PAIRS: Physics-Enabled AI for Real-Time Simulations Surrogates

Zeinab Alfaytarouni and Hamza Ben Ammar
Capgemini Engineering
Toulouse, France
e-mail: {zeinab.alfaytarouni|hamza.ben-ammar}@capgemini.com

***Abstract*—The development and improvement of complex engineering systems increasingly depend on virtual and hybrid test benches for validating new designs or modifications to existing ones. Central to these test benches are simulation models, which are essential but time-consuming to develop due to their reliance on domain expertise. Full-fledged simulation models can also be slow, impeding the validation process that requires real-time simulation. Conversely, AI surrogate models, derived from sensor data, face constraints due to insufficient training data and potentially lacking physical sense. To address these challenges, we propose the use of physics-enabled AI models as surrogates, which strike a balance by integrating underlying physical laws through model equations, thereby requiring significantly less data for training. Once trained, these models operate in real-time, expediting the validation process. In this work, we introduce a Physics-enabled AI surrogate model development process that augments to the existing Machine Learning Operations (MLOps) workflow. Our approach employs an ontological framework to align user needs with a model template. We leverage Physics-Informed Neural Networks (PINNs) as the core building block for this template. Once a model structure is selected, the traditional MLOps process is applied to train and validate the AI surrogate. This methodology simplifies the model development process and hence accelerates the overall system development.**

Keywords-Physics-enabled AI; Physics-Informed Neural Networks; Ontology; Simulations.

I. INTRODUCTION

Advances in Artificial Intelligence (AI) have had a profound impact across numerous disciplines, including engineering. One particularly transformative development is surrogate modeling, especially approaches based on AI techniques [1]. Surrogate models offer simplified, fast, and reliable alternatives to complex, costly, and time-consuming multiphysics simulations or physical experiments.

The creation of an AI-based simulation model typically follows a common set of steps, regardless of the specific AI technique employed. The process begins with a clear definition of the problem that the model is intended to solve. This is followed by the acquisition of the necessary data, as well as the identification of performance requirements and constraints. The collected data must then be prepared, potentially merged if sourced from different origins, and cleaned to ensure quality and relevance. The next step involves selecting an appropriate model, designing its architecture, fine-tuning its parameters, and carrying out the training, validation, and testing phases. The process concludes with the deployment of the model in its target environment. Once deployed, the model can be monitored to assess its performance under real-world conditions and to detect any anomalies. One of the most complex tasks in this process is selecting the appropriate AI model. While data-driven Machine Learning (ML) algorithms have demonstrated success in many surrogate modeling tasks, they are often criticized for operating as "black boxes", their internal decision-making processes are difficult to interpret. This raises concerns about transparency and trust [2]. Additionally, such models tend to require large volumes of high-quality training data and significant computational resources, which may not be feasible in domains where data is scarce or expensive to acquire [3]. Furthermore, purely data-driven models can struggle to generalize beyond the specific conditions seen during training [2][3]. In response to these challenges, numerous studies have emphasized the importance of integrating domain-specific knowledge into machine learning [4] [5]. This integration improves interpretability, reduces data requirements, and increases consistency with known physical laws or constraints. This approach, which combines domain knowledge with data, is known as a hybrid approach and is gaining increasing popularity. Among these methods, Physics-Informed Neural Networks (PINNs) [6] [7] have recently attracted significant attention and have been applied to a wide range of applications across various fields. These models incorporate physical laws, typically in the form of differential equations, directly into the loss function during training. As a result, PINNs not only offer better generalization from limited data but also provide more transparent and physically consistent predictions.

Since the introduction of PINNs by Raissi et al. in 2019 [6], most studies have employed PINNs with feedforward neural network architectures. However, some researchers have explored alternative types of neural networks to assess their impact on overall model performance [8]–[14]. The effectiveness of a PINN architecture depends on both the data and the task at hand. This means that certain architectures are better suited to specific situations, depending on the characteristics of the data and the nature of the problem to be solved, while others may perform better in different contexts. Therefore, selecting a suitable architecture for a PINN is challenging due to the wide range of available options, from the type of neural network to various model parameter choices.

To simplify this task, we aim to develop an ontology-based Recommendation System (RS) to assist in selecting an appropriate PINN model. This work is part of a broader project that seeks to streamline and automate the entire process of developing physics-enabled AI surrogates designed to replace complex simulations. The primary goal of the project is to enable domain or simulation experts to build their AI Surrogate Models (SMs) without requiring extensive expertise in AI.

The remainder of this paper is organized as follows. In

 ISBN: 978-1-68558-300-2

Section II, we introduce the concept of Physics-Enabled AI, with a focus on PINNs. We present an analysis of the key parameters and structural choices that influence PINN's performance. Section III provides a literature review of existing PINN architectures across various applications and domains. In Section IV, we introduce the concept of ontology-based recommender systems. Section V details the development and implementation of our proposed ontology-driven recommendation system, including the ontology construction process and the recommender interface that suggests suitable PINN architectures based on user input. Finally, Section VI concludes the paper with a summary of our findings and directions for future work.

II. Physics-Enabled AI

A. Principle of Physics-Informed Neural Networks

PINNs [6] are a specialized category of deep learning algorithms designed to tackle both forward and inverse problems. Unlike traditional neural networks that rely solely on data, PINNs incorporate prior knowledge of the system, typically in the form of Partial Differential Equations, directly into the training process. This is achieved by embedding the governing equations into the network's loss function, effectively constraining the solution space and guiding the model toward physically consistent predictions. The primary motivation for developing these algorithms is that incorporating prior knowledge or physical constraints can lead to more interpretable machine learning models that require less data and remain robust in the presence of imperfect data.

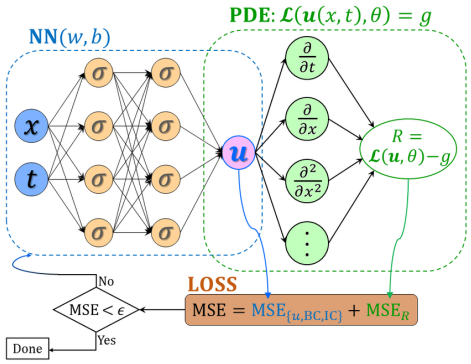

Figure 1. Principle of Physics-Informed Neural Networks [15].

As shown in Figure 1, a PINN consists of three main components:

- An approximation module: the neural network $NN(w,b)$, parameterized by weights w and biases b, takes as inputs x and t. Through nonlinear transformations governed by the activation function σ, the network outputs an approximation of the solution $u(x,t)$.
- A physics-informed module, where the predicted solution is inserted into the governing Partial Differential Equation (PDE), expressed as $\mathcal{L}(u(x,t),\theta) = g$. The derivatives of u with respect to t and x are computed automatically through differentiation of the neural network. These derivatives allow evaluation of the residual $R = \mathcal{L}(u,\theta) - g$, which measure the discrepancy between the network prediction and the governing equation.
- An optimization module, responsible for minimizing the loss and ensuring convergence toward a physically consistent solution. The loss is expressed as the Mean Squared Error (MSE), defined as: $MSE = MSE_{\{u,BC,IC\}} + MSE_R$. The term $MSE_{\{u,BC,IC\}}$ measures the discrepancy between the predicted solution and the available data, including Boundary Conditions (BC) and Initial Conditions (IC). The second term, MSE_R, evaluates the residual of the PDE. By combining both contributions, the optimization process balances fidelity to the observed data with consistency to the underlying physics, and training continues iteratively until the overall loss drops below a prescribed tolerance ϵ.

While this structure provides a powerful framework, choosing an appropriate NN architecture and parameters can be a significant challenge due to the vast number of available options. As highlighted in Section III, a wide variety of PINN architectures have been proposed across different industries and engineering domains. Without extensive experimentation, it is often difficult to determine the most effective configuration, making the process time-consuming. To address this, the present article explores various possible architectures to help guide the selection process based on the available data and input types.

B. Structure and Parameters

To select the most suitable PINN architecture for a given problem, several key aspects must be considered:

- Network Type: The architecture should match the task and the nature of the data.
- Depth and Width: The number of layers and neurons per layer, along with the choice of activation functions, significantly influence the model's ability to learn complex representations.
- Optimization Methods: The choice of optimizer and hyperparameters, such as the learning rate, plays a crucial role in the model's convergence.
- Batch Size and Regularization: Batch size, regularization strategies (e.g., dropout, L2 regularization), and data normalization are essential for efficient and stable training.
- Performance Enhancements: Techniques like data augmentation and early stopping can improve model performance and generalization.
- Hyperparameter Tuning: Methods such as Grid Search, Random Search, and Bayesian Optimization help in finding the optimal network configuration.
- Hardware Considerations: Available computational resources (e.g., GPUs, TPUs) and compatibility with deep learning frameworks like TensorFlow or PyTorch are important for scalability and efficiency.

In summary, designing an effective PINN architecture requires a strategic combination of these elements to ensure optimal performance.

III. Literature Review on Physics-Informed Neural Network Architectures

Several neural network architectures have been adapted for use in PINNs, including: Fully Connected Neural Networks (FCNNs) [16], Convolutional Neural Networks (CNNs) [8], Recurrent Neural Networks (RNNs) [13], Long Short-Term Memory networks (LSTMs) [12], Autoencoders (AEs) [17], Generative Adversarial Networks (GANs) [10], Bayesian Neural Networks (BNNs) [9], Graph Neural Networks (GNNs) [11], and Residual Networks (ResNets) [14], among others.

Among these, FCNNs are the most widely used. These networks are frequently employed to approximate solutions to scalar or vector-valued functions, such as PDEs and Ordinary Differential Equations (ODEs).

CNNs are designed to efficiently process grid-based data, such as images. They are widely used in tasks like image classification, object detection, and image segmentation, and are also applicable to problems involving spatially structured data, such as velocity fields in fluid mechanics or temperature distributions.

RNNs are a type of network where connections between nodes form directed cycles, allowing information to persist over time. This architecture is particularly well-suited for tasks involving sequential data, such as time series forecasting, natural language processing, and speech recognition. By retaining a form of memory, RNNs can use past information to influence current predictions, which is essential for understanding and generating sequences where context and order matter.

LSTMs, a specialized type of RNN, are designed to address the vanishing gradient problem in traditional RNNs. They incorporate gating mechanisms to better capture long-term dependencies in sequential data, making them especially effective for tasks such as speech recognition, machine translation, and sentiment analysis.

GNNs are designed to process data structured as graphs and are applied to systems where relationships between entities are important, such as transportation networks or molecular interactions.

Autoencoders are often used for dimensionality reduction and anomaly detection in complex physical systems, as well as for data denoising and generative modeling.

GANs are widely used to generate realistic images, videos, and other types of data, or to produce physically plausible solutions that respect physical constraints.

Bayesian Deep Learning (BDL) is an approach that integrates Bayesian methods into deep neural networks to quantify uncertainty in predictions. Unlike traditional neural networks, which provide deterministic predictions, Bayesian models generate probability distributions over model parameters, allowing the uncertainty associated with each prediction to be assessed. BDL is particularly useful in domains where decisions must be made cautiously and where prediction uncertainty can have significant consequences.

Finally, ResNets are used for tasks requiring very deep networks, enabling more stable training and improved performance through residual connections.

IV. Ontology-Based Recommender Systems

In an increasingly data-driven world, organizing knowledge in a structured and meaningful way is essential for understanding, sharing, and reusing information. Whether in science, business, or technology, we need systems that help us make sense of complex domains. To achieve this, various techniques have been developed to organize knowledge, each with different levels of complexity and expressiveness. These include taxonomies, ontologies, and knowledge graphs [18].

A taxonomy is the simplest form of knowledge organization. It arranges concepts in a hierarchical structure, typically from general to specific, using parent-child relationships [18]. Ontologies provide a more expressive and formal way to represent knowledge. They define concepts, properties, relationships, and rules within a domain, enabling both humans and machines to reason about the data. Ontologies are essential in fields like artificial intelligence, semantic web, and biomedical informatics [18]–[20]. Knowledge graphs extend ontologies by linking entities and their relationships in a graph structure [20].

For our project, we aim to develop an ontology-driven recommender system that suggests the most suitable Neural Network (NN) architecture based on the user's data type and task. This system combines a formally structured ontology, representing relationships between NN types, data modalities, and task categories, with a Python-based reasoning and querying engine. The ontology enables semantic inference, while the Python system handles user input, executes reasoning logic, and delivers recommendations.

Despite the growing interest in semantic technologies for intelligent systems, no published scientific work to date appears to directly implement an ontology-based recommender system specifically designed to suggest NN architectures based on the type of data and the task to be accomplished.

While there are ontologies that describe machine learning concepts, such as the Machine Learning Schema (MLS) [21], these are primarily intended for metadata annotation, experiment tracking, or model documentation. They do not aim to support reasoning or recommendation of neural network architectures based on task and data characteristics, which is the focus of our work.

V. Results

A. Guiding PINN Architecture Selection Through Input Analysis

The developed workflow, named PAIRS (for Physics-enabled AI for Real-time simulations Surrogates), allows users to input differential equations along with ICs and BCs, which can be integrated into the loss function. In addition, users provide data and specify the type of task to be accomplished.

Differential equations play a significant role in optimizing the learning process. However, the type of these equations (whether ODEs, PDEs, or Integro-Differential Equations (IDEs)) does not influence the choice of the PINN architecture. In contrast, the types and properties of the data, along with the tasks to be performed, play a crucial role in this choice.

Data can be classified into quantitative and qualitative types, as well as more advanced forms. Quantitative data refers to measurable information expressed numerically, such as discrete or continuous numerical data, and time series data (linear or nonlinear). Qualitative data includes descriptive information that cannot be measured numerically but can be categorized or described, such as categorical data, text, images, audio, and video. Sensor data may be either quantitative or qualitative, depending on what is being measured. Graph data is represented as graphs composed of nodes (or vertices) and edges (or links) connecting them. This format is particularly useful for representing complex relationships between entities.

These data types may exhibit various characteristics that influence their processing and analysis, such as:

- Temporal dependency (Dynamic Data): Data evolves over time, with short-term or long-term dependencies. This distinction is crucial for choosing between models like LSTM and RNN.
- Probabilistic nature: Data may contain uncertainty or variability.
- High dimensionality: Data with many features or variables, making processing more complex.
- Noise: Data may include errors, inconsistencies, or irrelevant information.
- Heterogeneity: Data from diverse sources and formats, requiring normalization for coherent analysis.
- Large volume: The data may be massive in scale.

PINNs are primarily applied to solving forward and inverse differential equations, including ODEs, PDEs, integro-differential, and stochastic equations, commonly encountered in physics and engineering. However, PINNs can also be applied to many other tasks. Some of these tasks influence the choice of the most appropriate PINN architecture, while others can be addressed with any architecture without a specific preference. Tasks that influence architecture choice include:

- Solving Differential Equations
- Inverse Problems:
  - Model Discovery: Identifying underlying models or physical laws from data
  - Parameter Estimation: Estimating unknown parameters in physical or statistical models from observed data
- Sequence Prediction: Forecasting future values or sequences based on time series or sequential data.
- Capturing Long-Term Dependencies: Modeling long-range dependencies in sequential or temporal data, important in time series forecasting or text analysis.
- Noise Reduction: Cleaning noisy data to recover the original signal, using techniques like autoencoders for image, audio, or other data types.
- Data Generation: Creating synthetic data from a learned model, especially when real data is scarce or expensive.
- Dimensionality Reduction: Reducing the number of variables while preserving important features
- Uncertainty Quantification: Estimating the uncertainty in model predictions.
- Preventing Degradation in Deep Neural Networks: Enhancing stability and performance in deep networks to avoid degradation during training, e.g., using ResNets to address vanishing gradients.

Other tasks, such as classification, predictive maintenance, and anomaly detection may also arise. For these, the choice of architecture primarily depends on the data type.

B. Ontology Development and Integration

Protégé [22], a free and open-source ontology editor, is used to develop the ontology. Created at Stanford University, Protégé is widely adopted in the semantic web community. It supports the creation and editing of ontologies in various formats, including RDF, RDFS, and OWL.

As shown in the Figure 2, in the first stage of ontology construction, we defined three main classes: Data, Task, and Neural Net Type. The Data class includes two subclasses: Type and Characteristics. The data types considered include: numerical data, sequences (e.g., time series), text, images, audio, video, and graphs. The characteristics include: temporal dependency, probabilistic nature, high dimensionality, heterogeneity, and data volume. For the Task class, only tasks that influence the choice of neural network architecture are considered. These include: solving differential equations and inverse problems, sequence prediction, capturing long-term dependencies, noise reduction, data generation, dimensionality reduction, uncertainty quantification, and preventing degradation in deep neural networks. This class represents the user's task preferences and requirements. The Neural Network class includes the architectures shown in Figure 2: FCNN, CNN, RNN, LSTM, AE, BNN, GAN, GNN, and ResNet. Each class is defined or described in detail using Annotations in Protégé. To establish relationships between these classes, four object properties were defined:

- areBestSuitedForData: Links a neural network type to data types or characteristics it is particularly well-suited for. For example, CNNs are best suited for image data, GNNs for graph-structured data, and RNNs for sequential data with temporal dependencies.
- canBeUsedForData: Also links neural networks to data types or characteristics they can be applied to, though not necessarily in an optimal way.
- areBestSuitedForTask: Indicates that a neural network is particularly well-suited for a specific task.
- canBeUsedForTask: Indicates that a neural network can be used for a given task.

The ontology is developed in OWL format, and the HermiT reasoner is used to validate its logical consistency by inferring implicit relationships and identifying contradictions. This involves checking that the defined classes, properties, and restrictions do not lead to inconsistencies in class hierarchies or instances. Queries are also executed using the DL Query tool in Protégé to ensure that the defined relationships and

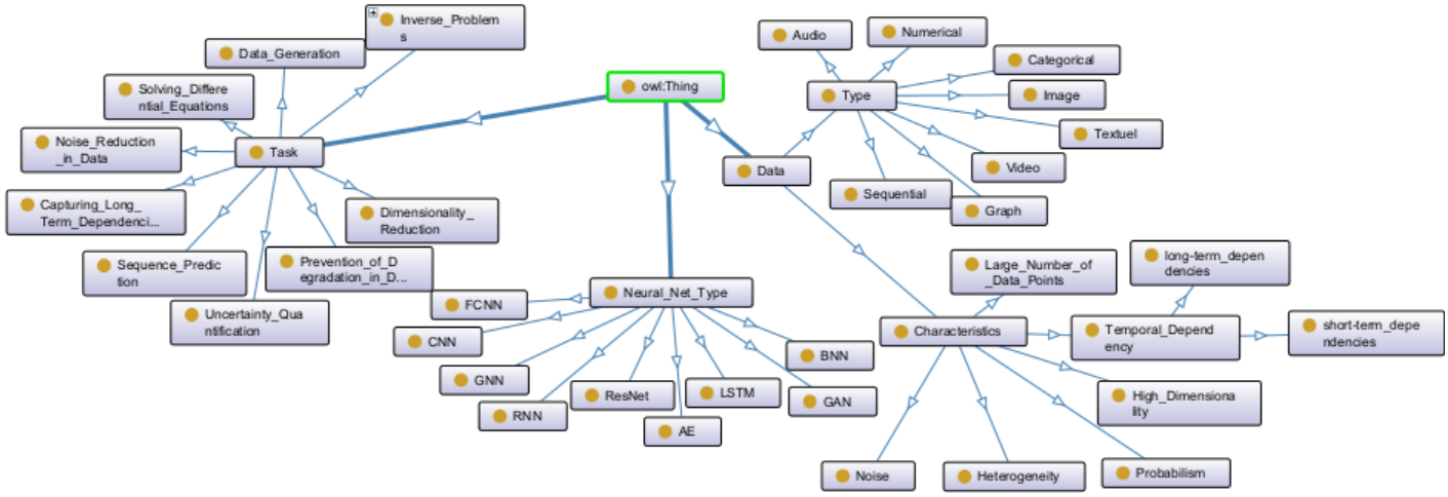

Figure 2. Ontology class diagram generated with OntoGraf, showing the main concepts: Data Types and Characteristics, Tasks, and Neural Network Types, along with their respective subclasses.

properties yield the expected results. An example of these queries is shown in Figure 3. All tests conducted during this phase confirmed that the ontology produces the expected results, both in terms of logical reasoning and query outcomes, reinforcing its reliability as the foundation for the recommendation system. Figure 4 illustrates the complexity of the ontology, with arrows indicating the links between classes based on the defined properties.

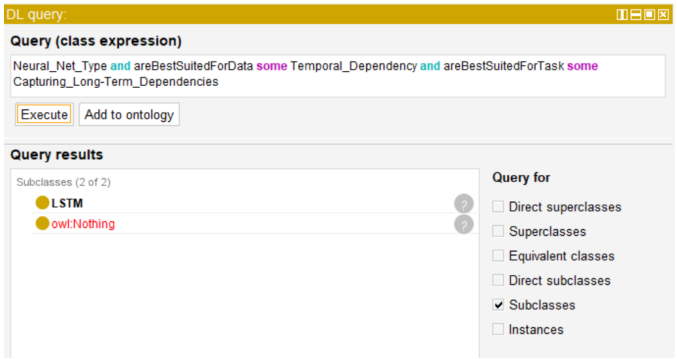

Figure 3. Example of DL queries used to test the ontology.

C. System Implementation and Recommender Workflow

To develop the recommendation system, user inputs are first connected to the ontology to extract relevant information. This linkage enables the system to derive meaningful insights and generate appropriate model recommendations. The implementation is carried out in Python, using the Owlready2 library to load and query the ontology.

The process begins with a user interface developed using Streamlit, which allows users to input their data, equations, IC, and BC. The interface also prompts users to specify the task they aim to accomplish by selecting from a set of predefined options.

Once the inputs are provided, the system analyzes the data to identify its type and specific characteristics. These are then mapped to corresponding concepts in the ontology. Based on this mapping and the selected task, the system generates a primary recommendation using the property areBestSuitedForData/Task, and suggests alternative models through the property canBeUsedForData/Task. Users can then select one of the recommended models that best fits their needs, guided by detailed descriptions that include relative levels of resource and time requirements.

After a model is selected, additional parameters—such as the number of layers, number of neurons, learning rate, and activation function—are either set to default values or defined as ranges for exploration through hyperparameter optimization techniques. The model is then implemented and trained using PyTorch. During training, validation, and testing, the user-provided equations and conditions (if available) are incorporated into the loss function alongside the data loss, ensuring that the model respects the underlying physics. These steps can follow a standard MLOps workflow. Once the model is trained and validated, it can be exported and deployed in its target environment for further testing and integration.

VI. Conclusion and Perspectives

This work presents a foundational version of an ontology designed to support the development of physics-informed AI surrogates, which aim to provide an alternative to complex, time-consuming simulations. While this initial version provides a structured framework, it remains a first iteration that will require further refinement. In particular, future enhancements should include the integration of additional types of hybrid models to better reflect the diversity of approaches in physics-enabled machine learning.

The ontology was constructed following an extensive literature review aimed at identifying the key factors influencing the selection of neural network architectures in PINNs. The

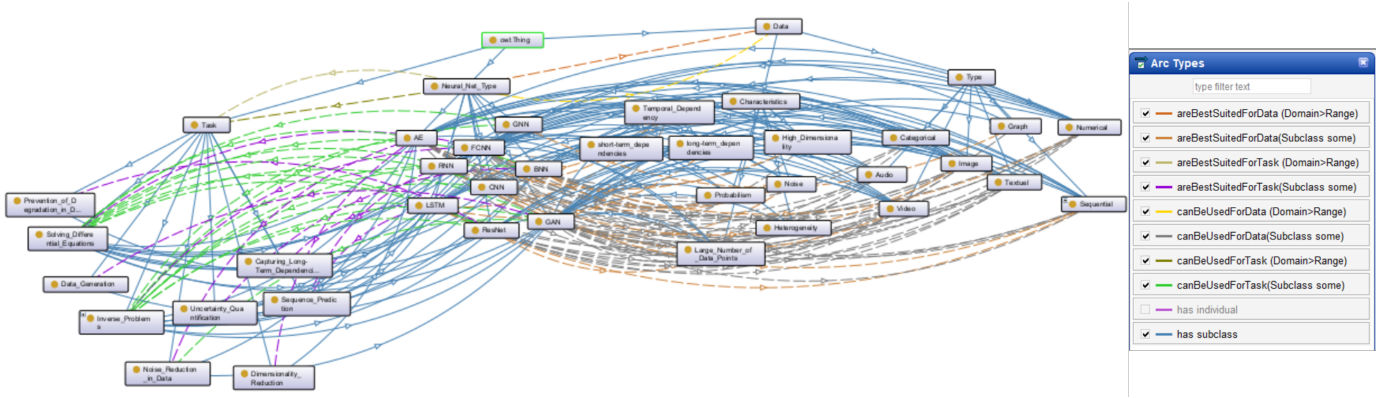

Figure 4. Diagram illustrating the complexity of the ontology, with arrows indicating the links between classes based on the defined properties.

review involved collecting and analyzing a wide range of studies that applied different architectures in various industrial and scientific contexts. Some of the relevant studies reviewed during this process are mentioned in Section III. Each study was examined based on architecture type, input/output structure, data characteristics, governing equations, task type, and application domain. Technical parameters such as activation functions, number of layers and neurons, optimization algorithms, learning rates, and performance metrics were also considered. This structured analysis enabled the classification of use cases and the identification of patterns linking specific architectures to particular contexts.

This study revealed that several aspects play a role in this decision, including the nature and characteristics of the data, the volume of available data, the complexity of relationships within the data (e.g., linear, nonlinear, or intricate patterns), the specific objectives of the modeling task, and the availability of computational resources and time constraints. The ontology serves as a structured framework that links the neural network, the data, and the task. Building on this foundation, a recommendation system suggests the most appropriate models to use, indicating the relative levels of resource and time requirements as well as the level of model complexity.

The ontology is currently in its first version and accessible only within our internal project environment. It is implemented in OWL format and can be edited using the Protégé tool, allowing for easy updates and integration with additional models and parameters. Although the development is ongoing and the full workflow is not yet complete, the ontology has been designed with extensibility in mind to support future enhancements and broader system integration. However, its current scope is limited to PINNs and does not yet cover the full spectrum of physics-informed machine learning models. Moreover, even within the PINNs domain, the rapid evolution of architectures presents a challenge for keeping the ontology up to date. Another challenge is integrating this approach into existing MLOps workflows, which can be complex due to the need to manage physics-based constraints and ensure alignment with the ontology.

This contribution is part of a broader initiative aimed at streamlining and automating the entire process of building physics-enabled AI surrogates. The primary goal is to empower domain experts, such as simulation engineers, to develop and deploy surrogate models without needing deep expertise in AI.

Several key directions are planned to extend this work:

- Automatic updating of the ontology using Generative AI (GenAI) to reflect evolving model types, as new models and approaches are constantly emerging in this rapidly evolving domain.
- Development of a database of pretrained models, facilitating reuse and accelerating deployment.
- Integration of GenAI and intelligent agents to assist users in model selection, configuration, and training.

These future developments aim to create a comprehensive, user-friendly ecosystem for building and managing physics-informed AI models, ultimately bridging the gap between simulation expertise and advanced machine learning capabilities.

REFERENCES

[1] S. Koziel and A. Pietrenko-Dąbrowska, "Basics of data-driven surrogate modeling," in *Performance-Driven Surrogate Modeling of High-Frequency Structures*. Springer International Publishing, 2020, pp. 23–58, ISBN: 978-3-030-38925-3. DOI: 10.1007/978-3-030-38926-0_2.

[2] G. Marcus, "Deep learning: A critical appraisal," Jan. 2018. DOI: 10.48550/arXiv.1801.00631. arXiv: 1801.00631.

[3] N. Thompson, K. Greenewald, K. Lee, and G. Manso, "The computational limits of deep learning," in *Ninth Computing within Limits 2023*, LIMITS, Jun. 2023. DOI: 10.21428/bf6fb269.1f033948.

[4] I. Pan, L. Mason, and O. Matar, "Data-centric engineering: Integrating simulation, machine learning and statistics. challenges and opportunities," *Chemical Engineering Science*, vol. 249, p. 117 271, Nov. 2021. DOI: 10.1016/j.ces.2021.117271.

[5] G. Karniadakis *et al.*, "Physics-informed machine learning," *Nature Reviews Physics*, pp. 1–19, May 2021. DOI: 10.1038/s42254-021-00314-5.

[6] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019. DOI: 10.1016/j.jcp.2018.10.045.

[7] J. Pateras, P. Rana, and P. Ghosh, "A taxonomic survey of physics-informed machine learning," *Applied Sciences*, vol. 13, no. 12, 2023, ISSN: 2076-3417. DOI: 10.3390/app13126892.

[8] X. Zhao, Z. Gong, Y. Zhang, W. Yao, and X. Chen, "Physics-informed convolutional neural networks for temperature field prediction of heat source layout without labeled data," Sep. 2021. DOI: 10.48550/arXiv.2109.12482.

[9] L. Yang, X. Meng, and G. E. Karniadakis, "B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data," *Journal of Computational Physics*, vol. 425, p. 109 913, Jan. 2021, ISSN: 0021-9991. DOI: 10.1016/j.jcp.2020.109913.

[10] W. Li, C. Zhang, C. Wang, H. Guan, and D. Tao, *Revisiting pinns: Generative adversarial physics-informed neural networks and point-weighting method*, 2022. DOI: 10.48550/arXiv.2205.08754. arXiv: 2205.08754.

[11] D. Dalton, D. Husmeier, and H. Gao, "Physics-informed graph neural network emulation of soft-tissue mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 417, p. 116 351, Dec. 2023. DOI: 10.1016/j.cma.2023.116351.

[12] F. Liu, J. Li, and L. Wang, "Pi-lstm: Physics-informed long short-term memory network for structural response modeling," *Engineering Structures*, vol. 292, DOI: 10.1016/j.engstruct.2023.116500.

[13] Y. Zheng, C. Hu, X. Wang, and Z. Wu, "Physics-informed recurrent neural network modeling for predictive control of nonlinear processes," *Journal of Process Control*, vol. 128, p. 103 005, 2023, ISSN: 0959-1524. DOI: https://doi.org/10.1016/j.jprocont.2023.103005.

[14] T. Shan *et al.*, "Physics-informed supervised residual learning for electromagnetic modeling," *IEEE Transactions on Antennas and Propagation*, vol. PP, pp. 1–1, Apr. 2023. DOI: 10.1109/TAP.2023.3245281.

[15] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, "Ppinn: Parareal physics-informed neural network for time-dependent pdes," *Computer Methods in Applied Mechanics and Engineering*, vol. 370, p. 113 250, Oct. 2020, ISSN: 0045-7825. DOI: 10.1016/j.cma.2020.113250.

[16] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112 789, Mar. 2020. DOI: 10.1016/j.cma.2019.112789.

[17] C. Tan, Y. Cai, H. Wang, X. Sun, and L. Chen, "Vehicle state estimation combining physics-informed neural network and unscented kalman filtering on manifolds," *Sensors*, vol. 23, no. 15, Jan. 2023. DOI: 10.3390/s23156665.

[18] T. Salatino A.and Aggarwal, A. Mannocci, F. Osborne, and E. Motta, "A survey of knowledge organization systems of research fields: Resources and challenges," *Quantitative Science Studies*, 2025. DOI: 10.1162/qss_a_00363.

[19] K. W. Fung and O. Bodenreider, "Knowledge representation and ontologies," *Clinical Research Informatics*, pp. 255–275, Jan. 2012. DOI: 10.1007/978-1-84882-448-5_14.

[20] Nature Research Intelligence, *Knowledge graphs and ontologies in semantic web applications*, https://www.nature.com/research-intelligence/nri-topic-summaries/knowledge-graphs-and-ontologies-in-semantic-web-applications-micro-92, Accessed: 2025-08-18.

[21] J. Braga, J. Dias, and F. Regateiro, *A machine learning ontology*, Oct. 2020. DOI: 10.31226/osf.io/rc954.

[22] M. A. Musen, "The protégé project: A look back and a look forward," *AI Matters*, vol. 1, no. 4, 2015. DOI: 10.1145/2557001.25757003.