be observed in the latent space.

Consider the DSVDD method (Ruff et al., 2019) where the parameters θ_E from $\mathcal{E}(\mathbf{X}_i|\theta_E)$ are refined such that data samples related to normal operation are brought close to a center c while letting data samples acquired during abnormal operation lie further apart from c.

Assume a set of N_n samples that are collected early in the lifetime of the asset where it is expected that it operates normally. Furthermore, assume a set of N_f samples collected when a (similar) asset had faulty behavior. Together, a data set of $N = N_n + N_f$ samples $\{(x_i, y_i)\}_{i=1}^{M}$ are provided, with $\mathbf{X}_i \in \mathbb{R}^{D \times T}$, and $y_i \in \{-1, 1\}$ with $y_i = -1$ referring to a normal sample and $y_i = 1$ to a faulty sample. The generalised DSVDD objective can then be written as:

$$\min_{\theta_E} \eta^{\frac{1+y_i}{2}} \Sigma_{i=1}^{N} (||\mathcal{E}(\mathbf{X}_i|\theta_E) - \mathbf{c}||^2)^{-y_i}. \tag{5}$$

As in (Ruff et al., 2019) using this objective function a sample of normal behavior will be mapped close to the center while a faulty sample will be mapped further away from the center due to the inverse in the function. Note that this objective can also be used when only samples that link to normal asset behavior are available, which makes it different from a binary classification objective. The hyper-parameter η can be used to balance the impact of the normal and faulty examples in the objective as the sizes of both sets (N_n, N_f) can differ much.

The encoder $\mathcal{E}$ notation from Section 2.1 was reused since typically an AE model structure is used in DSVDD with the weights initialised with values that were pretrained using an AE loss function (prior to and decoupled from the DSVDD training cycle). The encoder weights are then further refined using the equation (5). The center c (in the latent space) is set by calculating the mean latent representation (derived by the encoder) of (a subset of) the normal data. This c is kept fixed during further training.

As discussed in detail in (Ruff et al., 2018), some important restrictions apply to deep DSVDD to obtain non-trivial solutions: (a) The center should not be initialized as all 0. This could lead to the model converging to a trivial all-zero weights solution. (b) The network should not have learnable bias terms. The bias terms could lead to the model learning a constant mapping to the center. (c) The network should not have bounded activation functions. These functions could be used as bias terms when the network learns to saturate them, leading to a constant mapping.

As with the AE, an equation similar to the learning objective is used to calculate the CI for DSVDD:

$$\text{CI}_{\text{DSVDD}}(\mathbf{X}) = ||\mathcal{E}(\mathbf{X}|\theta_E) - \mathbf{c}||^2. \tag{6}$$

3. ADAPTIVE DSVDD

The data (source domain) used to train some generic DSVDD model can have a distributional shift compared to the data measured on the asset under test (target domain). Such mismatch causes suboptimal performance of the DSVDD model. Adapting the SVDD model towards the target asset is expected to boost performance. As was argued in the introduction fine-tuning methods are chosen for this purpose. In this work two models are considered: (a) a generic model trained on data that include samples recorded in both normal and faulty conditions from a set of different assets (from the same type) and (b) a run-specific model, which is the generic model that is fine-tuned to the target asset (using only data recorded during normal asset operation) such as is also done in (Mao et al., 2020). Bearing in mind computational efficiency which can be important when DSVDD is deployed on computing hardware with limited processing capabilities, for DSVDD specifically, two additional adaptation strategies are considered: (a) fine-tuning of only the final layer of the generic model instead of the full model as was e.g. done in (Li, Jiang, Zhang, & Shu, 2021); (b) adapting the center c to be the mean of the target latent data (z), which is expected to be normal (target asset is expected to behave normal when the monitoring system is setup). Using the latter the target c is determined by calculating the mean of the latent representation of the target data without needing any retraining of model parameters (using back propagation).

4. DATASETS

As mentioned earlier two datasets will be used in this study: 1) the publicly available IMS dataset, 2) a novel and unique run-to-failure bearing dataset.

4.1. IMS dataset

The IMS dataset (Qiu et al., 2006) consists of data collected from three run-to-failure experiments, without any acceleration. The setup consisted of four double row bearings, as shown in Figure 1, with one accelerometer for each bearing for datasets 2 and 3 and two for dataset 1.

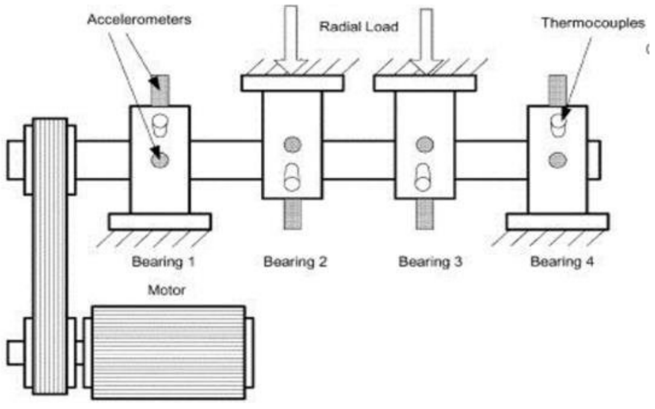

Figure 1. Example of a bearing test rig setup (Qiu et al., 2006).

During the experiments the rotation speed of the shaft was fixed at 2000 RPM and a radial load of 6000 lbs was applied. For each experiment 1 second of data was collected every 10 minutes at a reported sampling rate of 20 kHz, however each file consists of 20480 points and (Gousseau, Antoni, Girardin, & Griffaton, 2016) believes the actual sampling rate to be 20.48 kHz. Each test was carried out until the amount of metal debris on a magnetic plug attached to the test bearings exceeded a set threshold. This dataset does not come with a readily available ground truth with regards to when a fault exactly happened. However, other algorithms have been used to estimate this fault point (Hasani, Wang, & Grosu, 2017), which we will call the cutoff point p_f, and hence can be used to create a ground truth, a summary is provided in Table 1. Do note that there is a noticeable difference in the estimated p_f between each algorithm, indicating a non-trivial problem. In this work it was opted to use the p_f based on the MAS-Kurtosis. Although this algorithm did not provide a p_f for bearing 3 of the third subset, other works have discussed difficulties with this specific experiment (Hasani et al., 2017; Gousseau et al., 2016), hence it was opted to not use data from this specific experiment. Additionally only the experiments where a fault happened are included in this work. The data after p_f is then considered as faulty for our experiments and the first 20% of the healthy data is considered as healthy, we will this the healthy cutoff point p_h.

Table 1. Listing of the faulty bearings in the IMS dataset and their respective p_f. S1B3: Set 1 Bearing 3, S1B4: Set 1 Bearing 4, S2B1: Set 2 Bearing 1, S3B3: Set 3 Bearing 3.

Algorithm	S1B3	S1B4	S2B1	S3B3
AEC	2027	1641	547	2367
MAS-Kurtosis	1910	1650	710	N/A
HMM-DPCA	2120	1760	539	N/A

AEC: auto-encoder-correlation-based prognostic algorithm
MAS-Kurtosis: moving average spectral kurtosis
HMM-DPCA: hidden Markov model with dynamic PCA

4.2. Private dataset

Next to the IMS data set also a novel data set, collected by Flanders Make, that consists out of measured acceleration signals during accelerated life time tests was used. In Figure 2 a bearing test rig setup, built by Flanders Make, is shown. An accelerometer was attached to the bearing housing to measure the accelerations with a sampling frequency of 50 kHz. During a life time test a high radial load of 9 kN was applied and the shaft of the setup was set to rotate at 2000 rpm. The test was stopped when the peak vibrations were at least 20g. A fleet of 7 bearing test rig setups was used and a total of 70 life time tests were performed, with for each test rig, 7 “faulty” runs with a small initial indent in the outer race of the bearing and 3 “healthy” runs with a healthy bearing. The former ran until the stopping condition was met and the latter ran for 2 hours. However, due to different stopping conditions 41 runs were retained. From these, 7 already had abnormal operation from (almost) the beginning of the measurement making

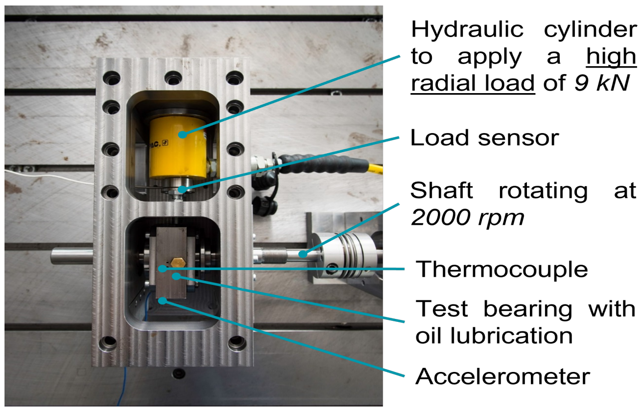
Hydraulic cylinder to apply a high radial load of 9 kN
Load sensor
Shaft rotating at 2000 rpm
Thermocouple
Test bearing with oil lubrication
Accelerometer

Figure 2. Example of a bearing test rig setup.

them unusable to tune run-specific models, which need samples that represent normal operation. In order to have a fair comparison between generic and run-specific models these 7 runs were discarded as well.
As no exact ground truth was provided, a run was manually segmented on the time axis into three regions by visually inspecting the time-frequency representation of the data The cutoff point p_h was chosen such that a stable and healthy behavior between the start and p_h is observed, the initial indent is not seen as faulty in this case. In the experiments p_h is set to 1000s (0.27h) unless the length of the measurement is less than 1.83h. In that case p_h is set to a smaller value such that from p_h until the end of the measurement 1.56h of data is available. In this way for each run a CI can be calculated for at least 1.56h of data, which translates to approximately 2 weeks of non-accelerated time. The second cutoff point p_f was chosen to have in between p_f and the end data that is clearly different from data before p_h. Between p_h and p_f the condition of the test rig is considered undefined.

5. EXPERIMENTAL SETUP

5.1. Preprocessing

Similar to (Jiang et al., 2019; Lee et al., 2021), in this work the acceleration signals are first transformed to log mel spectra before inputting them to a DL model. However, in this paper there is no further processing to Mel Frequency Cepstral Coefficients (MFCC) since log mel spectra enable DL methods to extract richer features compared to when MFCCs are used. From the raw acceleration signals the log mel spectra are extracted, using a window size of 1s and a hop size of 1s for both datasets. A total of 64 and 512 mel bands were extracted for the IMS and private dataset respectively, which were then log scaled. Finally, the log mel spectra of 4, for the IMS dataset, and 8, for the private dataset, consecutive seconds were stacked together to create a frame, with shape (64,4) and (512,8) respectively, that also contains some temporal information, these seconds will receive the same CIs. The exact configuration of the preprocessing was determined during preliminary experiments. After this extraction each run is standardized such that the data prior to p_h has zero mean and unit variance. As a result the data from different runs are expected to be more similar to each other.

5.2. Performance Metrics

As mentioned earlier, (Kim et al., 2016) evaluated the correlation of the CI and the time using the Spearman's ρ (Kokoska & Zwillinger, 1999). The formula used to calculate the Spearman's correlation is shown in Equation 7,

$$\rho = \frac{cov(C_k, T_k)}{\sigma_{C_k}\sigma_{T_k}} \tag{7}$$

with C_k and T_k being the rank of the kth observation in C and T, after both have been ranked from smallest to largest, cov and σ being the covariance and the standard deviations of the rank variables respectively.

This evaluation is performed using the undefined segment of a run, between p_h and p_f. The CI prior to p_h is omitted since the run-specific models use this data for training and the CI after p_f can be dropped since the behavior of the CI after the fault is less important than prior to the fault.

In addition to this metric, we also evaluated the diagnostic ability of the CI, i.e. to use it to discriminate between healthy and faulty data. For this evaluation the Area Under the Receiver Operating Characteristic (AUROC)[2] was used. We calculated the AUROC for the faulty runs and assumed the data before the cutoff p_h to be healthy and all data after the cutoff p_f was considered faulty. As the data before p_h was also used during training for the adaptation strategies, this metric will only be calculated for the generic models. To evaluate the computational complexity the amount of FLoating OPerations (FLOPs), calculated by Tensorflow, will be examined.

5.3. Model architectures and learning parameters

DSVDD was compared to: a) a kurtosis baseline and b) an AE model that is trained using an one-class learning approach using only examples that are considered healthy.

Kurtosis was chosen since p_f for the IMS dataset is based on the MAS-kurtosis, however as this algorithm used data from specific frequencies based on prior knowledge, we opted for the standard kurtosis as the aim of this work is to be fully data driven. Equation 8 was used to calculate the kurtosis, per second, on the raw vibration data, using the implementation of scipy[3],

[2]https://scikit-learn.org/stable/modules/generated /sklearn.metrics.roc_auc_score.html
[3]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kurtosis.html

$$Kurtosis = \frac{\frac{1}{N}\Sigma_{i=1}^{N}(x_i - \mu)^4}{\sigma^4} \quad (8)$$

with x_i being the various samples of the raw vibration data, μ the average of this data and σ the standard deviation.

To allow for a comparison, the architectures of all models were kept as similar as possible. All models were trained with the Adam optimizer (Kingma & Ba, 2014). When building generic models 100 epochs were used and to learn run-specific models an additional 100 epochs were used. The learning rate was set to $1e^{-3}$ for the generic models. For the run-specific models the learning rate was set to $1e^{-4}$ when they were learned by refining a generic model. The batch size was set to 8 for the IMS dataset and to 64 for our private dataset, however this was lowered to 4 when training a run-specific model without added faulty data. If a model did not improve, which we considered as the validation loss lowering by 1%, for 10 consecutive epoch the learning rate was halved and training was stopped early if it did not improve for 15 consecutive epochs. L_2 weight regularisation was applied with a factor (λ) of $5e^{-3}$ and $5e^{-6}$ for the IMS and private dataset respectively.

First we will discuss the architectures for the private dataset, as the IMS dataset uses the same general structure, but a smaller version. An important note here is that all models use a fixed random initialization.

The architecture of the Auto Encoder (AE) used in this study consists of 3 convolutional layers with 32, 32, 16 filters respectively followed by a FC layer with 8 neurons as the encoder and a FC layer with 1024 neurons followed by 3 deconvolutional layers with 16, 32, 32 filters and a final deconvolutional layer with a single filter as the output layer in the decoder. A batch normalization layer follows all the convolutional layers, except for the final one. All layers use leaky ReLu activation functions, except for the final layers in the encoder and decoder which use linear activation functions. The filters are all of size (3,3) and move with a stride of 2 in both directions, except in the final layer a stride of 1 is used. To make a more direct comparison with DSVDD (where bias terms need to be deactivated) the bias terms in all layers were removed. For the batch normalization layers this means the β term has been turned off. This model has 47,392 parameters.

The Deep Support Vector Data Description (DSVDD) model used the same architecture as the AE. This AE model was pretrained for 50 epochs. Then the decoder part was removed and an additional FC layer with 16 neurons was added. The DSVDD objective was then used to fine-tune the encoder network. When assessing models with different amount of labels the fine-tuning cycle always starts from the same pretrained encoder. For the generic model η was experimentally determined and setting it to 0.1 attained the best results. For the run-specific model η was set to 0.1 in order to place more importance on the healthy data, since this data is from the run itself. Since DSVDD does not use the decoder of the AE and adds a single layer, this model has 22,672 parameters.

As mentioned previously, the IMS dataset uses smaller versions of these architectures. Specifically this is implemented by removing the first layer of the encoder and the second to last layer of the decoder and additionally the neurons in the convolutional layers are halved. These adjustments combined with the reduced input features result in the AE model having 7,408 parameters, the DSVDD 3,544.

6. EXPERIMENTS

As explained earlier the use of DSVDD for the purpose of estimating CI scores is studied. For this purpose two datasets were used and two sets of experiments were carried out on each dataset. First the results on the public IMS dataset will be discussed and later the results on the private dataset. The first set of experiments studies the models' ability to generalize by evaluating them in a leave-one-run out manner. The second set of experiments evaluates different adaptation strategies to refine the DSVDD model, and hence build various run-specific DSVDD models, to cope with the distributional shift of data that is present between runs.

6.1. IMS dataset

As mentioned in Section 4.1, this work only included the experiments of the IMS dataset that included a fault. This leads to a set of 3 experiments, which were split into 3 folds using a leave-one-run-out scheme for the generic models. For the run-specific models, as discussed in Section 3 three adaptation strategies for DSVDD are considered, the generic model trained in the 3-fold CV was adapted to a run-specific model using the data prior to p_h of the hold out set as normal data and the faulty data of the training 2 runs. The run-specific model was then validated on the remaining samples of the hold-out set. Note that data from both accelerometers in the first set of experiments was used, but was seen as a single set for the creation of the folds.

Prior to the calculation of the metrics the estimated CI scores were smoothed using a running median of 21 lags.

The prognostic performance of the models in terms of ρ, as introduced in Section 5.2, in function of the percentage of faulty data used during training is shown in Figure 3 for both the generic and run-specific models as well as the kurtosis baseline and the exact values can be found in Table 3.

It can be seen that the results show a high standard deviation, however this is due to the nature of the result calculation. Each result shows the mean and standard deviation of a metric on multiple runs, 5 for the IMS dataset and 34 for the private dataset, and some runs show less complex behavior while others show more complex behavior, resulting in larger differences between the metrics for these runs.

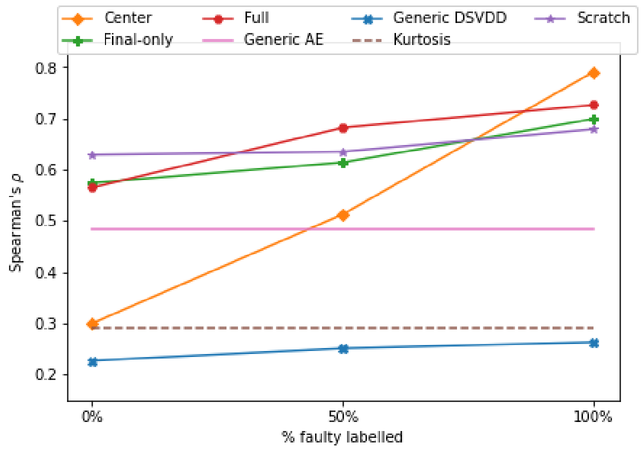

Figure 3. The mean spearman's ρ for generic models, the kurtosis baseline and the various adaptation strategies for DSVDD in function of the amounts of added faulty data for the IMS dataset.

First the results using the generic models will be discussed. It can be seen that AE (`Generic AE`) does outperform DSVDD (`Generic DSVDD`), however do note that AE utilizes a decoder (which gives additional modeling flexibility) in combination with an encoder, while DSVDD only utilizes the latter, which roughly halves the model complexity (and hence the modeling flexibility). The kurtosis baseline (`Kurtosis`) also outperforms DSVDD, which is unexpected. This could be due to the small amount of training data not leading to a good training optimum for this model.

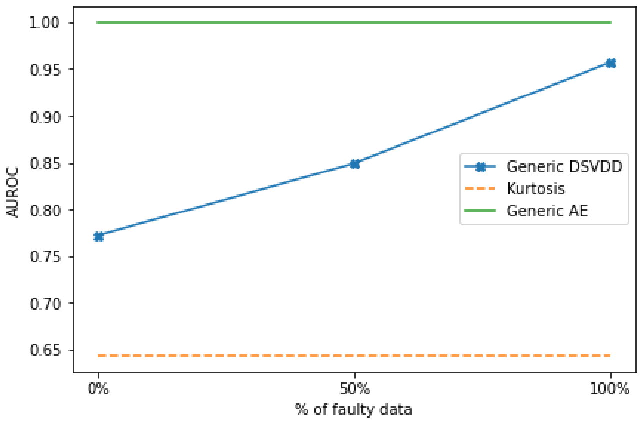

Figure 4. The AUROC comparison for the generic AE and DSVDD models and a kurtosis baseline for the IMS dataset in comparison with the amount of added faulty data.

When looking at the CI evaluation it can be seen that the DSVDD model does not improve much when more faulty data is added, however when looking at the diagnostic behavior, as seen in Table 2 and Figure 4, it can be seen that DSVDD does improve significantly with more faulty data, approaching a (near) perfect AUROC, and it outperforms the kurtosis baseline with regards to diagnostics. The AE showed perfect diagnostic behavior.

Table 2. The AUROC comparison for the generic AE and DSVDD models and a kurtosis benchmark for the IMS dataset in comparison with the amount of added faulty data.

	0%	50%	100%
AE	1.000 ± 0.000		
DSVDD	0.772 ± 0.312	0.849 ± 0.291	0.957 ± 0.087
Kurtosis	0.644 ± 0.174		

Second the results for the run-specific DSVDD models will be discussed. The generic model (`Generic DSVDD`) is compared to a run-specific model that is either build by (a) only adapting the center (`Center`), (b) adapting the weights of the final layer of the model (`Final-only`), and (c) adapting all model parameters (`Full`). As a reference a model was also learned from scratch (`Scratch`), which is similar to the experiment where all model parameters are adapted except that the model weights are randomly initialised (hence, there is no adaptation of a generic model that was previously learned).

Table 3. The mean spearman's ρ with standard deviation for generic models, the kurtosis baseline and the various adaptation strategies for DSVDD in function of the amounts of added faulty data for the IMS dataset.

	0%	50%	100%
Kurtosis	0.290 ± 0.319		
AE	0.484 ± 0.191		
DSVDD	0.227 ± 0.446	0.251 ± 0.354	0.263 ± 0.572
Center	0.300 ± 0.449	0.513 ± 0.326	0.791 ± 0.154
Final-only	0.574 ± 0.116	0.614 ± 0.235	0.699 ± 0.203
Full	0.565 ± 0.249	0.682 ± 0.232	0.726 ± 0.212
Scratch	0.629 ± 0.249	0.635 ± 0.222	0.679 ± 0.271

As was expected, all adaptation strategies outperform the generic model. Final-only, full and scratch performed the best. However, the center adjustment as used by DSVDD already shows a significant increase in performance, even surpassing the other strategies when using all data, in comparison with the generic model. This performance increase might be due to the test run mainly having a mismatched mean in the latent space while the overall shape of the latent data distribution is similar to that of the training data. When the center is then mapped to the center of the latent distribution of the target data the performance is improved significantly. The increasing trend of this boost could be due to the model learning the distribution of other fault types and therefore learning to shape the latent space in a way that is more suitable to an unseen fault type. In contrast to this the final-only scheme adjusts the model to map the test run onto the center. While this does obtain a better performance, the cost of adapting the model in this way is significantly higher compared to only adjusting the center, which can be done during run-time. Additionally, it can be seen that training the model from scratch shows a slight improvement when no faulty data is used, this might be due to the models that adapt the generic model already having learned features on data from other runs, whereas the model that is trained from scratch has only learned features for the specific run.

It can also be seen that adding faulty data does, generally, improves the model, with the best performance being attained when using all the faulty data. This improvement is possibly due to the dataset containing multiple fault types.

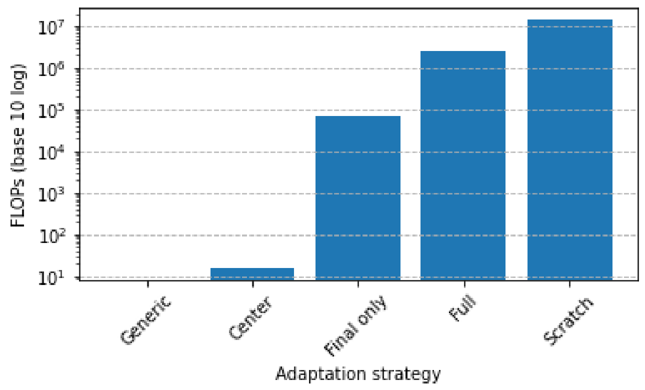

Figure 5. The FLOPs (in base 10 log) needed, per input sample, to adapt the generic DSVDD model for the IMS dataset for each adaptation strategy.

An important aspect to consider when creating these run-specific models is the cost associated with each adaptation strategy. The amount of FLOPs needed, per input sample, to adapt the generic DSVDD model is shown in Figure 5. It should be noted that the lack of necessity for DL training for the center adjustment is not taken into account, which would further increase the difference in the computational cost between these adaptation strategies. It was mentioned earlier that the cost of adapting the model is significantly higher than only adjusting the center, this is shown here, with the center adjustment needing only 16 FLOPS, the amount of values in the latent representation, and adapting the full model needing roughly 13.8 million FLOPs per input.

6.2. Private dataset

As explained in Section 4.2, a total of 34 faulty runs were combined with the 21 healthy runs to form a total of 55 runs, these were then split into 55 folds using a leave-one-run-out scheme to train the generic models. In each realisation 54 runs were used for training and validation by randomly sampling 75% and 25% from these runs for training and validation respectively and the hold out run served as an independent test sample. Similar to 6.1, for each iteration in the 55-fold CV a generic model is adapted to a run-specific model using the first 1000s of the hold out set of the fold as normal data and the faulty data of the training 54 folds. The run-specific model is then validated on the remaining samples of the hold-out set.

As was done for the IMS dataset, the estimated CI scores were smoothed using a running median of 21 lags.

For this dataset the achieved AUROC scores were (near) perfect for both AE and DSVDD with no significant differences, hence these results were not further discussed.

Similar to 6.1, the prognostic performance of both the generic and run specific models as well as the kurtosis baseline in terms of ρ in function of the percentage of faulty data that was used during training is shown in Figure 6 and the exact values can be found in Table 4. Additionally, the performance based on the Ball Pass Frequency Inner (BPFI) is also provided. This metric calculates the envelope spectrum and uses the amplitude for the BPFI frequency as a HI. This could be calculated as we had detailed knowledge about the bearings.

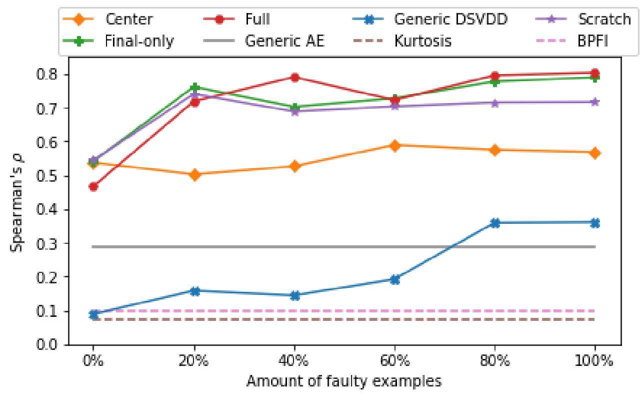

Figure 6. The mean spearman's ρ for the kurtosis baseline, the generic AE and DSVDD models and various adaptation strategies for DSVDD for the private dataset in comparison with the amount of added faulty data.

Again the results for the generic models will be discussed first. AE does outperform DSVDD again when up to 60% of faulty data was added, however, different from the results on the IMS dataset, DSVDD performs better than AE when 80% or more faulty data is used. It is observed that when the number of faulty examples is increased the ρ increases as well, this was less noticeable for the IMS dataset. The kurtosis baseline also does not outperform DSVDD, in comparison to the IMS dataset, likely due to the increased amount of data allowing the models to find better optima and the kurtosis also performs worse in general.

The results for the run-specific results will be discussed second again. In line with the results on the IMS dataset, all adaptation strategies outperform the generic model and the final-only and full strategies performed the best, with scratch only performing slightly worse. However it can be seen that, while all the strategies benefit from the added faulty data,

Table 4. The mean spearman's ρ and standard deviation for the kurtosis baseline, the generic AE and DSVDD models and various adaptation strategies for DSVDD for the private dataset in comparison with the amount of added faulty data.

	0%	20%	40%	60%	80%	100%
Kurtosis	0.071 ± 0.344					
BPFI	0.096 ± 0.244					
AE	0.287 ± 0.485					
Generic	0.089 ± 0.489	0.158 ± 0.485	0.144 ± 0.446	0.192 ± 0.394	0.359 ± 0.489	0.361 ± 0.497
Center	0.537 ± 0.317	0.502 ± 0.376	0.526 ± 0.369	0.589 ± 0.259	0.575 ± 0.345	0.568 ± 0.339
Final-only	0.541 ± 0.376	0.761 ± 0.161	0.702 ± 0.230	0.727 ± 0.221	0.777 ± 0.166	0.789 ± 0.183
Full	0.467 ± 0.344	0.718 ± 0.196	0.790 ± 0.134	0.722 ± 0.255	0.795 ± 0.149	0.803 ± 0.133
Scratch	0.544 ± 0.321	0.740 ± 0.241	0.688 ± 0.280	0.703 ± 0.257	0.715 ± 0.214	0.716 ± 0.247

most of the performance is gained when adding the first 20% to 40%, in comparison to the IMS dataset, where the best performance was attained when using all available data. This could be due to only one fault type being present in this dataset, making this amount of data is probably already sufficient to adapt the algorithms ability to provide a good CI, with additional data only providing a slight further refinement.

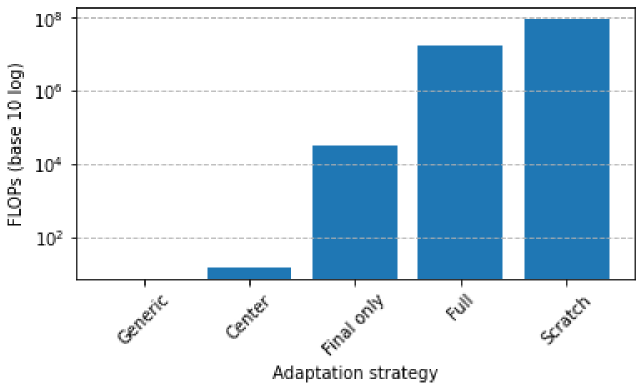

Figure 7. The FLOPs (in base 10 log) needed, per input sample, to adapt the generic DSVDD model for each adaptation strategy.

The amount of FLOPs needed for each adaptation strategy is shown in Figure 7, with the general trend being very similar to the IMS dataset. As the model used for this dataset is larger, the difference between adjusting the center, which needs 16 FLOPs, and training the model from scratch, needing roughly 90 million FLOPs, is much larger.

7. CONCLUSION AND FUTURE WORK

In this work we evaluated and compared a kurtosis basline and AE and DSVDD models that take log mel spectra as their input for the purpose of CI estimation for bearing fault prediction on a custom dataset, that includes accelerometer data, and the publicly available IMS dataset. Apart from this comparison it was also investigated to what extent added labeled faulty data improves the model performance and different adaptation strategies for DSVDD were evaluated in terms of performance increase and computational cost.

For the generic models, AE does outperform DSVDD, likely due to the increased complexity of the model, however adding faulty data lowers this difference and even causes DSVDD to outperform AE on the private dataset. However, the performance of DSVDD already significantly improves when the center is adapted to the target data, which can be done at nearly no cost. When more computational power is available, the diagnostic performance can be further improved by adapting the parameters of the final layer of the DSVDD model. Adapting all model parameters did only give a relatively small improvement. This implies that when there is a certain amount of faulty data available DSVDD is a better choice than AE, however if faulty data is scarce DSVDD either needs to be adapted or AE provides a better alternative.

It also indicated that, depending on whether there are different fault types in the dataset, the amount of faulty data that attains the best performance changes. This is less noticeable for the generic model, but becomes more clear for the run-specific models. When there are more types, as is in the IMS dataset, using all faulty data performs the best. However, when there is only one, as is in the private dataset, adapting the model with only 20% to 40% already attains the best performance.

In future research we will study the inclusion of multi-modal measurements and modified versions of the DSVDD algorithm that can cope with dynamic conditions that can be present in practice such as a changing rpm and/or load.

Acknowledgements

This research received funding from the Flemish Government (AI Research Program). The work presented here was partially supported by the project "ISAAC: Intelligent Sensors for Anomaly Detection in Harsh Environments" funded by the Belgian Government (FOD economie) and Flanders Make.

REFERENCES

Benkedjouh, T., Medjaher, K., Zerhouni, N., & Rechak, S. (2013, 08). Remaining useful life estimation based on nonlinear feature reduction and support vector regression. *Engineering Applications of Artificial Intelli-*

gence, *26*, 1751–1760.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., … Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, *17*(1), 2096–2030.

Gousseau, W., Antoni, J., Girardin, F., & Griffaton, J. (2016). Analysis of the rolling element bearing data set of the center for intelligent maintenance systems of the university of cincinnati. In *Cm2016.*

Hasani, R. M., Wang, G., & Grosu, R. (2017). An automated auto-encoder correlation-based health-monitoring and prognostic method for machine bearings. *arXiv preprint arXiv:1703.06272*.

Hoang, D.-T., & Kang, H.-J. (2019). A survey on deep learning based bearing fault diagnosis. *Neurocomputing*, *335*, 327 - 335.

Jiang, Q., Chang, F., & Sheng, B. (2019). Bearing fault classification based on convolutional neural network in noise environment. *IEEE Access*, *7*, 69795-69807. doi: 10.1109/ACCESS.2019.2919126

Jiaxian, C., Wentao, M., & Yuejian, C. (2020). Transferable health indicator for rolling bearings: A new solution of cross-working condition monitoring of degradation process. In *2020 asia-pacific international symposium on advanced reliability and maintenance modeling (aparm)* (p. 1-6). doi: 10.1109/APARM49247.2020.9209439

Kim, S., Park, S., Kim, J.-W., Han, J., An, D., Kim, N. H., & Choi, J.-H. (2016). A new prognostics approach for bearing based on entropy decrease and comparison with existing methods. In *Annual conference of the phm society* (Vol. 8).

Kingma, D., & Ba, J. (2014, 12). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kokoska, S., & Zwillinger, D. (1999). Crc standard probability and statistics tables and formulae, student edition..

Kouw, W. M. (2018). An introduction to domain adaptation and transfer learning. *CoRR*, *abs/1812.11806*.

Lee, S., Yu, H., Yang, H., Song, I., Choi, J., Yang, J., … Kwon, J. (2021). A study on deep learning application of vibration data and visualization of defects for predictive maintenance of gravity acceleration equipment. *Applied Sciences*, *11*(4).

Li, Y., Jiang, W., Zhang, G., & Shu, L. (2021). Wind turbine fault diagnosis based on transfer learning and convolutional autoencoder with small-scale data. *Renewable Energy*, *171*, 103-115.

Liu, C., & Gryllias, K. (2020). Unsupervised domain adaptation based remaining useful life prediction of rolling element bearings. PHM SOCIETY.

Mao, W., Ding, L., Tian, S., & Liang, X. (2020). Online detection for bearing incipient fault based on deep transfer learning. *Measurement*, *152*, 107278.

Nabhan, A., Ghazaly, N., Samy, A., & M.O, M. (2015, 01). Bearing fault detection techniques - a review. *Turkish Journal of Engineering, Sciences and Technology*, *3*.

Nectoux, P., Gouriveau, R., Medjaher, K., Ramasso, E., Chebel-Morello, B., Zerhouni, N., & Varnier, C. (2012, 06). Pronostia: An experimental platform for bearings accelerated degradation tests. In (p. 1-8).

Pan, S., & Yang, Q. (2010, 11). A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, *22*, 1345 - 1359.

Qiu, H., Lee, J., Lin, J., & Yu, G. (2006, 02). Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics. *Journal of Sound and Vibration*, *289*, 1066-1090.

Razavian, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, *abs/1403.6382*.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., … Kloft, M. (2018, 10–15 Jul). Deep one-class classification. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 4393–4402). Stockholmsmässan, Stockholm Sweden: PMLR.

Ruff, L., Vandermeulen, R., Görnitz, N., Binder, A., Müller, E., & Kloft, M. (2019, 06). Deep support vector data description for unsupervised and semi-supervised anomaly detection..

She, D., Jia, M., & Pecht, M. G. (2020, aug). Sparse auto-encoder with regularization method for health indicator construction and remaining useful life prediction of rolling bearing. , *31*(10), 105005.

Su, C., Li, L., & Wen, Z. (2020). Remaining useful life prediction via a variational autoencoder and a time-window-based sequence neural network. *Quality and Reliability Engineering International*, *36*(5), 1639-1656.

Wen, B. c., Xiao, M. q., Wang, X. q., Zhao, X., Li, J. f., & Chen, X. (2021, September). Data-driven remaining useful life prediction based on domain adaptation. *PeerJ Computer Science*, *7*, e690.

Wu, C., Feng, F., Wu, S., Jiang, P., & Wang, J. (2019). A method for constructing rolling bearing lifetime health indicator based on multi-scale convolutional neural networks. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *41*(11), 526.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *CoRR*, *abs/1411.1792*.

Zhao, M., Tang, B., & Tan, Q. (2016, 05). Bearing remaining useful life estimation based on time–frequency representation and supervised dimensionality reduction. *Measurement*, *86*, 41-55.