Enhancing Siamese Neural Networks through Expert Knowledge for Predictive Maintenance *

Patrick Klein[1], Niklas Weingarz[1], and Ralph Bergmann[1,2]

[1] Business Information Systems II, University of Trier, 54296 Trier, Germany
{kleinp, s4niwein, bergmann}@uni-trier.de
[2] German Research Center for Artificial Intelligence (DFKI), Branch University of Trier, Behringstraße 21, 54296 Trier, Germany
ralph.bergmann@dfki.de

Abstract. The data provided by cyber-physical production systems (CPPSs) to monitor their condition via data-driven predictive maintenance is often high dimensional and only a few fault and failure (FaF) examples are available. These FaFs can usually be detected in a (small) localized subset of data streams, whereas the use of all data streams induces noise that could negatively affect the training and prediction performance. In addition, a CPPS often consists of multiple similar units that generate comparable data streams and show similar failure modes. However, existing approaches for learning a similarity measure generally do not consider these two aspects. For this reason, we propose two approaches for integrating expert knowledge about class or failure mode dependent attributes into siamese neural networks (SNN). Additionally, we present an attribute-wise encoding of time series based on 2D convolutions. This enables that learned knowledge in the form of filters is shared between similar data streams, which would not be possible with conventional 1D convolutions due to their spatial focus. We evaluate our approaches against state-of-the-art time series similarity measures such as dynamic time warping, NeuralWarp, as well as a feature-based representation approach. Our results show that the integration of expert knowledge is advantageous and combined with the novel SNN architecture it is possible to achieve the best performance compared to the other investigated methods.

Keywords: Siamese Neural Network · Predictive Maintenance · Expert Knowledge · 2D Convolution · Time Series Similarity

1 Introduction

The current transition to Industry 4.0 transforms production environments into complex cyber-physical production systems (CPPSs) consisting of several separate cyber-physical systems (CPSs) [20]. Each CPS contains a large number of

* The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-66770-2_6

different sensors and actuators, whereby instances of the same type can occur multiple times in the CPPS. The data recorded by these sensors is enriched with the control commands of the actuators and provides the opportunity to monitor the current condition in order to detect fault and failures (FaFs). For prognostic and health management in a CPS, Lee et al. [19] proposed a similarity-based approach which is generally favored if numerous run-to-failure (R2F) recordings are available [14].

The architecture of siamese neural networks (SNNs) is able to learn a similarity measure by distinguishing training examples of similar pairs from dissimilar ones, even if only few examples are available [10]. However, recent surveys on data-driven predictive maintenance (PredM) approaches [8,12,24] do not consider the use of SNNs, although they are well suited for additional challenges, such as the need of explainability. To the authors' best knowledge, only Zhang et al. [27] applied SNNs on vibration data of ball bearings for similarity-based PredM. While effects resulting from long-term degradation processes can be well monitored at the component level, sensor faults or incorrect control commands due to unexpected situations which are typical for CPSs [21] can only be analyzed by considering the CPPS as a whole. Thus, unlike monitoring a single component, a large number of data streams from the CPPS are potentially relevant. Still, for identifying a specific failure mode, only a relatively small, specific subset is useful. Although SNNs or deep learning, in general, have their strength in learning expressive representations from high-dimensional data, this requires a large amount of examples for each failure mode, which are often not available for most FaFs. However, an engineer is usually able to identify (or at least restrict) the relevant data streams w.r.t. a failure mode, based on general knowledge about the design and mechanics of the CPS. We expect that integrating this knowledge can support an SNN during training and inference. For this reason, we investigate how to infuse prior expert knowledge about the detectability of a failure mode in the form of a restriction on relevant data streams of sensors and actuators into an SNN model, which is also the main contribution of our work. Therefore, we present two approaches for infusing manually defined expert knowledge and compare them with various state-of-the-art methods for time series similarity-based classification on a data set generated by a physical model factory that imitates relevant characteristics of a CPPS.

In the following section the foundations regarding distance-based time series classification with different representations, similarity measures, and SNNs for PredM are presented. Then, our approaches for infusing expert knowledge are introduced in Sect. 3. Next, we evaluate them against various other approaches on a self-created CPPS research data set. Finally, Sect. 5 concludes the results and discusses future work.

2 Foundations and Related Work

2.1 Distance-based Time Series Classification

In a distance-based time series classification scenario, let $D_{train}\{(X_i, y_i)\}$ be our labeled training data set with pairs consisting of an input $X_i \in \mathbb{R}^{a \times t}$ in the form of a multivariate time series with a attributes (streams) and length t as well as its respective label $y_i \in \{1, ..., c\}$. The classification of a k-nearest neighbour (NN) classifier with $k = 1$ for an input $Q \in D_{test}$ is the label y_j of X_j which has the minimal distance from all examples in D_{train}. More formally,

$$y_j \text{ with } j = \arg\min_i \{d(Q, X_i)\}\,, \tag{1}$$

where $d(\cdot, \cdot)$ is a distance function. Since simple measures – such as any form of Minkowski distance functions – cannot handle the alignment of contraction and expansion along the time axis, typically elastic measures such as dynamic time warping (DTW) [4] are used on raw time series data [15]. Therefore, a distance matrix between each time step of Q and X_i is computed in order to find a path with the minimal cumulative distance under some constraints. It has been shown that a k-NN classifier combined with DTW results in the best accuracy on various benchmark data sets [1]. This method can also be used for establishing a similarity-based PredM approach by using the k most similar NNs between a time window Q and a stored time series X_i from previous R2F recordings to estimate the condition of a system. For example, Mai and Chevalier [22] developed a complex similarity function that includes DTW for time series comparison to support diagnosis, corrective actions, and remaining useful life estimations in case of an abnormal event.

2.2 Feature-based Time Series Representation

A well-known problem of DTW, however, is its relatively bad run time behaviour, making it unsuitable for analyzing sensor data of a CPPS in real time. For this reason, it is often preferable to transform the input from the problem space into a more suitable representation in terms of its utility for solving the given problem [3]. A common transformation of time series consists of the automatic computation of a large number of predefined features and the selection of significant ones, e.g. through usage of the TSFresh algorithm [6]. We denote this representation as $f_{TSF}(X) \in R^n$ where n is the number of computed features for a time series X. This type of representation intends to take advantage of the fact that similar time series are represented by similar features, resulting in a small distance between similar instances. For example, kurtosis and root mean square are typical features for monitoring the vibration of bearings and higher values are generally associated with a worse condition.

Other work has shifted to the usage of deep neural networks to learn f in order to extract relevant features that result in an expressive representation of a time series. Typically, autoencoders (AEs) are used to generate a latent representation $H = f_{AE}(X)$ that fuses and compresses multi-sensory data and is learnt by reconstruction of its input [24].

2.3 Siamese Neural Networks for Time Series Similarity

In general, an SNN architecture [5] consists of two deep neural networks that both share the same parameters θ as shown in Fig. 1. These networks are referred to as an encoder that extracts (deep) features $H = f_{SNN}(X)$ from an input X by a neural network $f_{SNN}(\cdot)$. In contrast to AEs, an encoder $f_{SNN}(\cdot)$ learns its parameters θ by distinguishing training examples of similar (X_q, X_p) from dissimilar (X_q, X_n) pairs of time series as shown by Pei et al. [23].

A pair is considered to be semantically similar $y = 1$, if both instances are of the same class and otherwise $y = 0$ for dissimilar pairs. The encoder of the SNN $f_{SNN}(\cdot)$ is trained to produce discriminative representations for X_q, X_p, X_n by minimizing the distance between positive pairs $d(f_{SNN}(X_q), f_{SNN}(X_p))$ and maximizing the distance between negative pairs $d\big(f_{SNN}(X_q), f_{SNN}(X_n)\big)$. Usually, the distance measure $d(\cdot, \cdot)$ can be any simple distance measure such as a Minkowski or cosine distance. The loss is usually computed for a mini batch of size b which consists of half positive and half negative pairs. Thereby, binary cross entropy (BCE) is used as follows to compute the loss value:

$$L(X_a, X_b, y) = -\frac{1}{b} \sum \big(y * log(sim(X_a, X_b)) + (1 - y) * log(1 - sim(X_a, X_b))\big), \tag{2}$$

where X_a, X_b are two time series and $sim(\cdot)$ is a function transforming the distance produced by the SNN into a similarity and $y \in \{0, 1\}$ is the labeled similarity of this pair.

2.4 Related Work

The closest to our work regarding SNNs is the approach by Zhang et al. [27] for bearing fault diagnosis using limited training data. Related to the integration of expert knowledge into deep neural networks is the work of Huang et al. [13] that replicates the hierarchical structure of a production system by the way layers are connected in a neural network. Somewhat related to our idea of manually restricting attributes is the work of Guan et al. [11], in which they included an additional branch to a deep neural network that only focuses on an excerpt of medical images since the most relevant information is located in a small part of the overall image. However, they do not use a manual restriction approach, which means that there is no guarantee that irrelevant noise from the whole image is excluded and that the focus of the analysis is performed on the most relevant excerpt. In contrast to approaches that transform time series into images as input for applying 2D convolutions [8], we use the approach proposed by Assaf and Schumann [2] so that each filter is applied only to one data stream at the same time.

3 Infusing Expert Knowledge About Attribute Relevance

If the amount of data per class is large, a neural network is expected to find meaningful relationships between input and output by itself. However, if the

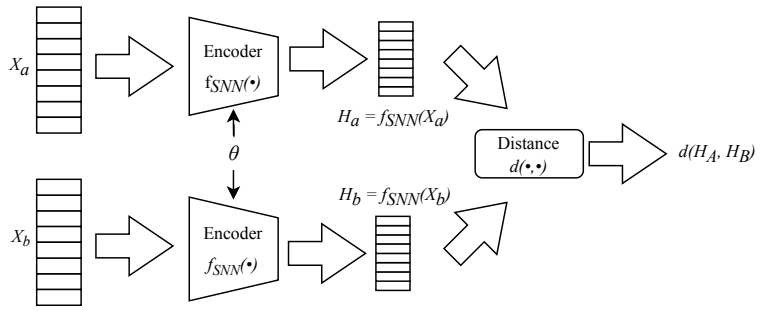

Fig. 1. Typical SNN Architecture

amount of examples per class is relatively small, we expect that – as previously shown on images by Lapuschkin et al. [18] – spurious or artifactual correlations learned from training data can negatively influence the usefulness in real-world applications where these are not present. With a manually defined attribute restriction – referred to as MAR throughout this paper – we want to minimize the influence of noise through irrelevant attributes and support the SNN to focus on the relevant aspects. Our two proposed approaches are driven by the idea that each failure mode has only a small subset of relevant attributes, i.e. sensor data streams, that are necessary for determining its existence. We assume that manually defining this subset and excluding irrelevant attributes should improve performance, robustness and trust of a deep learning model. Moreover, this predefined subset can be further specified into: i) attributes that contain directly measurable patterns which indicate a FaF and ii) attributes that are necessary to establishing the relevant context in the sense of broader situational awareness, e.g. actuator commands and additional sensors. With regard to Turney's classification of attributes [25], we can think of i) as primary ones, ii) as contextual ones and the excluded ones as irrelevant ones.

In the following, we present two approaches that are developed with the purpose to infuse expert knowledge about the attribute relevance by forcing neural networks to use only relevant attributes. Our first approach is more generally applicable regarding the used types of neural network layers and it applies the MAR at the input level, whereas the second one is based on a specialized architecture to apply MAR at a later stage in the architecture.

3.1 Infusing Expert Knowledge at the Input Level

This approach – hereafter referred to as MS-SNN – is based on the assumption that, firstly, many failure modes have common characteristics and, secondly, that prior expert knowledge allows them to be grouped regarding their relevant attributes. More precisely, each label $y_i \in \{1, ..., c\}$ with similar relevant attributes is merged into a group $g_j \in \{1, ..., l\}$ so that $l \ll c$. Then for each group g_j, a separate SNN encoder $f_{SNN,g_j}(\cdot)$ is trained only receiving attributes as input

that are manually defined as relevant for each label y_i of group g_j. Each encoder can be of a different type (e.g. LSTM, CNN) and uses different hyperparameters based on the characteristics of group g_j. To estimate the current state during inference, the class y_i of the example with the highest similarity of all SNNs is selected. In our experiments, the labels are categorized into 8 groups based on the union of relevant features and for each group a CNN as encoder with FC layers is used.

3.2 Infusing Expert Knowledge With 2D CNNs

In this section, we present a modification of the SNN architecture as our second approach – hereafter referred to as CNN2D + MAR – to infuse expert knowledge about the attribute relevance at a deep level. This SNN architecture is designed to ensure two objectives: firstly, preserve the attribute dimension as long as possible to enable attribute-wise access at deep layers. Secondly, to ensure that only manually defined attributes are considered for generating the encoding and assessing the similarity. The modifications conducted on input, encoder, and distance measure w.r.t. a standard SNN are presented below.

Input The normal input of a time series $X_1 \in \mathbb{R}^{a \times t}$ is extended by $x_2 \in \{0,1\}^{2a}$ such that $x = \{X_1, [x_{2_1}, x_{2_2}]\}$. This additional input is based on the training example's label y_i and consists of two vectors $x_{2_1}, x_{2_2} \in \{0,1\}^a$ indicating the manually defined attribute's relevance for a failure mode. The vector x_{2_1} defines which attributes should be compared in a univariate way, whereas x_{2_2} defines which attributes are taken into account for defining the context which ensures that also multivariate aspects are considered.

Encoder The encoder consists of two distinct branches. The first branch encodes time series and the second one learns a weighting value z between both outputs of the first branch.

The first branch applies 2D convolutions (convs) to $X_1 \in \mathbb{R}^{a \times t}$. Therefore, l layers of 2D convs with a filter size of $(1 \times k)$ followed by batch normalization and a ReLU activation function are applied. The filter size causes that each filter is applied on k time steps of 1 attribute at the same time. The lth layer uses a (1×1) convolution to reduce the number of generated feature maps to 1 which means that we finally get a feature map $M^l \in \mathbb{R}^{a \times t_y}$ at layer l where t_y is the length of the time dimension. If strides $s > 1$ are used in the convs, then $t_y < t$ holds, which corresponds to a compressed time dimension. Traditionally, the conv layers are followed by one or more fully connected (FC) layers to connect the different features (from the feature maps). Since we want to obtain a separate feature representation for each attribute, so-called time-distributed FC layers are applied attribute-wise which means that each row $m_l^i \in M^l$ is processed by the same FC layers. Finally, this results in our first output $H_1 \in \mathbb{R}^{a \times n}$ where n is the number of units of the last FC layer. The purpose of this transformation is to obtain a representation for each attribute i which corresponds to the row

vector of H_1, denoted as $h_1^i \in \mathbb{R}^n$, and enables the use of an attribute-weighted distance measure.

A remaining deficiency of H_1 is that this representation does not consider patterns that occur across different attributes which is typically computed by 1D convs. Especially in CPPSs where a lot of failures can only be determined in its context, i.e. in relation of several data streams, this perspective on the data is important. For this purpose, H_1 is multiplied element-wise ($\odot$) with the relevance vector $x_{2_2} \in \{0,1\}^a$, which serves as a gate so that only features of relevant attributes are considered for generating a "contextual" representation. This "contextual" representation $h_2 = g(H_1 \odot x_{2_2})$ is obtained by processing the gated input with several FC layers that are referred to as $g(\cdot)$. This results in the second output $h_2 \in \mathbb{R}^j$ where j is the size of the last FC layer. In summary, the first branch generates two outputs of which the first $H1$ allows an association to the input attributes and the second h_2 only considers the attributes that are manually defined as relevant.

The second branch is responsible for a further output z, which is used to weight the distances resulting from both previous outputs based on the failure mode of the training example used for comparison. Therefore, z is obtained by $z = k(x_{2_2})$ where k represents several FC layers. The last layer consists of one neuron with a sigmoid activation function so that $z \in [0,1]$. Since both encoders receive the same input x_{2_2} based on the known training example, both encoders generate the same value for z.

Distance The distance between a pair (a,b) using the output of the encoder $f_{SNN}(X_1, [x_{2_1}, x_{2_2}]) = \{H_1, h_2, z\}$ where $H_1 \in \mathbb{R}^{a \times n}$, $h_2 \in \mathbb{R}^j$, $z \in [0,1]$ is calculated as follows:

$$d(a,b) = z * d_1\big(H_{1,a}, H_{1,b}\big) + (1-z) * d_2\big(h_{2,a}, h_{2,b}\big)\,. \tag{3}$$

The distance function $d_1(\cdot,\cdot)$ is an attribute weighted distance and would be calculated for the Manhattan distance as follows:

$$d_1\left(H_{1,a}, H_{1,b}\right) = \frac{1}{r}\sum_{i=1}^{r} \left|w_i * (H_{1,a}^i - H_{1,b}^i)\right|, \tag{4}$$

with $\sum_{i=1}^{r} w_i = 1$ and $r = a * n$ is the number of entries of output matrix H_1. The weighting is based on x_{2_1} of the training example and restricts the distance calculation to only relevant attributes. Since the amount of attributes to be considered can vary greatly, we normalize each weight $w_i = \frac{1}{m \times n}$ where m is the sum of relevant attributes multiplied by the length n of each attribute vector h_1^i. In the case of an irrelevant attribute, we use $w_i = 0$ so that this distance does not influence our assessment.

For the Manhattan distance, $d_2(\cdot,\cdot)$ would be calculated as follows:

$$d_2\left(h_{2,a}, h_{2,b}\right) = \frac{1}{j}\sum_{i=1}^{j} \left|(h_{2,a}^i - h_{2,b}^i)\right|. \tag{5}$$

Since the input $h_{2,a}$ and $h_{2,b}$ are vectors that represent the current context, $d_2(\cdot,\cdot)$ has the purpose of a contextual distance measure that considers the multivariate aspects. This corresponds to the typically used distance measure for SNNs, but with the difference that its input $h_{2,a}$ and $h_{2,b}$ is exclusively generated of features from attributes that have been manually classified as relevant.

4 Evaluation

4.1 Fischertechnik Model Factory Data Set

We use the Fischertechnik (FT) factory model presented in [16] for the simulation of a CPPS. It consists of five workstations (WSs) such as a sorting line with colour recognition, a high-bay warehouse, two processing stations and a vacuum gripper robot, which are connected to each other in order to simulate the processing of workpieces. Each of these WSs is made up of FT parts, which means that actuators and sensors of the same type occur multiple times within the factory. A detailed description of the relation and structure is available in form of an ontological knowledge base [17]. The raw data is generated by multiple runs of this factory model, in which one (or no) failure mode is simulated at a time. During each run, 61 attributes are recorded, representing sensor data streams and actuator control commands. The data set contains 29 classes that correspond to different failure modes, components and conditions, which can be summarized into four groups:

- False signals from control sensors are simulated over a period of approx. 10 s continuously or intermittently for light barriers and position switches at five different positions.
- Wear is simulated by artificially induced vibrations on two conveyor belt motors with intermittent or exponential progression over random durations. For the simulation of an exponential degradation progression, a state deviating from normal is classified as degraded first and then critical before the failure occurs, while the intermittent failure mode simulation only has the critical phase. The motors are monitored by three-axis acceleration sensors.
- Leakages are simulated in pneumatic systems, which are monitored by differential pressure sensors.
- Other FaFs such as a broken tooth of a gear or slippage on the conveyor belt were fabricated. Additionally, incorrect transport processes of the vacuum gripper due to a missing workpiece were simulated.

For processing the raw data into examples, a sliding window approach with a time window of 4 s, an overlap of 1 s, and a sampling rate of 4 ms is used, resulting in a shape of $X \in R^{61\times1000}$. Our training data set consists of 25,550 examples, of which 24,908 are labeled as a normal state and 642 labeled as FaFs. The test data set is composed of 18 classes and consists of 3,389 examples, of which 2907 are labeled as normal state and 482 labeled as FaFs. The splitting into both sets was done in a way that all examples labeled as FaF are separated based

on a complete R2F recording, so that all examples of single R2F are either only included in the test or training data set. A detailed overview of the classes and their distribution in the data sets can be found in appendix A.

The selection of relevant attributes for each failure mode is based on expert knowledge with an average of about 4 attributes per FaF label for x_{2_2} and around 1.4 for x_{2_1}. In our case, this knowledge was obtained by examining the (visualized) data streams. Alternatively, this could be acquired by means of a failure mode effect analysis (FMEA) [7]. For example, to monitor a leakage in our pneumatic systems, we have identified its differential pressure and the control parameters of their valves and compressors as relevant attributes. For the healthy state (no failure simulated), the union of features selected for all failures modes is used, resulting in a count of 41 used for x_{2_1} as well as x_{2_2}.

4.2 Approaches for Measuring Time Series Similarity

Baseline Methods We report three baseline methods that are not based on neural networks. The first one combines a feature-based representation $f_{TSF}(X_1)$ with 27,969 significant features with the Euclidean distance. The other variants are based on DTW using a FastDTW implementation, whereby i) DTW is directly applied to X_1 or ii) X_1 is reduced to the relevant attributes according to x_{2_2} of the labeled training example before DTW is applied.

Siamese Neural Networks As part of the evaluation, we use three encoder types, namely a standard CNN with 1D convs, 2D convs with a filter size of $(1 \times k)$ and one with LSTM layers. Each conv operation used valid padding and is followed by batch normalization and processed by a ReLU activation function. A dropout regularization function is added as final layer. For all CNN2D variants (except CNN2D + MAR) an additional 1D conv operation with 61 filters, a filter size and stride of 1 on the input X_1 is used to obtain a feature map $m \in R^{a \times t}$. This feature map contains (contextual) information across a time step which is then concatenated with X_1 so that the input into the 2D conv encoder is $\in R^{a \times t \times 2}$. Additionally to these core encoders, we considered two extensions which are applied subsequently. First, FC layers which are typically used in CNNs to merge different features into a feature vector that is then passed to the distance measure. Second, the approach of NeuralWarp (NW) [9] that uses an additional feed forward neural network for similarity determination between each time step of two time series based on their deep representations instead of a standard distance measure. In both extensions, the input to each FC layer is batch normalized and ReLU is used as activation function, except the last neuron of NW, which uses a sigmoid activation function.

The configurations which (to the best of our knowledge) yield the highest performance are shown in Tab. 1.

Table 1. Models Selected for Evaluation

		CNN1D	CNN1D FC	CNN1D NW	CNN2D	CNN2D FC	CNN2D + MAR	LSTM NW
Training	Batch Size	64	64	64	64	64	64	16
	Learning Rate	0.001	0.00001	0.001	0.001	0.001	0.001	0.001
	Dropout	0.05	0.05	0.05	0.05	0.05	0.1	0.05
Encoder	Layer Type	Conv1D	Conv1D	Conv1D	Conv2D	Conv2D	Conv2D	LSTM
	Units per Layer	256, 64, 32	256, 64, 32	256, 64, 32	128, 64, 16, 1	128, 64, 16, 1	128, 64, 16, 1	128, 64, 32
	Filter Size	7, 5, 3	7, 5, 3	7, 5, 3	5,5,3,1	5,5,3,1	5,5,3,1	-
	Stride	2, 2, 1	2, 2, 1	2, 2, 1	2,2,2,1	2,2,2,1	2,2,2,1	-
	FC-Layer	-	1024, 768, 512	-	-	1024, 768, 512	(128, 64, 32) (256, 128, 64)	-
NW	Units per Layer	-	-	32, 16, 1	-	-	-	32, 16, 1
Dis.	Distance	MH	MH	NW	MH	MH	Weighted MH & MH	NW

4.3 Experimental Setup and Training Procedure

Across all variants, a batch size of 64 is used, meaning that 128 (64 positive and 64 negative) pairs are processed in each batch. Due to high VRAM requirements, the batch size for the LSTM encoder has to be reduced to 16 though. The composition of batches follows the approach that each one should contain one example of all 29 classes and reflect the data distribution. For this reason, 32 pairs are sampled according to the data set distribution, which therefore mostly leads to pairs of healthy condition while the remaining 32 pairs are selected equally distributed over all classes. Furthermore, if not stated otherwise, the Manhattan (MH) Distance is applied.

The selection of hyper parameters, such as layer size, learning and dropout rate, is performed manually by expert knowledge by means of achieving a steep loss during the first 300 epochs of training. We use early stopping to determine the termination of the training process if there is no improvement to the loss in an interval of i) 500 or ii) 1000 epochs. Among all models saved every 10 epochs in a single training, the one with the best loss is selected. To avoid overfitting of the CNN2D + MAR variant due to the attribute restrictions, the dropout rate is increased from 0.05 to 0.1 in contrast to CNN2D and CNN2D FC. In addition to the early stopping described above, a fixed limit of 0.03 is set for the loss and the number of epochs is limited to a maximum of 2500.

The evaluation is conducted on a partial section (case base) of the training data set, which is identical for all experiments. It contains a maximum of 150 examples per class that are randomly selected and only reached by the class representing the healthy state. In total, the case base therefore consists of 792 examples. All approaches are evaluated using the k-NN method with $k = 1$, which is consistent with the procedure of related work [1,9,15].

4.4 Predictive Maintenance Related Quality Measures

In order to examine the suitability of our approaches for the application in PredM, we propose several quality measures that are shown in Tab. 2. First of all, we consider the standard measures Precision (Prec.), Recall (Rec.) as well as the

F1-Score (F1) as weighted average based on the number of examples per class. In the context of PredM, it makes sense to take a closer look at the normal condition class because too many false positives would lead to unnecessary inspections and too many false negatives would result in undetected FaFs. Therefore, we report the F1-Score of this class – referred to as Health F1 – as major criterion for distinguishing between normal and faulty states and indicating the overall utility of an approach.

A disadvantage of these standard measures, however, is that they do not consider the value of a wrong prediction. In the case of the real world application of PredM, a wrong classification could still be useful for a human maintenance engineer in order to locate the source of an issue, even if the specific failure mode is classified wrongly. For this reason, we report self-defined quality measures that evaluate the usefulness of a prediction based on the factors location, condition and failure mode. Location (Loc.) assesses the position in the factory at which a deviation from the normal state is detected. Condition (Cond.) distinguishes between different stages of the fault progression and is divided into normal, degraded, critical and failure, whereby a subset of these is used for different failure modes. The diagnosis (Diag.) score represents the similarity between different ways in which a component may fail. In case of a label that diverges from normal condition, the quality of each factor is calculated as $\frac{\sum_i^n u_i}{n}$ where n is the number of test examples labeled as FaFs and u_i is the predefined usefulness for humans corresponding to the label of the test example i. If the classified class is equal to the label of example i then $u_i = 1$, otherwise $u_i \in [0, 1)$. Moreover, the measure H+F is an extension of the previously described measures by also considering examples with normal condition in the calculation described above. The resulting three factors are averaged to provide a more comprehensive measure.

4.5 Results

For each SNN, we perform multiple training runs with variation of the loss function and early stopping limit (as per section 4.3). Using an additional validation set for parameter optimization and model selection was not possible due to the restricted amount of available examples of failures. Hence, we selected the parameter combination with the highest overall Health F1 score (as per Sect. 4.4. If these scores are equal, the decision is made according to the overall F1 score, since it weights precision and recall equally. In most cases the variants with BCE as loss and an early stopping limit of 1000 epochs are selected (except NeuralWrap (CNN1D) with a limit of 500 epochs and Neural Warp (LSTM) with a mean squared error as loss function), although there are only slight deviations in the scores of the other combinations. In contrast, the CNN2D + MAR SNN showed larger deviations for the performance measures, which we could not resolve by hyperparameter tuning, so the average of six consecutive training runs is reported. The aim was to ensure that an appropriate reference point for comparison is used, which does not overstate the results but also does not disadvantageously reflect them. Our evaluation results are shown in Tab. 2.

In the evaluation we especially focus on the consideration of two hypotheses:

H1 The integration of expert knowledge about the relevance of individual attributes (MAR) can improve similarity-based classification if for each failure mode only a small subset of attributes is relevant.

H2 Learning methods (e.g. SNNs) are superior compared to static methods (e.g. DTW) because of their ability to adapt to the problem domain.

We can confirm H1 with the exception of the MS-SNNs approach. Both DTW + MAR and CNN2D + MAR provide better results compared to their respective variants without attribute relevance knowledge. The results also confirm H2 since approaches without learning capabilities (1st block of Tab. 2) generally achieve worse results than those with learning capabilities (2nd and 3rd block of Tab. 2), again with the exception of MS-SNNs. Another disadvantage of these static approaches is their poor run time behaviour, which reduces their applicability in real-time applications. For instance, in our computing environment the classification of an example with DTW takes about 18.63 s (using multithreaded computation) and is therefore slower than our 2D-CNN + MAR approach (1.36 s/example, using a single NVIDIA Tesla V100 GPU) by a factor of about 13.7.

Furthermore, the results show that joint learning of all classes performs significantly better than using several specialized SNNs (MS-SNNs approach). It can be assumed that the larger number of classes to be delimited requires the learning of more distinct representations. Overall, our proposed CNN2D + MAR architecture is able to achieve the best performance by leveraging expert knowledge about the attribute relevance. For validating the significance of our results we used a non-parametric stratified shuffling test [26] with $p < 0.01$ for Prec., Rec. and F1 against CNN1D FC and CNN2D FC, since these also achieve high Health F1 values of 0.97. Considering the results of the six trained CNN2 + MAR models, a significance is shown especially in comparison to the CNN1D FC. The best two trained models show significantly better results compared to the other two models for all metrics. Only the recall is not sufficiently significant compared to CNN2D FC.

Furthermore, CNN2D and CNN2D + MAR provide the best results for locating fault locations (see Loc. score, Table 2). A possible explanation might be that the deep representations used for similarity comparisons always retains the attribute dimension. The CNN2D + FC variant loses this link in the FC layer, which could explain the weaker performance for this measure.

5 Conclusion and Future Work

This research has shown that the integration of expert knowledge about relevant attributes (data streams) can enhance the performance of similarity-based deep learning approaches for high-dimensional time series classification. Additionally, restricting the similarity-based classification to manually defined attributes could help to improve confidence into the predictions. Furthermore, a general suitability for applying 2D convolutions on multivariate time series consisting of data streams with similar characteristics has been shown. Our proposed model, which

Table 2. Evaluation Results of Different Time Series Similarity Measures and Standard Deviation of Multiple Runs for CNN2D + MAR

Approach	Health	Failure			H+F	Overall		
	F1	Diag.	Loc.	Cond.		Prec.	Rec.	F1
Feature-based + Eucl.	0.76	0.48	0.44	0.59	0.62	0.81	0.57	0.67
DTW	0.85	0.70	0.66	0.74	0.76	0.87	0.70	0.77
DTW + MAR	0.94	0.44	0.39	0.57	0.87	0.85	0.84	0.84
CNN1D	0.96	0.73	0.64	0.76	0.91	0.89	0.85	0.87
CNN1D FC	**0.97**	0.66	0.55	0.73	0.92	0.88	0.87	0.87
NeuralWarp (CNN1D)	0.96	0.70	0.64	0.79	0.91	0.87	0.84	0.85
NeuralWarp (LSTM)	0.95	0.71	0.59	0.77	0.90	0.88	0.83	0.85
CNN2D	0.95	0.75	**0.76**	0.79	0.90	0.89	0.85	0.87
CNN2D FC	**0.97**	0.72	0.59	0.75	0.92	0.89	0.87	0.88
CNN2D + MAR	**0.97** *±0.00*	**0.78** *±0.04*	**0.76** *±0.05*	**0.84** *±0.03*	**0.93** *±0.01*	**0.91** *±0.01*	**0.88** *±0.01*	**0.89** *±0.01*
MS-SNN + MAR	0.68	0.60	0.51	0.76	0.54	0.86	0.47	0.60

combines these two approaches, is able to achieve the best results compared to all other methods.

Future work could examine the usefulness of our attribute-wise time series representations for transfer learning purposes, e.g. for comparing the same failure modes between different instances of similar components. Furthermore, it could be investigated whether the integration of more knowledge through the use of different 2D CNNs based on sensor data properties, such as sampling rate or attribute value range, could further enhance performance. Finally, to promote further research of time series analysis for PredM, we provide our implementation, data set as well as supplementary resources at https://github.com/PredM/SiameseNeuralNetwork.

References

1. Abanda, A., Mori, U., Lozano, J.A.: A review on distance based time series classification. Data Mining and Knowledge Discovery **33**(2), 378–412 (2018)
2. Assaf, R., Schumann, A.: Explainable Deep Neural Networks for Multivariate Time Series Predictions. In: Proc. of the Twenty-Eighth Int. Joint Conf. on Artif. Intell., IJCAI-19. pp. 6488–6490 (2019)
3. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-based Applications. Springer-Verlag (2002)
4. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD workshop. vol. 10, pp. 359–370. Seattle, WA, USA: (1994)
5. Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., Shah, R.: Signature Verification Using A "Siamese" Time Delay Neural Network. IJPRAI **7**(4), 669–688 (1993)
6. Christ, M., Braun, N., Neuffer, J., Kempa-Liehr, A.W.: Time Series Feature extraction on Basis of Scalable Hypothesis Tests. Neurocomputing **307**, 72–77 (2018)

7. Douglas Goodman, James P. Hofmeister, F.S.: Prognostics and Health Management: A Practical Approach to Improving System Reliability Using Condition-Based Data. Wiley (2019)
8. Fink, O., Wang, Q., Svensén, M., Dersin, P., Lee, W.J., Ducoffe, M.: Potential, challenges and future directions for deep learning in prognostics and health management applications. Engineering Applications of Artif. Intell. **92**, 103678 (2020)
9. Grabocka, J., Schmidt-Thieme, L.: NeuralWarp: Time-Series Similarity with Warping Networks. CoRR **abs/1812.08306** (2018)
10. Gregory Koch, Richard Zemel, R.S.: Siamese Neural Networks for One-Shot Image Recognition. In: ICML, Lille, France (2015)
11. Guan, Q., Huang, Y., Zhong, Z., Zheng, Z., Zheng, L., Yang, Y.: Thorax Disease Classification with Attention Guided Convolutional Neural Network. Pattern Recog. Lett. **131** (11 2019)
12. Guo, J., Li, Z., Li, M.: A Review on Prognostics Methods for Engineering Systems. IEEE Transactions on Reliability (2019)
13. Huang, X., Zanni-Merk, C., Crémilleux, B.: Enhancing Deep Learning with Semantics: an Application to Manufacturing Time Series Analysis. Procedia Comput. Sci. **159**, 437–446 (2019)
14. Jia, X., Cai, H., Hsu, Y.M., Li, W., Feng, J., Lee, J.: A Novel Similarity-based Method for Remaining Useful Life Prediction Using Kernel Two Sample Test. In: Proc. of the Annu. Conf. of the PHM Society (2019)
15. Jiang, W.: Time series classification: nearest neighbor versus deep learning models. SN Applied Sciences **2**(721) (2020)
16. Klein, P., Bergmann, R.: Generation of Complex Data for AI-Based Predictive Maintenance Research With a Physical Factory Model. In: 16th Int. Conf. on Inform. in Control Automat. and Rob. pp. 40–50. SciTePress (2019)
17. Klein, P., Malburg, L., Bergmann, R.: FTOnto: A Domain Ontology for a Fischertechnik Simulation Production Factory by Reusing Existing Ontologies. In: Proc. of the Conf. LWDA. vol. 2454, pp. 253–264. CEUR-WS.org (2019)
18. Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., Müller, K.R.: Unmasking Clever Hans predictors and assessing what machines really learn. Nat. Commun. **10**(1) (2019)
19. Lee, J., Bagheri, B., Kao, H.A.: A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manufacturing letters **3**, 18–23 (2015)
20. Lezoche, M., Panetto, H.: Cyber-Physical Systems, a new formal paradigm to model redundancy and resiliency. Enterprise Information Systems pp. 1–22 (2018)
21. Luo, Y., Xiao, Y., Cheng, L., Peng, G., Yao, D.D.: Deep Learning-Based Anomaly Detection in Cyber-Physical Systems: Progress and Opportunities. CoRR **abs/2003.13213** (2020)
22. Mai, C.K., Chevalier, R.: Equipment Diagnostics based on Comparison of Past Abnormal Behaviors Using a Big Data Platform. In: Proc. of the Annu. Conf. of the PHM Society (2016)
23. Pei, W., Tax, D.M.J., van der Maaten, L.: Modeling Time Series Similarity with Siamese Recurrent Networks. CoRR **abs/1603.04713** (2016)
24. Ran, Y., Zhou, X., Lin, P., Wen, Y., Deng, R.: A Survey of Predictive Maintenance: Systems, Purposes and Approaches. CoRR **abs/1912.07383** (2019)
25. Turney, P.D.: The Management of Context-Sensitive Features: A Review of Strategies. CoRR **abs/cs/0212037** (2002)
26. Yeh, A.: More accurate tests for the statistical significance of result differences. arXiv preprint cs/0008005 (2000)

27. Zhang, A., Li, S., Cui, Y., Yang, W., Dong, R., Hu, J.: Limited Data Rolling Bearing Fault Diagnosis With Few-Shot Learning. IEEE Access **7**, 110895–110904 (2019)

A Dataset

Table 3. Overview of the classes and their distribution contained in the data set. Note that the case base used for evaluation, as described in Sect. 4.3, contains up to 150 examples from the training data set for each class, i.e. 150 for no_failure and every example of all other classes. The "txt" part of the label indicates the location of the component, i.e. the CPS, where the failure was simulated.

Failure mode	Train	Test	Total
no_failure	24908	2907	27815
txt15_conveyor_failure_mode_driveshaft_slippage...	11	0	11
txt15_i1_lightbarrier_failure_mode_1	7	0	7
txt15_i1_lightbarrier_failure_mode_2	6	18	24
txt15_i3_lightbarrier_failure_mode_2	8	5	13
txt15_m1_t1_high_wear	82	88	170
txt15_m1_t1_low_wear	112	79	191
txt15_m1_t2_wear	42	58	100
txt15_pneumatic_leakage_failure_mode_1	11	0	11
txt15_pneumatic_leakage_failure_mode_2	12	0	12
txt15_pneumatic_leakage_failure_mode_3	10	0	10
txt16_conveyor_failure_mode_driveshaft_slippage...	11	12	23
txt16_conveyor_big_gear_tooth_broken_failure	13	11	24
txt16_conveyor_small_gear_tooth_broken_failure	3	0	3
txt16_i3_switch_failure_mode_2	9	0	9
txt16_i4_lightbarrier_failure_mode_1	31	42	73
txt16_m3_t1_high_wear	42	26	68
txt16_m3_t1_low_wear	12	20	32
txt16_m3_t2_wear	63	43	106
txt17_i1_switch_failure_mode_1	17	15	32
txt17_i1_switch_failure_mode_2	24	11	35
txt17_pneumatic_leakage_failure_mode_1	24	9	33
txt17_workingstation_transport_failure_mode_wou...	20	18	38
txt18_pneumatic_leakage_failure_mode_1	11	9	20
txt18_pneumatic_leakage_failure_mode_2	27	10	37
txt18_pneumatic_leakage_failure_mode_2_faulty	11	8	19
txt18_transport_failure_mode_wout_workpiece	6	0	6
txt19_i4_lightbarrier_failure_mode_1	9	0	9
txt19_i4_lightbarrier_failure_mode_2	8	0	8
Total	25550	3389	28939