Expert Knowledge Induced Logic Tensor Networks: A Bearing Fault Diagnosis Case Study

Maximilian-Peter Radtke[1], Jürgen Bock[2]

[1,2] *Technische Hochschule Ingolstadt, Ingolstadt, Germany*
maximilian-peter.radtke@thi.de
juergen.bock@thi.de

ABSTRACT

In the recent past deep learning approaches have achieved some remarkable results in the area of fault diagnostics and anomaly detection. Nevertheless, these algorithms rely on large amounts of data, which is often not available, and produce outputs, which are hard to interpret. These deficiencies make real life applications difficult. Before the broad success of deep learning machine faults were often classified using domain expert knowledge based on experience and physical models. In comparison, these approaches only require small amounts of data and produce highly interpretable results. On the downside, however, they struggle to predict unexpected patterns hidden in data. Merging these two concepts promises to increase accuracy, robustness and interpretability of models. In this paper we present a hybrid approach to combine expert knowledge with deep learning and evaluate it on rolling element bearing fault detection. First, we create a knowledge base for fault classification derived from the expected physical attributes of different faults in the envelope spectrum of vibration signals. This knowledge is used to derive a similarity function for comparing input signals to expected faulty signals. Afterwards, the similarity measure is incorporated into different neural networks using a Logic Tensor Network (LTN). This enables logical reasoning in the loss function, in which we aim to mimic the decision process of an expert analyzing the input data. Further, we extend LTNs by weight schedules for axiom groups. We show that our approach outperforms the baseline models on two bearing fault data sets with different attributes and directly gives a better understanding of whether or not fault signals are influenced by other effects or behave as expected.

1. INTRODUCTION

The increased connectivity of machines and whole production halls enabled by the Industrial Internet of Things (IIoT) has made gathering data and monitoring machines easier. We can collect condition monitoring data about machines, which in turn make new maintenance strategies possible. Before, machines were often run until they failed or maintained on a regular predefined schedule. Now, predictive maintenance, where maintenance activities are based on the remaining useful life of a machine, is widely possible. In comparison to traditional approaches this can reduce machine down times, prolong machinery lifetime and reduce resource consumption (Selcuk, 2017).

To make this possible we must diagnose and predict faults correctly based on available data. In the recent past deep learning (DL) in particular has shown great success in doing so (Fink et al., 2020; Liu, Yang, Zio, & Chen, 2018; Arinez, Chang, Gao, Xu, & Zhang, 2020; Zhao et al., 2019). Despite its broad success DL has a number of known issues. One of the major drawbacks is the necessity of large amounts of data for training DL algorithms, without which generalization is not possible (Fink et al., 2020). Even though IIoT can provide us with plenty of data, the nature of the required data often makes collection very costly or even prohibitive. For example, to gather fault data, on which diagnostics and prognostics can be performed, a machine needs to suffer that fault. Especially for large special machinery this can be very expensive and interrupt the production flow. Another disadvantage is the difficulty with which DL output can be explained (Arrieta et al., 2020). However, this explainability is essential for building trust in the results, which can only be achieved by understanding their occurrence.

According to different authors these shortcoming could be reduced by combining purely data driven models with domain knowledge and symbolic AI techniques to hybrid models (Fink et al., 2020; Arinez et al., 2020; Garcez & Lamb, 2020; Marcus, 2020). To achieve this multiple approaches have been studied in the domain of fault diagnostics and prognostics. For instance, the combination has shown to increase explainability and enable root cause analysis by taking into account input from human experts (Steenwinckel et al., 2021).

Moreover, it has shown to make models more robust and decrease the amount of "real" data needed through high-quality simulation of data based on expert knowledge (Wang, Taal, & Fink, 2021), and increase overall performance by extending the feature space with physics inspired features (Chao, Kulkarni, Goebel, & Fink, 2022).

These approaches all have in common that the knowledge and data based parts can be clearly distinguished from one another. A different way of creating hybrid models is when both approaches are completely integrated into one another. One such idea is neurosymbolic AI, where traditional rule-based AI is merged with recent developments in DL (Garcez & Lamb, 2020). Logic Tensor Networks (LTN) (Badreddine, Garcez, Serafini, & Spranger, 2022), which use a logic language to create a loss function for training arbitrary neural networks, are a realization of this. This approach has shown to be effective on different tasks like semantic image interpretation (Donadello, Serafini, & Garcez, 2017) or deductive reasoning (Bianchi & Hitzler, 2019). To the best of our knowledge, this method has not yet been applied for classifying faults.

Our contribution is to apply the neurosymbolic framework LTN to the fault diagnostics domain. This is demonstrated by showing its suitability for incorporating knowledge into DL in the special case of bearing fault diagnostics with constant shaft speed. For this, we propose a new scoring function based on physical knowledge for classifying bearing faults. The scoring function acts as domain expertise for the LTN loss function, which can thereby be injected into classic DL models. Additionally, we extend the LTN framework by assigning weights to different axiom groups and propose weight schedules along the training process for doing so. We show that our approach outperforms the baseline methods in terms of test accuracy.

In the remainder of this paper we will first lay the foundations for Logic Tensor Networks in Sec. 2. Next, in Sec. 3 our approach for combining knowledge with DL based on LTNs for bearing fault classification is outlined. Finally, we present our experimental setup in Sec. 4 and evaluate our approach on two well known bearing fault data sets in Sec. 5.

2. FOUNDATIONS

A Logic Tensor Network (LTN) is a neurosymbolic framework, which uses a fully differentiable first-order logic language called Real Logic in the loss function for training a neural network (NN). Here we give an overview of Real Logic and how LTNs learn. For an in depth treatment of these topics we refer to the original work (Badreddine et al., 2022).

Real Logic is defined on a first-order logic language $\mathcal{L} = (\mathcal{C}, \mathcal{P}, \mathcal{F}, \mathcal{X})$, where $\mathcal{C}$ is a set of constant symbols, $\mathcal{P}$ a set of relational symbols (predicates), $\mathcal{F}$ a set of functional symbols and $\mathcal{X}$ a set of variable symbols. $\mathcal{L}$ allows for defining logical formulas, e.g., $\forall x((x > \mathrm{thr}) \rightarrow \mathrm{isFault}(x, c))$, which states that all x, that are greater than thr, yield the fault c.

Due to the desired applicability to real world problems fuzzy semantics is used, meaning that the truth value of a logical formula is between 0 and 1 in comparison to just being *true* or *false*. One of the main benefits of Real Logic is its differentiability, which is achieved by grounding onto the real plane, i.e., all logical expressions are mapped onto $\mathbb{R}$. For this to work all elements of $\mathcal{L}$ are typed and attributed to a domain, e.g. the constant *Ingolstadt* is of the domain *City*.

The functions $\mathbf{D}$, $\mathbf{D}_{\mathrm{in}}$ and $\mathbf{D}_{\mathrm{out}}$ return the domains of the elements of $\mathcal{L}$ and are defined as

$$\begin{aligned}\mathbf{D}&\colon \mathcal{X} \cup \mathcal{C} \mapsto \mathcal{D},\\ \mathbf{D}_{\mathrm{in}}&\colon \mathcal{F} \cup \mathcal{P} \mapsto \mathcal{D}^*,\\ \mathbf{D}_{\mathrm{out}}&\colon \mathcal{F} \mapsto \mathcal{D},\end{aligned}$$

where $\mathcal{D}$ is a non-empty set of symbols called domain symbols and $\mathcal{D}^*$ is the Kleene Star of $\mathcal{D}$, which is defined as the set of all finite sequences of symbols in $\mathcal{D}$. Thus, $\mathbf{D}$ outputs the domain of a constant or variable, $\mathbf{D}_{\mathrm{in}}$ gives us the domain of the input for a function or predicate and $\mathbf{D}_{\mathrm{out}}$ returns the output domain for functions.

Grounding means that domains are interpreted concretely as tensors in the real field, constants and variables as tensors of real values, functions as real functions or tensor operations and predicates as functions or tensor mappings to a value in the interval $[0, 1]$. Formally, a grounding $\mathcal{G}$ satisfies the conditions

$$\begin{aligned}&\forall x \in \mathcal{X} \cup \mathcal{C} : \mathcal{G}(x) \in \prod_{i=1}^{k} \mathcal{G}(\mathbf{D}(x)),\\ &\forall f \in \mathcal{F}\colon \mathcal{G}(f) \in \mathcal{G}(\mathbf{D}_{\mathrm{in}}(f)) \mapsto \mathcal{G}(\mathbf{D}_{\mathrm{out}}(f)),\\ &\forall p \in \mathcal{P}\colon \mathcal{G}(p) \in \mathcal{G}(\mathbf{D}_{\mathrm{in}}(p)) \mapsto [0, 1],\end{aligned}$$

with k being the number of instances of x. A grounding depending on a set of parameters θ is depicted as $\mathcal{G}(\cdot|\theta)$. This definition can be expanded to also include all first-order terms and atomic formulas by consecutive application of the grounding function. See (Badreddine et al., 2022) for details.

To ground a complete atomic formula connectives and quantifiers are necessary. We define these in accordance to Product Real Logic as proposed by (Badreddine et al., 2022). Hence, connectives, i.e., conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$) and negation ($\neg$), are defined on first-order fuzzy logic semantics (Hájek, 2013) and are associated with a t-norm (T), t-conorm (S), fuzzy implication (I) or fuzzy nega-

tion (N) respectively. These are defined as

$$\begin{aligned} T(a,b) &= ab, \\ S(a,b) &= a+b-ab, \\ I(a,b) &= 1-a+ab, \\ N(a) &= 1-a, \end{aligned}$$

with fuzzy logic values $a, b \in [0,1]$. Additionally, quantifiers are defined via an aggregation operator $A : \bigcup_{n\in\mathbb{N}}[0,1]^n \mapsto [0,1]$. The for-all ($\forall$) quantifier is defined as the p-mean error

$$A_{\mathrm{pME}}(a_1,...,a_n) = 1 - \left(\frac{1}{n}\sum_{i=1}^{n}(1-a_i)^p\right)^{1/p}, \quad (1)$$

and exists ($\exists$) as the p-mean

$$A_{\mathrm{pM}}(a_1,...,a_n) = \left(\frac{1}{n}\sum_{i=1}^{n}a_i^p\right)^{1/p},$$

given the fuzzy truth values $a_1,...,a_n$ and $p \geq 1$.

Using these definitions we can now ground an exemplary atomic formula $\phi = \forall x((x > \mathrm{thr}) \rightarrow \mathrm{isFault}(x,c))$ with variable x, constants thr, c and predicate $\mathrm{isFault}(x,c)$, which depends on parameters θ, and $(x > \mathrm{thr})$. The respective domains are given by $\mathbf{D}(\cdot)$ and $\mathbf{D}_{\mathrm{in}}(\cdot)$. Therefore, the formula is grounded through

$$\mathcal{G}(\phi|\theta) = A_{\mathrm{pME}}(I(\mathcal{G}((x > \mathrm{thr})), \mathcal{G}(\mathrm{isFault}(x,c)|\theta)).$$

Real logic can be used to create a knowledge base, which is defined by the triple $\mathcal{T} = \langle \mathcal{K}, \mathcal{G}(\cdot|\theta), \Theta \rangle$. Here $\mathcal{K}$ is a set of closed first-order logic formulas (axioms) defined on the set of symbols $\mathcal{S} = \mathcal{C} \cup \mathcal{P} \cup \mathcal{F} \cup \mathcal{X} \cup \mathcal{D}$ and Θ is the hypothesis space for the parameters θ. The goal is to learn some set of parameters that maximizes the satisfiability of the knowledge base

$$\theta^* = \underset{\theta\in\Theta}{\mathrm{argmax}} \ \underset{\phi\in\mathcal{K}}{\mathrm{SatAgg}}\, \mathcal{G}(\phi|\theta), \quad (2)$$

where SatAgg is some aggregation operator. In the following we will use the p-mean error defined in Eq. (1). This formulation can then be used as the loss function of a NN, which searches for an optimal set of parameters that maximizes the satisfiability. A NN with a loss function based on Real Logic is called a LTN.

Due to the differentiability of Real Logic and therefore of the optimization problem in Eq. (2), the loss function can be applied like any other loss function to a NN and all established optimization techniques and network architectures can be used.

An implementation of LTNs in Python based on Tensorflow is available.[1]

3. Approach

3.1. Scoring Function for Bearing Faults

For identifying bearing faults we propose to create a heuristic scoring function, that acts similar to the way an expert would analyze a bearing vibration signal. The function is inspired by the analysis done on the CWRU bearing data set by (Smith & Randall, 2015) and how the authors achieved their assessment by relying on the expected frequencies for specific faults. Depending on the location of the fault these are called ball pass frequency, outer race (BPFO), ball pass frequency, inner race (BPFI) and ball (roller) spin frequency (BSF) and are described by

$$\mathrm{BPFO} = \frac{n_r}{2}(1 - \frac{d_r}{d_p}\cos\psi), \quad (3)$$

$$\mathrm{BPFI} = \frac{n_r}{2}(1 + \frac{d_r}{d_p}\cos\psi), \quad (4)$$

$$\mathrm{BSF} = \frac{d_p}{2d_r}(1 - (\frac{d_r}{d_p}\cos\psi)^2), \quad (5)$$

where n_r denotes the number of rolling elements, d_r the roller diameter, d_p the pitch diameter and ψ the contact angle of the bearing. These frequencies are multiplied by multiples of the shaft speed v_r to obtain the expected fault frequencies over the complete spectrum.

These frequencies are more easily identified when inspecting the envelope spectrum of the vibration signal in comparison to the spectrum of the raw signal. Therefore, we first transform the raw signal into the envelope signal using the Hilbert transform (Bonnardot, Randall, Antoni, & Guillet, 2004). Afterwards, the envelope signal is transferred to the corresponding envelope spectrum. Here the different fault types outer race fault (OR), inner race fault (IR) and ball fault (B) can be recognized by the peaks at the respective fault frequencies BPFO, BPFI and BSF (Smith & Randall, 2015). The fault frequencies are especially visible in lower frequency areas (Randall & Antoni, 2011). Therefore, we only analyze frequencies of up to 500 Hz. To ensure, that the remnants of a fractional pass of a faulty bearing does not disturb our score calculation, we additionally limit ourselves to frequencies higher than one and a half times the shaft speed. We write the operation of transforming a signal x into its envelope spectrum consisting of frequencies $h = (h_1,...h_n)$ and corresponding amplitude values $d = (d_1,...,d_n)$, with $h_n \leq 500$ and $h_1 \geq 1.5v_r$, as $\mathrm{EnvSpec}(x)$.

In the envelope spectrum we can identify peaks by taking the moving average along a predefined window to see which points (h_i, d_i) vary strongly from the norm. We use the well-

[1] https://github.com/logictensornetworks/logictensornetworks

known definition of the moving average

$$\mathrm{MA}^{(w)}(d) = \frac{1}{w}\sum_{i=0}^{w-1} d_{t-i},$$

over a window of w values. We set this window to the shaft speed v_r and omit the superscript.

Let $\mathbb{1}_{\{A\}}$ be the indicator function, which is one if A is true and zero otherwise. Then, if a frequency value h_i has a corresponding amplitude d_i of 2.5 times the moving average, we can mark these as peaks and count them with

$$N_{\mathrm{P}}(d) = \sum_{i=1}^{n} \mathbb{1}_{\{d_i \geq 2.5\mathrm{MA}(d,i)\}}.$$

The identified peaks are then classified into whether or not they are expected, i.e., they are at the expected fault frequencies derived from Eqs. (3), (4) and (5). The expected fault frequencies are given by $F^{(l)} = (F_1^{(l)}, ..., F_m^{(l)})$ for each fault type $l \in \{\mathrm{IR}, \mathrm{OR}, \mathrm{B}\}$. In the following we will omit the superscript l and remark that the rest of the score calculation procedure is applied for all l independently.

Due to the imperfections of the real world we relax the condition of a peak being exactly at the expected fault frequency by the value $\delta = v_r/2$ and get the number of expected peaks

$$N_{\mathrm{EP}}^{(j)}(h,d) = \sum_{i=1}^{n} \mathbb{1}_{\{d_i \geq 2.5\mathrm{MA}(d,i)\}} \mathbb{1}_{\{F_j-\delta \leq h_i \leq F_j+\delta\}}.$$

If multiple peaks fall into the same $F_j \pm \delta$ section the weighted average is taken by calculating the value

$$V_{\mathrm{EP}}^{(j)}(h,d) = \sum_{i=1}^{n} \mathbb{1}_{\{d_i \geq 2.5\mathrm{MA}(d,i)\}} \mathbb{1}_{\{F_j-\delta \leq h_i \leq F_j+\delta\}} h_i,$$

and dividing it through the amount of peaks. Combining this for all expected frequencies $F_1, ..., F_m$ this gives us

$$\bar{V}_{EP}(h,d) = \sum_{j=1}^{m} V_{\mathrm{EP}}^{(j)}(h,d)/N_{\mathrm{EP}}^{(j)}(h,d).$$

The value of all peaks, which do not fall in the expected category, is calculated by

$$V_{\mathrm{PNE}}^{(j)}(h,d) = \sum_{i=1}^{n} \mathbb{1}_{\{F_{j-1}+\delta \leq h_i \leq F_{j-1}-\delta\}} d_i,$$

with $F_0 = -\delta$ and $F_{m+1} = \inf$. By combining the previous equations we obtain the score function

$$f_l(h,d) = \frac{\bar{V}_{EP}(h,d)}{\bar{V}_{EP}(h,d) + \sum_{j=1}^{m+1} V_{\mathrm{PNE}}^{(j)}(h,d)} \tag{6}$$

for each fault type l, which returns a value between 0 and 1. An exemplary visualization for how the score is calculated is given in Fig 1.

The value can be interpreted as the share of peaks, which were as expected for a specific fault. Therefore, we set the threshold $\mathrm{thr}_{\mathrm{score}} = 0.49$ for determining whether one is confident in the occurrence of the respective fault. Hence, we obtain the logical formula

$$\forall x \left((f_l(\mathrm{EnvSpec}(x)) > \mathrm{thr}_{\mathrm{score}}) \rightarrow P(x, c_l) \right), \tag{7}$$

with the respective fault label l and predicate P for fault classification. With this formula we aim to mimic the way an expert would analyze a vibration signal and identify a certain kind of fault. It will serve as an input to our knowledge base defined in the next section. To unclutter notation we will omit $\mathrm{EnvSpec}(x)$ and directly write $f_l(x)$ for the rest of the paper.

Notice, that even though we describe a specific function for bearing fault diagnostics on vibration data, any function f can be used to create the axiom in Eq. (7). Therefore, arbitrary knowledge can be induced and the axiom works for different classification settings.

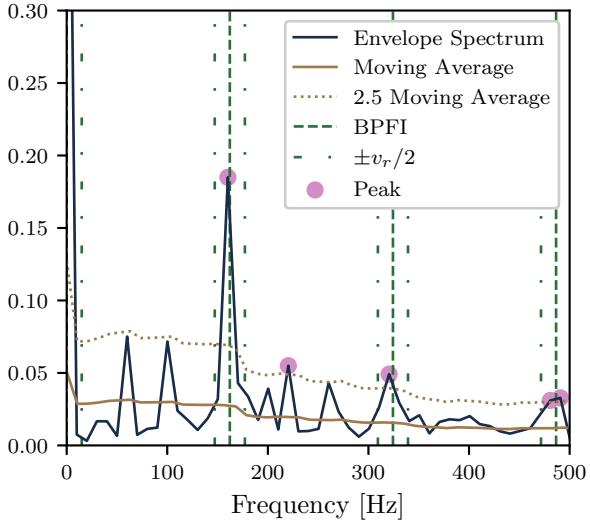

Figure 1. Visualization of the components of the f_{IR} score function for an exemplary signal with an inner race fault (IR). The fault scores of this signal are $f_{\mathrm{OR}} = 0.30$, $f_{\mathrm{IR}} = 0.83$ and $f_{\mathrm{B}} = 0.16$. It would therefore be classified as IR by axiom Eq. (7).

3.2. Identifying Normal Bearings

Healthy bearings can be identified more easily. In comparison to the impulsive signal of a faulty bearing the signal of a normal bearing is rather smooth, see Fig. 2. This difference can be measured by the kurtosis, which is defined as the

fourth standardized moment

$$\mathrm{Kurt}(x) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s}\right)^4, \tag{8}$$

with mean $\bar{x}$ and standard deviation s of the signal x (Randall & Antoni, 2011). We use this knowledge as a further input to our knowledge base and write the corresponding formula as

$$\forall x\,((\mathrm{Kurt}(x) > \mathrm{thr}_{\mathrm{kurt}}) \to P(x, c_{\mathrm{N}})), \tag{9}$$

where we set the threshold $\mathrm{thr}_{\mathrm{kurt}} = 0.1$.

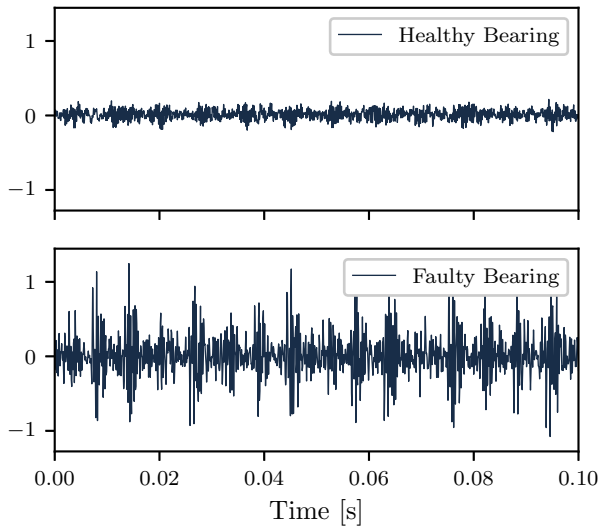

Figure 2. Vibration signal of a healthy and a faulty bearing with a kurtosis of -0.19 and 2.20 respectively.

3.3. Knowledge Base for the LTN

Using these means of diagnosing bearing signals we create a knowledge base as per the definitions for LTNs in Sec. 2.

Domains $\mathcal{D}$:

$$\text{data}, \text{labels}$$

Variables $\mathcal{X}$:

$$x, x_{\mathrm{N}}, x_{\mathrm{OR}}, x_{\mathrm{IR}}, x_{\mathrm{B}}$$
$$\mathbf{D}(x) = \mathbf{D}(x_{\mathrm{N}}) = \mathbf{D}(x_{\mathrm{OR}}) = \mathbf{D}(x_{\mathrm{IR}}) = \mathbf{D}(x_{\mathrm{B}}) = \text{data}$$

Constants $\mathcal{C}$:

$$c_{\mathrm{N}}, c_{\mathrm{OR}}, c_{\mathrm{IR}}, c_{\mathrm{B}}$$
$$\mathbf{D}(c_{\mathrm{N}}) = \mathbf{D}(c_{\mathrm{OR}}) = \mathbf{D}(c_{\mathrm{IR}}) = \mathbf{D}(c_{\mathrm{B}}) = \text{labels}$$

Predicates $\mathcal{P}$:

$$P(x, c)$$
$$\mathbf{D}_{\mathrm{in}}(P) = \text{data}, \text{labels}$$

Axioms $\mathcal{K}$:

Data axioms $\mathcal{K}_{\mathrm{D}}$:

$$\forall x_{\mathrm{N}}\, P(x_{\mathrm{N}}, c_{\mathrm{N}})$$
$$\forall x_{\mathrm{OR}}\, P(x_{\mathrm{OR}}, c_{\mathrm{OR}})$$
$$\forall x_{\mathrm{IR}}\, P(x_{\mathrm{IR}}, c_{\mathrm{IR}})$$
$$\forall x_{\mathrm{B}}\, P(x_{\mathrm{B}}, c_{\mathrm{B}})$$

Knowledge axioms $\mathcal{K}_{\mathrm{K}}$:

$$\begin{aligned}
&\forall x\,((\mathrm{Kurt}(x) > \mathrm{thr}_{\mathrm{kurt}}) \to P(x, c_{\mathrm{N}}))\\
&\forall x\,((f_{\mathrm{OR}}(x) > \mathrm{thr}_{\mathrm{score}}) \to P(x, c_{\mathrm{OR}}))\\
&\forall x\,((f_{\mathrm{IR}}(x) > \mathrm{thr}_{\mathrm{score}}) \to P(x, c_{\mathrm{IR}}))\\
&\forall x\,((f_{\mathrm{B}}(x) > \mathrm{thr}_{\mathrm{score}}) \to P(x, c_{\mathrm{B}}))
\end{aligned} \tag{10}$$

Groundings $\mathcal{G}$:

$$\mathcal{G}(\text{data}) = \mathbb{R}^n,\ \mathcal{G}(\text{labels}) = \mathbb{N}^4$$
$$\mathcal{G}(x_{\mathrm{N}}) \in \mathbb{R}^{m_{\mathrm{N}} \times n},\ \mathcal{G}(x_{\mathrm{OR}}) \in \mathbb{R}^{m_{\mathrm{OR}} \times n}$$
$$\mathcal{G}(x_{\mathrm{IR}}) \in \mathbb{R}^{m_{\mathrm{IR}} \times n},\ \mathcal{G}(x_{\mathrm{B}}) \in \mathbb{R}^{m_{\mathrm{B}} \times n}$$
$$\mathcal{G}(x) \in \mathbb{R}^{(m_{\mathrm{N}}+m_{\mathrm{OR}}+m_{\mathrm{IR}}+m_{\mathrm{B}}) \times n}$$
$$\mathcal{G}(c_{\mathrm{N}}) = [1,0,0,0],\ \mathcal{G}(c_{\mathrm{OR}}) = [0,1,0,0]$$
$$\mathcal{G}(c_{\mathrm{IR}}) = [0,0,1,0],\ \mathcal{G}(c_{\mathrm{B}}) = [0,0,0,1]$$
$$\mathcal{G}(P|\theta) : x, c \mapsto c^T \cdot \mathrm{softmax}(\mathrm{CLF}_\theta(x))$$
$$\mathrm{thr}_{\mathrm{score}} = 0.49,\ \mathrm{thr}_{\mathrm{kurt}} = 0.1$$
$$\mathrm{Kurt}(\mathcal{G}(x)) \in \mathbb{R}$$
$$\forall l \in \{\mathrm{OR}, \mathrm{IR}, \mathrm{B}\}\colon f_l(\mathcal{G}(x)) \in [0,1]$$

The subscripts of the variables in $\mathcal{X}$ indicate, that only the fraction, where data has the same label as the subscript, is used. If there is no subscript, then all data is used. This also explains grounding of the different variables onto different dimensional real spaces.

Notice the grounding of predicate P in some classifier CLF, which in turn relies on the parameter configuration θ. The softmax function ensures that a truth value between 0 and 1 is returned by this grounding. As mentioned in Sec. 2 this classifier can be any kind of NN and the parameters of the NN are optimized so that the satisfiability of the axioms $\mathcal{K}$ is maximized.

The knowledge base is specific to our case study of bearing fault diagnosis. But, the general formulation of the whole knowledge base and specifically the knowledge axioms $\mathcal{K}_{\mathrm{K}}$ make it possible to apply this approach to any other problem setting in the fault diagnosis domain.

3.4. Weights for Axiom Groups

Of course, our end goal is not to satisfy the knowledge base developed in Sec. 3.3, but to achieve the highest accuracy on some test set. To improve the performance of the underlying classifier CLF we optimize the satisfiability of the knowledge base, which consists of two kinds of axioms, namely data axioms $\mathcal{K}_\mathrm{D}$ and knowledge axioms $\mathcal{K}_\mathrm{K}$. Data axioms rely only on the provided labeled data and are always true in terms of classification accuracy. In contrast the knowledge axioms consist of the heuristics formulated in Sec. 3.1 and 3.2, which are not always true but are based on physical certainties.

If we contemplate on how we as humans learn new things, it mainly starts off with somebody explaining to us how something should work in theory. After understanding this, we go out into the world to apply this knowledge and quickly realize that the theory is based on assumptions, which don't always hold in reality. Therefore we update our beliefs based on new experiences and observations. According to (Fitts & Posner, 1967) this is reflected in the learning phases "cognitive", "associative" and "autonomous". We want to apply this way of learning to LTNs.

By weighting the importance of the satisfiability for certain axiom groups differently during the course of training, we can mimic exactly this process. To incorporate this idea we introduce the idea of weighted axiom groups. For this we need to rewrite the satisfiability optimization problem from Eq. (2) to

$$\theta^* = \underset{\theta\in\Theta}{\operatorname{argmax}} \sum_{k=1}^{K} w_k^{(e)} \underset{\phi\in\mathcal{K}_k}{\operatorname{SatAgg}} \mathcal{G}(\phi|\theta), \tag{11}$$

with K weights $w_k^{(e)}$ and $\sum_k^K w_k^{(e)} = 1$, which are dependent on the current training epoch e and a set of K axiom groups $\mathcal{K}_k$. For our purposes we group axioms into knowledge and data axioms, but other more granular groupings are also possible. We call a weight schedule the setting of weights w_k per axiom group $\mathcal{K}_k$ as a function of epochs e and propose the two weight schedules first knowledge then data (FKTD) and up and down (UAD). The corresponding weight schedules are given in Table 1.

Table 1. Definition of the weight schedules UAD and FKTD for data axioms (D) and knowledge axioms (K) along the training process of N epochs.

Epochs (e)	**UAD**	**FKTD**
$e \leq \frac{1}{4}N$	$w_\mathrm{D} = 0,\ w_\mathrm{K} = 1$	$w_\mathrm{D} = 0,\ w_\mathrm{K} = 1$
$\frac{1}{4}N < e \leq \frac{1}{2}N$	$w_\mathrm{D} = \frac{1}{4},\ w_\mathrm{K} = \frac{3}{4}$	$w_\mathrm{D} = 0,\ w_\mathrm{K} = 1$
$\frac{1}{2}N < e \leq \frac{3}{4}N$	$w_\mathrm{D} = \frac{3}{4},\ w_\mathrm{K} = \frac{1}{4}$	$w_\mathrm{D} = 1,\ w_\mathrm{K} = 0$
$e > \frac{3}{4}N$	$w_\mathrm{D} = 1,\ w_\mathrm{K} = 0$	$w_\mathrm{D} = 1,\ w_\mathrm{K} = 0$

3.5. Extending the Feature Space

We can also incorporate our knowledge without using the Real Logic machinery of LTNs by extending the feature space with inputs describing this knowledge (Chao et al., 2022). Hereby we include the score functions ($f_\mathrm{OR}, f_\mathrm{IR}, f_\mathrm{B}$) and the Kurtosis of the signals as additional features in the data and train the network based on this enhanced data set. In the following we will use this approach in combination with LTNs and independently as a further baseline on top of networks without any induced knowledge.

4. Experimental Setup

4.1. Data

We used two bearing fault data sets with different attributes and constant shaft speed for evaluating the proposed approach.

4.1.1. CWRU Data Set

The Case Western University (CWRU) bearing data set[2] is a widely used benchmark data set for bearing fault classification (Neupane & Seok, 2020). A thorough analysis of the data is given by (Smith & Randall, 2015). The data set consists of vibration signal measurements of the four different classes healthy/normal (N), inner race fault (IR), outer race fault (OR) and rolling (ball) element fault (B). The faults have the different sizes 0.07 in, 0.14 in, 0.21 in or 0.28 in and are measured at different positions. The machine was run on constant shaft speeds of 1730, 1750, 1772 and 1792 rotations per minute (RPM) for different motor loads. For our experiments we will use the drive end data sampled at a frequency of 12 kHz and the baseline data with a frequency of 48 kHz. For the drive end data a SKF 6205-2RS JEM deep groove ball bearing was used, which has the ball pass frequencies $\mathrm{BPFI} = 5.415$, $\mathrm{BPFO} = 3.585$ and $\mathrm{BSF} = 2.357$, that we use to create the score from Sec. 3.1. We will exclude the 0.28 in fault data, because a different bearing is used for which neither ball pass frequencies nor measurements for calculating these are available.

4.1.2. MFPT Data Set

As a second data set we used the data provided by the Society for Machinery Failure Prevention Technology[3] (MFPT). MFPT provides us with data for healthy bearings and outer race fault conditions with 98 kHz sample frequency and a constant shaft speed of 1560 RPM. Additionally, the data includes inner and outer race fault conditions with 48 kHz sample frequency with the same shaft speed. The test rig used for data generation is equipped with a NICE bearing with the ball pass frequencies $\mathrm{BPFI} = 4.755$ and $\mathrm{BPFO} = 3.245$.

[2] https://engineering.case.edu/bearingdatacenter, accessed 02/04/2022
[3] https://www.mfpt.org/fault-data-sets/, accessed: 02/04/2022

4.2. Data Preprocessing

For the CWRU data set we excluded all faults with size 0.28 in because no information about the used bearing is available. From the signal data we extracted time series of length 1200. For the faults, which were sampled at a frequency of 12 kHz, this amounts to an observation time of 0.1 seconds. The baseline data of non faulty bearings was sampled at 48 kHz, therefore we downsampled the data to 12 kHz, so that we have the same time interpretation.

The MFPT data is sampled at different frequencies. The smallest fault frequency is 48 kHz, therefore we downsampled all other signals to this sample rate. Afterwards we extracted time series of length 4800 to also have an observation time of 0.1 seconds.

After the extraction of the time series the scores for different classes and the kurtosis were calculated based on these segments. Since the MFPT data set does not include any ball faults, we excluded the score calculation and the respective axiom from the knowledge base. Depending on whether or not we used feature space extension, we included or excluded the scores and kurtosis from the data used for training and testing of the networks.

4.3. Model Configurations

We used two different kinds of NNs as inputs to the LTN and as baselines for comparison: A Multi Layer Perceptron (MLP) with three fully connected hidden layers, which consist of 32, 32 and 16 nodes, and a CNN consisting of one convolutional layer with kernel size 9 and a fully connected hidden layer with 64 nodes. Both use the exponential linear unit (elu) activation function (Clevert, Unterthiner, & Hochreiter, 2016) with $\alpha = 1$, the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.001 and are trained with batches of 32 for 100 epochs. The baseline NNs use the categorical cross entropy loss function. Combining these two NNs with the different means of inducing knowledge and weight schedules, we obtain 16 different models listed in Table 2.

4.4. Experiments

All experiments were run and averaged over the same 7 random seeds. Each model was trained on 10 %, 30 %, 50 %, 70 % and 90 % of the data and test accuracies were calculated on an unseen 10 % of the data. The underlying NNs of the LTNs and for the pure DL approaches were completely identical and were initialized with the same random seed. Thereby both methods suffered or profited from equal initial weight settings likewise. All code was implemented in Python 3 using the Tensorflow and Logic Tensor Networks packages.

5. RESULTS

The results of all experiments for different model configurations for the CWRU data set and the MFPT data set are given in Table 3 and Table 4 respectively. We continue by evaluating the knowledge and the proposed models independently. For each section we discuss both the performance on the CWRU data set and on the one provided by MFPT.

5.1. Pure Knowledge Based Classification

First, we evaluate how the proposed knowledge axioms perform independent of DL on both data sets. We do this by dividing the number of correctly identified labels of a class through the number of all labels of a class. The labels were identified by using the knowledge axioms $\mathcal{K}_{\mathrm{K}}$ (10) from the proposed knowledge base. The results are shown in Table 5. We see that the proposed knowledge axioms work very well on MFPT, with an average value of 0.9 and less so on the CWRU data with an average value of 0.62. From these values we can follow that the MFPT data is very well behaved in terms of characteristic signals for different fault types and that these are not significantly masked by other influences. For the CWRU data on the other hand, there appear to be multiple other influences on the signals, which corresponds to one of the key findings by (Smith & Randall, 2015). Additionally, we see that faults of type B are very hard to classify with only the proposed scoring function.

For MFPT the purely knowledge based approach even outperformed all analyzed neural networks regardless of the amount of data used, which further underlines the validity of the proposed logical axioms for clear unmasked bearing fault data. On the other hand, when using the CWRU data set the purely knowledge based classification is outperformed by all analysed models if 30 % or more of the data was used.

5.2. Real Logic in the Loss Function

Next, we evaluate the performance of LTNs (LtnMlpNoF & LtnCnnNoF) given the proposed knowledge base from Sec. 3.3 against the pure DL approaches with a MLP or CNN (NnMlpF & NnCnnF) .

On the CWRU data set LTNs outperformed their respective NN counterparts along all data fractions. The difference is especially noticeable for the MLP, where the average test accuracy of LTNs was increased by 0.02.

When using the MFPT data set no clear difference in performance can be read from the experimental data. Depending on the data fraction either LTNs or NNs performed better.

5.3. Extending the Feature Space

Now, we extend the feature spaces for both LTNs (LtnMlpNoT & LtnCnnNoT) and NNs (NnMlpT & NnCnnT). With

Table 2. Names and configurations of the models used for experiments.

Name	Model	Neural Network	Weight Schedule	Extend Features Space
LtnMlpNoT	LTN	MLP	None (NO)	True
LtnMlpNoF	LTN	MLP	None (NO)	False
LtnMlpUadT	LTN	MLP	Up And Down (UAD)	True
LtnMlpUadF	LTN	MLP	Up And Down (UAD)	False
LtnMlpFktdT	LTN	MLP	First Knowledge Then Data (FKTD)	True
LtnMlpFktdF	LTN	MLP	First Knowledge Then Data (FKTD)	False
LtnCnnNoT	LTN	CNN	None (NO)	True
LtnCnnNoF	LTN	CNN	None (NO)	False
LtnCnnUadT	LTN	CNN	Up And Down (UAD)	True
LtnCnnUadF	LTN	CNN	Up And Down (UAD)	False
LtnCnnFktdT	LTN	CNN	First Knowledge Then Data (FKTD)	True
LtnCnnFktdF	LTN	CNN	First Knowledge Then Data (FKTD)	False
NnMlpT	NN	MLP	-	True
NnMlpF	NN	MLP	-	False
NnCnnT	NN	CNN	-	True
NnCnnF	NN	CNN	-	False

Table 3. Test accuracies for different models and data fractions used for training on the CWRU data set. The standard deviation is given in parenthesis. Bold entries mark the best performing model for the respective data fraction.

Model	10 % of Data	30 % of Data	50 % of Data	70 % of Data	90 % of Data
LtnMlpNoT	0.595 (0.032)	0.731 (0.016)	0.8 (0.021)	0.84 (0.014)	0.874 (0.015)
LtnMlpNoF	0.526 (0.026)	0.678 (0.015)	0.74 (0.015)	0.782 (0.012)	0.799 (0.016)
LtnMlpUadT	0.598 (0.028)	0.748 (0.021)	0.81 (0.014)	0.859 (0.019)	0.891 (0.013)
LtnMlpUadF	0.533 (0.042)	0.675 (0.018)	0.734 (0.019)	0.777 (0.008)	0.807 (0.016)
LtnMlpFktdT	0.597 (0.035)	0.74 (0.016)	0.799 (0.019)	0.845 (0.016)	0.868 (0.02)
LtnMlpFktdF	0.552 (0.022)	0.668 (0.017)	0.727 (0.019)	0.773 (0.016)	0.8 (0.012)
LtnCnnNoT	0.581 (0.018)	0.778 (0.011)	0.855 (0.014)	0.896 (0.011)	0.925 (0.012)
LtnCnnNoF	0.545 (0.027)	0.727 (0.022)	0.818 (0.011)	0.858 (0.017)	0.89 (0.019)
LtnCnnUadT	0.635 (0.027)	0.82 (0.012)	**0.89 (0.012)**	**0.917 (0.01)**	**0.934 (0.011)**
LtnCnnUadF	0.584 (0.029)	0.747 (0.017)	0.814 (0.017)	0.863 (0.009)	0.876 (0.015)
LtnCnnFktdT	**0.641 (0.024)**	**0.826 (0.016)**	0.886 (0.018)	0.904 (0.018)	0.915 (0.023)
LtnCnnFktdF	0.591 (0.023)	0.759 (0.02)	0.824 (0.01)	0.856 (0.015)	0.882 (0.012)
NnMlpT	0.551 (0.025)	0.69 (0.027)	0.762 (0.011)	0.807 (0.016)	0.847 (0.009)
NnMlpF	0.496 (0.031)	0.649 (0.028)	0.721 (0.022)	0.762 (0.01)	0.787 (0.011)
NnCnnT	0.535 (0.02)	0.738 (0.02)	0.851 (0.017)	0.902 (0.013)	0.933 (0.012)
NnCnnF	0.477 (0.036)	0.72 (0.021)	0.802 (0.017)	0.854 (0.008)	0.884 (0.011)

feature space extension, we see a similar picture as in Sec. 5.2, but the differences are more pronounced.

For the CWRU data set LtnMlpNoT outperformed its counterpart NnMlpT by between 0.03 and 0.04 points, where outperformance is especially high for smaller data fractions and lower when using more data. When examining LtnCnnNoT and NNCnnT we see a similar dynamic. The LTN has a higher accuracy for data fractions 0.1 and 0.3 but the gap is closed for larger data fractions, where the NN even outperformed the LTN slightly.

Again, MFPT data does not show as much of a difference between the models. We see, that LtnMlpNoT outperformed NnMlpNoT marginally along all data fractions. For the CNN based model on the other hand NnCnnT performed better on less training data. Starting with a data fraction of 0.5 the LtnCnnNoT increased its accuracy in comparison to the NN and had a 0.04 higher accuracy when all data was used.

In general, just by extending the feature space the performance of all models increased significantly. For the CWRU data the inclusion of our proposed knowledge features into the benchmark NN increased test accuracy by 0.04 on average and for the MFPT data by 0.02.

5.4. Weight Schedules for Axioms Groups

Finally, we compare different weight schedules for data and knowledge axioms in LTNs (LtnMlpUadT, LtnMlpFktdT & LtnCnnUadT, LtnCnnFktdT).

By using the weight schedule UAD the performance of LtnMlpNoT was further improved by an average of 0.01 on the CWRU data set. FKTD on the other hand did not seem to increase performance. The same can be said when the fea-

Table 4. Test accuracies for different models and data fractions used for training on the MFPT data set. The standard deviation is given in parenthesis. Bold entries mark the best performing model for the respective data fraction.

Model	10 % of Data	30 % of Data	50 % of Data	70 % of Data	90 % of Data
LtnMlpNoT	0.547 (0.032)	0.63 (0.031)	0.682 (0.029)	0.732 (0.021)	0.772 (0.036)
LtnMlpNoF	0.5 (0.019)	0.596 (0.036)	0.636 (0.033)	0.698 (0.032)	0.741 (0.022)
LtnMlpUadT	0.548 (0.03)	0.63 (0.03)	0.679 (0.031)	0.719 (0.032)	0.777 (0.027)
LtnMlpUadF	0.521 (0.042)	0.58 (0.028)	0.653 (0.029)	0.698 (0.017)	0.733 (0.029)
LtnMlpFktdT	**0.552 (0.045)**	**0.649 (0.032)**	0.688 (0.031)	0.723 (0.023)	0.781 (0.047)
LtnMlpFktdF	0.494 (0.019)	0.583 (0.024)	0.645 (0.03)	0.697 (0.018)	0.74 (0.029)
LtnCnnNoT	0.475 (0.054)	0.613 (0.056)	**0.692 (0.02)**	**0.751 (0.025)**	**0.802 (0.033)**
LtnCnnNoF	0.461 (0.071)	0.551 (0.043)	0.668 (0.034)	0.725 (0.036)	0.778 (0.056)
LtnCnnUadT	0.491 (0.07)	0.537 (0.063)	0.631 (0.029)	0.634 (0.109)	0.64 (0.179)
LtnCnnUadF	0.469 (0.06)	0.531 (0.048)	0.534 (0.056)	0.569 (0.099)	0.601 (0.116)
LtnCnnFktdT	0.479 (0.047)	0.56 (0.039)	0.616 (0.046)	0.648 (0.081)	0.664 (0.139)
LtnCnnFktdF	0.49 (0.028)	0.477 (0.056)	0.519 (0.083)	0.534 (0.054)	0.571 (0.089)
NnMlpT	0.542 (0.041)	0.618 (0.024)	0.677 (0.027)	0.706 (0.028)	0.756 (0.036)
NnMlpF	0.512 (0.03)	0.58 (0.028)	0.643 (0.023)	0.689 (0.034)	0.739 (0.033)
NnCnnT	0.502 (0.059)	0.615 (0.032)	0.684 (0.019)	0.72 (0.027)	0.757 (0.036)
NnCnnF	0.503 (0.055)	0.604 (0.029)	0.657 (0.025)	0.692 (0.021)	0.752 (0.031)

Table 5. Fraction of correctly classified data when adhering to the underlying axioms of the proposed knowledge base per class and data set.

Underlying Axiom	CWRU	MFPT
$\forall x\,((\mathrm{Kurt}(x) > \mathrm{thr}_{\mathrm{Kurt}}) \rightarrow P(x, c_{\mathrm{N}}))$	0.91	0.88
$\forall x\,((f_{\mathrm{OR}}(x) > \mathrm{thr}_{\mathrm{score}}) \rightarrow P(x, c_{\mathrm{OR}}))$	0.49	1.00
$\forall x\,((f_{\mathrm{IR}}(x) > \mathrm{thr}_{\mathrm{score}}) \rightarrow P(x, c_{\mathrm{IR}}))$	0.76	0.83
$\forall x\,((f_{\mathrm{B}}(x) > \mathrm{thr}_{\mathrm{score}}) \rightarrow P(x, c_{\mathrm{B}}))$	0.32	-

ture space was not extended. Without feature extension we only see an increase in the standard deviation of results, especially for the weight schedule FKTD. For LTNs based on the CNN the addition of weighted axioms shows stronger impact on accuracy. Especially the lower data fractions profited from the weights, but this improvement gradually levels out when using more training data. This is true for both extended and not extended feature spaces. Again, the UAD weight schedule performed best and increased the performance by up to 0.05.

The MFPT data gives us a rather different picture. For MLPs there are close to no performance gains for both extended and not extended features spaces when weighted axioms were used. CNNs even show a drastic deterioration of model performance when weights were included. This observation is paired with a sharp increase of the standard deviation to a value of up to 0.18 over the seven analyzed runs, which shows that the performance was highly dependent on the random seed and data composition. Fig. 3 shows that the LTN seems to have gotten stuck in a local optimum when trained on one of the low performing seeds. A similar training progress was observed for all other low performing seeds. It seems, that the concentration on the knowledge axioms at the beginning of the training found a local optimum, which is hard to escape.

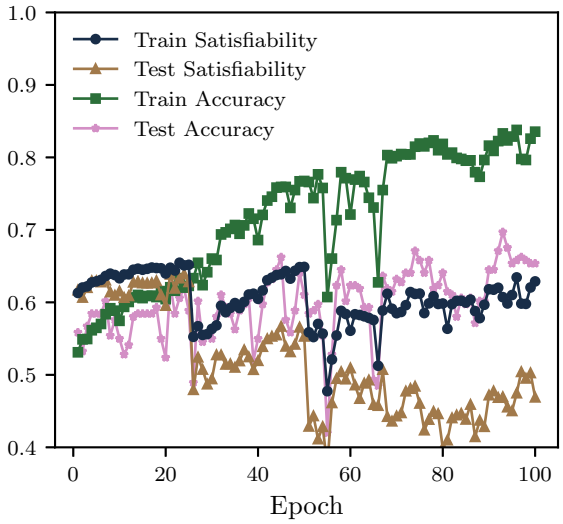

Figure 3. Training of LtnCnnUadT for 100 epochs on 90 % of the MFPT data set for a bad performing random seed. The train satisfiability seems to be stuck in a local optimum.

5.5. Discussion

Although the improvement of each single aspect (real logic in the loss function, extending the feature space and weight schedules) of LTNs in comparison to the pure DL approaches does not seem large, the combination of these show a significant increase in accuracy for LTNs. We identify the optimal models LtnMlpUadT and LtnCnnUadT on the CWRU data set with an increase in accuracy of up to 0.10 for small amounts of training data in comparison to the feature ex-

tended NNs. On the MFPT data set the best performing models where LtnMlpNoT and LtnCnnNoT with a maximum performance increase of 0.04. Note that compared to the networks without the use of our proposed knowledge features the performance increased by up to 0.15.

Especially on the CWRU data we see, that the gain decreases with the amount of data used. This aligns with one of the main motivations for hybrid models, namely the reduction of necessary labeled data and further underlines the strength of purely data driven DL, when enough data is available. Weight schedules, which we motivated by the way humans learn, also increased the test accuracy. Notably, the strongest improvement could be observed when focusing on knowledge first and putting more and more weight on the contribution given by the data.

We also observe a surprising difference between the performance increase of the analyzed data sets. Even though the induced knowledge was better in terms of per class accuracy for the MFPT then for the CWRU data set, the gains in test accuracy were mainly seen for the CWRU data. An explanation for this can maybe be found in the very bad performance of weighted schedules on the MFPT data set. It seems like the focus on the knowledge brought the LTN to a local optimum, which is hard to get out of. Maybe the nearly perfect knowledge limits the search space for optimal parameter configurations to a degree, that it can't be optimized through more data. Or the underlying models where not appropriate for the data set and the weight schedules.

6. Conclusion

We proposed an application of LTNs for fault diagnostics in the special case of bearing faults with constant shaft speed. To this end, we introduced a scoring function based on physical attributes of the bearings as a representation of expert knowledge and extended the LTN framework by weight schedules. Because of the general formalization of the LTN knowledge base the method can readily be applied to various other diagnostic tasks with domain-specific adjustments of the scoring function. Our proposed approach was evaluated on two different data sets and showed an increase in test accuracy in comparison to the benchmark NNs, which were, in some experiments, also enhanced with the created knowledge features. We demonstrated that inducing knowledge increased performance, particularly when less data was available and can also be used to gain a better understanding of the data set, e.g., getting an indication on whether fault signals are masked by other influences. However, we also showed that the performance gains of LTNs are not completely intuitive and that LTNs with weight schedules can get stuck in local minima during training for certain data compositions.

The findings from this work confirm the hypothesis that a combination of DL with expert knowledge in the domain of fault diagnosis is a promising direction, and motivate further studies and future research. Most notably, the proposed weight schedules need to be analyzed in more detail, since they have a positive impact on the training performance in most situations, but cause a stagnation of the learning curve in other situations. In this context, the interplay between different NN architectures, weight schedules, axioms and data sets needs to be studied further to leverage the full potential of LTNs. Also a factor that could further strengthen the performance is the utilization of more complex logical formulas and logical reasoning in the loss function, which is a feature of logic-based knowledge representation but has not yet been exploited in this research. Currently the scoring function represents physical knowledge from the domain of bearing fault diagnosis, but it can be reconfigured to incarnate physical knowledge from other domains as well. A more abstract representation with clear pathways to instantiate the scoring function in different domains would also be part of future research.

References

Arinez, J. F., Chang, Q., Gao, R. X., Xu, C., & Zhang, J. (2020). Artificial intelligence in advanced manufacturing: Current status and future outlook. *Journal of Manufacturing Science and Engineering*, *142*(11).

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., ... others (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, *58*, 82–115.

Badreddine, S., Garcez, A. d., Serafini, L., & Spranger, M. (2022). Logic tensor networks. *Artificial Intelligence*, *303*.

Bianchi, F., & Hitzler, P. (2019). On the capabilities of logic tensor networks for deductive reasoning. In *Aaai spring symposium: combining machine learning with knowledge engineering.*

Bonnardot, F., Randall, R., Antoni, J., & Guillet, F. (2004). Enhanced unsupervised noise cancellation using angular resampling for planetary bearing fault diagnosis. *International journal of acoustics and vibration*, *9*(2), 51–60.

Chao, M. A., Kulkarni, C., Goebel, K., & Fink, O. (2022). Fusing physics-based and deep learning models for prognostics. *Reliability Engineering & System Safety*, *217*.

Clevert, D., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus). In *4th international conference on learning representations, conference track proceedings.*

Donadello, I., Serafini, L., & Garcez, A. d. (2017). Logic tensor networks for semantic image interpretation. In *Proceedings of the twenty-sixth international joint con-*

ference on artificial intelligence, IJCAI-17 (pp. 1596–1602).

Fink, O., Wang, Q., Svensen, M., Dersin, P., Lee, W.-J., & Ducoffe, M. (2020). Potential, challenges and future directions for deep learning in prognostics and health management applications. *Engineering Applications of Artificial Intelligence*, *92*.

Fitts, P. M., & Posner, M. I. (1967). *Human performance*. Brooks/Cole Publishing Company.

Garcez, A. d., & Lamb, L. C. (2020). Neurosymbolic ai: the 3rd wave. *arXiv preprint arXiv:2012.05876*.

Hájek, P. (2013). *Metamathematics of fuzzy logic* (Vol. 4). Springer Science & Business Media.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd international conference on learning representations, conference track proceedings.*

Liu, R., Yang, B., Zio, E., & Chen, X. (2018). Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, *108*, 33–47.

Marcus, G. (2020). The next decade in ai: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.

Neupane, D., & Seok, J. (2020). Bearing fault detection and diagnosis using case western reserve university dataset with deep learning approaches: A review. *IEEE Access*, *8*.

Randall, R. B., & Antoni, J. (2011). Rolling element bearing diagnostics—a tutorial. *Mechanical systems and signal processing*, *25*(2), 485–520.

Selcuk, S. (2017). Predictive maintenance, its implementation and latest trends. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *231*(9), 1670–1679.

Smith, W. A., & Randall, R. B. (2015). Rolling element bearing diagnostics using the case western reserve university data: A benchmark study. *Mechanical systems and signal processing*, *64*, 100–131.

Steenwinckel, B., De Paepe, D., Hautte, S. V., Heyvaert, P., Bentefrit, M., Moens, P., … others (2021). Flags: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning. *Future Generation Computer Systems*, *116*, 30–48.

Wang, Q., Taal, C., & Fink, O. (2021, 07). Integrating expert knowledge with domain adaptation for unsupervised fault diagnosis. *IEEE Transactions on Instrumentation and Measurement*.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213–237.

BIOGRAPHIES

Maximilian-Peter Radtke studied business mathematics at the University of Mannheim, Germany and graduated in 2018. After his studies he worked as a data science consultant in various industries for two and half years before returning to academia. Since 2021 he has been working at the Technische Hochschule Ingolstadt (THI) as part of AIMotion Bavaria and the research group AI applications for innovative production and logistic systems. His research interests include the combination of symbolic and sub-symbolic AI approaches and the incorporation of knowledge into deep learning in the area of fault diagnostics and prognostics.

Jürgen Bock is a computer scientist, who graduated as Diplom-Informatiker from Ulm University, Germany, and as Bachelor of Information Technology with Honours from Griffith University, Brisbane, Australia, in 2006. He began his research career at the FZI Research Center for Information Technology in Karlsruhe, Germany, and received his PhD from the Karlsruhe Institut of Technology (KIT) in 2012. After 2 years as post doc and team leader at the FZI, he joined the corporate research department of KUKA Robotics in Augsburg, Germany as developer and later leader of the team Smart Data and Infrastructure. In 2020 he joined the Technische Hochschule Ingolstadt (THI) as research professor in the area of AI applications in innovative production and logistics systems.