Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 232 (2024) 1388–1399

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Industry 4.0 and Smart Manufacturing

Taxonomy-Informed Neural Networks for Smart Manufacturing

Vagan Terziyan [a,*], Oleksandra Vitko [b]

[a] Faculty of Information Technology, University of Jyväskylä, 40014, Jyväskylä, Finland
[b] Department of Artificial Intelligence, Kharkiv National University of Radio Electronics, 61166, Kharkiv, Ukraine

Abstract

A neural network (NN) is known to be an efficient and learnable tool supporting decision-making processes particularly in Industry 4.0. The majority of NNs are data-driven and, therefore, depend on training data quantity and quality. The current trend in enhancing data-driven models with knowledge-based models promises to enable effective NNs with less data. So-called physics-informed NNs use additional knowledge from computational science to improve NN training. Quite much of the knowledge is available as logical constraints from domain ontologies, and NNs may benefit from using it. In this paper, we study the concept of Taxonomy-Informed NN (TINN), which combines data-driven training of NNs with ontological knowledge. We study different patterns of NN training with additional knowledge on class-subclass hierarchies and instance-class relationships with potential for federated learning. Our experiments show that additional knowledge, which influences TINNs' training process through the loss function at backpropagation, improves the quality of trained models.

Keywords: neural networks; machine learning; informed machine learning; physics-informed neural networks; taxonomy; Industry 4.0

1. Introduction

New challenges related to digitalization, automation and intellectualization of industrial processes make Industry 4.0 (also 5.0) and smart manufacturing the key adopters of recent discoveries in artificial intelligence (AI) in general and machine learning (ML) in particular [1], [2]. Current progress in ML is associated with data-driven modelling, deep learning and neural networks (NNs). ML, however, may benefit from also exploring other AI assets (knowledge and behavior) in addition to data to enable better models. Modern industry has already collected useful knowledge formulated in terms of computational science (scientific computing) by combining mathematics with physics and

* Corresponding author. *E-mail address:* vagan.terziyan@jyu.fi

1877-0509
10.1016/j.procs.2024.01.137

other disciplines into analytical and computational models. The current "big thing" in ML, deep learning and NNs training is updating the NNs' training algorithms with respect to available knowledge. Such "informed" ML [3] trend has already made a huge impact on developing efficient physics-informed NNs (PINNs) for a variety of industrial applications [4].

The majority of emergent informed ML models (particularly PINNs) combine NNs with knowledge in the form of differential or integral equations and other typical for scientific computing mathematical constraints. What is missing is the knowledge constraints formulated using the conceptual modelling terms (e.g., first-order logic) and available as domain ontologies (or particularly taxonomies). *Could knowledge from ontologies (taxonomies) be used to guide and improve NNs' training from data? Would potential taxonomy-informed NNs (TINNs) become a reasonable concept and architecture for the informed ML? How can TINNs be potentially useful for Industry 4.0 and smart manufacturing?* We formulated these questions as objectives of our study described in this paper. In the context of these objectives, we are going to show the similarities and differences between emergent PINNs and potential TINNs suggested in this paper.

We have contextualized our work within the broader landscape of Industry 4.0 (driven by automation and ML) and its evolution towards Industry 5.0 (with enhanced value of human knowledge). Our research converges at the juncture of cutting-edge NN methodologies and the imperative for intelligent decision-making in these transformative industrial paradigms. By bridging data-driven NNs with knowledge-based models, we empower these industries to harness ontological insights for enhanced decision-making even in scenarios with limited data. The concept of TINN that we explore finds its resonance in Industry 4.0's drive for more efficient and effective decision support systems. One potential application example could revolve around predictive maintenance within a smart manufacturing environment. Imagine a scenario where a manufacturing facility seeks to optimize maintenance schedules for critical machinery. By integrating TINNs with domain ontologies and historical sensor data, the facility could create a more accurate predictive maintenance model. This model would not only consider the machinery's data-driven health metrics but also leverage ontological knowledge to anticipate potential failure modes based on known relationships between component attributes. The resulting model would empower the facility to proactively schedule maintenance interventions, minimizing downtime and maximizing operational efficiency.

The rest of the paper is organized as follows: Section 2 describes a general NN model enhancement mechanism used in PINNs and similar informed ML models; Section 3 describes how to use class-subclass hierarchy with multiple inheritance in known taxonomies to improve training of NNs from data, i.e., enabling TINNs; Section 4 presents some motivation examples (scenarios and experiments) with simple TINNs; In Section 5, we present couple of potential industrial scenarios with TINNs, which utilize knowledge on instance-class relationships for diagnostics (both individually and group informed) of industrial assets; and we conclude in Section 6. Related work will be cited throughout the paper.

2. From Neural Networks to Physics-Informed Neural Networks

Habitual opposition of knowledge-based and data-driven modelling frameworks has recently changed to capable hybrids. Among current trends is the transformation of ML in general and NNs, in particular, into physics-informed ML and corresponding physics-informed NNs. We will briefly introduce these in this section.

2.1. A neural network and its general training schema

A very generic representation of a NN is shown in Fig. 1 (the part within the dashed line). If you consider a NN as a black box, then it, actually, represents a complex function $y = f(\vec{x}, \vec{\theta})$ (aka intended model), which takes a sample of data (a vector $\vec{x}$ of numeric attributes' values $x_1, x_2, \dots, x_n$ of the sample) as an input and outputs the target numeric value (regression case) or a label (classification case). Inside of such a black box, there are hyperparameters [5], which determine the structure (complexity) of the function (i.e., the number, the size and the type of hidden layers of the NN), and vector $\vec{\theta}$ of the parameters $\theta_1, \theta_2, \dots, \theta_k$ of that function, which are the NN's weights [6]. Learning NN means finding the values for both hyperparameters and parameters of the NN such that the resulting function will fit the best to the available (for training) data. Typically, the hyperparameters are chosen and managed semi-automatically with the support from data scientists and the parameters (weights) are discovered automatically using a backpropagation algorithm. Training by backpropagation is an iterative optimization algorithm (with roots from

neuroscience [7]), which is searching for a best-fit function (intended model with optimal $\vec{\theta}$). The algorithm starts with some arbitrary weights (initialization of NN [8]) and continuously updates these weights after processing groups (epochs or batches) of training data samples. At each iteration, the training algorithm measures the opposite to the (data-model) fitness function value (i.e., "loss", or "error", or "cost" function [9] computed by comparing the predicted value $\hat{y}((\vec{x})_i)$ by the NN and the actual known value y_i for each i-th training sample from the batch) and computes corresponding updates for the parameters (weights) according to the gradient descent schema aiming to get smaller loss at the next iteration. Therefore, by training a NN, we seek to minimize the loss function. One may see that such training makes NNs biased towards the training data [10] (including natural or adversarial noise within it). Therefore, quality and quantity of training data samples (in addition to various regularization tricks used during training to avoid overfitting [11]) matters a lot to get well performing (high prediction accuracy) and well generalized (capability to address unseen data) NNs.

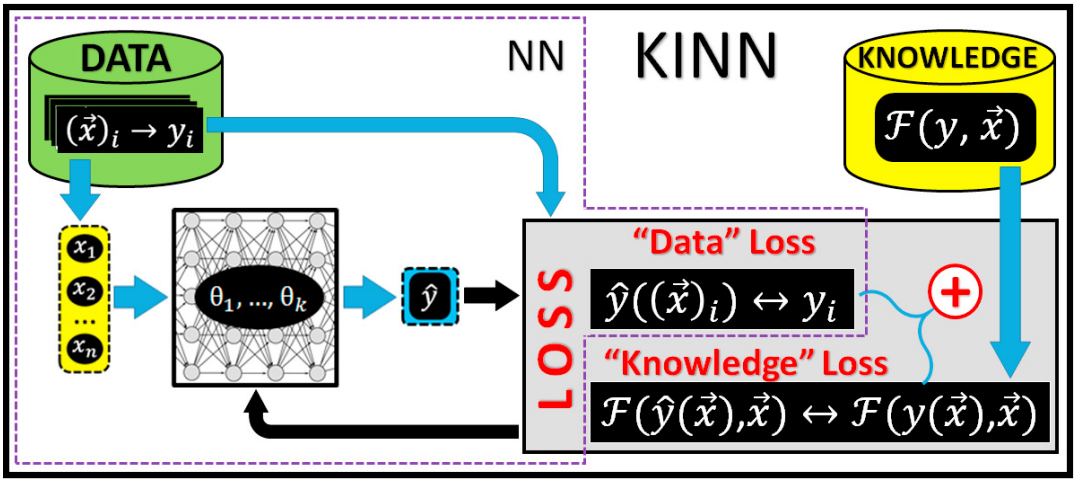


Fig. 1. Illustrating a generic NN schema (the part within the dashed line) and its extension towards KINN. One may see that KINN adds an extra summand to the NN's loss function, which, in addition to the traditional "Data" Loss, takes into account also the "Knowledge" Loss related to fitness of the current NN's predictions regarding the target function with the available knowledge on this function.

2.2. A knowledge-informed neural network and its training specifics

The recent trend in updating NN architectures towards better performance and less dependency solely on data is related to the so-called "informed ML" in general and "informed NNs" in particular [3]. Different variations of informed NNs use available prior knowledge of the target model in addition to training data. Such NNs can be integrated into one broad category under the common umbrella term "Knowledge-Informed NN" (KINN). The very generic architecture and training schema of KINN is shown in Fig. 1 as an update of traditional NN architecture. Let us assume that we have some prior knowledge of the target function in the form of some constraint $\mathcal{F}(y, \vec{x})$ from which it is impossible (or unfeasible) to directly derive the target function (model) $y = f(\vec{x})$. The objective of KINN training will be still discovering the target function from available data in the form $y = f(\vec{x}, \vec{\theta})$, however, with respect to the prior knowledge constraint $\mathcal{F}(y, \vec{x})$. This can be achieved by making a complex loss function in backpropagation training, which sums up (with defined coefficients of importance) the loss related to the training data fitness to the target model ("Data" loss) and the loss related to the prior knowledge ("Knowledge" loss) as shown in Fig 1.

The most popular subcategory of informed ML nowadays is physics-informed ML [12] with a variety of Industry 4.0 applications (see, e.g., [13], [14], [15]). This subcategory is based on PINNs [4] and it combines scientific computing with computational intelligence, aka a hybrid. The prior knowledge constraints in PINNs are usually differential or integral equations, which involve a target function. The balance between being "data-driven" and "knowledge-based" can be controlled by the corresponding adaptive weights in the loss function [16]. PINNs have been quickly adopted by industry (see, e.g., [17], [18]).

In addition to physics constraints, industrial applications of KINNs may also benefit from knowledge and fuzzy analytics on a variety of human factors [19], knowledge from chemistry [20], biology [21], etc.

3. Adding Awareness of Class-Subclass Hierarchies to Neural Networks

The success of current KINNs implementations shows that additional knowledge constraints from computational modelling analytics may improve traditional data-driven modelling by NNs (e.g., in PINNs). However, would similar effects take place if we use constraints from conceptual modelling frameworks (e.g., ontology engineering [22])? Can we augment ML in general and NNs in particular by conceptual models and get benefits from the explainability of these models [23]? In this section, we will start with simple taxonomies based on class-subclass hierarchies and multiple inheritance [24] to add taxonomy awareness to NNs, i.e., enabling TINNs.

An approach to adding taxonomy awareness to NNs is demonstrated in Fig.2. Assume that we have a simple taxonomy, which can be defined using conceptual modelling terms from RDF/RDFS (N3 notation; https://www.w3.org/TeamSubmission/n3/) language as follows:

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
:DOMAIN a rdfs:Class . :CLASS1 a rdfs:Class ; rdfs:subClassOf :DOMAIN .
:CLASS2 a rdfs:Class ; rdfs:subClassOf :DOMAIN .
:CLASS1.1 a rdfs:Class ; rdfs:subClassOf :CLASS1 . :CLASS1.2 a rdfs:Class ; rdfs:subClassOf :CLASS1 .
:CLASS1.3 a rdfs:Class ; rdfs:subClassOf :CLASS1 .
:CLASS2.1 a rdfs:Class ; rdfs:subClassOf :CLASS2 . :CLASS2.2 a rdfs:Class ; rdfs:subClassOf :CLASS2 .
:CLASS2.3 a rdfs:Class ; rdfs:subClassOf :CLASS2 . :CLASS2.4 a rdfs:Class ; rdfs:subClassOf :CLASS2 . …

Fig. 2a provides a visual representation of this taxonomy.

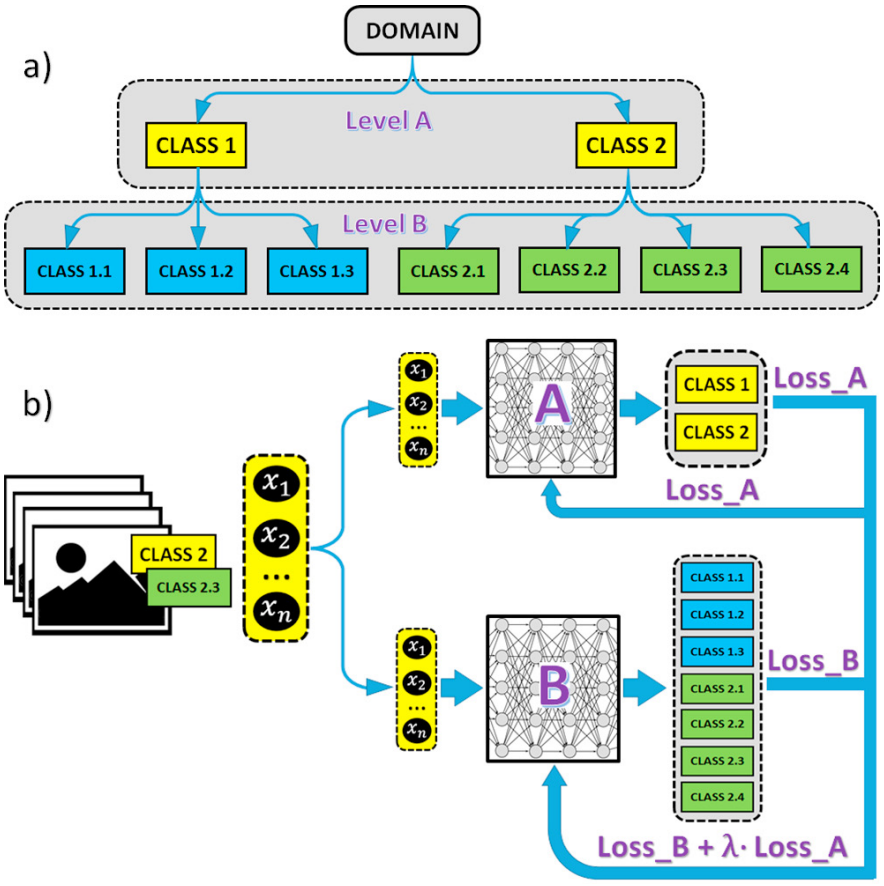


Fig. 2. The training schema of TINN originated from simple taxonomy awareness is illustrated: (a) example of a taxonomy (class-subclass hierarchy of classes in the taxonomy); (b) the NN classifier B, which is currently being trained to label samples into 7 classes (lower level of the taxonomy) benefits from the NN classifier B trained to label the same instances into 2 classes of the upper level of the taxonomy. Classifier A contributes to the performance of classifier B by sharing part of its loss regarding the same input sample (e.g., some input training sample with two true labels, CLASS2 and CLASS2.3).

Assume that, according to the scenario in Fig. 2b, we want to train NN ("B") to label DOMAIN instances into seven classes {CLASS1.1 – CLASS1.3; CLASS2.1 – CLASS2.4} from Level B of the taxonomy. This could be done in a traditional way by feeding training instances into the network and learning (updating weights of) it by backpropagation until it converge. However, in such a case, we will not benefit from complete information provided by the taxonomy. It is known that an instance of a subclass is also an instance of its superclass. If, e.g., some data sample in our example is labeled with CLASS2.3 (at level B of the taxonomy), then this instance would also have another label, CLASS2 (at Level A of the taxonomy). Let us assume that we synchronously train also the classifier (NN "A") to be capable of labeling input samples to two classes {CLASS1; CLASS2} from Level A of the taxonomy. Both classifiers ("A" and "B") will address any training input sample with its own probability distribution (among output labels) and, therefore, error (loss). Therefore, when (during training) NN "B" gets an input sample with, e.g., the true label CLASS2.3 (explicit) and, therefore, with the label CLASS2 (implicit), then the error (Loss_B) made by NN "B" (regarding CLASS2.3) could be updated if to get also the error (Loss_A) from NN "A" regarding the same sample and its true label CLASS2. We have here the possibility to train NN "B" in the context of being "informed" on the outputs from NN "A" operating on a higher level. A modified loss function in this case would contain actual Loss_B plus part (defined by λ) of Loss_A (Fig. 2b). Update of the loss function is expected to have an impact on the training (and potentially on testing) performance of NN "B". Such informed NN "B" is the simplest case of TINN.

Let us consider a more complex taxonomy (Fig. 3a). Here we can see that some classes of the lowest Level C belong to the intersection of some classes from the higher Level B so that, e.g.:

… :CLASS_C_i rdfs:subClassOf :CLASS_$B1_j$; rdfs:subClassOf :CLASS_$B2_k$.
:CLASS_$B1_j$ rdfs:subClassOf :CLASS_$A1_s$. :CLASS_$B2_k$ rdfs:subClassOf :CLASS_$A2_t$.
:CLASS_$A1_s$ rdfs:subClassOf :$DOMAIN_1$. :CLASS_$A2_t$ rdfs:subClassOf :$DOMAIN_2$. …

Therefore, for the class CLASS_C_i, we have a multiple inheritance case because it happens to be at the intersection of two of taxonomy branches (one with the root from $DOMAIN_1$ and another one from $DOMAIN_2$). This means that, if you apply the same TINN logic as above for the potential NN, we have two "channels to be additionally informed". In Fig. 3b, we may see how the NN "C", which is trained to classify instances at the lowest Layer C, may benefit from the additional loss information from the classifiers NN "B1" and NN "B2" operating at the Level B, and even deeper – from the classifiers NN "A1" and NN "A2" operating at the Level A of the taxonomy. If, for example, during backpropagation training, the current training sample has label CLASS_C_i and, additionally (due to taxonomy), we know that it also has labels CLASS_$B1_j$; CLASS_$B2_k$; CLASS_$A1_s$; and CLASS_$A2_t$, then the error (loss) Loss_C made (after processing this training sample) by the target classifier NN "C" can be updated with the losses (Loss_B1, Loss_B2, Loss_A1, and Loss_A2) from the higher layers' classifiers NNs "B1", "B2", "A1", and "A2" as follows:

$$LOSS = Loss_C + \lambda 1 \cdot (Loss_{B1} + Loss_{B2}) + \lambda 2 \cdot (Loss_{A1} + Loss_{A2}). \quad (1)$$

Formula (1) is derived by recursively applying the basic TINNs' logic from the previous example (across all the taxonomy layers) as follows:

$$LOSS = Loss_C + \lambda_B \cdot Loss_{B1} + \lambda_B \cdot Loss_{B2} = Loss_C + \lambda_B \cdot (Loss_{B1} + Loss_{B2}) =$$
$$= Loss_C + \lambda_B \cdot [(Loss_{B1} + \lambda_A \cdot Loss_{A1}) + (Loss_{B2} + \lambda_A \cdot Loss_{A2})] =$$
$$= Loss_C + \lambda_B \cdot [(Loss_{B1} + Loss_{B2}) + \lambda_A \cdot (Loss_{A1} + Loss_{A2})] =$$
$$= Loss_C + \lambda_B \cdot (Loss_{B1} + Loss_{B2}) + \lambda_B \cdot \lambda_A \cdot (Loss_{A1} + Loss_{A2}),$$

… which after substitutes $\lambda 1 \leftarrow \lambda_B$ and $\lambda 2 \leftarrow \lambda_B \cdot \lambda_A$ gives us formula (1).

One can see that, in such and similar multiple inheritance cases, the losses coming from the same layer of the taxonomy are summed up and then each sum is taxed with the appropriate coefficient (different for different layers).

4. A Couple of Motivating Examples

Consider the example in Fig. 4. The pictures of black or white squares or circles of different sizes represent the DOMAIN in the example. The final task was to train NN "B" to label any of such instances into one of four classes {BLACK_SQUARE; BLACK_CIRCLE; WHITE_SQUARE; WHITE_CIRCLE}. Using the terminology of taxonomy, we may say that our DOMAIN is a disjoint union of these four classes. Another important taxonomy awareness would be the knowledge that DOMAN, on the one hand, is also a disjoint union of BLACK and WHITE samples, and, on the other hand, DOMAIN is a disjoint union of SQUARE and CIRCLE samples. To make the task a

bit more challenging for a convolutional NN we added some noise to all the images in the dataset. The classifier after training gave us 91.3% accuracy. We repeated the experiment from scratch, but now we try to train NN "B" as a TINN. For that, we simultaneously also train two other classifiers: NN "A1" (to label samples to two classes, BLACK and WHITE) and NN "A2" (to label samples to two classes, SQUARE and CIRCLE). Therefore, it was possible to train NN "B" with additional loss information coming from NN "A1" and NN "A2". This TINN's trick has surprisingly improved the accuracy of the NN "B" classifier from 91.3% (without taxonomy awareness) to 93.7 (with taxonomy awareness).

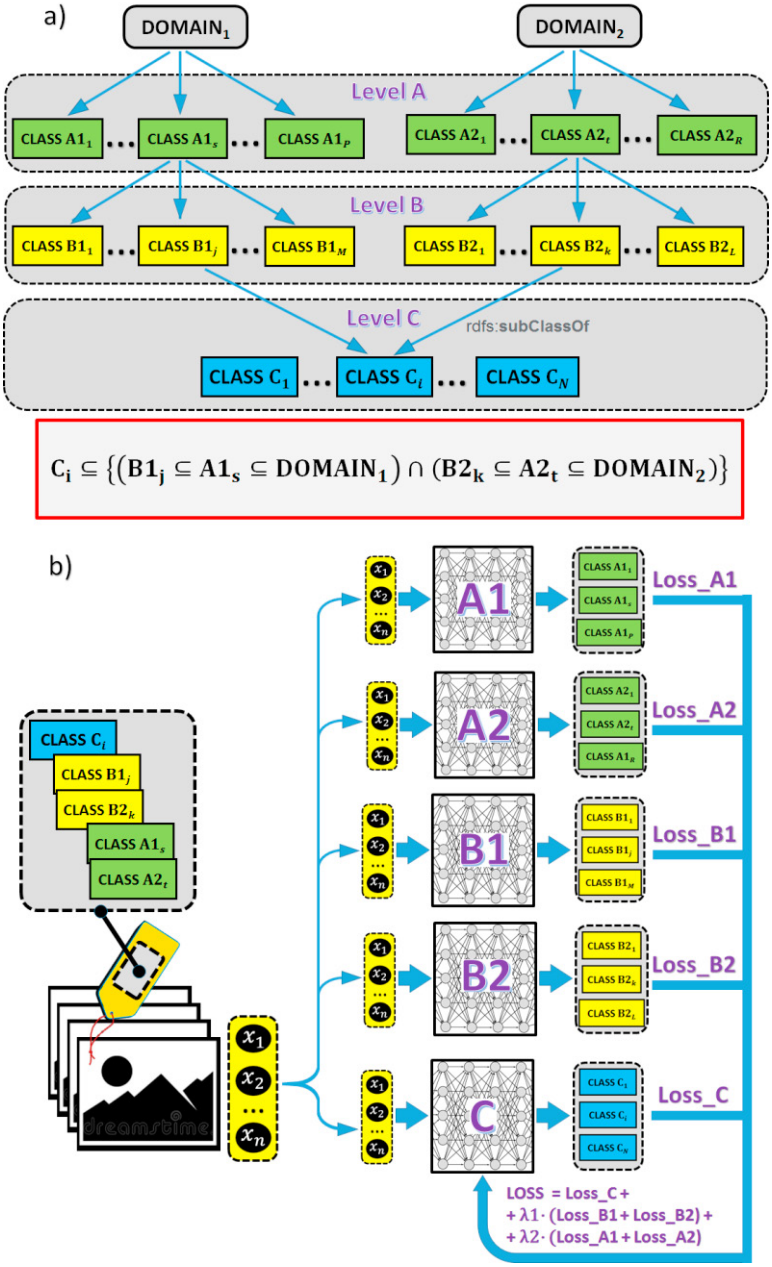


Fig. 3. The training schema of the TINN originated from taxonomy (with multiple inheritance) awareness is illustrated: (a) example of a taxonomy (class-subclass hierarchy of classes in the taxonomy); (b) the NN classifier C, which is currently being trained to label data samples into 3 classes (lower level of the taxonomy) benefits from the NN classifiers B1 and B2 trained to label the same instances into two different groups of 3 classes each of the upper level of the taxonomy. In addition, the classifiers A1 and A2 from the higher level contribute indirectly to classifier C. The construction of overall loss (LOSS) for C is illustrated.

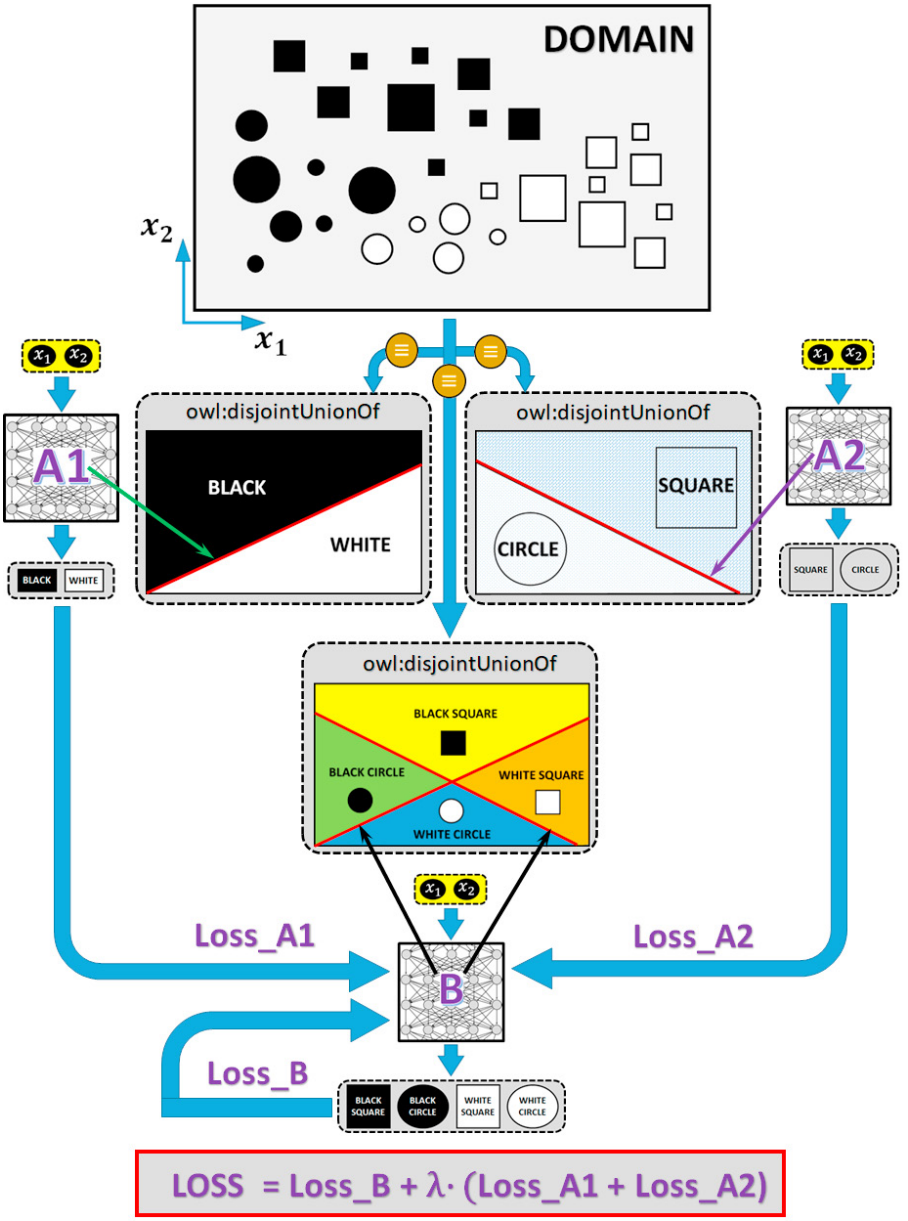


Fig. 4. The experiment of building NN classifier "B" for the DOMAIN, which is a disjoint union of four classes {BLACK_SQUARE; BLACK_CIRCLE; WHITE_SQUARE; WHITE_CIRCLE}. Additional knowledge, that DOMAN is also a disjoint union of BLACK and WHITE samples and a disjoint union of SQUARE and CIRCLE samples, makes it possible to use TINN architecture for the target classifier "B". The loss of "B" during training is updated with the loss coming from two other synchronously trained classifiers, "A1" (BLACK vs. WHITE) and "A2" (SQUARE vs. CIRCLE), which improves the performance of "B" to correctly classify the DOMAIN samples into four classes.

The influence of taxonomy awareness on the accuracy of the convolutional NN classifier in the example is understandable. The features (color, shape), which are the basic ones for the class-subclass hierarchy in the taxonomy, are actually visually apparent. Therefore, to further check the concept of TINNs, we will try to construct the next example in a more provocative way so that the actual features behind the taxonomy are not explicitly visible in the pictures (training samples). We take MNIST (handwritten digits; http://yann.lecun.com/exdb/mnist/) image dataset and we split it to two disjoint unions of subclasses: (a) EVEN and ODD digits; and (b) "< 5" and "> 4" digits (see Fig.5). The features of being even or odd, as well as being "> 4" or "< 5" do not have explicit visual appearance in the pictures of the digits. Therefore, we expected TINN to fail to improve NN's performance in this case.

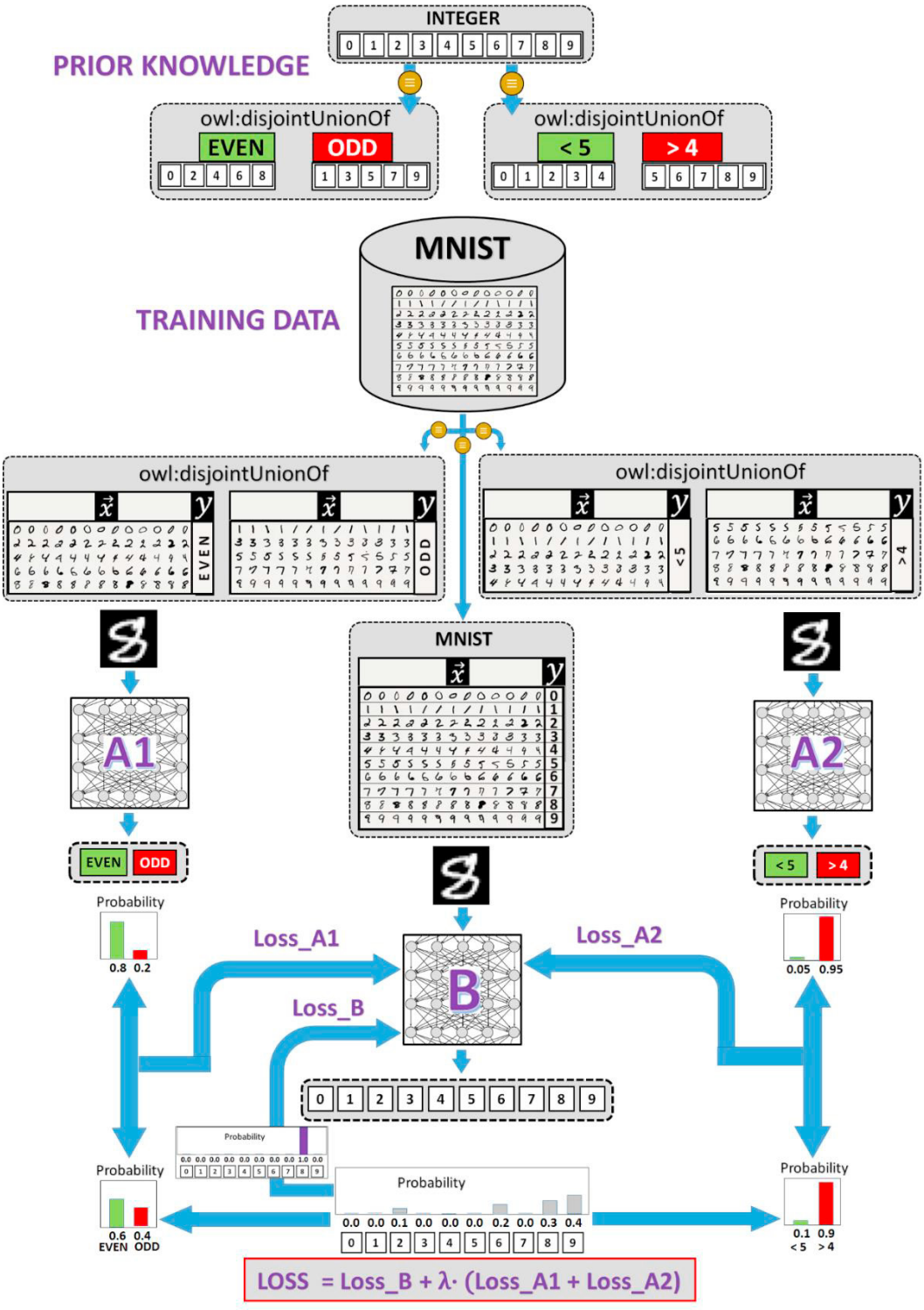


Fig. 5. The unusual taxonomy awareness effect is explored on top of the MNIST dataset. The target convolutional NN classifier "B" is trained synchronously with the two other ones: "A1", which distinguishes between odd and even digits, and "A2", which distinguishes between "<5" and ">4" digits. For each portion of input training samples, the loss of "B" during backpropagation training is updated also with the corresponding losses from "A1" and "A2", and such a TINN trick has surprisingly improved the performance of "B".

First, we add noise to the MNIST samples to make a challenge (reduce classification accuracy) for all the potential classifiers. Then we train the target NN classifier "B" (labelling digits to 10 classes) without taxonomy awareness, achieving 87.2% of test accuracy. After that, we start from scratch training of "B" but synchronously with two other classifiers: "A1" (labelling among ODD and EVEN) and "A2" (labelling among "<5" and ">4"). According to suggested TINN logic, we update the loss calculation for "B" with the awareness of losses from "A1" and "A2" as shown in Fig. 5. We were actually surprised to see that the new "taxonomy-informed" classifier "B" after training shows better performance comparably to its ignorant version (improved from 87.2 to 88.1%). The effect of taxonomy awareness seems to work even in cases when the features hidden behind the taxonomy hierarchy are not explicitly visible in the training samples. The presence of noise in training data makes such an awareness impact even bigger.

5. Instance-Class Relationship Awareness and its (Knowledge-Based and Federated) Effect

Consider a typical example of industrial assets' diagnostics. Assume that a batch (aka class) of identical complex and smart (with self-monitoring infrastructure) industrial machines has been produced. These assets are sold and used in different processes, environments, places and other contexts. Assume that, according to the design features of this class of assets, the state (vector $\vec{Y}$) of some particular machine can be derived as some function over the measurable (by embedded sensors) vector of parameters $\vec{X}$. If the function is known or can be learned by ML, then the state diagnostics for each asset can be done automatically. In this section, we consider two different options of benefit from knowing instance-class relationships (aka taxonomy awareness) among the assets to improve diagnostics by using TINN architectures.

The first scenario is shown in Fig. 6. Here we assume that the (general for the class) function $\vec{Y} = F(\vec{X})$ is known just after design and pilot testing and it is supposed to be applied to all the instances of the assets within the class during their operation within particular industrial processes. However, each machine may have some small individual constructive specifics before use and, definitely, more individual features may be acquired during operation due to different operational environments. Therefore, the histories of $\vec{X}$ and $\vec{Y}$ are collected as training data separately for each i-th machine and the individual diagnostic functions $\vec{Y} = F_i(\vec{X})$ can be learned using ML (particularly NNs).

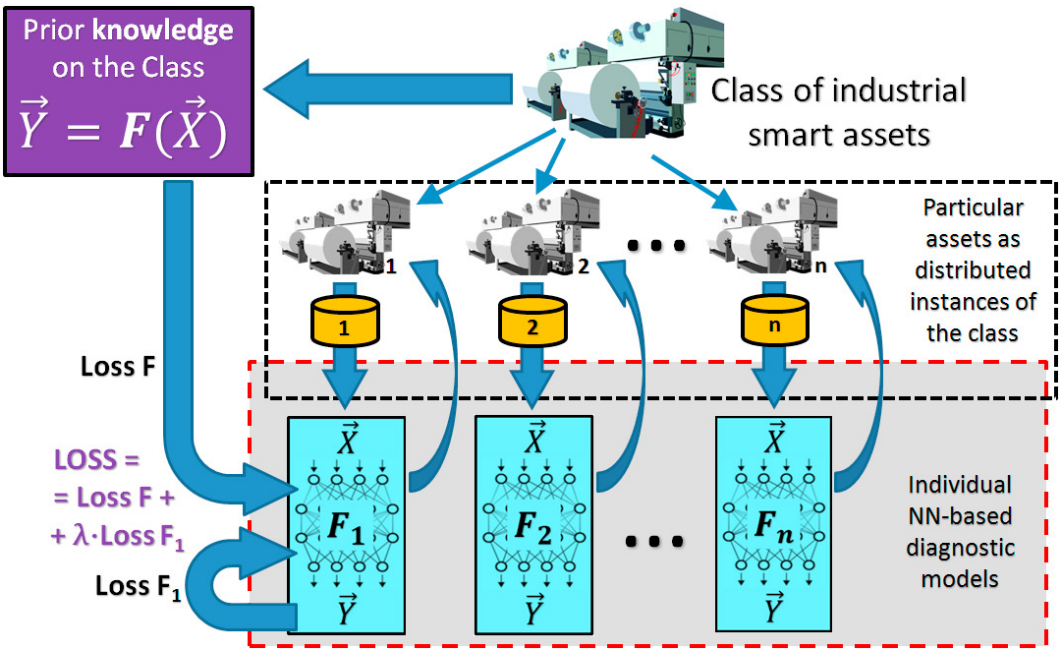
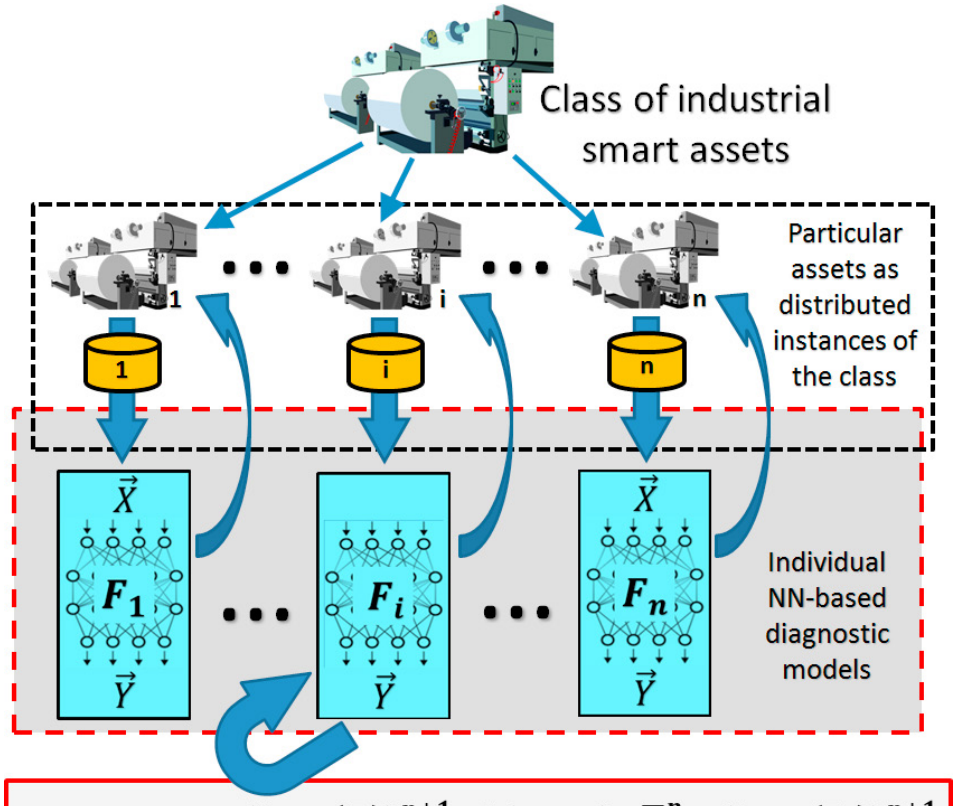


Fig. 6. The scenario is shown when each individual asset diagnostics model (NN) is trained based on the data collected from the particular asset history including awareness of the model constructed for the whole class of assets. Like in TINNs, we can update the loss function of each individual model during its backpropagation training with the loss regarding the overall class model.

Naturally, the data collected independently regarding each asset may not be big enough for constructing a reliable diagnostic model. Therefore, combining it with the diagnostic knowledge on the class of identical assets will make each individual model more trustful. This can be done by applying TINNs' logic and updating the loss value calculation for each individual NN (F_i) training by the loss related to the known and common-for-the-class function F. Taxonomy-awareness here is simply the known fact that i-th asset (with target diagnostic model F_i) is an instance of a class of similar assets with a general diagnostic model F.

The second scenario is shown in Fig. 7. This scenario is an attempt to combine the TINN approach with federated learning. Here we do not have any prior knowledge of the common-for-the-class diagnostic function F. We suppose that each individual asset will collect its own history as training data, learn individual diagnostic models from the data and, in addition, they will benefit from the similar (same class) assets' model construction experiences. Following the TINNs' logic, we need such a loss function for backpropagation training of each individual model that takes into account the loss associated with its own asset as well as the losses associated with other assets.



$$\mathbf{LOSS_{i,p,n,\alpha,t}} = \frac{(\mathbf{Loss(F_i)})^{\mathbf{p+1}} + \boldsymbol{\lambda}(\mathbf{n},\boldsymbol{\alpha},\mathbf{t}) \cdot \sum_{\substack{\mathbf{k=1} \\ \mathbf{k \neq i}}}^{\mathbf{n}} (\mathbf{Loss(F_k)})^{\mathbf{p+1}}}{(\mathbf{Loss(F_i)})^{\mathbf{p}} + \boldsymbol{\lambda}(\mathbf{n},\boldsymbol{\alpha},\mathbf{t}) \cdot \sum_{\substack{\mathbf{k=1} \\ \mathbf{k \neq i}}}^{\mathbf{n}} (\mathbf{Loss(F_k)})^{\mathbf{p}}}$$

$$\boldsymbol{\lambda}(\mathbf{n},\boldsymbol{\alpha},\mathbf{t}) = \frac{\mathbf{n-1}}{\mathbf{n} \cdot \log_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}+\mathbf{t}-\mathbf{1})}$$

Fig. 7. The federated learning scenario with TINNs is presented. Several individual NN diagnostic models (one for each smart asset) are trained simultaneously, each on its own training data. Each particular batch of training samples from the i-th NN goes through all the NNs, then a joint loss is computed from the individual losses as shown, and finally backpropagated only through the i-th network. This process repeats for each network. In this way, each network updates its weights with the awareness of the reaction of other networks to the same training samples.

For each i-th NN model during its backpropagation training, we suggest constructing the loss by applying the weighted Lehmer mean (LM) function over the individual losses from other "partner" models according to the general schema: $LOSS_i = LM[Loss(F_i), \lambda \cdot Loss(F_1), \lambda \cdot Loss(F_2), \dots, \lambda \cdot Loss(F_{i-1}), \lambda \cdot Loss(F_{i+1}), \dots, \lambda \cdot Loss(F_n)]$.

The Lehmer mean for some set of positive values $\{l_1, l_2, \dots, l_n\}$ is known to be a generalized mean function controlled by parameter p as follows:

$$LM_p(l_1, l_2, \dots, l_n) = \frac{l_1^{p+1} + l_2^{p+1} + \dots + l_n^{p+1}}{l_1^{p} + l_2^{p} + \dots + l_n^{p}} = \frac{\sum_{i=1}^{n} l_i^{p+1}}{\sum_{i=1}^{n} l_i^{p}}.$$

In the target loss function, which is the weighted Lehmer mean, we pay special attention to the λ value as follows:

$$LOSS_{i,p,n,\alpha,t} = \frac{(Loss(F_i))^{p+1} + \lambda(n,\alpha,t) \cdot \sum_{\substack{k=1 \\ k \neq i}}^{n} (Loss(F_k))^{p+1}}{(Loss(F_i))^{p} + \lambda(n,\alpha,t) \cdot \sum_{\substack{k=1 \\ k \neq i}}^{n} (Loss(F_k))^{p}}, \text{ where } \lambda(n,\alpha,t) = \frac{n-1}{n \cdot log_{\alpha}(\alpha + t - 1)}. \quad (2)$$

In formula (2): i indicates a particular model (F_i) being trained; n – the number of simultaneously trained models (i.e., the number of assets in the class); parameter t is the current training epoch number (i.e., the influence of other "partner" models on the target model will decrease during the training process, and parameter α can be used to control the decrease function); parameter p controls the Lehmer mean itself, i.e., the biases related to the distribution of averaged losses.

We experimented with both of these use-case scenarios for condition monitoring, diagnostics and predictive maintenance of industrial assets. For that, we have implemented and embedded both schemas as new additional features into our UBIWARE platform [25], which is a semantic (taxonomy-aware) middleware for smart industrial assets. Experiments show that "informed" TINNs as diagnostic models in both scenarios outperform the ignorant NNs, giving some updates on the accuracy from 0.9% to 5.8% (depending on the type of assets and other settings).

6. Conclusions

In this paper, we explored the possibility to adapt the modern "informed ML" [3] approach and emerging informed NNs architectures (such as PINNs [12] or causality-aware NNs [26]) to deal with conceptual models (ontologies/taxonomies). We show that taxonomy awareness during NNs' training could be as useful as analytical constraints in PINNs. What has been taken from the PINNs' experience is the structure of the loss function, which includes traditional fitness-to-training-data loss and some portion of the loss related to taxonomy awareness. We recommended and made preliminary exploration of TINNs as a new subclass of informed NN architectures, which can be used in many domains including smart manufacturing.

We limited our study to the basic taxonomic relationships: (hierarchical) "class-subclass" regarding the structure of training data and "instance-class" regarding physical objects from which the training data has been collected. Knowledge of complex taxonomies, i.e., with deep class hierarchies and multiple inheritance can be mapped to an appropriate structure of loss function applied during backpropagation training of TINNs.

We experimented with some image datasets and noticed that TINNs improve performance comparably to traditional convolutional NNs and this performance difference is more visible when dealing with noisy data.

A couple of TINN architectures have been explored regarding condition monitoring and diagnostics of smart assets in the Industry 4.0 context. Exploring the first architecture, we show that learning diagnostic models for individual assets from limited data will benefit from adding available knowledge on the class of identical assets. The second architecture can be applied as a kind of federated learning system where each individual diagnostic model (for the instance of the class of similar assets) learns from limited available data but benefits from exchanging training loss information with the other "partner" models. In both cases, TINN architectures outperformed ignorant NN solutions.

Quite a lot of experimentation is still needed to explore the full potential of TINNs. In addition, other complex ontological constraints (regarding taxonomy classes) must be studied to explore other possible options for TINN architecture. Robustness of TINN models against noise and adversarial attacks is still the plan for future studies.

References

[1] Elbasheer, M., Longo, F., Nicoletti, L., Padovano, A., Solina, V., and Vetrano, M. (2022). "Applications of ML/AI for decision-intensive tasks in production planning and control". *Procedia Computer Science*, **200**: 1903-1912. https://doi.org/10.1016/j.procs.2022.01.391

[2] Golovianko, M., Terziyan, V., Branytskyi, V., and Malyk, D. (2023). "Industry 4.0 vs. Industry 5.0: Co-existence, transition, or a hybrid". *Procedia Computer Science*, **217**: 102-113. https://doi.org/10.1016/j.procs.2022.12.206

[3] Von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., …, and Schuecker, J. (2021). "Informed Machine Learning–A taxonomy and survey of integrating prior knowledge into learning systems". *IEEE Transactions on Knowledge and Data Engineering*, **35(1)**: 614-633. https://doi.org/10.1109/TKDE.2021.3079836

[4] Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. (2022). "Scientific machine learning through physics–informed neural networks: where we are and what's next". *Journal of Scientific Computing*, **92(3)**: 88. https://doi.org/10.1007/s10915-022-01939-z

[5] Andonie, R. (2019). "Hyperparameter optimization in learning systems". *Journal of Membrane Computing*, **1(4)**: 279-291. https://doi.org/10.1007/s41965-019-00023-0

[6] Dayhoff, J. E., and DeLeo, J. M. (2001). "Artificial neural networks: opening the black box". *Cancer: Interdisciplinary International Journal of the American Cancer Society*, **91(S8)**: 1615-1635. https://doi.org/10.1002/1097-0142(20010415)91:8+%3C1615::AID-CNCR1175%3E3.0.CO;2-L

[7] Whittington, J. C., and Bogacz, R. (2019). "Theories of error back-propagation in the brain". *Trends in Cognitive Sciences*, **23(3)**: 235-250. https://doi.org/10.1016/j.tics.2018.12.005

[8] Narkhede, M. V., Bartakke, P. P., and Sutaone, M. S. (2022). "A review on weight initialization strategies for neural networks". *Artificial Intelligence Review*, **55(1)**: 291-322. https://doi.org/10.1007/s10462-021-10033-z

[9] Wang, Q., Ma, Y., Zhao, K., and Tian, Y. (2020). "A comprehensive survey of loss functions in machine learning". *Annals of Data Science*, **9**: 187–212. https://doi.org/10.1007/s40745-020-00253-5

[10] Sa-Couto, L., and Wichert, A. (2021). "Simple convolutional-based models: Are they learning the task or the data?". *Neural Computation*, **33(12)**: 3334-3350. https://doi.org/10.1162/neco_a_01446

[11] Bejani, M. M., and Ghatee, M. (2021). "A systematic review on overfitting control in shallow and deep neural networks". *Artificial Intelligence Review*, **54**: 6391–6438. https://doi.org/10.1007/s10462-021-09975-1

[12] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). "Physics-informed machine learning". *Nature Reviews Physics*, **3(6)**: 422-440. https://doi.org/10.1038/s42254-021-00314-5

[13] Ma, Z., Liao, H., Gao, J., Nie, S., and Geng, Y. (2023). "Physics-informed machine learning for degradation modeling of an electro-hydrostatic actuator system". *Reliability Engineering & System Safety*, **229**: 108898. https://doi.org/10.1016/j.ress.2022.108898

[14] Du, Y., Mukherjee, T., and DebRoy, T. (2021). "Physics-informed machine learning and mechanistic modeling of additive manufacturing to reduce defects". *Applied Materials Today*, **24**: 101123. https://doi.org/10.1016/j.apmt.2021.101123

[15] Xu, Y., Kohtz, S., Boakye, J., Gardoni, P., and Wang, P. (2022). "Physics-informed machine learning for reliability and systems safety applications: State of the art and challenges". *Reliability Engineering & System Safety*, **230**: 108900. https://doi.org/10.1016/j.ress.2022.108900

[16] Xiang, Z., Peng, W., Liu, X., and Yao, W. (2022). "Self-adaptive loss balanced physics-informed neural networks". *Neurocomputing*, **496**: 11-34. https://doi.org/10.1016/j.neucom.2022.05.015

[17] Huang, B., and Wang, J. (2023). "Applications of physics-informed neural networks in power systems - a review". *IEEE Transactions on Power Systems*, **38(1)**: 572-588. https://doi.org/10.1109/TPWRS.2022.3162473

[18] Cai, S., Wang, Z., Wang, S., Perdikaris, P., and Karniadakis, G. E. (2021). "Physics-informed neural networks for heat transfer problems". *Journal of Heat Transfer*, **143(6)**: 060801. https://doi.org/10.1115/1.4050542

[19] Longo, F., Nicoletti, L., and Padovano, A. (2019). "Modeling workers' behavior: A human factors taxonomy and a fuzzy analysis in the case of industrial accidents". *International Journal of Industrial Ergonomics*, **69**: 29-47. https://doi.org/10.1016/j.ergon.2018.09.002

[20] Xing, J., Kurose, R., Luo, K., and Fan, J. (2022). "Chemistry-Informed Neural Networks modelling of lignocellulosic biomass pyrolysis". *Bioresource Technology*, **355**: 127275. https://doi.org/10.1016/j.biortech.2022.127275

[21] Daneker, M., Zhang, Z., Karniadakis, G. E., and Lu, L. (2022). "Systems Biology: Identifiability analysis and parameter identification via systems-biology informed neural networks". arXiv preprint arXiv:2202.01723.

[22] Amaral, G., Baiao, F., and Guizzardi, G. (2021). "Foundational ontologies, ontology-driven conceptual modeling, and their multiple benefits to data mining". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **11(4)**: e1408. https://doi.org/10.1002/widm.1408

[23] Lukyanenko, R., Castellanos, A., Storey, V. C., Castillo, A., Tremblay, M. C., and Parsons, J. (2020). "Superimposition: augmenting machine learning outputs with conceptual models for explainable AI". In: *Advances in Conceptual Modeling: Proceedings 39* (pp. 26-34). Springer. https://doi.org/10.1007/978-3-030-65847-2_3

[24] Guizzardi, G., Fonseca, C. M., Almeida, J. P. A., Sales, T. P., Benevides, A. B., and Porello, D. (2021). "Types and taxonomic structures in conceptual modeling: A novel ontological theory and engineering support". *Data & Knowledge Engineering*, **134**: 101891. https://doi.org/10.1016/j.datak.2021.101891

[25] Katasonov A., Kaykova O., Khriyenko O., Nikitin S., and Terziyan V. (2008). "Smart Semantic Middleware for the Internet of Things". In: *Proceedings of the Fifth International Conference on Informatics in Control, Automation and Robotics* (vol. 1, pp. 169-178). https://doi.org/10.5220/0001489001690178

[26] Terziyan, V., and Vitko, O. (2023). "Causality-aware convolutional neural networks for advanced image classification and generation". *Procedia Computer Science*, **217**: 495-506. Elsevier. https://doi.org/10.1016/j.procs.2022.12.245