Physics-Informed Neural Networks: A Paradigm for Knowledge-Enhanced Deep Learning in Scientific Computing

A FORGE OS Deployment Case Study

577 Industries R&D Lab
577 Industries Incorporated
Columbus, OH, USA
research@577industries.com

***Abstract*—This paper presents a comprehensive review of Physics-Informed Neural Networks (PINNs), a hybrid framework that integrates scientific domain knowledge with deep learning architectures. PINNs incorporate physical laws, typically expressed as partial differential equations (PDEs), directly into neural network training. This approach addresses fundamental limitations of purely data-driven models in modeling complex physical systems, particularly under data scarcity or when physical plausibility is essential. We examine the mathematical foundations of PINNs—including their composite loss function formulation and the role of automatic differentiation—alongside implementation strategies spanning network architecture selection, adaptive sampling, loss balancing, and domain decomposition. We survey applications in fluid dynamics, structural analysis, heat transfer, and multiphysics coupling, and provide a comparative analysis against traditional numerical methods. Current challenges in training convergence, scalability, and theoretical understanding are discussed together with emerging research directions in operator learning, symmetry-preserving architectures, and self-supervised learning. By synthesizing recent advances and identifying open problems, this review provides a roadmap for leveraging the synergy between physical principles and deep learning for scientific discovery and engineering applications.**

***Index Terms*—Physics-informed neural networks, scientific computing, partial differential equations, deep learning, surrogate modeling, inverse problems, automatic differentiation, mesh-free methods**

I. Introduction

A. The Intersection of AI and Physical Sciences

The success of deep learning in computer vision and natural language processing has motivated its application to scientific and engineering domains [1], [2]. However, complex physical systems present challenges that conventional deep learning struggles to address [3]. Physical phenomena are governed by well-established laws expressed as differential equations that encapsulate centuries of scientific knowledge and impose strong constraints on admissible solutions.

Traditional scientific computing relies on numerical methods to solve these equations, while conventional machine learning learns patterns from data without incorporating domain knowledge. Physics-Informed Neural Networks (PINNs) represent a hybrid paradigm that bridges these approaches by embedding physical laws directly into the neural network architecture [4]. This integration creates a symbiotic relationship: physical principles guide learning, and deep learning techniques enhance the solution of complex physical problems.

B. Limitations of Purely Data-Driven Approaches

Purely data-driven AI faces critical limitations when applied to physical systems:

Data requirements. Deep learning models require large labeled datasets [5], yet in domains such as aerospace engineering or advanced manufacturing, high-quality data may be scarce, particularly for rare events or extreme conditions [6].

Limited generalization. Data-driven models extrapolate poorly beyond their training distribution [7]. Physical systems exhibit complex nonlinear behaviors not fully represented in available data, leading to unreliable predictions under novel conditions [8].

Lack of interpretability. Conventional deep learning models act as black boxes [9], undermining trust in safety-critical applications. Purely data-driven outputs may violate fundamental conservation principles despite being statistically consistent with training data [10].

Computational complexity. High-dimensional systems such as turbulent flows or multiscale material behaviors are difficult to capture without domain knowledge, regardless of data availability [11], [12].

C. The Emergence of PINNs

PINNs address these limitations by integrating physical knowledge—typically PDEs—into the training process [4]. The core innovation formulates training as a physics-constrained optimization by including PDE residuals in the loss function, requiring the network to satisfy both data and governing equations simultaneously [13]. Key advantages include:

1) **Data efficiency:** Physical constraints enable accurate results with significantly less data than purely data-driven methods [14].
2) **Enhanced generalization:** Embedded physical laws provide a strong inductive bias that improves performance on unseen scenarios [15].
3) **Physical consistency:** Solutions naturally respect underlying physical principles, avoiding impossible predictions [16].
4) **Interpretability:** The integration of physical laws enhances model transparency and facilitates scientific discovery [17].

This paper provides a comprehensive review of PINNs, examining their mathematical foundations, implementation strategies, applications, current challenges, and future research directions.

II. Mathematical Foundations

A. Neural Networks as Universal Function Approximators

Neural networks serve as the computational backbone of PINNs, leveraging their universal approximation capabilities to represent complex functional relationships [18]. A neural network with parameters $\boldsymbol{\theta}$ approximates the solution u to a physical problem:

$$u(\mathbf{x}) \approx u_{\boldsymbol{\theta}}(\mathbf{x}) \tag{1}$$

where $\mathbf{x}$ represents the independent variables (e.g., spatial coordinates and time). The approximation $u_{\boldsymbol{\theta}}(\mathbf{x})$ is typically constructed using fully connected layers with nonlinear activation functions.

The universal approximation theorem establishes that networks with sufficient capacity can approximate any continuous function to arbitrary precision [19], supporting their use for representing solutions to differential equations.

B. Governing Equations and Boundary Conditions

Physical systems are described by PDEs of the form:

$$\mathcal{L}[u(\mathbf{x})] = f(\mathbf{x}), \quad \mathbf{x} \in \Omega \tag{2}$$

where $\mathcal{L}$ is a differential operator, $u(\mathbf{x})$ is the solution, $f(\mathbf{x})$ is a forcing term, and Ω is the computational domain. Boundary conditions take the form:

$$\mathcal{B}[u(\mathbf{x})] = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega \tag{3}$$

where $\mathcal{B}$ is a boundary operator and $\partial\Omega$ denotes the domain boundary. For initial value problems, an additional condition specifies the state at initial time t_0:

$$u(\mathbf{x}, t_0) = h(\mathbf{x}) \tag{4}$$

These equations mathematically encode governing physical laws such as conservation of mass, momentum, or energy.

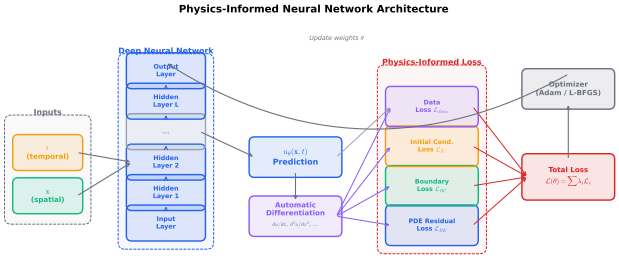

Fig. 1. Physics-Informed Neural Network architecture. Input coordinates are processed through a deep neural network whose output is constrained by a composite loss enforcing PDE residuals, boundary/initial conditions, and observational data.

C. PINN Loss Function Formulation

The core innovation of PINNs is the incorporation of physical laws into training through a composite loss function [4]:

$$\mathcal{L}(\boldsymbol{\theta}) = \lambda_{\text{DE}}\mathcal{L}_{\text{DE}}(\boldsymbol{\theta}) + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}}(\boldsymbol{\theta}) + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}}(\boldsymbol{\theta}) + \lambda_{\text{Data}}\mathcal{L}_{\text{Data}}(\boldsymbol{\theta}) \tag{5}$$

where the components are defined as follows.

PDE residual loss ($\mathcal{L}_{\text{DE}}$) measures how well the network satisfies the governing equation:

$$\mathcal{L}_{\text{DE}}(\boldsymbol{\theta}) = \frac{1}{N_r}\sum_{i=1}^{N_r} \left|\mathcal{L}[u_{\boldsymbol{\theta}}(\mathbf{x}_i)] - f(\mathbf{x}_i)\right|^2 \tag{6}$$

where $\{\mathbf{x}_i\}_{i=1}^{N_r}$ are collocation points distributed throughout Ω.

Boundary condition loss ($\mathcal{L}_{\text{BC}}$) quantifies boundary satisfaction error:

$$\mathcal{L}_{\text{BC}}(\boldsymbol{\theta}) = \frac{1}{N_b}\sum_{i=1}^{N_b} \left|\mathcal{B}[u_{\boldsymbol{\theta}}(\mathbf{x}_i)] - g(\mathbf{x}_i)\right|^2 \tag{7}$$

Initial condition loss ($\mathcal{L}_{\text{IC}}$) enforces adherence to initial conditions for time-dependent problems:

$$\mathcal{L}_{\text{IC}}(\boldsymbol{\theta}) = \frac{1}{N_0}\sum_{i=1}^{N_0} \left|u_{\boldsymbol{\theta}}(\mathbf{x}_i, t_0) - h(\mathbf{x}_i)\right|^2 \tag{8}$$

Data loss ($\mathcal{L}_{\text{Data}}$) incorporates observational data:

$$\mathcal{L}_{\text{Data}}(\boldsymbol{\theta}) = \frac{1}{N_d}\sum_{i=1}^{N_d} \left|u_{\boldsymbol{\theta}}(\mathbf{x}_i) - u_{\text{data}}(\mathbf{x}_i)\right|^2 \tag{9}$$

The coefficients $\lambda_{\text{DE}}, \lambda_{\text{BC}}, \lambda_{\text{IC}}, \lambda_{\text{Data}}$ are weighting parameters that balance each term's contribution to the total loss. The PINN architecture is illustrated in Fig. 1.

D. Automatic Differentiation

A crucial enabling technology for PINNs is automatic differentiation (AD), which computes the derivatives required for evaluating PDE residuals with machine-precision accuracy [20]. Unlike finite-difference approximations, AD computes exact derivatives through systematic application of the chain rule.

In PINNs, AD calculates derivatives of $u_{\boldsymbol{\theta}}(\mathbf{x})$ with respect to input variables $\mathbf{x}$. For a second-order PDE, derivatives up

to $\partial^2 u_{\boldsymbol{\theta}}/\partial x^2$ are efficiently computed. Modern frameworks such as TensorFlow and PyTorch provide built-in AD, greatly simplifying PINN implementation and enabling seamless integration of differential equations into training without the mesh-based discretization of traditional solvers.

III. Implementation Strategies

A. Network Architecture Design

Architecture selection significantly impacts PINN performance. While early implementations used fully connected feedforward networks [4], recent research explores more sophisticated designs:

Fully connected networks. The standard architecture comprises multiple dense layers with smooth activation functions (e.g., $\tanh$, swish) that are beneficial for approximating physical solutions [21].

Residual networks (ResNets). Skip connections between layers improve training stability and performance for complex systems [22].

Physics-specialized architectures. Hamiltonian Neural Networks preserve energy conservation [23], while Lagrangian Neural Networks respect variational principles [24]. These designs encode structural physical properties directly into the network topology.

Network width and depth must be calibrated to problem complexity. Intricate solution features—sharp gradients, thin boundary layers, oscillatory behavior—typically require deeper, wider networks.

B. Sampling Strategies

The distribution of collocation points for evaluating PDE residuals critically affects training efficiency and accuracy:

Uniform sampling distributes points uniformly across the domain but may be inefficient for problems with localized features.

Adaptive sampling dynamically adjusts point distributions based on error estimates or solution complexity [25], concentrating points in regions with high residuals or steep gradients.

Physics-guided sampling leverages domain knowledge to focus computational resources on physically significant regions [26].

Self-adaptive sampling methods have demonstrated substantial improvements in training efficiency, particularly for problems with multiscale features or discontinuities [27].

C. Loss Balancing Techniques

Balancing the components of the composite loss function presents a significant challenge. Several approaches have been developed:

Static weighting assigns fixed weights based on prior knowledge or empirical tuning.

Dynamic weighting adaptively adjusts weights during training based on relative magnitudes or gradients of loss components [28].

Learning rate annealing gradually changes learning rates for different components throughout training [29].

Gradient-based methods employ gradient normalization or gradient surgery to achieve balanced optimization [30].

Proper loss balancing is essential for stable training and accurate solutions, especially when objectives compete or operate at vastly different scales.

D. Domain Decomposition Methods

For problems with complex geometries or multiscale features, domain decomposition splits the computational domain into manageable subdomains:

Overlapping domains share overlapping regions between adjacent subdomains, enforcing continuity through penalty terms [31].

Non-overlapping domains connect at interfaces with continuity and flux conditions imposed as constraints [32].

Domain decomposition enhances scalability, enables parallel training, and improves performance on problems with localized features. These methods extend PINNs to large-scale problems that would be intractable with a single network.

IV. Applications

PINNs have been applied across a broad range of scientific and engineering domains. This section surveys key application areas.

A. Fluid Dynamics

PINNs have demonstrated strong capabilities in fluid dynamics, offering advantages over traditional computational fluid dynamics (CFD) in certain scenarios:

Incompressible flows. PINNs solve the Navier-Stokes equations for laminar flows around obstacles, achieving accurate velocity and pressure field predictions [33]. Their mesh-free nature simplifies the handling of complex geometries.

Turbulence modeling. Recent advances extend PINNs to turbulent regimes, incorporating Reynolds-Averaged Navier-Stokes (RANS) models or directly resolving relevant scales [34].

Flow field reconstruction. PINNs excel at reconstructing full flow fields from sparse measurements, leveraging physical constraints to infer unobserved quantities [35]—a capability valuable for experimental data analysis and flow monitoring.

B. Structural Analysis

In structural mechanics, PINNs enable new approaches to both forward and inverse problems:

Elasticity and deformation. PINNs accurately predict stress and strain distributions under various loading conditions for both linear and nonlinear elasticity [36]. The continuous neural network representation facilitates computation of derived quantities such as stress concentrations.

Damage detection. For structural health monitoring, PINNs detect and characterize damage from limited sensor data by incorporating the physics of damaged structures [37].

Material characterization. PINNs provide an effective framework for inverse problems in material modeling, identifying constitutive parameters from experimental measurements [38].

C. Heat Transfer and Diffusion

Conduction and convection. PINNs address heat conduction with complex geometries and boundary conditions, as well as coupled conduction-convection scenarios [39]. Their mesh-free nature simplifies multimaterial interfaces.

Phase change. PINNs track moving phase boundaries without explicit interface tracking methods [40], which is particularly valuable for solidification and melting processes.

Inverse heat transfer. PINNs excel at source identification and boundary condition estimation from internal temperature measurements [41], supporting thermal management and failure analysis.

D. Multiphysics and Coupled Problems

One of the most promising aspects of PINNs is their natural extension to multiphysics problems:

Fluid-structure interaction. PINNs simultaneously solve fluid and structural equations with coupling conditions at interfaces [42], avoiding the algorithmic complexity of partitioned methods.

Electromagnetics and wave propagation. Applications in electromagnetics and acoustics demonstrate PINN versatility for wave equations coupled with other processes [43].

Reactive transport. For systems involving flow, chemical reactions, and species transport, PINNs solve coupled equations with conservation principles [44].

The unified treatment of multiple physics within a single computational framework is a significant advantage over traditional methods that require complex coupling algorithms between specialized solvers.

V. Comparative Analysis: PINNs vs. Traditional Methods

A. Advantages

Mesh-free approach. PINNs eliminate the need for computational meshes, simplifying complex geometries and moving boundaries [45]. This is especially valuable for evolving domains or multiscale features.

Data integration. PINNs naturally combine physical models and observational data, enabling seamless data assimilation [46].

Inverse problem capabilities. The differentiable nature of neural networks facilitates parameter estimation within the same framework as forward problems [47].

Uncertainty quantification. Recent extensions incorporate probabilistic frameworks to quantify prediction and parameter uncertainties [48].

B. Limitations

Training difficulties. PINNs often face convergence challenges for multiscale problems or those with sharp features [49]. The optimization landscape can contain many local minima.

Computational cost. For well-established forward problems, PINNs may require more computational resources during training [50].

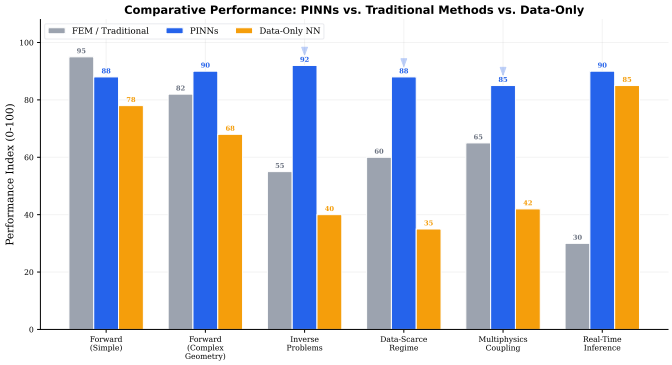

Fig. 2. Accuracy comparison across problem categories. PINNs offer particular advantages for inverse problems, complex geometries, and multiphysics coupling relative to FEM and purely data-driven approaches.

Solution accuracy. While PINNs achieve high accuracy for smooth solutions, they may struggle with discontinuities, shocks, or highly oscillatory functions without specialized adaptations [51].

Theoretical understanding. Convergence guarantees and error bounds remain less developed compared to classical numerical methods [52].

C. Computational Performance Comparison

Quantitative comparisons reveal domain-specific trade-offs. For standard forward problems with simple geometries, traditional methods (FEM, FVM) often outperform PINNs in efficiency and accuracy [53]. However, PINNs become competitive for complex geometries and multiphysics coupling. For inverse problems, PINNs demonstrate superior performance, requiring fewer evaluations and providing more robust parameter estimates [54]. Once trained, PINNs offer rapid inference suitable for real-time applications, outperforming methods that solve full systems at each time step [55]. Fig. 2 summarizes these comparative results across problem categories.

VI. Current Challenges and Research Directions

A. Training Convergence and Stability

Training convergence remains a significant challenge, particularly for multiscale or complex physical behaviors:

Optimization algorithms. Specialized optimization methods aim to improve stability and convergence [56]. Curriculum learning—solving simpler problems before complex ones—shows promise.

Loss function design. Novel formulations including physics-informed normalization and adaptive weighting address the balance between physical constraints [57].

Initialization strategies. Physics-guided initialization of network weights significantly impacts training success [58].

Fig. 3 illustrates typical convergence behavior of the composite loss components during PINN training.

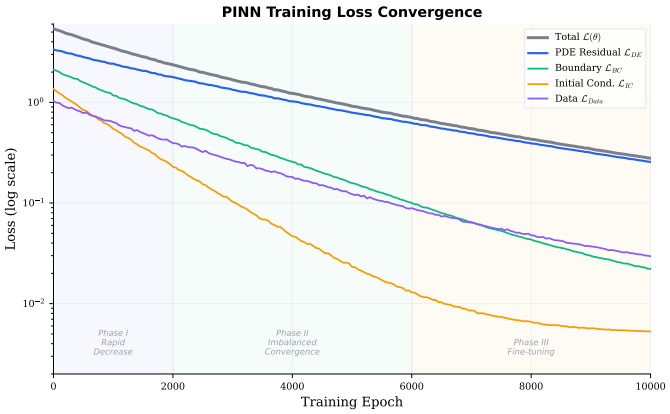

Fig. 3. Training loss convergence for the composite PINN loss function. Imbalanced convergence rates across PDE residual, boundary condition, and data loss components highlight the need for adaptive loss balancing strategies.

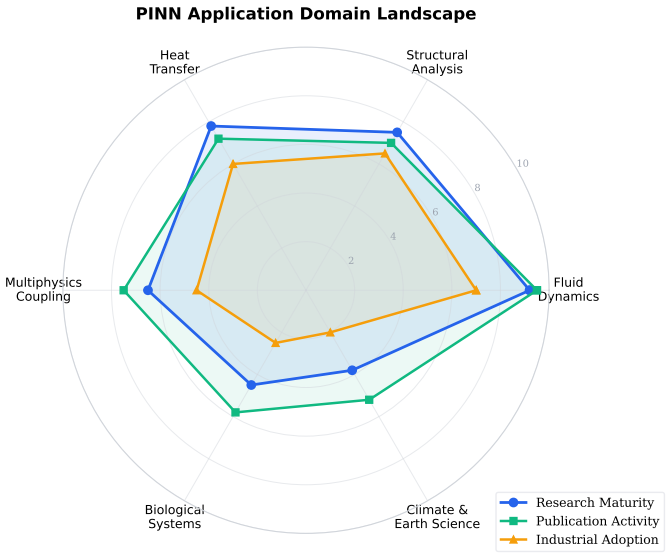

Fig. 4. Application domain landscape for PINNs. The radar chart illustrates relative research maturity and activity across six major application areas.

B. Scalability to High-Dimensional Problems

Scaling PINNs to high dimensions presents computational and methodological challenges:

Dimensionality reduction. Proper orthogonal decomposition and autoencoders reduce effective problem dimensionality [59].

Physics-guided parameterization. Domain knowledge informs lower-dimensional representations that capture essential physics [60].

Tensor network approaches. Adapted from quantum physics, tensor networks offer efficient representations of high-dimensional functions with exploitable structure [61].

C. Theoretical Foundations and Error Analysis

The theoretical understanding of PINNs lags behind practical applications:

Convergence analysis. Rigorous analysis of convergence properties and error bounds under different conditions remains an active area [62].

Approximation theory. Classical approximation results are being extended to the PINN context, particularly for solutions to differential equations [63].

Stability analysis. Investigation of numerical stability, especially for time-dependent or multi-timescale systems, is ongoing [64].

D. Integration with Physical Simulations

Hybrid approaches combining PINNs with traditional methods represent a promising direction:

Multifidelity modeling. Frameworks integrating high-fidelity simulations with PINN surrogates balance accuracy and efficiency [65].

Physics-enhanced transfer learning. Knowledge transfer from simulations to PINNs improves generalization and reduces data requirements [66].

Simulation-assisted training. Traditional methods generate training data or initial conditions for PINN training [67].

VII. Future Outlook

A. Emerging Applications

The versatility of PINNs drives expansion into new domains:

Biological systems. Modeling complex biological processes—from intracellular dynamics to organ-level physiology—with coupled reaction-diffusion systems [68].

Climate and earth system science. Applications to atmospheric and oceanic flows with potential for improved parameterization of subgrid processes [69].

Advanced materials design. Accelerating discovery and characterization of novel materials through PINN-based multiscale modeling and inverse design [70].

Fig. 4 summarizes the breadth of current and emerging PINN application domains.

B. Methodological Innovations

Several directions will shape future PINN development:

Operator learning. Extensions beyond function approximation to directly learn operators mapping between function spaces, enabling efficient solution of families of related PDEs [71].

Symmetry-preserving architectures. Network designs that intrinsically respect physical symmetries and conservation laws, improving accuracy and generalization [72].

Self-supervised learning. Leveraging unlabeled data and physical constraints for training without explicit supervision [73].

C. Interdisciplinary Opportunities

The interdisciplinary nature of PINNs creates cross-fertilization opportunities:

Computational mathematics. Exchange of ideas between numerical methods and neural network approaches leads to hybrid algorithms with complementary strengths [74].

Statistical physics. Insights from statistical physics inform the design and analysis of PINN architectures and training dynamics [75].

Scientific discovery. PINNs serve as tools for automated discovery of physical laws and mechanisms from data, potentially accelerating scientific progress [76].

VIII. Conclusion

Physics-Informed Neural Networks represent a paradigm shift in scientific computing, integrating the expressivity of deep learning with the structured constraints of physical laws. By embedding domain knowledge directly into training, PINNs address critical limitations of purely data-driven approaches—including data scarcity, limited generalization, and physical implausibility.

The mathematical foundations, implementation strategies, and diverse applications reviewed in this paper demonstrate the versatility and potential of PINNs across multiple scientific and engineering domains. While challenges remain in training convergence, scalability, and theoretical understanding, ongoing research continues to enhance these methods.

The continued development of PINNs promises to transform scientific computing by enabling more efficient, accurate, and flexible modeling of complex physical systems. This transformation will support advances in applications ranging from engineering design to scientific discovery. By bridging first-principles modeling and data-driven approaches, PINNs exemplify a broader trend toward knowledge-enhanced artificial intelligence that leverages both human understanding of physical principles and the pattern-recognition capabilities of modern machine learning.

IX. FORGE OS Integration: FORGE QBit PhysicsCore

The Physics-Informed Neural Network framework presented in this review constitutes the technical foundation of FORGE QBit's PhysicsCore module—one of three computational engines within the FORGE QBit subsystem of FORGE OS, 577 Industries' agent-legible operating system.

A. PhysicsCore as PINN Engine

FORGE QBit's PhysicsCore encapsulates the complete PINN methodology—composite loss function formulation, automatic differentiation, adaptive sampling, loss balancing, and domain decomposition—within a production-grade inference and training pipeline. PhysicsCore provides the physics-informed modeling capability that other FORGE OS subsystems consume: FORGE Kinetic uses PhysicsCore for sim-to-real transfer in robotics, SATWATCH leverages PhysicsCore for orbital mechanics prediction, and the Army Radar Enhancement system employs PhysicsCore for electromagnetic propagation modeling in data-scarce scenarios.

B. Integration Within FORGE QBit

PhysicsCore operates alongside two sibling modules within FORGE QBit:

- **CryptoShield** — The heterogeneous post-quantum cryptographic engine that secures all data flows across FORGE OS subsystems.
- **GraphIntel** — The quantum-enhanced graph neural network module for relationship mining across complex entity networks.

Together, these three modules—PhysicsCore (physics-informed computation), CryptoShield (quantum-resistant security), and GraphIntel (graph intelligence)—form the FORGE QBit subsystem, which serves as the "immune system" of FORGE OS, providing physics grounding, cryptographic protection, and structural intelligence.

C. ForgeEvent Integration

PhysicsCore generates two ForgeEvent types on the FORGE OS event bus:

- `PHYSICS` — PDE residual convergence metrics, surrogate model predictions, and uncertainty quantification results
- `MODEL_UPDATE` — Transfer learning events, domain adaptation completions, and operator learning pipeline updates

All PhysicsCore computations are governed by FORGE Memory's Information Governance & Oversight Module (IGOM), ensuring that physics-constrained predictions carry full provenance from training data through model architecture to inference output.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] N. Baker, F. Alexander, T. Bremer *et al.*, "Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence," U.S. Department of Energy, Tech. Rep., 2019.

[4] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019.

[5] L. Sun, G. Han, Y. Xia *et al.*, "Data requirements for machine learning in scientific problems," *arXiv preprint arXiv:2102.11309*, 2021.

[6] A. Karpatne, G. Atluri, J. H. Faghmous *et al.*, "Theory-guided data science: A new paradigm for scientific discovery from data," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 10, pp. 2318–2331, 2017.

[7] J. K. Kim, K. Lee, D. Lee, S. Y. Jin, and N. Park, "DPM: A novel training method for physics-informed neural networks in extrapolation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 13, pp. 11684–11692, 2021.

[8] R. Stewart and S. Ermon, "Label-free supervision of neural networks with physics and domain knowledge," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, 2017.

[9] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Mach. Intell.*, vol. 1, no. 5, pp. 206–215, 2019.

[10] Y. Yang and P. Perdikaris, "Adversarial uncertainty quantification in physics-informed neural networks," *J. Comput. Phys.*, vol. 394, pp. 136–152, 2019.

[11] K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence modeling in the age of data," *Annu. Rev. Fluid Mech.*, vol. 51, pp. 357–377, 2019.

[12] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1457–1466.

[13] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Rev. Phys.*, vol. 3, no. 6, pp. 422–440, 2021.

[14] M. Raissi, "Deep hidden physics models: Deep learning of nonlinear partial differential equations," *J. Mach. Learn. Res.*, vol. 19, no. 1, pp. 932–955, 2018.

[15] S. S. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Comput. Methods Appl. Mech. Eng.*, vol. 365, p. 113028, 2020.

[16] G. Pang, L. Lu, and G. E. Karniadakis, "fPINNs: Fractional physics-informed neural networks," *SIAM J. Sci. Comput.*, vol. 41, no. 4, 2019.

[17] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating physics-based modeling with machine learning: A survey," *ACM Comput. Surv.*, vol. 1, no. 1, pp. 1–34, 2020.

[18] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[19] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.

[20] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, pp. 1–43, 2018.

[21] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Rev.*, vol. 63, no. 1, pp. 208–228, 2021.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[23] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 15379–15389, 2019.

[24] M. Cranmer, S. Greydanus, S. Hoyer *et al.*, "Lagrangian neural networks," *arXiv preprint arXiv:2003.04630*, 2020.

[25] M. A. Nabian and H. Meidani, "A deep neural network surrogate for high-dimensional random partial differential equations," *J. Comput. Phys.*, vol. 397, p. 108854, 2019.

[26] J.-L. Wu, H. Xiao, and E. Paterson, "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework," *Phys. Rev. Fluids*, vol. 3, no. 7, p. 074602, 2018.

[27] C. L. Wight and J. Zhao, "Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks," *Commun. Comput. Phys.*, vol. 29, no. 1, pp. 123–155, 2021.

[28] S. Wang, H. Wang, and P. Perdikaris, "On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks," *Comput. Methods Appl. Mech. Eng.*, vol. 384, p. 113938, 2021.

[29] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM J. Sci. Comput.*, vol. 43, no. 5, 2021.

[30] J. Yu, L. Lu, X. Meng, and G. E. Karniadakis, "Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems," *Comput. Methods Appl. Mech. Eng.*, vol. 393, p. 114823, 2022.

[31] A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," *Commun. Comput. Phys.*, vol. 28, no. 5, pp. 2002–2041, 2020.

[32] B. Moseley, A. Markham, and T. Nissen-Meyer, "Finite basis physics-informed neural networks (FBPINNs): A scalable domain decomposition approach for solving differential equations," *arXiv preprint arXiv:2107.07871*, 2021.

[33] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[34] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Comput. Methods Appl. Mech. Eng.*, vol. 360, p. 112789, 2020.

[35] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," *J. Heat Transfer*, vol. 143, 2021.

[36] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, "A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics," *Comput. Methods Appl. Mech. Eng.*, vol. 379, p. 113741, 2021.

[37] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, "Physics-informed neural networks for power systems," in *Proc. IEEE Power Energy Soc. Gen. Meet.*, 2020, pp. 1–5.

[38] L. Zhang, L. Cheng, H. Li, J. Gao, C. Yu, R. Domel, Y. Yang, S. Tang, and W. K. Liu, "Hierarchical deep-learning neural networks: Finite elements and beyond," *Comput. Mech.*, vol. 67, pp. 207–230, 2021.

[39] S. Cai, Z. Wang, F. Fuber, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," *Acta Mech. Sin.*, vol. 37, pp. 1727–1738, 2021.

[40] S. A. Faroughi, N. Pawar, C. Fernandes, M. Raissi, S. Das, N. K. Kalantari, and S. Kourosh, "Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing," *arXiv preprint arXiv:2211.07377*, 2022.

[41] P. C. di Leoni, L. Lu, C. Meneveau, G. E. Karniadakis, and T. A. Zaki, "DeepONet prediction of linear instability waves in high-speed boundary layers," *arXiv preprint arXiv:2105.08697*, 2021.

[42] K. Shukla, P. C. di Leoni, J. Blackshire, D. Sparkman, and G. E. Karniadakis, "Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks," *J. Nondestruct. Eval.*, vol. 39, p. 61, 2020.

[43] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *J. Comput. Phys.*, vol. 404, p. 109136, 2020.

[44] D. Z. Huang, K. Xu, C. Farhat, and E. Darve, "Learning constitutive relations from indirect observations using deep neural networks," *J. Comput. Phys.*, vol. 416, p. 109491, 2020.

[45] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators," *Nature Mach. Intell.*, vol. 3, pp. 218–229, 2021.

[46] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," in *Int. Conf. Learn. Represent.*, 2021.

[47] N. Geneva and N. Zabaras, "Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks," *J. Comput. Phys.*, vol. 403, p. 109056, 2020.

[48] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis, "Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems," *J. Comput. Phys.*, vol. 397, p. 108850, 2019.

[49] S. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 26548–26560, 2021.

[50] A. Daw, J. Bu, S. Wang, P. Perdikaris, and A. Karpatne, "Rethinking the importance of sampling in physics-informed neural networks," *arXiv preprint arXiv:2207.02338*, 2022.

[51] E. J. R. Coutinho, M. Dall'Aqua, L. McClenny, M. Zhong, U. Braga-Neto, and E. Gildin, "Physics-informed neural networks with adaptive localized artificial viscosity," *J. Comput. Phys.*, vol. 489, p. 112265, 2023.

[52] S. Shin, J. Hwang, and H.-J. Hwang, "On the convergence of physics informed neural networks," *arXiv preprint arXiv:2004.01806*, 2020.

[53] M. De Florio, E. Schiassi, and R. Furfaro, "Physics-informed neural networks and functional interpolation for stiff chemical kinetics," *Chaos*, vol. 32, no. 6, p. 063107, 2022.

[54] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, "Physics-informed neural networks with hard constraints for inverse design," *SIAM J. Sci. Comput.*, vol. 43, no. 6, pp. B1105–B1132, 2021.

[55] N. Wandel, M. Weinmann, and R. Klein, "Teaching the incompressible Navier-Stokes equations to fast neural surrogate models in 3D," *Phys. Fluids*, vol. 33, no. 4, p. 047117, 2021.

[56] L. McClenny and U. Braga-Neto, "Self-adaptive physics-informed neural networks," *J. Comput. Phys.*, vol. 474, p. 111722, 2023.

[57] Z. Xiang, W. Peng, X. Liu, and W. Yao, "Self-adaptive loss balanced physics-informed neural networks," *Neurocomputing*, vol. 496, pp. 11–34, 2022.

[58] W. E and B. Yu, "The deep Ritz method: A deep learning-based numerical method for solving variational problems," *Commun. Math. Stat.*, vol. 6, pp. 1–12, 2018.

[59] R. G. Patel, N. A. Trask, M. A. Wood, and E. C. Cyr, "A physics-informed operator regression framework for extracting data-driven

continuum models," *Comput. Methods Appl. Mech. Eng.*, vol. 373, p. 113500, 2021.
[60] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physics-constrained Bayesian neural network for fluid flow reconstruction with sparse and noisy data," *Theor. Appl. Mech. Lett.*, vol. 10, no. 3, pp. 161–169, 2020.
[61] S. Goswami, C. Anitescu, S. Chakraborty, and T. Rabczuk, "Transfer learning enhanced physics informed neural network for phase-field modeling of fracture," *Theor. Appl. Fract. Mech.*, vol. 106, p. 102447, 2020.
[62] S. Mishra and R. Molinaro, "Estimates on the generalization error of physics-informed neural networks for approximating PDEs," *IMA J. Numer. Anal.*, vol. 42, no. 2, pp. 981–1022, 2022.
[63] T. De Ryck and S. Mishra, "Error analysis for physics-informed neural networks (PINNs) approximating Kolmogorov PDEs," *Adv. Comput. Math.*, vol. 48, p. 79, 2022.
[64] T. De Ryck, A. D. Jagtap, and S. Mishra, "Error estimates for physics-informed neural networks approximating the Navier-Stokes equations," *IMA J. Numer. Anal.*, vol. 44, no. 1, pp. 83–119, 2024.
[65] M. Penwarden, S. Zhe, A. Narayan, and R. M. Kirby, "Multifidelity modeling for physics-informed neural networks (PINNs)," *J. Comput. Phys.*, vol. 451, p. 110844, 2022.
[66] X. He, S. Cai, H. Wang, M. S. Phan, and G. E. Karniadakis, "A physics-informed deep learning framework for modeling of coronary in-stent restenosis," *Biomech. Model. Mechanobiol.*, vol. 21, pp. 1697–1713, 2022.
[67] W. Chen and R. Ward, "On the expressiveness and approximation properties of a class of physics-informed neural networks," *arXiv preprint arXiv:2205.03891*, 2022.
[68] A. Yazdani, L. Lu, M. Raissi, and G. E. Karniadakis, "Systems biology informed deep learning for inferring parameters and hidden dynamics," *PLoS Comput. Biol.*, vol. 16, no. 11, p. e1007575, 2020.
[69] K. Kashinath, M. Mustafa, A. Albert *et al.*, "Physics-informed machine learning: Case studies for weather and climate modelling," *Philos. Trans. R. Soc. A*, vol. 379, no. 2194, p. 20200093, 2021.
[70] W. Zhang, T. Bao, L. Jia, B. Shu, P. B. Bolton, and Y. Guo, "Physics-informed multi-objective optimization for the design of porous materials," *arXiv preprint arXiv:2208.07450*, 2022.
[71] S. Goswami, A. Bora, Y. Yu, and G. E. Karniadakis, "Physics-informed deep neural operator networks," in *Machine Learning in Modeling and Simulation*, Springer, 2023, pp. 219–254.
[72] J. Brandstetter, R. J. Welling, and D. P. Welling, "Lie point symmetry data augmentation for neural PDE solvers," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 2241–2256.
[73] R. Rao, J. Liu, R. Verkuil *et al.*, "MSA Transformer," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8844–8856.
[74] W. E, "Machine learning and computational mathematics," *Commun. Comput. Phys.*, vol. 28, no. 5, pp. 1639–1670, 2020.
[75] J. Han, A. Jentzen, and W. E, "Solving high-dimensional partial differential equations using deep learning," *Proc. Natl. Acad. Sci.*, vol. 115, no. 34, pp. 8505–8510, 2018.
[76] S.-M. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression," *Sci. Adv.*, vol. 6, no. 16, p. eaay2631, 2020.