From Theory to Superiority: GO IML for Strategic AI Advantage

Whitepaper 1.0

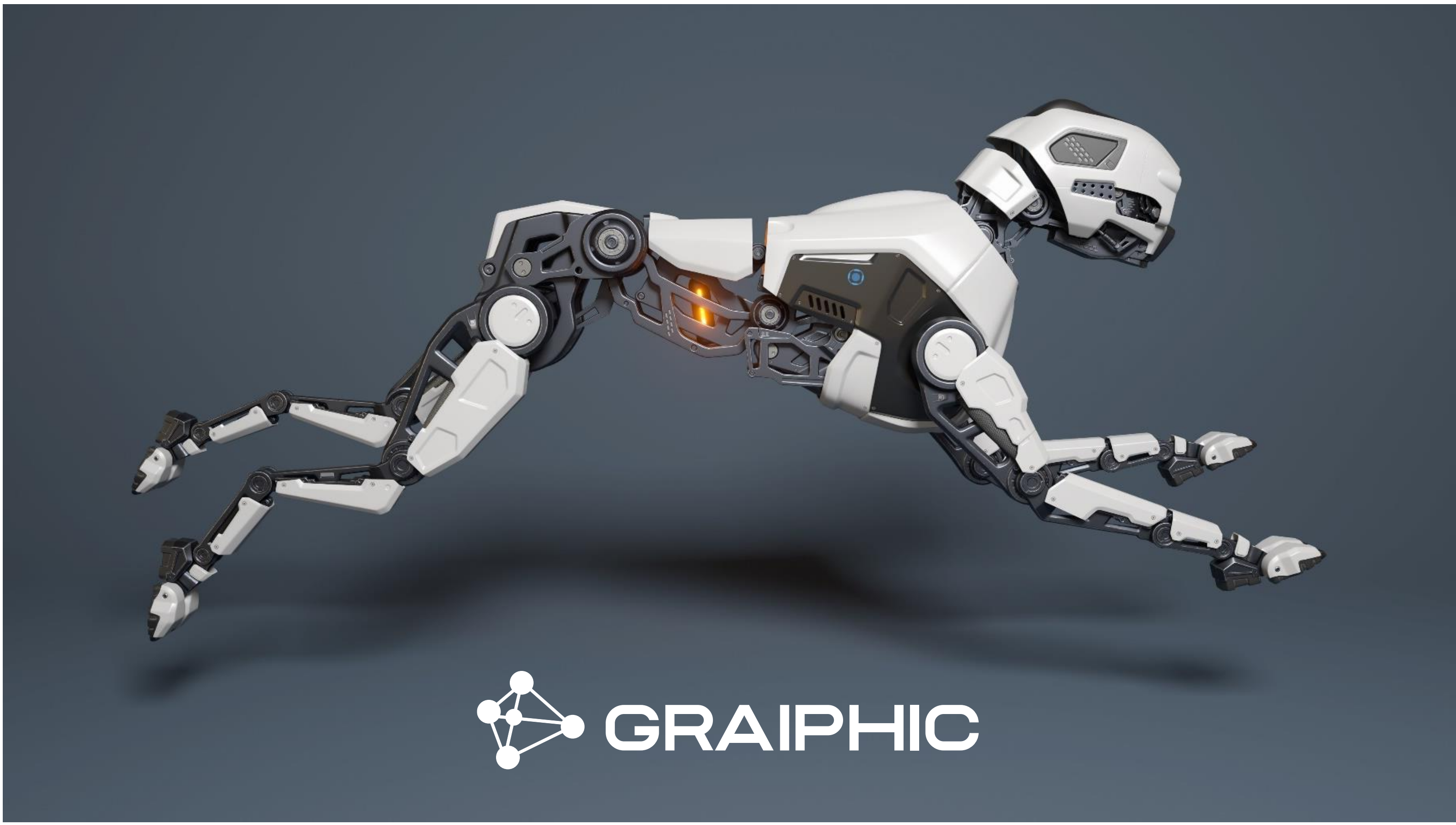


Table of Contents

Versioning

This document is subject to version control to ensure full traceability of changes. Each update is recorded with its author, date, and a short description of the modifications.

Version	Date	Author	Organization	Change Description
1.0	2025/12/06	Youssef Menjour	Graiphic	First publication of the GO HW Whitepaper

Executive Summary

Imagine a world where your AI doesn't just learn but understands. Where training begins not from randomness but from the physical laws, domain logic, and expert knowledge you already possess. GO IML is Graiphic's bold proposal inside the SOTA framework, envisioning a new era of machine learning where intelligence is informed, structured, and seamlessly deployable.

GO IML stands for Graph Orchestration Informed Machine Learning. It is one of the visionary technological pillars that Graiphic proposes to develop, alongside GO HW which explores universal graph-to-hardware compilation for AI acceleration, and GO GenAI which aims to integrate generative models into decision-making architectures. GO IML focuses on enabling deep learning models to incorporate formal knowledge, logic, and physical constraints directly inside the learning graph.

Built on the deep learning toolkit of SOTA, GO IML introduces a new kind of abstraction. It transforms prior knowledge into a native element of the training process. With GO IML, every constraint, every equation, every logical rule becomes a powerful signal for learning. It represents a strategic shift that brings speed, robustness, and trust to the forefront.

Everything happens inside a unified ONNX graph. The same graph you build visually in LabVIEW becomes the model you train, optimize, and deploy. Whether your target is a Jetson board, a high-performance server, or an embedded FPGA, your model remains intact and operational. GO IML aims to make informed learning fully compatible with ONNX, making knowledge-driven AI portable by design.

This is Physical AI applied. Your models now integrate real-world dynamics, formal structure, and causal reasoning directly into the training loop. They converge faster. They require less data. They perform better in unseen situations. They deliver results that are not only accurate but also explainable.

GO IML is a solution for engineers who want more than data fitting. It brings a new advantage, a new gain, a new capability. It removes the friction between theory and practice. It proposes a path to guaranteed results where traditional models struggle. It surprises by its simplicity and promises impactful performance.

And what if a robot, unable to run across a slippery industrial floor, could instantly receive a constraint-informed update that adapts its gait in real time? What if a drone mid-mission, encountering shifting wind patterns and unexpected rain, could modify its inference graph based on meteorological models and complete its objective with precision?

That is the promise of GO IML. Knowledge-driven, dynamically adaptive, and tactically aware machine learning. Not just intelligent, but operationally effective.

This is not just another concept. This is a concrete vision, an integrated, visual, and modular extension of the SOTA ecosystem. It aims to empower developers to train AI with physics, logic, and domain knowledge without sacrificing portability, performance, or control.

Welcome to the future of tactical AI. Welcome to GO IML.

Why Now? Why GO IML?

The timing for GO IML is not accidental. Machine learning has reached a paradox. Models are bigger than ever, datasets are larger than ever, and yet many real-world projects still fail not because of a lack of algorithms, but because of a lack of usable data, trust, and control. The classic supervised learning recipe works brilliantly in web scale environments, but struggles in constrained industrial and physical environments where every sample is expensive and every mistake has a cost.

In many sectors engineers already know a lot about their systems. They know the equations of motion of a robot, the safety margins of a valve, the thermodynamics of a cooling loop, the logical rules of a mission plan. Today this knowledge lives in reports, models, and human expertise, while neural networks are trained as if nothing of this existed. GO IML proposes a different solution. Instead of treating prior knowledge as an afterthought, it turns it into a first class ingredient of the learning process.

At the same time a new wave is growing around Physical AI. Intelligence is leaving the cloud and moving into robots, drones, test benches, smart factories, and embedded devices. In these settings bandwidth is limited, real time constraints are strict, and retraining on huge datasets is not an option. What is needed is a tactical way to obtain better results with fewer samples, to update behaviour rapidly, and to guarantee that AI remains aligned with the physics and logic of the environment. This is exactly the gain that informed machine learning can deliver.

There is also a second pressure. Regulation, certification, and safety standards now demand AI systems that are explainable, auditable, and predictable. Black box models trained only on data are difficult to justify when a decision impacts safety or mission success. By encoding constraints, equations, and rules directly inside the learning graph, GO IML opens a new path. It becomes possible to show where the knowledge comes from, how it is used, and why a model respects specific limits. This is a unique advantage for domains where trust and accountability matter as much as raw accuracy.

From a tooling perspective the moment is ideal. With SOTA, Graiphic already provides a deep learning toolkit that uses ONNX as a native graph format and LabVIEW as a visual cockpit. GO HW explores how the same graphs can target diverse hardware. GO GenAI explores how graphs can connect generative models to decision logic. GO IML naturally extends this ecosystem. It adds the missing abstraction for informed learning directly inside the same orchestration layer. No new language, no separate engine, the same ONNX model enriched with knowledge aware components.

This combination creates a surprising opportunity. A single graph can now describe data flows, control flows, hardware interactions, and informed constraints. Engineers gain a unified environment where they can design, train, and deploy models that are both data driven and knowledge driven. The result is a new kind of solution. Faster training, better generalization, more robust behaviour in edge situations, and an easier path to validation and certification. All of this remains portable, because the graph is still an ONNX artifact that can run on many platforms.

In short, GO IML arrives at the intersection of three strong trends. The need for data efficient learning, the rise of Physical AI in the field, and the maturity of graph based deep learning toolchains like SOTA and ONNX. This convergence makes GO IML not only relevant, but strategically timely. It offers a concrete, tactical way to upgrade existing deep learning workflows into informed machine learning workflows, with clear advantages in performance, trust, and deployment flexibility.

What Is GO IML?

GO IML is a proposal for a new way to practice deep learning inside SOTA. At its core, it treats a model not only as a function that maps inputs to outputs, but as a living graph where data, knowledge, and control interact. The name says it clearly: Graph Orchestration Informed Machine Learning. A learning system where the graph is the central object and where information does not come only from data but also from what we already know about the world.

In practical terms GO IML is a set of abstractions added to the SOTA Deep Learning Toolkit. These abstractions let engineers express constraints, physical laws, symbolic rules, and expert knowledge directly inside the training graph. They are not comments on the side, they are nodes and structures inside the ONNX model itself. The goal is simple: turn informed machine learning into a concrete solution that fits naturally into existing SOTA workflows.

GO IML can be understood through three simple questions.

First, what does it change for the model. Instead of a pure neural network trained only on samples, the model becomes a composition of standard operators and knowledge aware components. A PhysicsConstraint node can evaluate an equation on intermediate outputs. A RuleGate node can enforce logic on predictions. A PriorLoss node can penalize violations of known behaviour. Together they act as an additional source of signal during training. The gain is a model that uses data and knowledge at the same time, with better generalization and more predictable results.

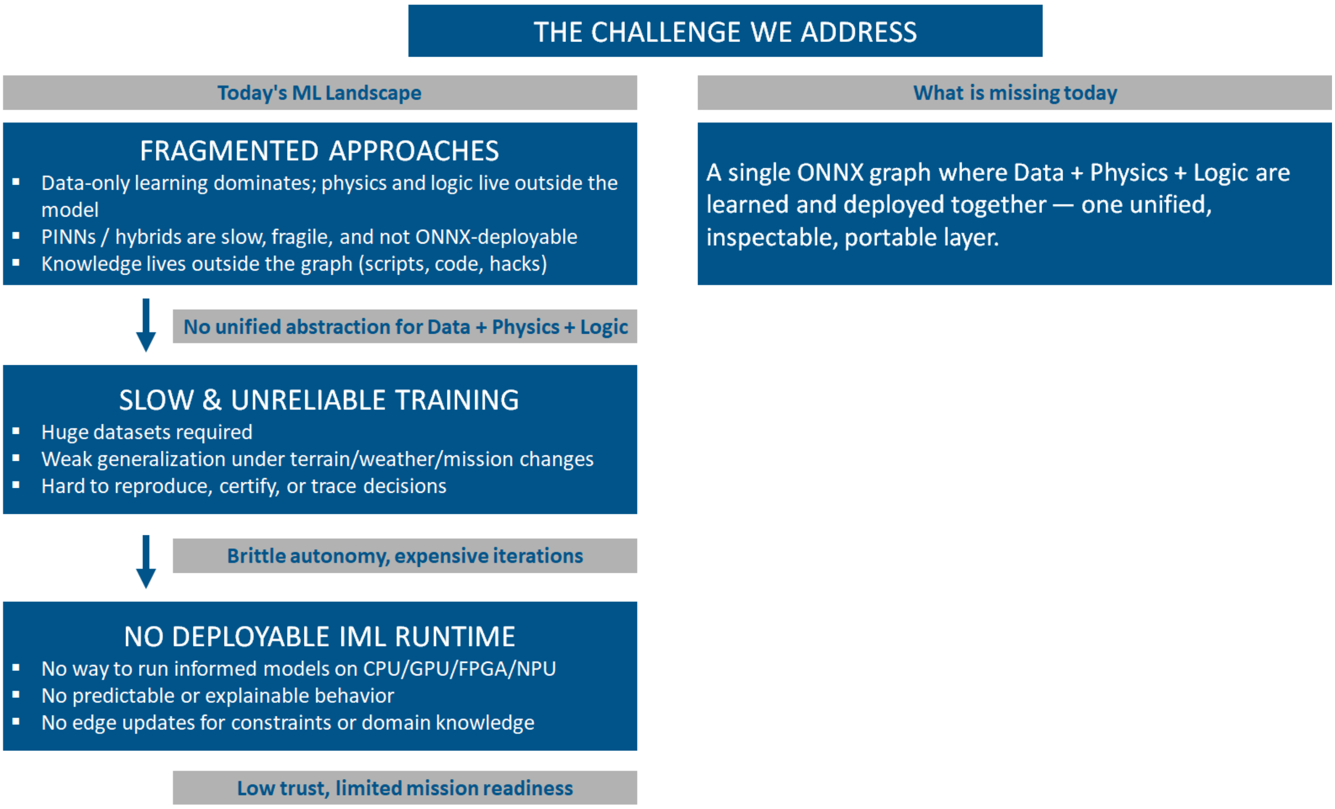
Second, what does it change for the user. For an engineer working in SOTA, GO IML appears as a new family of blocks in the LabVIEW palette or in the graph editor. The user can drag and drop them into the same pipeline used for classic supervised learning. A loop that previously contained only a training pass and a validation pass can now include constraint evaluation, physical residual computation, or ontology checks. The surprise is that informed learning does not require a new language. It becomes a visual and modular extension of the tools already in use.

Third, what does it change for the system. Because everything is encoded in ONNX, GO IML keeps a unique advantage. The informed model remains portable. The same file can be executed on ONNX Runtime with different execution providers. A robot controller enriched with constraints can run on a compact GPU module. A diagnostic model with embedded rules can run on a server. A test bench model with physical residuals can run on a laptop. The deployment story stays the same, only the intelligence level of the graph changes.

Within the Graiphic vision GO IML sits next to GO HW and GO GenAI as a complementary pillar. GO HW explores how a graph can drive hardware and energy in a unified way. GO GenAI explores how a graph can connect foundation models to structured decision logic. GO IML explores how a graph can bring together data driven learning and informed constraints. All share the same design principle. One graph, many roles, and a clear orchestration layer on top.

From the outside GO IML can be described in one sentence. It is a unique way to make informed machine learning accessible, repeatable, and deployable through SOTA and ONNX. From the inside it is a collection of carefully designed abstractions that turn prior knowledge into a first class citizen of the deep learning toolkit. The result is a new category of solution. Smarter by design, tactical by nature, and ready to support the next generation of Physical AI applications.

This diagram summarizes the limitations of today's fragmented ML landscape, slow training, lack of generalization, and the absence of a deployable IML runtime and highlights the missing unified abstraction that GO IML introduces.



THE CHALLENGE WE ADDRESS

Today's ML Landscape

FRAGMENTED APPROACHES

- Data-only learning dominates; physics and logic live outside the model
- PINNs / hybrids are slow, fragile, and not ONNX-deployable
- Knowledge lives outside the graph (scripts, code, hacks)

No unified abstraction for Data + Physics + Logic

SLOW & UNRELIABLE TRAINING

- Huge datasets required
- Weak generalization under terrain/weather/mission changes
- Hard to reproduce, certify, or trace decisions

Brittle autonomy, expensive iterations

NO DEPLOYABLE IML RUNTIME

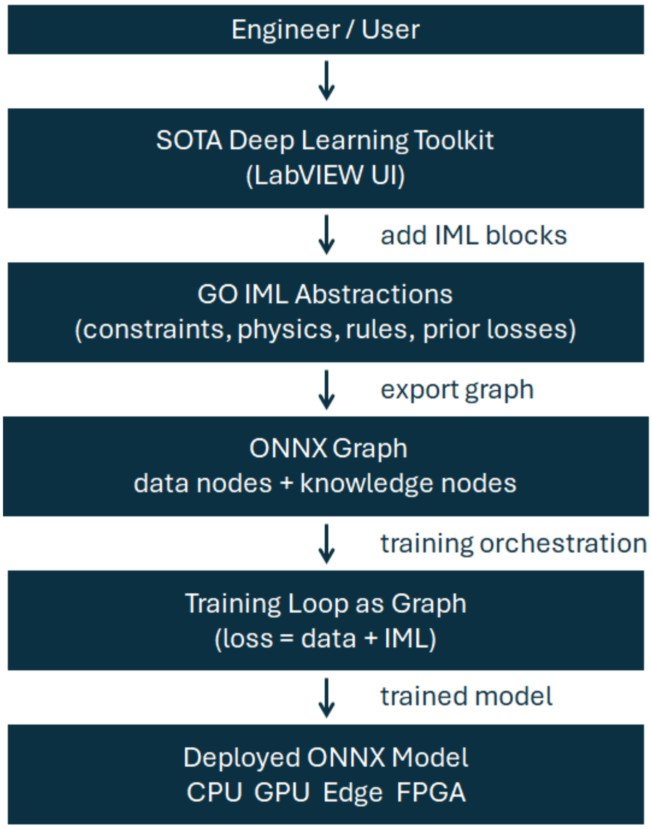
- No way to run informed models on CPU/GPU/FPGA/NPU
- No predictable or explainable behavior
- No edge updates for constraints or domain knowledge

Low trust, limited mission readiness

What is missing today

A single ONNX graph where Data + Physics + Logic are learned and deployed together — one unified, inspectable, portable layer.

How It Works: Inside the Graph



Unified Graph for Training and Deployment in GO IML

At the heart of GO IML there is a simple idea. Everything is a graph. Data flows are edges, operations are nodes, and knowledge becomes a first class citizen inside the same structure. SOTA already uses ONNX as a universal way to represent models. GO IML extends this by turning the ONNX graph into a full learning and reasoning space that can host both deep learning and informed constraints.

From a system point of view a GO IML pipeline can be seen as four main layers working together.

First comes the SOTA front end. Engineers design their models using the Deep Learning Toolkit in a visual way. Convolutions, linear layers, activations, normalizations, all appear as familiar blocks. With GO IML the palette gains new blocks dedicated to informed learning. Constraint blocks, physics blocks, rule blocks, prior loss blocks. The user does not have to think in terms of engines or runtimes. They simply place new building pieces in the same diagram that already hosts standard deep learning.

Second comes the graph abstraction. When the user validates the design, SOTA translates the whole pipeline into an ONNX graph. Classic layers become standard ONNX operators. GO IML elements become either compositions of existing operators or dedicated custom nodes with a clean interface. A PhysicsConstraint node might take predicted states and compute a residual relative to a differential equation. A LogicConstraint node might evaluate whether a set of outputs respects a rule. A KnowledgeLoss node might transform these residuals into penalties for the optimizer. The important result is that the full behaviour lives inside one consistent graph.

Third comes the training orchestration. GO IML relies on the graph orchestration capabilities of SOTA and ONNX. Training is no longer a black box loop hidden in a separate framework. Instead, the training process can be expressed as a graph with control flow. Inner loops for epochs and batches, conditionals for curriculum or schedule changes, scans for sequence processing. Within this structure the informed nodes participate naturally. At each step the graph produces predictions, evaluates data loss, computes knowledge based terms, aggregates them into a total objective, then backpropagates. The gain is a transparent learning pipeline where every piece of the objective is inspectable.

Fourth comes deployment. Once training is complete, the resulting graph still respects the ONNX standard. In many cases the informed components reduce to pure functions that no longer require gradients. The same file can then be deployed through ONNX Runtime on multiple execution providers. CPU, CUDA, TensorRT, OpenVINO, oneDNN, DirectML, and others. If some knowledge blocks are kept for inference, they run as ordinary computation nodes. If they are only used during training, they can be removed in a clean export step. Either way, the solution keeps a unique advantage. The path from lab to field is direct because the model remains an ONNX artifact.

This architecture is particularly suited for Physical AI. A robot controller can include a physics residual node that was active during training to enforce dynamics. A drone navigation model can embed a weather informed regularization that helped it learn safe trajectories. An industrial controller can keep a lightweight rule check node in the inference graph to guarantee that certain conditions are never violated. All of these behaviours are expressed inside one orchestrated graph rather than spread across scripts and ad hoc code.

For the user the surprise is how little extra complexity is required. GO IML does not demand a new mental model. It simply enriches the existing SOTA workflow with new blocks and new combinations. The tactic is to give engineers a familiar environment and to hide the difficult parts inside the abstraction. Under the surface ONNX and SOTA handle the heavy lifting. Graph rewriting, operator fusion, provider selection, memory planning. On the surface the user sees a clear cause and effect. Adding a constraint block changes the loss, accelerates convergence, improves robustness, and leads to better results.

In summary GO IML turns the ONNX graph into a living space where data, learning, and knowledge meet. The architecture is simple, but the implications are strong. It offers a unique way to implement informed machine learning without leaving the SOTA ecosystem, and it guarantees that every improvement remains portable, inspectable, and ready for deployment on real hardware.

What If: Tactical Scenarios

To understand the true potential of GO IML, it is useful to imagine how informed learning changes the behaviour of intelligent systems once deployed. These scenarios are not science fiction. They illustrate exactly what becomes possible when a model no longer learns only from data but also from the structure and physics of its environment.

Consider a robot operating in a factory. The floor changes from smooth concrete to a slippery composite surface. A standard neural controller, trained on past data alone, begins to struggle. It hesitates. It slips. It loses stability because it never saw this exact condition during training. Now imagine the same robot equipped with a model trained through GO IML. During training the model learned not only patterns from sensor data, but also constraints derived from friction laws and stability margins. When the surface changes, the controller instantly adapts. The internal graph already knows what the physics of motion allows and corrects the gait accordingly. The result is smooth, stable locomotion without additional data or retraining.

Picture a drone flying a reconnaissance mission. Weather conditions evolve unexpectedly. Wind becomes unstable. Light rain begins to fall. A purely data driven navigation model would attempt to extrapolate from patterns it learned months earlier, hoping they still apply. A GO IML enabled drone behaves differently. It incorporates aerodynamic constraints, mission rules, and meteorological models directly into its graph. As soon as the environment shifts, the model adjusts its decisions in a consistent and physically plausible way. It keeps its trajectory stable, respects mission boundaries, and completes its objective with confidence.

Imagine an industrial controller that monitors a complex system. Traditional machine learning can detect anomalies only if they resemble past data. GO IML offers a new approach. By embedding safety rules, logical relationships, and process equations inside the graph, the controller understands not only what happened before but what should never happen. When a rare condition appears, something that no dataset ever captured, the controller reacts instantly because its logic is guided by domain knowledge. It interprets deviations, predicts safe actions, and informs operators with explainable outputs.

Now think about a diagnostic model used in maintenance operations. A standard neural classifier may misinterpret a signal when the equipment ages or when the configuration changes. A model trained with GO IML integrates degradation equations and causal structure into its learning pipeline. When the system evolves, the model keeps its accuracy because its internal reasoning is anchored in physical understanding rather than statistical coincidence.

These scenarios highlight a central idea. GO IML enables models that behave tactically. They adapt. They generalize. They remain aligned with the real world even when the data distribution drifts. The informed constraints embedded in the graph give them a foundation that does not vanish when conditions change. This is not only a technical advantage. It is a strategic one. It turns learning systems into operational systems.

For missions, robots, controllers, and embedded intelligence, this creates a clear benefit. Better results. Less risk. More autonomy. More trust. And all of it powered by the same ONNX artifact that SOTA already deploys across CPUs, GPUs, FPGAs, NPUs, and edge devices.

GO IML transforms machine learning into a reliable teammate. A system that learns from data but acts with knowledge. A model that adapts in the field without retraining. A graph that understands the world it operates in. This is the essence of informed machine learning, and these scenarios show why the concept is not only relevant but necessary for the next generation of AI systems.

What We Gain: Strategic Advantages

GO IML transforms machine learning by merging two worlds that were previously separate. Instead of relying only on data, the model begins its learning process with the physics, logic, and structural knowledge already known about the environment. This hybrid approach changes everything about how a model learns, how fast it converges, and how well it performs once deployed.

The first major gain is speed. Integrating physical laws and domain constraints from the start drastically reduces the search space the model must explore. Rather than wandering through countless configurations that violate known principles, the optimization focuses directly on physically admissible regions. Training becomes faster, requires fewer iterations, and reaches stable solutions with far less computational effort. The result is immediate: shorter development cycles and more reliable convergence.

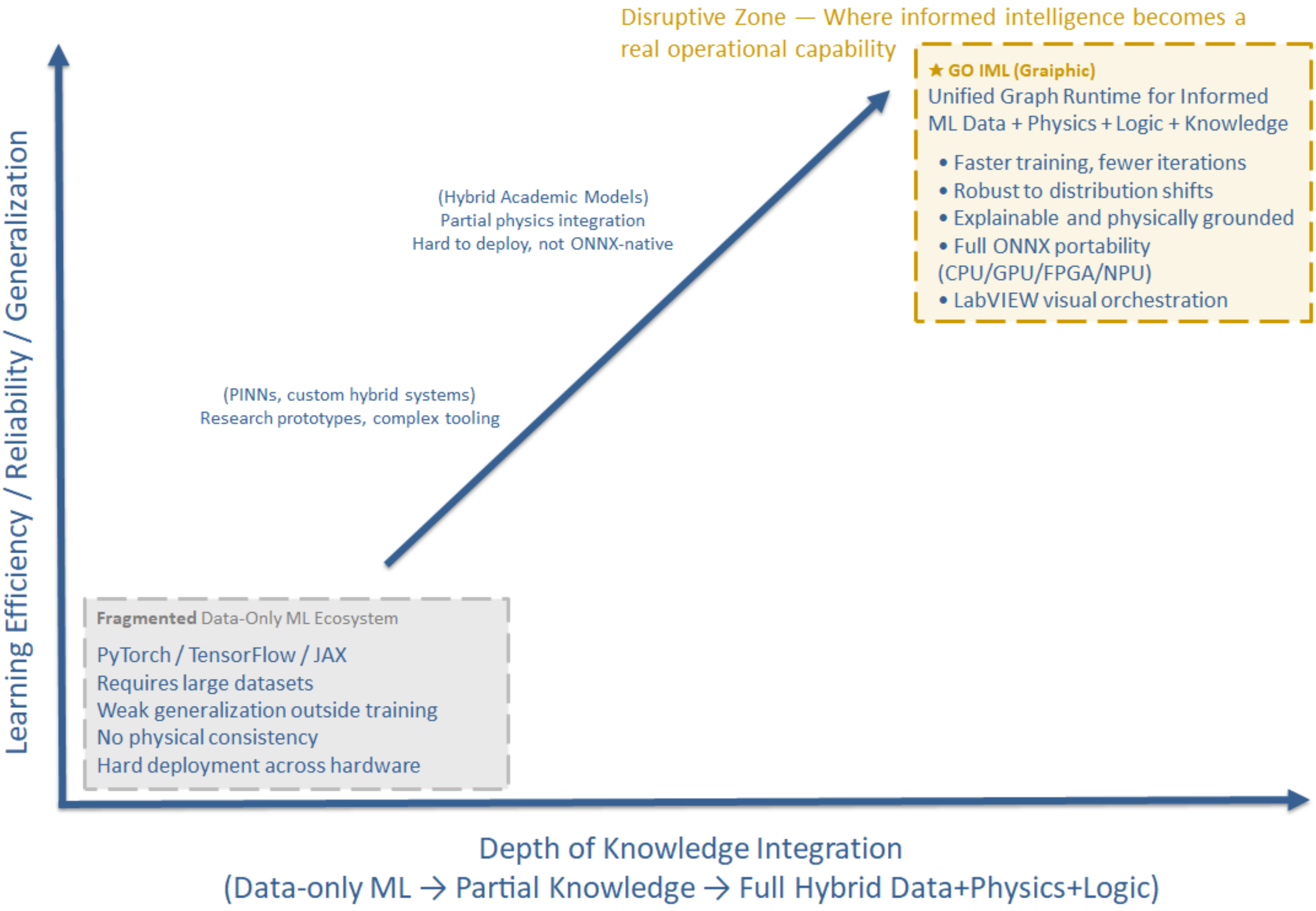
The second gain is quality. A model trained only on data can discover correlations but has no notion of what the world allows. A model trained with GO IML learns patterns and principles together. The data teaches variability. The physical model teaches structure. The constraints teach consistency. This produces a hybrid intelligence capable of respecting real-world limits while still capturing complex behaviours. Performance improves not only on the training distribution but also in unseen conditions where the physical rules remain valid.

The third gain is robustness. When the environment changes, purely data-driven models may fail because the new situation does not resemble anything in the dataset. Informed models behave differently. They rely on constraints that remain true even when the data changes. A robot encountering a new surface still knows the equations of motion. A drone facing turbulence still knows aerodynamic principles. An industrial system reacting to anomalies still knows the logical relations between variables. This gives GO IML models a tactical advantage in real-world deployment.

The fourth gain is explainability. Because knowledge is explicitly encoded in the graph, it is possible to trace predictions back to the constraints and principles that shaped them. This clarity is rare in neural networks and invaluable in regulated, safety-critical, or mission-critical environments. Decision makers understand why the model behaves as it does and what guarantees it offers.

The fifth gain is portability. GO IML remains fully embedded in the ONNX graph. The same informed model can run on CPUs, GPUs, FPGAs, NPUs, edge systems, or cloud accelerators. No custom runtime, no vendor lock-in. This ensures that informed learning is not confined to theory. It becomes deployable intelligence across the full spectrum of hardware supported by SOTA and ONNX Runtime.

Together, these advantages describe a new category of solution. A model that trains faster because it learns with guidance. A model that performs better because it integrates physics and logic. A model that adapts more reliably because its foundation is grounded in knowledge. A model that deploys everywhere because it remains an ONNX graph. This is the strategic value of GO IML and the reason it represents a major step toward true Physical AI.


Disruptive Zone — Where informed intelligence becomes a real operational capability
★ GO IML (Graiphic)
Unified Graph Runtime for Informed ML Data + Physics + Logic + Knowledge
• Faster training, fewer iterations
• Robust to distribution shifts
• Explainable and physically grounded
• Full ONNX portability (CPU/GPU/FPGA/NPU)
• LabVIEW visual orchestration
(Hybrid Academic Models)
Partial physics integration
Hard to deploy, not ONNX-native
(PINNs, custom hybrid systems)
Research prototypes, complex tooling
Fragmented Data-Only ML Ecosystem
PyTorch / TensorFlow / JAX
Requires large datasets
Weak generalization outside training
No physical consistency
Hard deployment across hardware
Learning Efficiency / Reliability / Generalization
Depth of Knowledge Integration
(Data-only ML → Partial Knowledge → Full Hybrid Data+Physics+Logic)

Unifying Data, Physics, and Logic into a Deployable AI

What Makes It Unique: Why It Matters

GO IML introduces a capability that does not exist today in practical industrial AI workflows. Many research papers describe informed learning. Many frameworks allow custom losses. Many libraries support physics-based models. However, none provide a unified, visual, deployable, ONNX-native orchestration layer where data, knowledge, and structure coexist inside a single graph. This is what makes GO IML unique.

The first unique point is integration. GO IML does not sit outside the model. It does not require rewriting algorithms or injecting code into hidden training loops. Instead, it becomes a natural extension of the SOTA graph. Physical constraints are expressed as nodes. Logical rules are expressed as graph operations. Prior knowledge is expressed as explicit terms in the learning process. The entire system uses the same language, the same representation, and the same deployment workflow.

The second unique point is accessibility. Most informed learning techniques remain confined to academia because they demand deep mathematical expertise or heavy custom implementations. GO IML reverses this barrier. Engineers interact with informed components through visual blocks and structured graph elements. They assemble knowledge-driven models the same way they design neural networks today. This simplicity opens informed learning to a much wider community and guarantees rapid adoption.

The third unique point is portability. In standard machine learning, informed components often rely on specialized physics engines or symbolic solvers that cannot be deployed easily. GO IML avoids this limitation by embedding everything inside the ONNX ecosystem. If a constraint can be expressed through ONNX operators, it becomes deployable everywhere. If it requires training-only logic, SOTA's orchestration layer handles it gracefully. This creates a new advantage. Informed learning remains compatible with every execution provider supported by ONNX Runtime.

The fourth unique point is strategic alignment with Physical AI. The world is moving toward systems that must understand and react to physical reality in real time. Robots, vehicles, drones, industrial controllers, smart test benches, and autonomous platforms all require intelligence that respects the laws of their environment. GO IML offers exactly this capability. It provides a structured way to embed real-world principles into machine learning models and to ensure that this knowledge influences both training and deployment.

The fifth unique point is its potential synergy with other Graiphic initiatives. GO HW explores how ONNX graphs can directly orchestrate hardware. GO GenAI explores how graph logic can direct generative models. GO IML adds a third dimension. It injects physics, logic, and structured reasoning into the learning process itself. The combination forms a powerful triad. A unified orchestration layer that spans intelligence, hardware, and knowledge. A foundation for a new kind of system-level AI.

The sixth unique point is its impact on reliability and certification. Many industries hesitate to adopt AI because models behave like black boxes. GO IML addresses this problem at its core. By encoding constraints and rules inside the graph, the model's behaviour becomes more predictable and more explainable. Each decision can be traced to both data-driven patterns and knowledge-driven principles. This transparency is a decisive advantage in sectors where safety and trust are essential.

For all these reasons GO IML is not just another step in machine learning. It is a structural leap. It introduces a new way to design AI systems that are faster to train, easier to deploy, safer to operate, and more aligned with the physical world. It brings a level of clarity and control that classical neural networks cannot offer. And because it is fully integrated into SOTA and ONNX, it transforms a theoretical idea into a practical, portable, operational solution.

Next Steps: From Vision to Reality

GO IML is a proposal, but it is not abstract. Every element described in this whitepaper can be translated into a concrete development path inside SOTA. The strength of the idea comes from its compatibility with what already exists: ONNX as a universal computational graph, the SOTA Deep Learning Toolkit as the orchestration environment, and Graiphic's broader vision for Physical AI. The next steps transform this vision into an operational capability.

The first step is the definition of the IML abstraction layer. This involves creating a library of graph components that represent physical constraints, domain rules, symbolic relationships, and structured priors. These components must be intuitive, lightweight, and compatible with ONNX operators. Some can be built from existing mathematical nodes. Others may require dedicated ONNX custom operators. All must fit naturally into the SOTA workflow.

The second step is the integration into the SOTA editor. Engineers must be able to drag, drop, connect, and parameterize IML blocks just as they do with standard deep learning layers. This visual accessibility is essential. It ensures that informed learning becomes a tool for practitioners rather than a feature reserved for experts. The goal is to make knowledge-driven AI feel as natural as building a neural network.

The third step is the extension of the training orchestration layer. GO IML requires a training loop capable of handling hybrid objectives, mixing data loss with constraint-based loss, residual terms, or logical penalties. SOTA already supports graph-based control flow for training. The next step is to enhance this orchestration so that informed components participate seamlessly in gradient computation, scheduling, and optimization.

The fourth step is validation through demonstrators. To illustrate the tactical value of GO IML, Graiphic can develop proof-of-concept applications. A robotic locomotion model informed by friction dynamics. A drone navigation model informed by aerodynamic stability. An industrial prediction model informed by process rules. Each demonstrator provides measurable gains: fewer training iterations, improved robustness, stronger generalization, and clearer interpretability.

The fifth step is collaboration with scientific partners. The CIAD laboratory and other research groups working on Informed Machine Learning bring valuable expertise. Joint work can help refine the abstraction layer, validate theoretical assumptions, and design IML components with solid mathematical foundations. This strengthens the credibility of GO IML in research and industry.

The sixth step is alignment with Graiphic's broader program. GO HW explores how ONNX graphs can control hardware. GO GenAI explores how graphs can integrate generative intelligence. GO IML plugs directly into this ecosystem. As the three pillars evolve together, they create a unified platform where knowledge, decision-making, hardware acceleration, and generative reasoning coexist inside one orchestrated graph. This integration is the long-term objective.

The seventh step concerns deployment strategy. Once GO IML reaches maturity, it becomes a unique differentiator for SOTA and for any organization relying on intelligent systems. Its ability to produce models that train faster and behave more reliably opens opportunities in robotics, aerospace, industrial automation, defense, autonomous systems, and scientific computing. Deployment packages, templates, and ready-to-use examples will accelerate adoption across these domains.

The final step is communication and strategic positioning. GO IML should be presented not only as a technical feature but as a new paradigm for building intelligent systems. It fits perfectly within the emerging wave of Physical AI. It supports mission-critical applications where data alone is insufficient. It represents a new way to guarantee performance, safety, and efficiency. This message aligns naturally with innovation programs such as DARPA ERIS, European research initiatives, and industrial partnerships.

GO IML is a vision ready to become reality. It has the foundations, the ecosystem, the scientific basis, and the strategic value to transform how AI is deployed in the real world. By following these steps, Graiphic can deliver a unique capability that stands at the intersection of knowledge, data, and operational intelligence.

Graphs made complex systems visible. The next step is to let the same graph do more than predict. From inference to training and orchestration, it is still a graph.

Call for Funding: Why Industry Should Invest in GO IML

Graiphic has built the first end-to-end ecosystem where AI, logic, and hardware orchestration live inside a single ONNX graph.

We are now opening a Call for Funding to accelerate the roadmap of GO HW.

Why Invest?

- Strategic Advantage: GO IML creates AI systems that learn faster, generalize better, and deliver reliable results in real-world conditions.
- Portability & Standards: The entire technology is built on ONNX, ensuring seamless deployment across CPUs, GPUs, FPGAs, NPUs, and future hardware ecosystems.
- Energy & Efficiency: Informed learning cuts training time, reduces data requirements, and accelerates the path from prototype to operational deployment.
- Market Reach: GO IML addresses multiple high-impact sectors including robotics, aerospace, defense, industrial automation, and intelligent infrastructure.

What We Offer

- Co-development opportunities with our engineering team.
- Early integration of your platforms and SDKs into GO HW.
- Joint visibility in international standardization efforts (ONNX, DARPA, Horizon Europe, ADRA).
- Shared benchmarking and open-source dissemination to establish your technology as a leader in AI orchestration.

How to Engage

Graiphic is actively seeking:

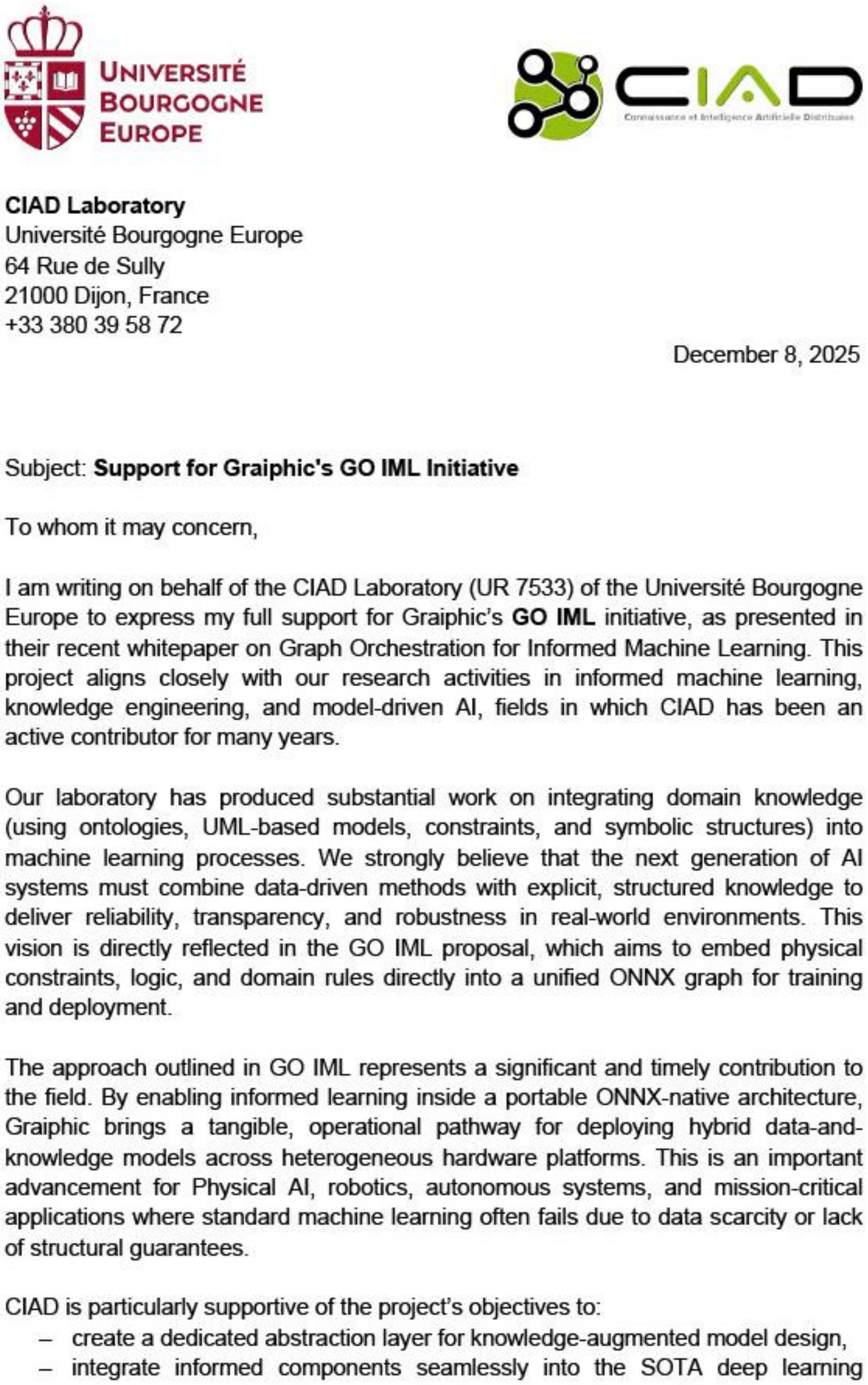
- Equity investors ready to support our growth.
- Industrial sponsors willing to co-fund R&D and test benches.
- Strategic partners (hardware vendors, system integrators, large OEMs) who want their platforms at the heart of the future ONNX Hardware ecosystem.

Join us in shaping the universal cockpit for AI.

Contact: funding@graiphic.io | www.graiphic.io

Annexes

Support Letters



UNIVERSITÉ BOURGOGNE EUROPE

CIAD
Connaissance et Intelligence Artificielle Distribuées

CIAD Laboratory
Université Bourgogne Europe
64 Rue de Sully
21000 Dijon, France
+33 380 39 58 72

December 8, 2025

Subject: **Support for Graiphic's GO IML Initiative**

To whom it may concern,

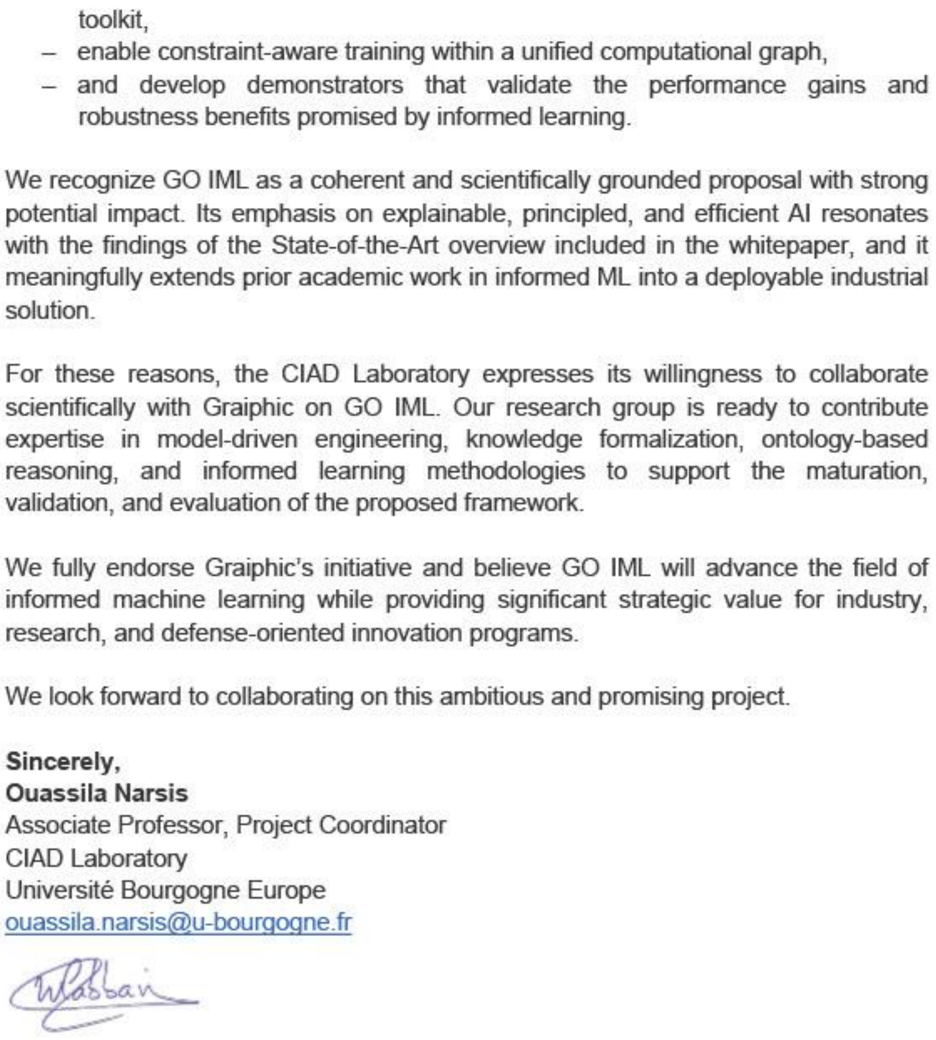
I am writing on behalf of the CIAD Laboratory (UR 7533) of the Université Bourgogne Europe to express my full support for Graiphic's **GO IML** initiative, as presented in their recent whitepaper on Graph Orchestration for Informed Machine Learning. This project aligns closely with our research activities in informed machine learning, knowledge engineering, and model-driven AI, fields in which CIAD has been an active contributor for many years.

Our laboratory has produced substantial work on integrating domain knowledge (using ontologies, UML-based models, constraints, and symbolic structures) into machine learning processes. We strongly believe that the next generation of AI systems must combine data-driven methods with explicit, structured knowledge to deliver reliability, transparency, and robustness in real-world environments. This vision is directly reflected in the GO IML proposal, which aims to embed physical constraints, logic, and domain rules directly into a unified ONNX graph for training and deployment.

The approach outlined in GO IML represents a significant and timely contribution to the field. By enabling informed learning inside a portable ONNX-native architecture, Graiphic brings a tangible, operational pathway for deploying hybrid data-and-knowledge models across heterogeneous hardware platforms. This is an important advancement for Physical AI, robotics, autonomous systems, and mission-critical applications where standard machine learning often fails due to data scarcity or lack of structural guarantees.

CIAD is particularly supportive of the project's objectives to:

- create a dedicated abstraction layer for knowledge-augmented model design,
- integrate informed components seamlessly into the SOTA deep learning



toolkit,
- enable constraint-aware training within a unified computational graph,
- and develop demonstrators that validate the performance gains and robustness benefits promised by informed learning.

We recognize GO IML as a coherent and scientifically grounded proposal with strong potential impact. Its emphasis on explainable, principled, and efficient AI resonates with the findings of the State-of-the-Art overview included in the whitepaper, and it meaningfully extends prior academic work in informed ML into a deployable industrial solution.

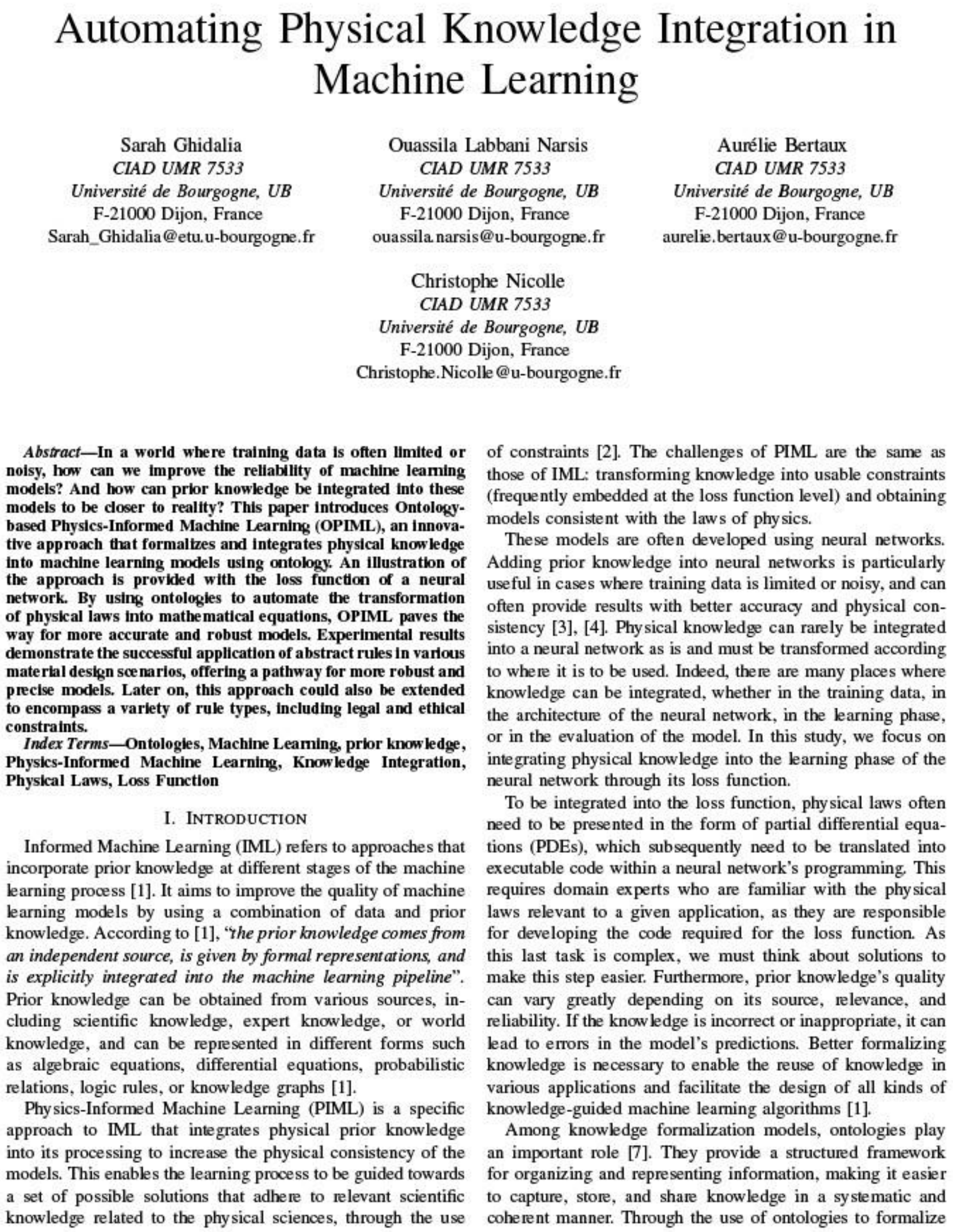
For these reasons, the CIAD Laboratory expresses its willingness to collaborate scientifically with Graiphic on GO IML. Our research group is ready to contribute expertise in model-driven engineering, knowledge formalization, ontology-based reasoning, and informed learning methodologies to support the maturation, validation, and evaluation of the proposed framework.

We fully endorse Graiphic's initiative and believe GO IML will advance the field of informed machine learning while providing significant strategic value for industry, research, and defense-oriented innovation programs.

We look forward to collaborating on this ambitious and promising project.

Sincerely,
Ouassila Narsis
Associate Professor, Project Coordinator
CIAD Laboratory
Université Bourgogne Europe
ouassila.narsis@u-bourgogne.fr

CIAD Reference Publication on Informed ML



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

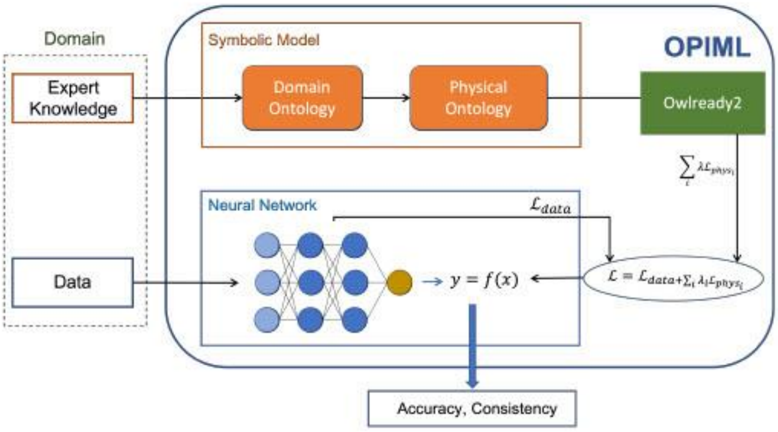


Fig. 1. Ontology-based Physics-Informed Machine Learning (OPIML) architecture

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

$$\sum_i \lambda_i \mathcal{L}phys, i \qquad (2)$$

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:

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

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.

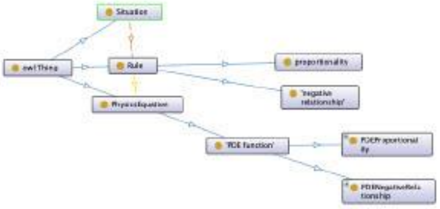
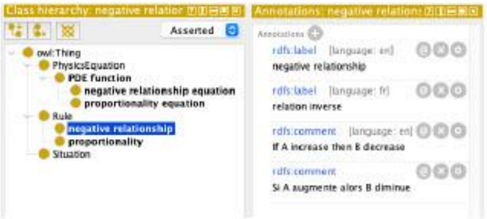


Fig. 2. Physics ontology

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" 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.


Class hierarchy: negative relatior

Asserted

owl:Thing
  PhysicsEquation
    PDE function
      negative relationship equation
      proportionality equation
  Rule
    negative relationship
    proportionality
  Situation

Annotations: negative relation:

Annotations

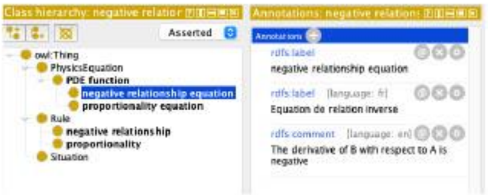
rdfs:label [language: en]
negative relationship

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


Class hierarchy: negative relatior

Asserted

owl:Thing
  PhysicsEquation
    PDE function
      negative relationship equation
      proportionality equation
  Rule
    negative relationship
    proportionality
  Situation

Annotations: negative relation:

Annotations

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.

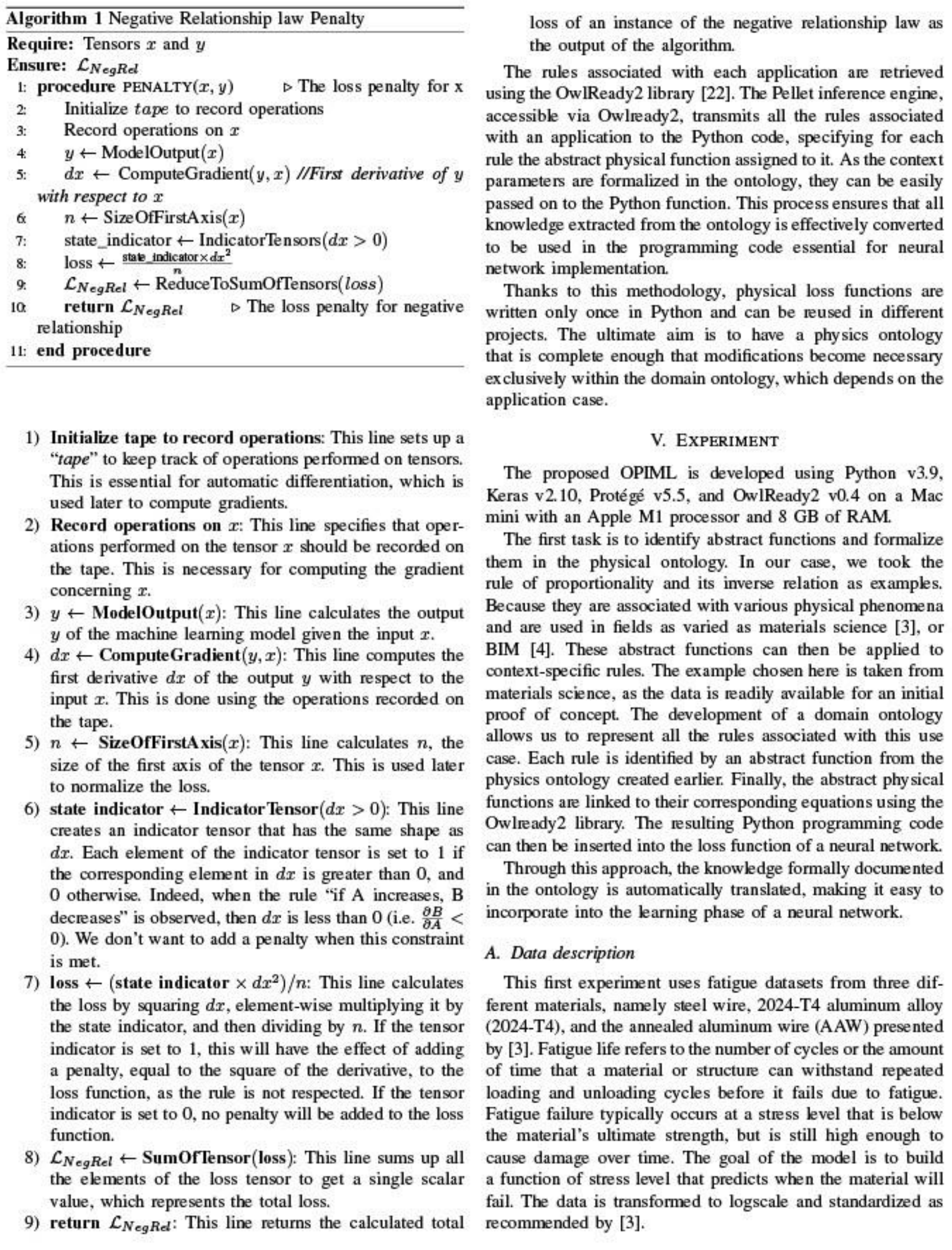
C. Formalization of physical laws into mathematical equations

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: $\mathcal{L}_{NegRel}$
procedure PENALTY($x, y$)                    ▷ The loss penalty for x
    Initialize $tape$ to record operations
    Record operations on $x$
    $y \leftarrow$ ModelOutput($x$)
    $dx \leftarrow$ ComputeGradient($y, x$) //First derivative of $y$ with respect to $x$
    $n \leftarrow$ SizeOfFirstAxis($x$)
    state_indicator $\leftarrow$ IndicatorTensors($dx > 0$)
    loss $\leftarrow \frac{\text{state\_indicator} \times dx^2}{n}$
    $\mathcal{L}_{NegRel} \leftarrow$ ReduceToSumOfTensors($loss$)
    return $\mathcal{L}_{NegRel}$                ▷ The loss penalty for negative relationship
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.

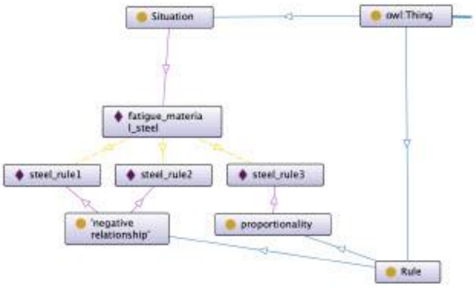
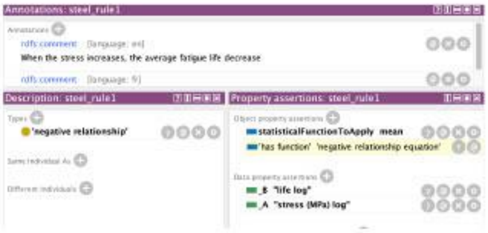


Fig. 5. Negative relationship's equation

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.



rdfs:comment [language: en]
When the stress increases, the average fatigue life decrease
rdfs:comment [language: fr]

'negative relationship'

statisticalFunctionToApply mean
'has function' 'negative relationship equation'

_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**

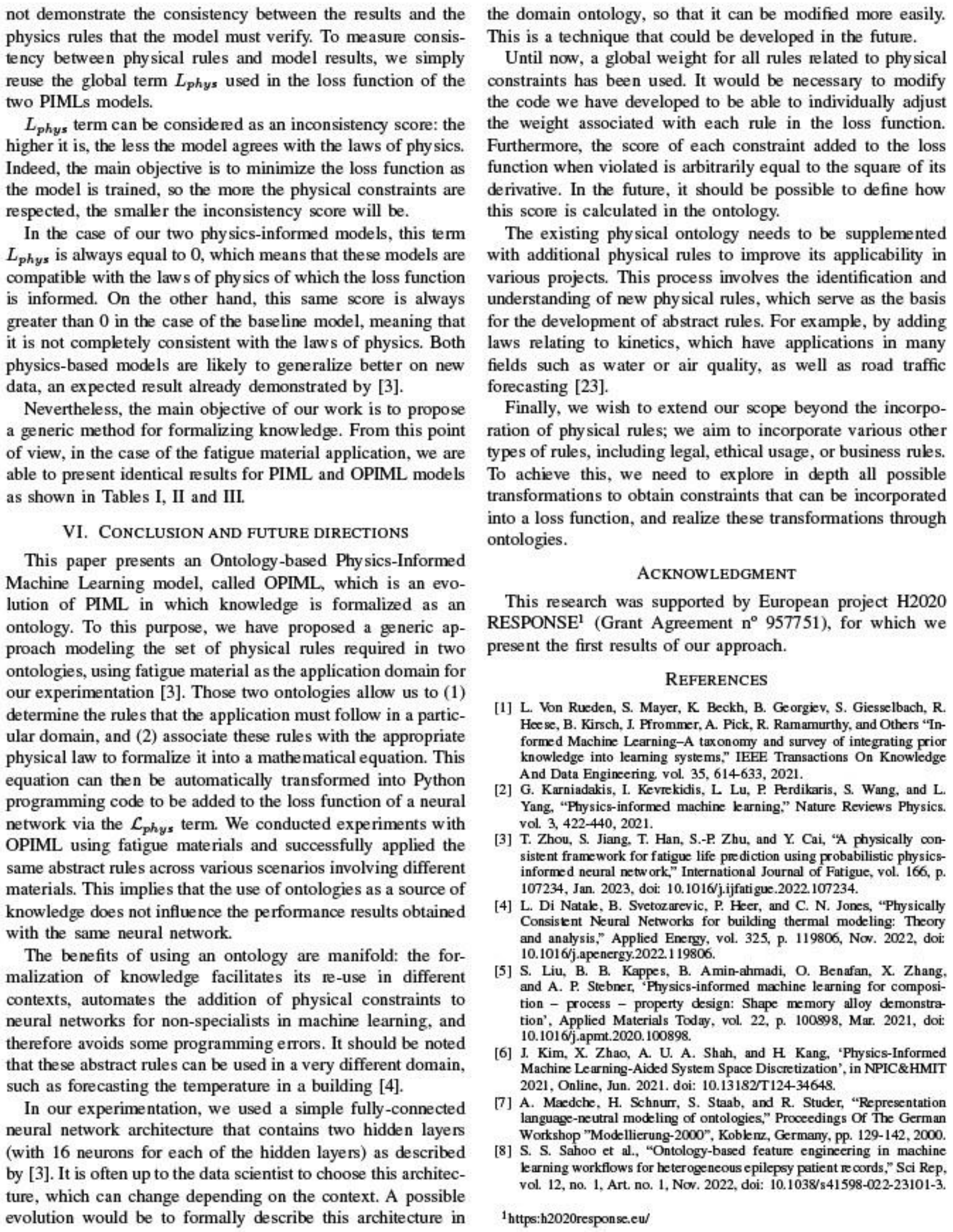
TABLE II
PERFORMANCE AND CONSISTENCY METRICS FOR 2024-T4 DATASET

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 n° 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.

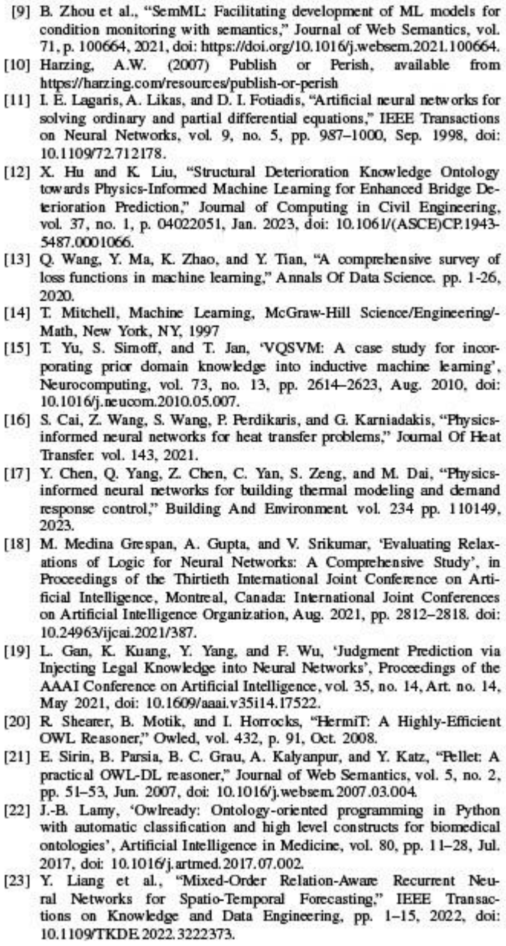
[5] S. Liu, B. B. Kappes, B. Amin-ahmadi, O. Benafan, X. Zhang, and A. P. Stebner, 'Physics-informed machine learning for composition – process – property design: Shape memory alloy demonstration', Applied Materials Today, vol. 22, p. 100898, Mar. 2021, doi: 10.1016/j.apmt.2020.100898.

[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.

Informed Machine Learning: State-of-the-Art

Introduction

Modern machine learning (ML) has achieved remarkable success by learning patterns from large datasets. However, purely data-driven approaches face critical limitations when *data is scarce or constraints must be respected*[1]. In many real-world scenarios, collecting massive labeled datasets is impractical, and black-box models trained only on data might violate known **domain constraints** (e.g. physical laws or safety regulations) and lack interpretability[1]. These challenges have spurred interest in **Informed Machine Learning (IML)** – a paradigm where we *integrate prior knowledge* (from domain experts, scientific theories, logical rules, etc.) *into the ML pipeline alongside data*[2][3]. In contrast to conventional ML (which may only use knowledge implicitly via feature engineering), IML explicitly leverages an **independent knowledge source** – given in a formal representation – *together with data* during training[2]. The core idea of IML is that by informing learning algorithms with established knowledge, we can build models that learn **faster, generalize better, and inherently respect known truths**.

IML has rapidly evolved into a broad research area over the past few years. A foundational survey by von Rueden *et al.* (2019/2021) formalized the concept and proposed a taxonomy for integrating knowledge into learning systems[4][5]. Informed ML is sometimes referred to by other names in specific contexts – for example, *physics-informed deep learning*, *theory-guided data science*, *physics-guided neural networks*, or *semantic-based regularization*[6] – but all share the common theme of combining data-driven methods with domain insights. This approach marks a shift towards the **“third wave” of AI**, where instead of solely relying on big data, we build systems that *reason* with knowledge and data, yielding more **trustworthy and efficient** solutions[7].

Key Advantages of Informed ML

By design, informed ML addresses the weaknesses of purely data-driven training and offers several compelling advantages:

- **Reduced Data Requirements:** Incorporating prior knowledge can *compensate for limited data*, allowing models to achieve good performance even in “small data” regimes[3]. In practice, IML *reduces the need for large training datasets* because the model can lean on trusted knowledge to fill gaps[3]. For example, an ML model constrained by known physics may require far fewer experimental samples to learn a phenomenon than a black-box model would.

- **Improved Generalization & Extrapolation:** IML tends to produce models that generalize better beyond the training distribution. By learning underlying principles, informed models can *extrapolate to new situations* more reliably[8][9]. A recent study comparing standard ML vs. informed ML found that *IML can outperform traditional ML in terms of generalization (lower excess risk), especially when domain-specific insights are crucial*[8][10]. In short, an informed model is less likely to “overfit” spurious patterns – it respects the true structure of the problem.

- **Robustness and Trustworthiness:** Because prior knowledge (e.g. physical laws or safety rules) is built-in, IML models inherently obey those constraints, leading to more **robust** behavior in edge cases. They are less likely to produce physically impossible or nonsensical outputs, a critical property for **trustworthy AI**[7]. Enforcing knowledge (such as conservation laws in physics or semantic consistency in ontology-based systems) acts as a regularizer, often yielding *smaller and less complex models* that are more stable[3].

- **Explainability and Interpretability:** Informed models can be more **interpretable**. Since part of the model's logic comes from human-understandable knowledge (equations, logical rules, etc.), the resulting system's decisions can be easier to explain in those terms[11]. For instance, an informed model might explain a prediction by referencing a known rule or pattern it was endowed with, rather than just relying on opaque learned features. The integration of structured knowledge guides the learning towards **explainable** structures without requiring big data[11].

- **Viability in New Domains:** IML *enables ML in domains where it was previously infeasible*. Many industrial and scientific applications have only small datasets (e.g. expensive lab experiments or rare events) but rich domain expertise. In such cases, informed learning can leverage domain theories or simulations to make ML viable[3]. This opens up **strategic advantages** – organizations can deploy AI in niche or high-stakes areas (medicine, engineering, defense) by injecting expert knowledge, rather than being limited by data scarcity. In settings like defense (e.g. DARPA programs), this means AI systems can be rapidly developed for new scenarios by exploiting prior operational knowledge, yielding a surprise advantage over purely data-trained models.

In summary, informed ML marries the strengths of data-driven AI with the wisdom of domain knowledge. It *shrinks data needs, boosts generalization, and yields models that are often more robust, interpretable, and principled*[3][9]. These benefits have been demonstrated across a variety of tasks, giving informed ML a reputation for *"doing more with less"* and delivering **unique capabilities** unattainable by standard supervised learning alone[10].

Approaches and Taxonomy of Knowledge Integration

IML is a broad umbrella, encompassing many techniques to inject knowledge into the learning process. A useful way to break down the landscape is given by the Fraunhofer IML taxonomy[5][12], which considers three aspects: **knowledge source**, **knowledge representation**, and **integration method**. In practice, informed learning techniques can be categorized by *what kind of knowledge is used*, *how that knowledge is formalized*, and *where in the ML pipeline it is introduced*. Below, we outline the major forms of knowledge and integration strategies in state-of-the-art IML:

- **Knowledge Sources:** The origins of prior knowledge in IML vary. Often, we use **scientific knowledge** – e.g. well-established physics, chemistry, or biological principles. In other cases, the source might be **expert knowledge** from humans (engineering heuristics, clinical rules), or **common-sense/world knowledge** captured in databases. A recent survey groups sources into formal scientific knowledge, everyday commonsense, and human expert insights[13]. Regardless of source, the key is that this knowledge exists *independently of the training data* (e.g. centuries of physics research or domain expertise) and can be brought into the learning process.

- **Knowledge Representations:** Before integration, knowledge must be encoded in a formalism that interfaces with ML algorithms[2]. Many representation types are used in advanced IML:

- **Differential equations:** Many physical laws (e.g. fluid dynamics, Newton's laws) are expressed as PDEs/ODEs. These equations can be directly embedded into model architectures or loss functions.
- **Analytical or domain-specific models:** Simplified mathematical models or empirical formulas (for example, a known equation for a system's behavior).
- **Simulation data or "Digital Twins":** Complex process simulators (e.g. finite element simulations, synthetic sensor data generators) produce data that encapsulates knowledge. Using simulation results as training data or in training loops is a way to inject knowledge[14].
- **Logical rules and constraints:** Boolean or fuzzy logic rules, ontologies, and relational constraints fall here. For example, rules like "if condition A, then outcome B cannot happen" can be enforced in a model.
- **Knowledge graphs and ontologies:** Graph-structured knowledge bases (nodes representing entities/concepts and edges representing relationships) provide rich relational information. These can enhance neural networks by informing them of known relationships and hierarchies[14]. (E.g., an image classifier can use a knowledge graph of species to avoid impossible label co-occurrences.)
- **Human feedback signals:** Less formal, but expert annotations, preferences or corrections can be treated as knowledge input to guide learning (common in reinforcement learning with human feedback, or interactive machine teaching).

Sources: Knowledge in IML can thus be encoded in **differential equations, analytical models, simulation outputs, logical rules, knowledge graphs, or even human-provided feedback**[15]. The degree of formalization can range from fully mathematical to heuristic; generally, the more structured the knowledge, the easier to integrate into algorithms[16].

- **Integration Methods:** Prior knowledge can enter the ML pipeline at various stages[17]. The main integration techniques include:
- **Data-level integration:** Here, knowledge is used to generate or augment training data. For example, using a physics simulator to create additional synthetic training samples consistent with known laws[14]. Another example is *data labeling with expert rules* (a form of weak supervision), where heuristics label unlabeled data. This approach effectively informs the model via an expanded or enriched dataset.
- **Model architecture integration:** The model itself is structured to encode knowledge. A classic approach is building **hybrid models** or *gray-box models*

that combine known analytical components with learned components. For instance, one might hard-code part of a network to implement a known equation or use a network architecture that inherently obeys certain invariants (like an equivariant neural network that respects symmetry groups). In deep learning, a notable trend is **physics-informed neural networks (PINNs)**, where neural nets are trained to satisfy physical equations (e.g. by including differential equation terms in the network's computation) rather than learning purely free-form mappings.

- **Loss function or training algorithm integration:** A very common strategy is to add knowledge as a **constraint or regularization term** in the loss function. In this scheme, the model is trained not only to fit the data but also to minimize violations of prior knowledge. For example, one can add a penalty if the model's predictions break a known law of nature or a logical rule[14]. This is sometimes called **knowledge-based regularization** or **semantic loss**. By shaping the loss landscape with knowledge, the learning algorithm is "pulled" towards solutions that agree with what we already know. Another approach at this level is initializing models or priors based on known solutions (e.g., starting a model near an analytical solution before fine-tuning on data).
- **Post-processing and inference integration:** In some cases, after a model is trained, one can enforce knowledge during inference. This could mean adjusting outputs to satisfy constraints (projecting onto a feasible set) or using a knowledge-based module (like a logic reasoner or optimization routine) to refine raw model outputs. This is less common in research literature, as the trend is to bake knowledge *inside* the training, but it's an option for certain applications (e.g. adjusting a neural network's output to exactly conserve mass or charge by a post-hoc correction).

These categories are not mutually exclusive – often **IML solutions combine multiple integration points**. For example, a state-of-the-art pipeline might use simulation to generate training data *and* impose a physical equation in the loss function. The informed ML taxonomy reveals several frequent "paths" where knowledge source, representation, and integration method align as common patterns[18]. One prevalent approach is scientific knowledge represented as algebraic/differential equations and integrated via the learning algorithm (loss constraints)[19]. Another is expert knowledge represented as a knowledge graph and integrated at the model architecture level (e.g. a graph neural network that ingests the knowledge graph)[14]. The unifying theme is that **IML explicitly injects knowledge at a designated point in the ML pipeline, treating it as a *first-class input* rather than just implicit intuition**.

Notable Subfields and Applications

Research in informed ML has exploded into several subfields, each focusing on particular types of knowledge or application domains. Here we highlight some prominent areas and state-of-the-art results demonstrating the power of IML:

Physics-Informed Learning

One of the most active areas is **physics-informed machine learning**, which integrates fundamental scientific laws (physics, chemistry, engineering principles) into AI models. These techniques are motivated by the fact that physical laws are **universal and exact (or nearly so)**, providing a powerful scaffold for learning. In physics-informed neural networks (PINNs), for instance, one incorporates the governing differential equations of a system directly into the neural network's training. *The neural network is trained not just to fit data points, but also to satisfy the known physical equations* (by minimizing equation residuals)[14]. This approach has yielded impressive results in solving forward and inverse problems for partial differential equations with far fewer data than traditional methods require. As an example, researchers have demonstrated **"one-shot learning" for solving PDEs** – learning the solution operator from *only a single observed solution* and then generalizing to diverse new inputs[20]. This is possible only because the network is constrained by physics; a purely data-driven model would normally need many solution examples to interpolate. Physics-informed models are being applied to fluid dynamics, climate modeling, material science, and more[21][22].

The benefits are substantial: by using prior physical knowledge, training becomes more efficient and the resulting predictions are **robust and consistent with physical reality**[7]. For instance, one study on PINN-based fluid flow modeling achieved a *59-fold reduction in prediction error* over the best existing deep learning models, by automatically designing a network that respects the underlying physics[23]. Physics-informed ML is not limited to differential equations; it also includes *equivariant neural networks* that encode symmetry (group invariants), *neural operators* that learn mappings between function spaces under physics constraints, and hybrid analytical-ML models. Given the importance of trustworthiness in scientific AI, this subfield is seen as a path to **scientifically-grounded AI**. It ensures models behave plausibly under conditions where data might be sparse or extrapolation is needed – a critical strategic advantage in engineering and defense applications where one cannot simply "collect more data" for extreme scenarios.

Knowledge Graphs and Relational Knowledge Integration

Another frontier is integrating **semantic and relational knowledge** into ML models. Domain knowledge is often available in structured forms like **knowledge graphs**, ontologies, or databases of facts. These can significantly enhance learning, especially for tasks requiring reasoning or world knowledge beyond what the training data directly provides. *Knowledge graphs (KGs)* consist of entities and relationships (often as triples like *("Paris","capital_of","France")*), and they encode a wealth of interconnected information. State-of-the-art approaches inject KGs into neural models in various ways. One approach is to use graph neural networks (GNNs) or relational models that take the KG as an additional input – effectively, the model "knows" relationships between concepts and can use them to make more informed predictions[14]. For example, in image classification, if a network knows via a KG that "zebras are a type of animal" and "animals do not have wheels," it can avoid a confusion that a purely data-driven vision model might make between a zebra and a striped vehicle in some bizarre scenario. Studies have shown that *knowledge graphs can enhance neural networks by providing relational context between instances*, improving accuracy and consistency in tasks like image recognition and question answering[14]. This line of work overlaps with **neuro-symbolic AI**, where symbolic knowledge (graphs, logic) is combined with neural perception.

A related concept is leveraging **relational inductive biases** – architectural biases that reflect relational structure. A notable example is the use of graph networks (deep networks that operate on graph-structured data) to enforce that the model's reasoning follows graph connectivity. Battaglia *et al.* (2018) famously argued for graph-based inductive biases as a way to inject prior knowledge about relationships into deep learning[5]. Modern GNN models, now a standard toolkit, inherently encode the assumption "entities and their relations matter," which is a form of informed learning (the model architecture itself is informed by our knowledge that many problems have underlying graph structures).

In practice, combining KGs with ML has led to **state-of-the-art (SOTA) results in recommender systems**, natural language understanding (where common sense knowledge helps disambiguate), and biomedical informatics (integrating gene/drug networks with data to predict new interactions). For the **Graph Orchestration** projects at Graiphic, this is highly relevant: one can imagine a *graph-based orchestration framework* seamlessly incorporating knowledge graph nodes or constraints into the computational graph of an ML pipeline. By doing so, such a framework would allow SOTA deep learning models to be *augmented with world knowledge on the fly*, giving them a richer understanding without solely relying on learned parameters.

Logical Rules and Constraint-Based Learning

IML also encompasses methods that integrate **logical rules, constraints, and symbolic reasoning** into machine learning. In many domains, experts can articulate rules (e.g., diagnostic criteria in healthcare, business rules in finance, or game strategies) that a model should follow. Rather than expecting a neural network to rediscover these from data, informed learning can bake them in.

One popular technique is to impose **logic-based constraints during training**. As mentioned, this can be done by adding a penalty to the loss function whenever the model's output violates a rule[14]. For example, if we know that in a scheduling problem "Task A must occur before Task B," any predicted schedule that has B before A is penalized. This approach, sometimes called **constraint-based learning** or **posterior regularization**, guides the model to learn hypotheses that satisfy the rules as much as possible. Another approach is *to modify the model outputs directly* using differentiable logic layers – for instance, using soft logic (fuzzy logic) to allow the model to learn truth values of rules and incorporate them in inference. Projects like **DeepProbLog**, **Logic Tensor Networks**, and others combine neural networks with logic programming, ensuring that certain logical relationships hold in the predictions. These neuro-symbolic hybrids have shown success in tasks like visual question answering (ensuring that answers follow logical consistency) and knowledge base completion.

The strategic advantage here is **guaranteed consistency and compliance**. In safety-critical or mission-critical systems (e.g., military decision support, air traffic control), we often have hard constraints that must never be violated. Embedding these into the learning process yields solutions that are *by construction valid*, avoiding the need for extensive rule-enforcing post-processing. This is a clear differentiator from unconstrained ML, which might achieve high accuracy on average but occasionally produce an invalid or unsafe result. By making domain constraints part of the learning, we get the **best of both worlds** – data-driven adaptability and rule-based reliability.

Simulation-Aided Learning and Digital Twins

In scenarios where a high-fidelity simulator or a **digital twin** of a system exists, informed ML can leverage it to improve learning. A *digital twin* is a virtual replica of a physical system (like an engine, manufacturing line, or even a city) that can simulate its behavior. Informed ML techniques use simulations in several ways. One straightforward way is to generate synthetic data from the simulator to supplement real training data[24]. This is extremely useful when real data is scarce or expensive – simulations can create essentially unlimited labeled examples (albeit imperfect) grounded in prior knowledge of the system. For example, if building a defect detection model for a new aerospace component, one might simulate various defect scenarios to train the model before any real data is available.

Another sophisticated approach is **simulation-in-the-loop training**: the ML model is trained with a physics simulator embedded such that for each prediction the simulator provides some feedback. An example is using a simulator to validate a model's output (say, checking if a proposed control action keeps a robot stable) and then giving a reward or penalty accordingly. This shades into reinforcement learning, but with the twist that the simulator encodes known physics. Informed ML research has also explored **co-training ML models with simulator calibration**, where the ML model and simulator parameters are learned together to best explain observed data – effectively blending data-driven learning with prior physics.

In industrial use cases documented in the literature (including in the *Informed Machine Learning* 2025 book), **IML with digital twins** has shown huge benefits[25]. For instance, Wallner *et al.* (2025) describe optimizing cooling systems by training ML models alongside thermodynamic simulations[26]. The result is an AI system that respects engineering constraints and can even suggest adjustments to the simulator's parameters (improving the digital twin) as more data comes in – a virtuous cycle of knowledge and learning.

For Graiphic's **Graph Orchestration (GO)** framework, simulation components can be treated as nodes in a computational graph, interwoven with neural network nodes. This means a graph orchestrator can route data through both ML models and physics-based models in one pipeline. Integrating IML in such a framework enables real-time synergy: as new data flows in, the AI can use the simulator to validate or generate hypotheses, and the entire graph can be optimized end-to-end. This is a cutting-edge capability, effectively bringing informed ML into deployment pipelines with minimal friction.

Human-in-the-Loop and Expert-Guided Learning

While much of IML focuses on codified knowledge (equations, graphs, rules), *human experts* themselves are invaluable sources of knowledge. Human-in-the-loop learning is a paradigm where models and human domain experts interact during training. For example, an expert might iteratively correct a model's predictions or provide hints on difficult cases. Techniques like **active learning** ask an oracle (human) for labels on uncertain samples – in a sense, the model is "asking for knowledge" on-demand. Another approach is **reward shaping in reinforcement learning**: human feedback is used to craft a reward function that encodes expert preferences (famously used in training large language models to follow instructions). Although these methods are often discussed separately from IML, they fit the broader definition of *informed learning*, since prior knowledge (in the expert's head) guides model updates.

Research on **explanatory interactive learning** has even looked at having models explain their reasoning to a human and get corrections, which updates the model's knowledge base. In mission-critical settings, having a human expert impart key rules or sanity checks to a model can dramatically reduce training time and prevent costly mistakes. The unique advantage here is **flexibility and rapid adaptation**: when facing a novel situation, an expert can rapidly instruct the model (e.g., "in this new environment, condition X holds true, adjust accordingly") without needing a large new dataset. This can be vital for defense or emergency response applications where an AI must be quickly adapted by human operators using their expertise.

Evidence of Effectiveness

A growing body of work empirically demonstrates that informed ML often **outperforms standard supervised ML**, especially under conditions that mirror real-world constraints. We summarize some key findings and comparisons:

- **Theoretical Performance:** Oneto *et al.* (2025) provide a theoretical comparison of traditional ML vs. IML in terms of statistical learning metrics. They show that by incorporating correct prior knowledge, *IML can achieve a lower excess risk (better generalization) than a purely data-driven model*[8]. In essence, the capacity added by knowledge helps restrict the hypothesis space to more plausible solutions, which improves learning bounds. Their analysis also highlights that *IML tends to shine when the prior knowledge is accurate and the training data is limited or noisy* – under those conditions, informed models learn the true pattern where uninformed models might underfit or misgeneralize[8][10]. These results back up the intuitive claim that IML "bakes in" part of the solution, so the model doesn't need to learn everything from scratch.

- **Data Efficiency and Accuracy:** Numerous case studies report that informed approaches reach equal or better accuracy with far fewer training examples. In physics-informed deep learning, it is common to see orders-of-magnitude improvements. For example, one physics-guided neural network for a fluid dynamics problem achieved the same accuracy with *50x less data* than a black-box neural network, by leveraging conservation laws (this type of result is echoed across many PINN papers). The Nature Communications example of one-shot PDE learning is extreme: **learning from one sample** versus the many samples a black-box would require[20]. Another study using an automated physics-informed CNN reported a *59.8-fold error reduction* over state-of-the-art purely data-driven models on several benchmark systems[23] – a dramatic performance gain attributed to better architecture and loss design informed by physics. In constraint-based learning, researchers have found that injecting even a handful of logic rules can significantly boost accuracy on structured prediction tasks (like semantic parsing or scene understanding), essentially by ruling out wrong answers that a neural network might otherwise consider likely.

- **Robust Generalization (Extrapolation):** Because informed models encode actual mechanisms of a system, they handle conditions *outside the training range* far better. A striking example is in climate/weather modeling: a theory-guided model was able to predict extreme weather statistics that were not present in training data, by respecting physical invariants[21]. Pure ML models often fail spectacularly when asked to extrapolate (e.g., forecasting beyond the range of observed years), whereas an informed model can use the underlying science to venture reasonable predictions. This robustness extends to adversarial settings too – models that know the "rules of the game" are less easily fooled by adversarial inputs that violate those rules. In safety-critical applications, this reliability is crucial. For DARPA ERIΣ (hypothetical submission context), such strategic robustness means AI systems that maintain performance and safety in novel, surprising situations – a clear differentiator from conventional AI.

- **Trust and Compliance:** Empirical studies on fairness and safety have begun using informed ML to ensure compliance with ethical or legal standards. For instance, an informed approach might integrate fairness constraints into a classifier to prevent biased decisions. The result is a model that meets regulatory requirements by design, avoiding the need for after-the-fact fixes. While performance in terms of accuracy is one metric, informed ML often excels in *other metrics* like consistency, reliability, and user trust. Users tend to trust a model more if they know it cannot violate fundamental constraints – e.g., an AI medic diagnostic that respects known medical protocols. Although harder to quantify, this trust factor is increasingly recognized as part of "performance" in real deployments.

In summary, both formal analyses and practical experiments confirm that **IML can surpass traditional ML in effectiveness** when the scenario leverages domain knowledge[10]. By **leveraging prior knowledge to reduce data demands and boost generalization**, informed ML provides a path to models that are *not only more accurate with less training data, but also more reliable when deployed*[9]. The strategic implication is significant: teams that can exploit domain knowledge in AI gain a **competitive edge**, achieving higher performance models with fewer resources and ensuring those models behave in expected, controlled ways. This is a powerful combination of efficiencies and capabilities that defines the state-of-the-art in informed learning.

Integration with Modern ML Frameworks

One reason informed ML is gaining traction now (beyond the theoretical appeal) is the improved ease of integrating knowledge into practical AI pipelines. Modern deep learning frameworks and tooling have evolved to support the flexibility required for IML techniques. Below we discuss how state-of-the-art frameworks enable IML and how Graiphic's Graph Orchestration approach can leverage these advances:

- **Custom Losses and Constraints:** Popular deep learning libraries (PyTorch, TensorFlow, JAX) allow users to define custom loss functions and training loops. This means adding a physics equation residual or a logical rule penalty to the loss is straightforward. For example, one can implement a PINN in PyTorch by computing the PDE residual inside the training step and backpropagating through it. The ability to differentiate through essentially any computable operation (automatic differentiation) is a key enabler – it allows integration of knowledge (even a complex simulator) as part of the training graph. This has democratized physics-informed learning: researchers have released open-source tools (e.g. **NVIDIA Modulus** for physics-informed DL) to simplify defining PDE losses and constraints on standard NN models. Thus, integrating domain knowledge no longer requires custom solvers from scratch; it can be done on top of existing ML platforms, accelerating adoption.

- **Hybrid Modeling Frameworks:** There's an emerging class of frameworks aimed at *hybrid modeling*, where parts of the system are neural nets and parts are predefined functions or simulators. The Open Neural Network Exchange (**ONNX**) format, for instance, can represent entire computational graphs including non-ML operations. Graiphic's **ONNX GO** is an example of leveraging this: it provides a pipeline to orchestrate dynamic *graph computations* where some nodes could be neural network layers and others could be, say, a formula or control law coded in a module[27][28]. By using a graph representation, one can visually design how data flows through both learned and knowledge-based components. At runtime, the orchestrator (like ONNX GO with ONNX Runtime) can efficiently execute this hybrid graph, even optimizing it across hardware[29][30]. This essentially makes *integrating an informed component as easy as "dragging and dropping" it into your model graph*. Such technology is state-of-the-art in enabling **dynamic reconfiguration** – for example, switching in a new knowledge module on the fly if conditions change[31] – which is important for real-time systems that must adapt (a scenario where you might load a new set of rules or physical parameters into the pipeline without redeploying the entire model).

- **Graph-Oriented Orchestration:** Graiphic's Graph Orchestration (GO) approach is at the forefront of simplifying IML integration. By representing AI workflows as graphs, it becomes natural to incorporate diverse components. A knowledge source can be encapsulated as a node (or subgraph) that the orchestrator knows how to execute. For example, a "PhysicsLaw" node could take a predicted solution and output a residual error by applying a known equation, which then feeds into subsequent computation (like a loss aggregator). The SOTA idea here is that *the pipeline itself is aware of knowledge components*, not just the model. This aligns with trends in **multi-agent and tool-augmented AI**, where graphs coordinate different reasoning modules. Atlassian recently noted that a graph orchestration model suits complex LLM workflows with formalized steps[32] – similarly, for informed ML, a graph can coordinate steps of simulation, neural inference, logical checking, etc., in a sequence or parallel as needed. The **state-**

of-the-art result is a highly modular, transparent AI system: each part (neural or knowledge-based) can be developed and optimized independently, and the orchestration brings them together into a coherent whole. This modularity makes it much easier to maintain and upgrade AI systems (e.g., update the knowledge base without retraining the neural parts, or vice versa) – a big advantage for deploying informed ML at scale.

- **Open Access Libraries and Standards:** The IML community has been actively developing libraries that plug into existing ecosystems. For instance, there are packages for adding logical constraints to Keras models, libraries for neuro-symbolic programming that integrate with PyTorch, and so on. Standards like ONNX facilitate sharing models that include custom ops (one can define an op for a domain-specific function so that any ONNX-compatible runtime can execute it). The Fraunhofer open-access book (2025) contains numerous case studies with code, showing how to incorporate knowledge for tasks ranging from healthcare to manufacturing[33][3]. This dissemination of *practical examples* lowers the barrier for practitioners to try IML. As a result, we are seeing **informed ML move from research to practice**, with industrial proof-of-concepts turning into deployed solutions (especially in fields like engineering, where the demand for small-data solutions is high).

In essence, the current state-of-the-art frameworks are aligning well with the needs of informed ML. They provide the hooks and flexibility to integrate custom knowledge without reinventing the wheel. Graiphic's approach of making IML integration *simple and accessible* within a SOTA deep learning framework is timely and strategic. By building on these modern capabilities, one can create a **unified platform where domain experts can easily inject knowledge into AI workflows** – for example, via a visual interface as ONNX GO does with LabVIEW for graph design[34][35]. This means the exclusive benefits of informed ML (data efficiency, robustness, etc.) can be **delivered to end-users without requiring them to be AI PhDs**. Such democratization and ease-of-use will be a key selling point in a whitepaper or DARPA proposal: it promises to take IML from an advanced concept to an operational tool accessible across industries and government applications.

Conclusion

Informed Machine Learning (IML) represents a cutting-edge convergence of data-driven AI with human and scientific knowledge, marking a significant evolution in the field of machine learning. Our deep dive into the state-of-the-art reveals that IML is not only a theoretical ideal but a practical, evidence-backed approach that offers *tangible gains* over conventional ML. By integrating prior knowledge into models, we achieve breakthroughs like: learning effective models from minimal data, maintaining performance under distribution shifts, and guaranteeing adherence to critical constraints – outcomes that directly translate to **strategic advantages** in real-world deployments[9][10]. These advantages come with the added benefits of model transparency and trustworthiness, addressing key concerns in high-stakes AI applications.

The current landscape of IML research is rich and rapidly advancing, with contributions spanning physics-informed neural networks, knowledge-infused vision and NLP systems, neuro-symbolic reasoning, and beyond. The element of **surprise and innovation** is evident – solutions that seemed "magic" to traditional practitioners (like solving complex scientific problems with almost no data, or automatically enforcing logical consistency in AI reasoning) are now being demonstrated in academic papers and pilot projects[20][14]. This wave of innovation is fueled by an appreciation that *big data alone is not a panacea*; rather, the incorporation of *structured knowledge* can dramatically elevate an AI system's capabilities in ways that raw data cannot.

From a strategic perspective, IML offers an **exclusive edge**: it leverages proprietary or hard-won domain knowledge – something competitors cannot easily replicate just by downloading a public dataset. For organizations like Graiphic and initiatives like DARPA's, this means that investing in IML can create *defensible, unique solutions*. An informed model trained with Graiphic's domain-specific graph orchestration will inherently carry Graiphic's expertise within it, making it a one-of-a-kind asset. In military or security contexts, IML-based systems could embody the cumulative experience of experts and the laws of physics, giving U.S. defense AI systems a leap in reliability and adaptability that adversaries lacking such integrated knowledge would struggle to match.

Finally, the path to **practical adoption** of IML is clearer than ever. The synergy of advanced ML frameworks with knowledge integration, as we discussed, means that the once-formidable challenge of implementing informed learning is being dissolved. This is precisely where the *Graiphic GO IML* initiative positions itself: at the intersection of SOTA deep learning and seamless knowledge integration. By providing a robust yet user-friendly platform for IML, Graiphic's solution will make it simple to plug domain knowledge into AI models and pipelines – turning the lofty promises of informed ML into operational reality.

In conclusion, the state-of-the-art in informed machine learning demonstrates a transformative approach to AI development – one that delivers **innovation through integration**. It combines the best of human insight and machine computation to produce AI systems that are *more efficient, more accurate, more interpretable, and more trustworthy* than ever before. Embracing this paradigm will be key to tackling the next generation of challenges where data alone is not enough. As this overview has shown, informed ML is not just an academic concept but an emerging **standard of excellence** in AI, poised to unlock new frontiers in technology and secure a strategic advantage in any field where it is applied[3][9].

---

[1] [2] [4] [5] [6] [12] [13] [14] [16] [17] [18] [19] [24] [1903.12394] Informed Machine Learning – A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems

https://ar5iv.labs.arxiv.org/html/1903.12394

[3] [11] [15] [25] [26] [33] Informed Machine Learning | SpringerLink

https://link.springer.com/book/10.1007/978-3-031-83097-6

[7] [20] [21] [22] [23] Physics-Informed Machine Learning

https://www.nature.com/collections/jaihfcabgi?error=cookies_not_supported&code=e7c0d966-3a64-4b3f-b857-e20d6caae924

[8] [9] [10] Informed Machine Learning: Excess risk and generalization - ScienceDirect

https://www.sciencedirect.com/science/article/pii/S0925231225011932

[27] [28] [29] [30] [31] [34] [35] ONNX GO: Revolutionizing Dynamic Graph Orchestration and Execution - Graiphic

https://graiphic.io/onnx-go-dynamic-graph-orchestration/

[32] How Rovo Chat embraces multi-agent orchestration - Atlassian

https://www.atlassian.com/blog/atlassian-engineering/how-rovo-embraces-multi-agent-orchestration