Automating Physical Knowledge Integration in Machine Learning

Sarah Ghidalia
CIAD UMR 7533
Université de Bourgogne, UB
F-21000 Dijon, France
Sarah_Ghidalia@etu.u-bourgogne.fr

Ouassila Labbani Narsis
CIAD UMR 7533
Université de Bourgogne, UB
F-21000 Dijon, France
ouassila.narsis@u-bourgogne.fr

Aurélie Bertaux
CIAD UMR 7533
Université de Bourgogne, UB
F-21000 Dijon, France
aurelie.bertaux@u-bourgogne.fr

Christophe Nicolle
CIAD UMR 7533
Université de Bourgogne, UB
F-21000 Dijon, France
Christophe.Nicolle@u-bourgogne.fr

***Abstract*—In a world where training data is often limited or noisy, how can we improve the reliability of machine learning models? And how can prior knowledge be integrated into these models to be closer to reality? This paper introduces Ontology-based Physics-Informed Machine Learning (OPIML), an innovative approach that formalizes and integrates physical knowledge into machine learning models using ontology. An illustration of the approach is provided with the loss function of a neural network. By using ontologies to automate the transformation of physical laws into mathematical equations, OPIML paves the way for more accurate and robust models. Experimental results demonstrate the successful application of abstract rules in various material design scenarios, offering a pathway for more robust and precise models. Later on, this approach could also be extended to encompass a variety of rule types, including legal and ethical constraints.**

***Index Terms*—Ontologies, Machine Learning, prior knowledge, Physics-Informed Machine Learning, Knowledge Integration, Physical Laws, Loss Function**

I. Introduction

Informed Machine Learning (IML) refers to approaches that incorporate prior knowledge at different stages of the machine learning process [1]. It aims to improve the quality of machine learning models by using a combination of data and prior knowledge. According to [1], "*the prior knowledge comes from an independent source, is given by formal representations, and is explicitly integrated into the machine learning pipeline*". Prior knowledge can be obtained from various sources, including scientific knowledge, expert knowledge, or world knowledge, and can be represented in different forms such as algebraic equations, differential equations, probabilistic relations, logic rules, or knowledge graphs [1].

Physics-Informed Machine Learning (PIML) is a specific approach to IML that integrates physical prior knowledge into its processing to increase the physical consistency of the models. This enables the learning process to be guided towards a set of possible solutions that adhere to relevant scientific knowledge related to the physical sciences, through the use of constraints [2]. The challenges of PIML are the same as those of IML: transforming knowledge into usable constraints (frequently embedded at the loss function level) and obtaining models consistent with the laws of physics.

These models are often developed using neural networks. Adding prior knowledge into neural networks is particularly useful in cases where training data is limited or noisy, and can often provide results with better accuracy and physical consistency [3], [4]. Physical knowledge can rarely be integrated into a neural network as is and must be transformed according to where it is to be used. Indeed, there are many places where knowledge can be integrated, whether in the training data, in the architecture of the neural network, in the learning phase, or in the evaluation of the model. In this study, we focus on integrating physical knowledge into the learning phase of the neural network through its loss function.

To be integrated into the loss function, physical laws often need to be presented in the form of partial differential equations (PDEs), which subsequently need to be translated into executable code within a neural network's programming. This requires domain experts who are familiar with the physical laws relevant to a given application, as they are responsible for developing the code required for the loss function. As this last task is complex, we must think about solutions to make this step easier. Furthermore, prior knowledge's quality can vary greatly depending on its source, relevance, and reliability. If the knowledge is incorrect or inappropriate, it can lead to errors in the model's predictions. Better formalizing knowledge is necessary to enable the reuse of knowledge in various applications and facilitate the design of all kinds of knowledge-guided machine learning algorithms [1].

Among knowledge formalization models, ontologies play an important role [7]. They provide a structured framework for organizing and representing information, making it easier to capture, store, and share knowledge in a systematic and coherent manner. Through the use of ontologies to formalize the domain-specific physical knowledge held by experts, it is possible to automate the generation of contextually relevant partial differential equations (PDEs) suitable for integration into a neural network's loss function. The Ontology-based Physics-Informed Machine Learning (OPIML), as presented in this paper, strives to achieve this goal. The purpose of this formalization work is to facilitate the design of informed neural network models by saving time for designers, for whom the transformation of business rules into constraints can be complicated at times.

Our paper is organized as follows: Section II presents work similar to ours and explains how our approach is different and more general than that of others. Section III shows how prior knowledge can be integrated into the loss function of neural networks. Section IV describes the methodology used to create an OPIML, in which physical knowledge is formalized in an ontology for easier integration into a neural network. Section V outlines the concrete implementation of the OPIML presented in the previous section. Section VI concludes with a review of the work accomplished and presents future challenges.

II. Related work

The machine learning pipeline comprises four fundamental elements: training data, a problem-specific architecture, a well-structured learning phase to optimize results, and a final evaluation [1]. It is important to note that it is possible to infuse domain knowledge into each of these four phases of the machine learning model.

Ontologies are frequently used in the construction of training datasets for feature engineering [8]. This application is particularly relevant because ontologies confer greater semantic meaning on data, which is advantageous for managing heterogeneous data [9].

Among the various forms of knowledge representation that can be applied in an IML model are knowledge graphs and logical rules, both of which can be represented in ontologies [1]. Nevertheless, it is relatively rare to find ontologies as primary knowledge sources for this category of models, and even rarer in the context of PIML models. The rise of PIML is relatively recent, as shown by a search on Google Scholar (via Publish or perish [10]) using the request *"intitle:"Physics-Informed Machine Learning""* which yields 413 articles, the oldest of them dating back to 2016. Previous work has mainly used physical knowledge in the form of partial differential equations (PDEs) in conjunction with machine learning [11]. However, it should be noted that this work may not have been explicitly classified under the term "Physics-Informed Machine Learning". In these studies, physical knowledge is often expressed as mathematical equations, which must then be translated into a programming language to enable their use in a learning model [2]. Existing works usually focus on specific scientific laws and are heavily dependent on the studied application, making it difficult to generalize these approaches and apply them to other fields or problems without significant modifications [3], [4].

Using a narrower search criterion in "publish or perish" with the query *"intitle: "Physics-Informed Machine Learning" AND ontology"* yields only three articles as results. [5] refer to ontologies as a data source for illustration purposes only, without using them in their work. [6] mention ontology as a source of knowledge, but do not use it in an automated way in their work. In [12], ontology is used to improve the activation function of a neural network, enhancing its ability to predict bridge deterioration more effectively. This last article is the only one to use ontology more extensively in a PIML.

To the best of our knowledge, there exists no prior research paper that has employed ontology to automatically transform a system's physical knowledge into programming code suitable for integration within a neural network's loss function.

III. Transforming knowledge into loss function constraint

In our approach, our central objective is to integrate physical knowledge into the learning algorithm by calculating the loss function of a neural network [13]. The loss function assesses the ability of a model to produce predictions that are consistent with the learning data by calculating the difference between the predicted values and the expected values, to minimize prediction errors [14]. Adding prior knowledge requires additional attention to produce predictions that are consistent not only with the training data but also with prior knowledge about the domain being studied [15].

A common approach to incorporate prior knowledge is to add additional terms to the loss function in machine learning models. In the case of PIML for neural networks, the loss function is often formulated as a weighted combination of two terms: a data loss term and a physics loss term [16]. The general formula for the loss function of this kind of neural networks can be expressed as:

$$\mathcal{L} = \mathcal{L}_{data} + \lambda \mathcal{L}_{phys} \tag{1}$$

where $\mathcal{L}_{data}$ is the data loss that measures the error between the model predictions and the training data, $\mathcal{L}_{phys}$ is the physics loss that measures the consistency error between the model predictions and the physical laws of the system, and λ is a weighting coefficient that controls the relative importance of the physics loss compared to the data loss [2], [3], [17]. The term for physical loss, $\mathcal{L}_{phys}$, can be formulated in different ways depending on the physical laws governing the system under study. The physical loss function can be associated with a global coefficient λ which controls the importance of the physical loss in the calculation of the total loss function.

It is also possible that the system being studied must respect multiple physical constraints. In this case, the physical loss function corresponds to the sum of the physical loss terms associated with each of the constraints. Adding these physical constraints to the loss function ensures that the model generates predictions that are not only accurate but also compliant with physical rules, which is particularly important

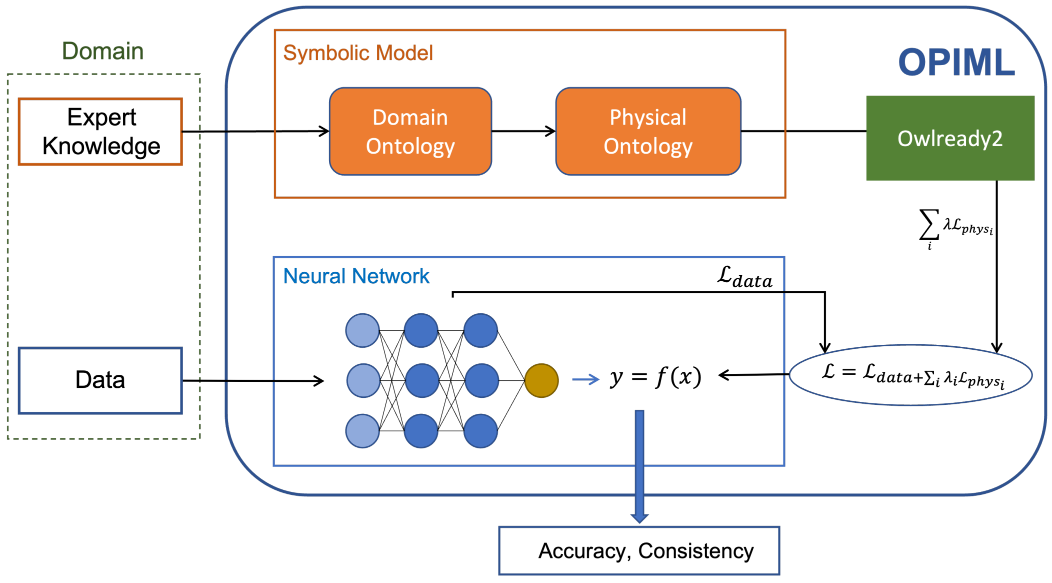


Fig. 1. Ontology-based Physics-Informed Machine Learning (OPIML) architecture

for ensuring the consistency of the model outputs. The physical loss term is given by:

$$\sum_i \lambda_i \mathcal{L}phys, i \tag{2}$$

where i represents each of the different physical constraints, λ_i is the weighting coefficient associated with each physical constraint, and $\mathcal{L}phys, i$ is the corresponding physical loss for each constraint. Thus, physical loss terms are incorporated into the total loss function to ensure that the model respects specific physical constraints while minimizing the overall loss function given by:

$$\mathcal{L} = \mathcal{L}data + \sum_i \lambda_i \mathcal{L}_{phys,i} \tag{3}$$

In this last equation, $\mathcal{L}data$ represents the loss associated with the data, which measures how far the model predictions are from the true values, while the weighted sum of the physical loss terms $\sum i \lambda_i \mathcal{L}_{phys,i}$ measures how well the model predictions comply with the different physical constraints.

In general, to integrate physical constraints into the loss function, it is necessary to formalize mathematical equations that reflect physical laws and include them in the loss function. The parameters of the learning model are then adjusted to minimize the loss function while respecting the imposed physical constraints. For example, we can model the behavior of a fluid by adding energy and mass conservation equations to the loss function. This approach guides the learning to ensure compliance with these physical constraints.

However, adding physical constraints requires specific domain knowledge and physical laws to be incorporated into the model. Although these approaches use prior knowledge to mitigate the shortcomings of purely data-driven methods, the technique remains artisanal and lacks the flexibility to identify and formalize the most appropriate physical knowledge.

PIMLs aim to increase the consistency of predictions by integrating prior knowledge [1]. However, this knowledge often requires preprocessing to convert it into usable constraints in the model. To optimize the integration of prior knowledge into the learning process, it is essential to define and formalize this knowledge.

To use knowledge in the loss function, this prior knowledge must be transformed into constraints. Depending on the form of the knowledge, this transformation will have to follow different processes [3], [18], [19]. First-order logic predicates cannot be integrated into a loss function as is; they must be processed before [18], [19]. Similarly, an equation or correlation rule is often transformed into a Partial Differential Equation (PDE) to be used in the loss function [3]. Formalizing knowledge aims to facilitate its transformation into constraints.

IV. Ontology-based Physics-Informed Machine Learning

We aim to formalize physical knowledge to integrate it more easily into the loss function of a neural network. To achieve this, it is necessary to (1) determine the rules that the application must follow in a particular domain, (2) associate these rules with the appropriate physical law and (3) formalize them into a mathematical equation to determine the $\mathcal{L}_{phys}$ term. To achieve this goal, we designed an Ontology-based Physics-Informed Machine Learning (OPIML), described in Figure 1, which incorporates two specialized ontologies adapted to the first two tasks and is complemented by Python code to handle the third task.

A. *Determination of rules for a particular domain*

The first ontology (“domain ontology” in Figure 1) concerns the knowledge related to the application being studied, as well as the associated rules. For example, an inverse relationship between air conditioning (C) and the temperature of a room (T): as the air conditioning increases, the room's temperature decreases [4]. In fatigue material, we find a similar relationship between the stress amplitude (S) imposed on the metal and its fatigue life (L): as stress increases, fatigue life decreases [3]. For the first example, the computation of the partial derivative $\frac{\partial T}{\partial C} < 0$ formalizes the inverse relationship between air conditioning (C) and the temperature of a room (T). In the second example, the computation of the partial derivative $\frac{\partial L}{\partial S} < 0$ formalizes the inverse relationship between stress (S) and fatigue (L). In two separate fields, such as building and materials science, some applications require very similar physical laws.

B. Association of rules with a physical law

The fundamental laws of physics can be presented as abstract rules that can be adapted to different contexts. Using the previous example, the abstract rule is "if A increases, then B decreases" with A and B being different objects depending on the application domain. This abstract rule, a partial derivative $\frac{\partial B}{\partial A} < 0$, is formalized in the second ontology, named "physics ontology" in Figure 1.

Thus, each specific rule present in the "domain ontology" is associated with at least one abstract physical law represented in the "physics ontology". The association is made through a subsumption link, i.e. the specific rule "as the air conditioning increases, the room's temperature decreases" is an instance of the class "if A increases, then B decreases". Each physical rule, specific to a particular context and associated with particular variables, is, in fact, an instantiation of a more generic physical law.

Our physics ontology (presented in Figure 2) is able to represent a situation with several rules that can be applied in the context of this situation. These rules are abstract rules that represent generic knowledge such as the law of proportionality ("if A increases, then B increases") or the law of negative relationship ("if A increases, then B decreases") as seen in Figure 3.

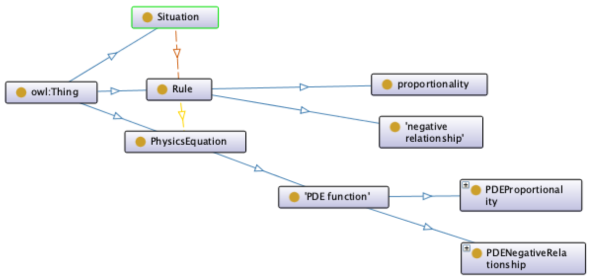
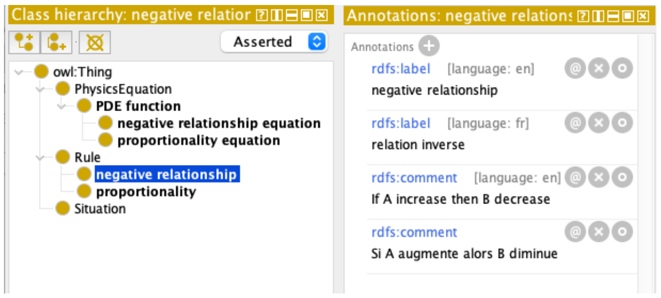


Fig. 2. Physics ontology

These rules are associated with specific functions that correspond to their transformation for use in a loss function. Thus, the law of proportionality becomes "the derivative of B concerning A is positive" and the law of negative relationship becomes "the derivative of B concerning A is negative"



rdfs:label [language: en]
negative relationship

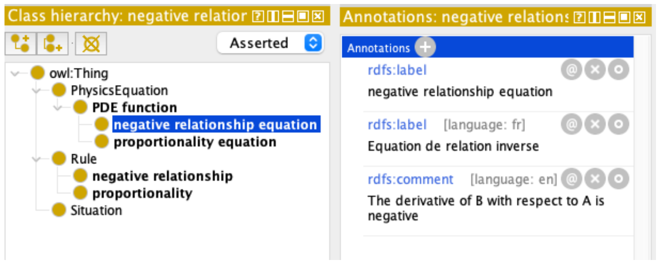
rdfs:label [language: fr]
relation inverse

rdfs:comment [language: en]
If A increase then B decrease

rdfs:comment
Si A augmente alors B diminue

Fig. 3. Negative relationship's rule

as seen in Figure 4. The link between the rules and their corresponding functions is realized thanks to the SWRL rules, so it can be inferred by an inference engine like HermiT [20] or Pellet [21]. The ontology is based on the OWL-Lite language, so it corresponds to the $\mathcal{SHIF}$ description logic.



rdfs:label
negative relationship equation

rdfs:label [language: fr]
Equation de relation inverse

rdfs:comment [language: en]
The derivative of B with respect to A is negative

Fig. 4. Negative relationship's equation

The domain ontology uses this second ontology to specify the general physical law that will then constrain the neural network through the loss function.

C. Formalization of physical laws into mathematical equations

To incorporate physics rules into the loss function through the $\mathcal{L}_{phys}$ term, it is imperative to express these rules as mathematical equations, frequently taking the form of partial differential equations.

For each abstract physical law formalized in the physics ontology, a corresponding Python function has been developed to represent it. This function accepts contextual variables from the domain ontology as input parameters to adapt to each specific rule. Through transitivity, this function can calculate the $\mathcal{L}_{phys,i}$ term corresponding to a rule i according to the parameters given by the domain ontology. The final loss function can therefore cover several different rules, each of which is added to the physical loss, as shown by Equation 2 in paragraph III.

For example, a Python function can calculate the equation $\frac{\partial B}{\partial A} < 0$ corresponding to the rule of negative relationship ("if A increases, then B decreases") when parameters A and B are known. Algorithm 1 is the pseudo-code of this function i.e. the calculated loss term for each rule about a negative relationship.

To explain the pseudo-code, consider the following:

Algorithm 1 Negative Relationship law Penalty

```
Require: Tensors $x$ and $y$
Ensure: $\mathcal{L}_{NegRel}$
 1: procedure PENALTY($x, y$)                 ▷ The loss penalty for x
 2:     Initialize $tape$ to record operations
 3:     Record operations on $x$
 4:     $y \leftarrow$ ModelOutput($x$)
 5:     $dx \leftarrow$ ComputeGradient($y, x$) //First derivative of $y$ with respect to $x$
 6:     $n \leftarrow$ SizeOfFirstAxis($x$)
 7:     state_indicator $\leftarrow$ IndicatorTensors($dx > 0$)
 8:     loss $\leftarrow \frac{\text{state\_indicator} \times dx^2}{n}$
 9:     $\mathcal{L}_{NegRel} \leftarrow$ ReduceToSumOfTensors($loss$)
10:     return $\mathcal{L}_{NegRel}$          ▷ The loss penalty for negative relationship
11: end procedure
```

1) **Initialize tape to record operations**: This line sets up a "*tape*" to keep track of operations performed on tensors. This is essential for automatic differentiation, which is used later to compute gradients.
2) **Record operations on** x: This line specifies that operations performed on the tensor x should be recorded on the tape. This is necessary for computing the gradient concerning x.
3) $y \leftarrow$ **ModelOutput**(x): This line calculates the output y of the machine learning model given the input x.
4) $dx \leftarrow$ **ComputeGradient**(y, x): This line computes the first derivative dx of the output y with respect to the input x. This is done using the operations recorded on the tape.
5) $n \leftarrow$ **SizeOfFirstAxis**(x): This line calculates n, the size of the first axis of the tensor x. This is used later to normalize the loss.
6) **state indicator** $\leftarrow$ **IndicatorTensor**($dx > 0$): This line creates an indicator tensor that has the same shape as dx. Each element of the indicator tensor is set to 1 if the corresponding element in dx is greater than 0, and 0 otherwise. Indeed, when the rule "if A increases, B decreases" is observed, then dx is less than 0 (i.e. $\frac{\partial B}{\partial A} < 0$). We don't want to add a penalty when this constraint is met.
7) **loss** $\leftarrow$ (**state indicator** $\times\, dx^2)/n$: This line calculates the loss by squaring dx, element-wise multiplying it by the state indicator, and then dividing by n. If the tensor indicator is set to 1, this will have the effect of adding a penalty, equal to the square of the derivative, to the loss function, as the rule is not respected. If the tensor indicator is set to 0, no penalty will be added to the loss function.
8) $\mathcal{L}_{NegRel} \leftarrow$ **SumOfTensor**(**loss**): This line sums up all the elements of the loss tensor to get a single scalar value, which represents the total loss.
9) **return** $\mathcal{L}_{NegRel}$: This line returns the calculated total loss of an instance of the negative relationship law as the output of the algorithm.

The rules associated with each application are retrieved using the OwlReady2 library [22]. The Pellet inference engine, accessible via Owlready2, transmits all the rules associated with an application to the Python code, specifying for each rule the abstract physical function assigned to it. As the context parameters are formalized in the ontology, they can be easily passed on to the Python function. This process ensures that all knowledge extracted from the ontology is effectively converted to be used in the programming code essential for neural network implementation.

Thanks to this methodology, physical loss functions are written only once in Python and can be reused in different projects. The ultimate aim is to have a physics ontology that is complete enough that modifications become necessary exclusively within the domain ontology, which depends on the application case.

V. Experiment

The proposed OPIML is developed using Python v3.9, Keras v2.10, Protégé v5.5, and OwlReady2 v0.4 on a Mac mini with an Apple M1 processor and 8 GB of RAM.

The first task is to identify abstract functions and formalize them in the physical ontology. In our case, we took the rule of proportionality and its inverse relation as examples. Because they are associated with various physical phenomena and are used in fields as varied as materials science [3], or BIM [4]. These abstract functions can then be applied to context-specific rules. The example chosen here is taken from materials science, as the data is readily available for an initial proof of concept. The development of a domain ontology allows us to represent all the rules associated with this use case. Each rule is identified by an abstract function from the physics ontology created earlier. Finally, the abstract physical functions are linked to their corresponding equations using the Owlready2 library. The resulting Python programming code can then be inserted into the loss function of a neural network.

Through this approach, the knowledge formally documented in the ontology is automatically translated, making it easy to incorporate into the learning phase of a neural network.

A. Data description

This first experiment uses fatigue datasets from three different materials, namely steel wire, 2024-T4 aluminum alloy (2024-T4), and the annealed aluminum wire (AAW) presented by [3]. Fatigue life refers to the number of cycles or the amount of time that a material or structure can withstand repeated loading and unloading cycles before it fails due to fatigue. Fatigue failure typically occurs at a stress level that is below the material's ultimate strength, but is still high enough to cause damage over time. The goal of the model is to build a function of stress level that predicts when the material will fail. The data is transformed to logscale and standardized as recommended by [3].

B. Context-specific rules about fatigue material

The domain ontology allows us to instantiate a new situation (e.g. the constraints related to the prediction of fatigue life on the steel) with the associated rules as illustrated in Figure 5.

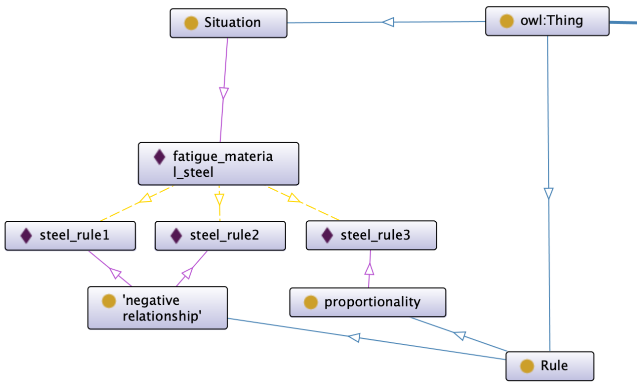
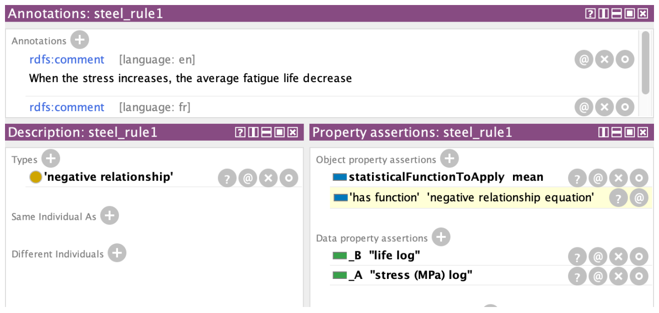


Fig. 5. Negative relationship's equation

Several rules can be associated with each situation. Here, we use the rules presented by [3]. The first rule that the model must verify is that as the stress increases, the average fatigue life decreases, as seen in Figure 6. The different parameters are given for each individual: the general rule that should apply (here "negative relationship"), the factor A (here "stress (Mpa) log"), and the target value B (here "life log") are given. The variables A and B correspond to the names of the columns in the dataset. As this first rule is identified as belonging to the "negative relationship" class, the inference engine deduces that it must use the "negative relationship equation" function to apply this constraint within the loss function.


Annotations: steel_rule1

Annotations

rdfs:comment [language: en]

When the stress increases, the average fatigue life decrease

rdfs:comment [language: fr]

Description: steel_rule1

Types

'negative relationship'

Same Individual As

Different Individuals

Property assertions: steel_rule1

Object property assertions

statisticalFunctionToApply mean

'has function' 'negative relationship equation'

Data property assertions

_B "life log"

_A "stress (MPa) log"

Fig. 6. First rule to predict the fatigue life of steel

This domain ontology, containing the situations and their associated rules, is loaded by the OwlReady2 library, which takes care of the inferences (thanks to the Pellet inference engine) to link them to the abstract physical functions and then by transitivity to the Python functions to be used to calculate the $\mathcal{L}_{phys,i}$ term of each (c.f. section IV-C).

This process enables the expert to describe his rules in the domain ontology without having to worry about coding each of the associated Python functions.

C. Results

1) prediction performance: We have created three different models for comparison with the same architecture: a fully connected neural network containing two hidden layers with 16 neurons for each hidden layer. The first is a baseline model whose loss function does not include a L_{phys} term. The second is a PIML model with a loss function defined by equation 1. Finally, the third is an OPIML model identical to the previous one, but for which the physical knowledge is provided by an ontology. The objective is to determine whether the automatic integration of physics rules within the loss function is costly in terms of performance.

Prediction performance, illustrated by the coefficient of determination (R^2), root mean square error ($RMSE$) and mean absolute percentage error ($MAPE$), shows that the two PIMLs (PIML model and OPIML model) for the Wire Steel and 2024-T4 datasets perform slightly better than the baseline model (c.f. Tables I, II). This is not the case for the AAW dataset, for which the baseline model performs slightly better (c.f. Table III).
It is important to note that the results of the two physics-based models have exactly the same performance. Thus the automation of the integration of mathematical functions in the loss function does not penalize the performance of the system in any way.

TABLE I
PERFORMANCE AND CONSISTENCY METRICS FOR STEEL WIRE DATASET

Steel Wire	**Baseline**	**PIML** ($\lambda = 770$)	**OPIML** ($\lambda = 770$)
R^2	0.76	**0.77**	**0.77**
$RMSE$	1.428	**1.415**	**1.415**
$MAPE$	**0.058**	0.059	0.059
L_{phys}	2.25	**0**	**0**

TABLE II
PERFORMANCE AND CONSISTENCY METRICS FOR 2024-T4 DATASET

2024-T4	**Baseline**	**PIML** ($\lambda = 500$)	**OPIML** ($\lambda = 500$)
R^2	0.92	**0.93**	**0.93**
$RMSE$	0.512	**0.459**	**0.459**
$MAPE$	0.02	**0.02**	**0.02**
L_{phys}	3.9	**0**	**0**

TABLE III
PERFORMANCE AND CONSISTENCY METRICS FOR AAW DATASET

AAW	**Baseline**	**PIML** ($\lambda = 1000$)	**OPIML** ($\lambda = 1000$)
R^2	**0.94**	0.93	0.93
$RMSE$	**0.374**	0.386	0.386
$MAPE$	**0.027**	0.028	0.028
L_{phys}	0.51	**0**	**0**

2) Consistency analysis: However, as mentioned above, these performance scores only show the correspondence between the results obtained and the input data. They do not demonstrate the consistency between the results and the physics rules that the model must verify. To measure consistency between physical rules and model results, we simply reuse the global term L_{phys} used in the loss function of the two PIMLs models.

L_{phys} term can be considered as an inconsistency score: the higher it is, the less the model agrees with the laws of physics. Indeed, the main objective is to minimize the loss function as the model is trained, so the more the physical constraints are respected, the smaller the inconsistency score will be.

In the case of our two physics-informed models, this term L_{phys} is always equal to 0, which means that these models are compatible with the laws of physics of which the loss function is informed. On the other hand, this same score is always greater than 0 in the case of the baseline model, meaning that it is not completely consistent with the laws of physics. Both physics-based models are likely to generalize better on new data, an expected result already demonstrated by [3].

Nevertheless, the main objective of our work is to propose a generic method for formalizing knowledge. From this point of view, in the case of the fatigue material application, we are able to present identical results for PIML and OPIML models as shown in Tables I, II and III.

VI. Conclusion and future directions

This paper presents an Ontology-based Physics-Informed Machine Learning model, called OPIML, which is an evolution of PIML in which knowledge is formalized as an ontology. To this purpose, we have proposed a generic approach modeling the set of physical rules required in two ontologies, using fatigue material as the application domain for our experimentation [3]. Those two ontologies allow us to (1) determine the rules that the application must follow in a particular domain, and (2) associate these rules with the appropriate physical law to formalize it into a mathematical equation. This equation can then be automatically transformed into Python programming code to be added to the loss function of a neural network via the $\mathcal{L}_{phys}$ term. We conducted experiments with OPIML using fatigue materials and successfully applied the same abstract rules across various scenarios involving different materials. This implies that the use of ontologies as a source of knowledge does not influence the performance results obtained with the same neural network.

The benefits of using an ontology are manifold: the formalization of knowledge facilitates its re-use in different contexts, automates the addition of physical constraints to neural networks for non-specialists in machine learning, and therefore avoids some programming errors. It should be noted that these abstract rules can be used in a very different domain, such as forecasting the temperature in a building [4].

In our experimentation, we used a simple fully-connected neural network architecture that contains two hidden layers (with 16 neurons for each of the hidden layers) as described by [3]. It is often up to the data scientist to choose this architecture, which can change depending on the context. A possible evolution would be to formally describe this architecture in the domain ontology, so that it can be modified more easily. This is a technique that could be developed in the future.

Until now, a global weight for all rules related to physical constraints has been used. It would be necessary to modify the code we have developed to be able to individually adjust the weight associated with each rule in the loss function. Furthermore, the score of each constraint added to the loss function when violated is arbitrarily equal to the square of its derivative. In the future, it should be possible to define how this score is calculated in the ontology.

The existing physical ontology needs to be supplemented with additional physical rules to improve its applicability in various projects. This process involves the identification and understanding of new physical rules, which serve as the basis for the development of abstract rules. For example, by adding laws relating to kinetics, which have applications in many fields such as water or air quality, as well as road traffic forecasting [23].

Finally, we wish to extend our scope beyond the incorporation of physical rules; we aim to incorporate various other types of rules, including legal, ethical usage, or business rules. To achieve this, we need to explore in depth all possible transformations to obtain constraints that can be incorporated into a loss function, and realize these transformations through ontologies.

Acknowledgment

This research was supported by European project H2020 RESPONSE[1] (Grant Agreement nº 957751), for which we present the first results of our approach.

References

[1] L. Von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, and Others "Informed Machine Learning–A taxonomy and survey of integrating prior knowledge into learning systems," IEEE Transactions On Knowledge And Data Engineering. vol. 35, 614-633, 2021.

[2] G. Karniadakis, I. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nature Reviews Physics. vol. 3, 422-440, 2021.

[3] T. Zhou, S. Jiang, T. Han, S.-P. Zhu, and Y. Cai, "A physically consistent framework for fatigue life prediction using probabilistic physics-informed neural network," International Journal of Fatigue, vol. 166, p. 107234, Jan. 2023, doi: 10.1016/j.ijfatigue.2022.107234.

[4] L. Di Natale, B. Svetozarevic, P. Heer, and C. N. Jones, "Physically Consistent Neural Networks for building thermal modeling: Theory and analysis," Applied Energy, vol. 325, p. 119806, Nov. 2022, doi: 10.1016/j.apenergy.2022.119806.

[5] S. Liu, B. B. Kappes, B. Amin-ahmadi, O. Benafan, X. Zhang, and A. P. Stebner, 'Physics-informed machine learning for composition – process – property design: Shape memory alloy demonstration', Applied Materials Today, vol. 22, p. 100898, Mar. 2021, doi: 10.1016/j.apmt.2020.100898.

[6] J. Kim, X. Zhao, A. U. A. Shah, and H. Kang, 'Physics-Informed Machine Learning-Aided System Space Discretization', in NPIC&HMIT 2021, Online, Jun. 2021. doi: 10.13182/T124-34648.

[7] A. Maedche, H. Schnurr, S. Staab, and R. Studer, "Representation language-neutral modeling of ontologies," Proceedings Of The German Workshop "Modellierung-2000", Koblenz, Germany, pp. 129-142, 2000.

[8] S. S. Sahoo et al., "Ontology-based feature engineering in machine learning workflows for heterogeneous epilepsy patient records," Sci Rep, vol. 12, no. 1, Art. no. 1, Nov. 2022, doi: 10.1038/s41598-022-23101-3.

[1]https:h2020response.eu/

[9] B. Zhou et al., “SemML: Facilitating development of ML models for condition monitoring with semantics,” Journal of Web Semantics, vol. 71, p. 100664, 2021, doi: https://doi.org/10.1016/j.websem.2021.100664.

[10] Harzing, A.W. (2007) Publish or Perish, available from https://harzing.com/resources/publish-or-perish

[11] I. E. Lagaris, A. Likas, and D. I. Fotiadis, “Artificial neural networks for solving ordinary and partial differential equations,” IEEE Transactions on Neural Networks, vol. 9, no. 5, pp. 987–1000, Sep. 1998, doi: 10.1109/72.712178.

[12] X. Hu and K. Liu, “Structural Deterioration Knowledge Ontology towards Physics-Informed Machine Learning for Enhanced Bridge Deterioration Prediction,” Journal of Computing in Civil Engineering, vol. 37, no. 1, p. 04022051, Jan. 2023, doi: 10.1061/(ASCE)CP.1943-5487.0001066.

[13] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, “A comprehensive survey of loss functions in machine learning,” Annals Of Data Science. pp. 1-26, 2020.

[14] T. Mitchell, Machine Learning, McGraw-Hill Science/Engineering/-Math, New York, NY, 1997

[15] T. Yu, S. Simoff, and T. Jan, ‘VQSVM: A case study for incorporating prior domain knowledge into inductive machine learning’, Neurocomputing, vol. 73, no. 13, pp. 2614–2623, Aug. 2010, doi: 10.1016/j.neucom.2010.05.007.

[16] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. Karniadakis, “Physics-informed neural networks for heat transfer problems,” Journal Of Heat Transfer. vol. 143, 2021.

[17] Y. Chen, Q. Yang, Z. Chen, C. Yan, S. Zeng, and M. Dai, “Physics-informed neural networks for building thermal modeling and demand response control,” Building And Environment. vol. 234 pp. 110149, 2023.

[18] M. Medina Grespan, A. Gupta, and V. Srikumar, ‘Evaluating Relaxations of Logic for Neural Networks: A Comprehensive Study’, in Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 2812–2818. doi: 10.24963/ijcai.2021/387.

[19] L. Gan, K. Kuang, Y. Yang, and F. Wu, ‘Judgment Prediction via Injecting Legal Knowledge into Neural Networks’, Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 14, Art. no. 14, May 2021, doi: 10.1609/aaai.v35i14.17522.

[20] R. Shearer, B. Motik, and I. Horrocks, “HermiT: A Highly-Efficient OWL Reasoner,” Owled, vol. 432, p. 91, Oct. 2008.

[21] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, “Pellet: A practical OWL-DL reasoner,” Journal of Web Semantics, vol. 5, no. 2, pp. 51–53, Jun. 2007, doi: 10.1016/j.websem.2007.03.004.

[22] J.-B. Lamy, ‘Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies’, Artificial Intelligence in Medicine, vol. 80, pp. 11–28, Jul. 2017, doi: 10.1016/j.artmed.2017.07.002.

[23] Y. Liang et al., “Mixed-Order Relation-Aware Recurrent Neural Networks for Spatio-Temporal Forecasting,” IEEE Transactions on Knowledge and Data Engineering, pp. 1–15, 2022, doi: 10.1109/TKDE.2022.3222373.