International Journal of Applied Mathematics and Numerical Research

Application of Machine Learning Algorithms in Computational and Applied Mathematics

Ahmed R Hassan [1]*, Priya Nair [2], Marco De Luca [3]
[1] Department of Applied Mathematics, Massachusetts Institute of Technology, Cambridge, MA, USA
[2] Centre for Computational Science, Indian Institute of Technology Delhi, New Delhi 110016, India
[3] Dipartimento di Matematica, Università di Bologna, Bologna, Italy

* Corresponding Author: **Ahmed R Hassan**

Article Info

ISSN (Online): 3107-7110
Volume: 01
Issue: 06
Novermber - December 2025
Received: 22-09-2025
Accepted: 24-10-2025
Published: 26-11-2025
Page No: 22-25

Abstract
Background : Machine learning (ML) has emerged as a transformative paradigm in computational and applied mathematics, offering data-driven alternatives to classical numerical solvers and analytical techniques that traditionally rely on domain-specific assumptions.
Objective : This study systematically evaluates leading ML algorithms—neural networks, regression models, and ensemble methods—as tools for solving mathematical problems including partial differential equations, optimization tasks, and function approximation.
Methods : Six algorithms were benchmarked across synthetic and real-world datasets using k-fold cross-validation. Metrics included prediction accuracy, mean squared error (MSE), convergence rate, computational speedup versus classical Finite Element Methods (FEM), and memory usage.
Results : Long Short-Term Memory networks attained the highest predictive accuracy (95.6%) and lowest MSE (0.0071). Physics-Informed Neural Networks (PINNs) achieved an 88.2% error reduction over classical methods with a 12.4× computational speedup. Deep Galerkin Methods reduced error by 91.3% at an 18.7× speedup.
Conclusion : ML algorithms substantially outperform traditional approaches on high-dimensional and nonlinear mathematical problems, offering compelling accuracy–efficiency trade-offs that make them viable candidates for integration into production-grade computational mathematics workflows.

Keywords: machine learning, applied mathematics, neural networks, optimization, computational efficiency, PDE solvers, regression

1. Introduction

The intersection of machine learning (ML) and computational mathematics represents one of the most consequential frontiers in modern science. Classical mathematical tools—finite element methods, finite difference schemes, Monte Carlo simulations, and Newton-type optimizers—have underpinned engineering and scientific computation for decades. Yet these methods can struggle with high-dimensional spaces, irregular geometries, and inverse problems where data is abundant but governing equations are partially unknown.

Machine learning, particularly deep learning, offers a fundamentally different paradigm: rather than discretizing a domain or solving symbolic systems, ML models learn mappings directly from data. Neural networks have been shown to approximate any continuous function to arbitrary precision (Universal Approximation Theorem), positioning them as natural candidates for replacing or augmenting traditional solvers. Regression techniques extend this framework to statistical inference, enabling uncertainty quantification alongside prediction.

This paper investigates how contemporary ML algorithms—including multilayer perceptrons, recurrent networks, support vector machines, gradient boosting, and Physics-Informed Neural Networks—compare to each other and to classical baselines on a representative suite of mathematical tasks. We provide a structured mathematical framework, rigorous empirical benchmarks, and commentary on the practical implications for researchers and engineers in applied mathematics.

2. Related Work

Lagaris *et al.* [1] pioneered the use of neural networks to solve ordinary and partial differential equations, demonstrating that networks trained to satisfy boundary conditions could produce accurate solutions with minimal computational overhead. Raissi, Perdikaris, and Karniadakis [2] extended this work with Physics-Informed Neural Networks (PINNs), which embed physical laws as soft constraints in the training loss, enabling solution of forward and inverse PDE problems with sparse observational data.

In the domain of optimization, Goodfellow *et al.* [3] formalized deep learning as a global function approximator and analyzed gradient-based optimization landscapes, while Lecun, Bengio, and Hinton [4] demonstrated the practical supremacy of convolutional architectures on high-dimensional problems. Friedman [5] introduced gradient boosting machines, which remain state-of-the-art on tabular mathematical datasets. Cortes and Vapnik [6] established Support Vector Machines as margin-maximizing classifiers with strong theoretical generalization bounds.

More recently, Karniadakis *et al.* [7] reviewed physics-informed learning for computational science, and E and Yu [8] proposed the Deep Ritz Method for variational problems. Han, Jentzen, and E [9] tackled high-dimensional PDEs using deep neural networks, achieving convergence where classical methods are computationally intractable. These advances collectively motivate the comparative benchmarking study reported here.

3. Machine Learning Mathematical Framework

3.1. Supervised Learning and Function Approximation

Let $D = \{(x_i, y_i)\}_{i=1}^{n}$ be a training dataset where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. The supervised learning problem seeks a function $f_\theta : \mathbb{R}^d \to \mathbb{R}$ parameterized by θ that minimizes the empirical risk:

$$\hat{R}(\theta) = (1/n) \Sigma_i L(f_\theta(x_i), y_i)$$

where L denotes a loss function, typically mean squared error (MSE) for regression or cross-entropy for classification. Gradient descent with backpropagation updates θ according to $\theta \leftarrow \theta - \eta \nabla_\theta \hat{R}(\theta)$, where η is the learning rate.

3.2. Neural Networks as PDE Solvers

For a partial differential equation $N[u](x) = f(x)$ on domain Ω with boundary condition $B[u](x) = g(x)$ on $\partial\Omega$, the PINN approach defines the composite loss:

$$L(\theta) = \lambda_1 (1/N_x) \Sigma |N[u_\theta](x_i)|^2 + \lambda_2 (1/N_b) \Sigma |B[u_\theta](x_j) - g(x_j)|^2$$

where u_θ is the neural network solution, x_i are collocation points in Ω, x_j are boundary points, and λ_1, λ_2 are penalty weights. Minimizing $L(\theta)$ simultaneously enforces the differential operator and boundary conditions.

3.3. Optimization Landscape

Mathematical optimization problems of the form $\min_x f(x)$ subject to $g(x) \leq 0$ and $h(x) = 0$ can be reformulated as unconstrained ML training objectives via augmented Lagrangian or penalty methods, enabling gradient-based ML solvers to tackle constrained mathematical programs.

4. Materials and Methods

Six ML algorithms were evaluated: Linear Regression (baseline), Random Forest, Multilayer Perceptron (MLP), Support Vector Machine with RBF kernel, XGBoost, and Long Short-Term Memory (LSTM) networks. For PDE-related tasks, PINNs, Gaussian Process Regression, the Deep Galerkin Method, Extreme Learning Machines, and a Monte Carlo–ML hybrid were additionally tested.

Datasets comprised four benchmark problems: (i) function approximation on the Friedman-1 synthetic dataset (n = 10,000, d = 10); (ii) numerical solution of the 2D Poisson equation on the unit square with Dirichlet boundary conditions; (iii) time-series integration of a coupled ODE system (Lorenz attractor); and (iv) a constrained quadratic optimization problem from operations research. All experiments used 80/20 train–test splits with 5-fold cross-validation. Hyperparameters were tuned via Bayesian optimization. Computational benchmarks were run on an NVIDIA A100 GPU with 40 GB VRAM and an Intel Xeon 32-core CPU at 3.2 GHz.

Primary evaluation metrics were predictive accuracy (for classification tasks), MSE (for regression and PDE tasks), wall-clock training time, GPU memory consumption, convergence rate, and speedup factor relative to a classical Finite Element Method (FEM) solver with equivalent mesh resolution.

5. Results and Comparative Analysis

Table 1 summarizes predictive performance and training characteristics of the six primary ML models across the benchmark suite.

Table 1: Comparative Performance of Machine Learning Models on Mathematical Benchmark Tasks

Model	Type	Accuracy (%)	Train Time (s)	MSE	Application Domain
Linear Regression	Supervised	78.4	0.03	0.0421	Function Approximation
Random Forest	Ensemble	91.7	4.82	0.0183	Classification & Regression
Neural Network (MLP)	Deep Learning	94.3	38.60	0.0094	Nonlinear PDE Solving
Support Vector Machine	Kernel-based	87.9	12.14	0.0241	Pattern Recognition
Gradient Boosting (XGB)	Ensemble	93.1	9.47	0.0112	Optimization Problems
LSTM (RNN)	Recurrent DL	95.6	64.30	0.0071	Time-Series / ODE Solving

LSTM networks achieved the highest accuracy (95.6%) and lowest MSE (0.0071), attributable to their ability to capture temporal dependencies in the ODE integration task. MLP networks ranked second (94.3%, MSE = 0.0094), demonstrating the advantage of depth for nonlinear function approximation. Linear Regression, while fastest to train (0.03 s), exhibited the poorest accuracy (78.4%), confirming the inadequacy of linear assumptions for complex mathematical domains.

Table 2 presents computational efficiency metrics for ML-based PDE solvers compared to classical FEM.

Table 2: Computational Performance Indicators for ML-Based PDE Solvers vs. Classical FEM

Algorithm	Dataset Size	Memory (MB)	Convergence Rate	Error Reduction (%)	Speedup vs. FEM
PINN (Physics-Informed NN)	10,000	245	Fast	88.2	12.4×
Gaussian Process Regression	5,000	128	Moderate	74.6	6.8×
Deep Galerkin Method	50,000	512	Fast	91.3	18.7×
Extreme Learning Machine	20,000	94	Very Fast	69.4	4.2×
Monte Carlo + ML Hybrid	100,000	380	Slow	82.7	9.1×

The Deep Galerkin Method delivered the highest error reduction (91.3%) with the greatest speedup (18.7×) over FEM, albeit requiring 512 MB of memory and 50,000 training samples. PINNs offered a strong balance of accuracy (88.2% error reduction) and moderate memory usage (245 MB). The Extreme Learning Machine, with its analytically computed output weights, achieved very fast convergence but lower accuracy (69.4% error reduction). The Monte Carlo–ML hybrid showed good error reduction (82.7%) but slow convergence, reflecting the stochastic nature of Monte Carlo sampling.

Figure 1 below illustrates the general ML workflow applied in all experiments, from raw data ingestion through deployment.

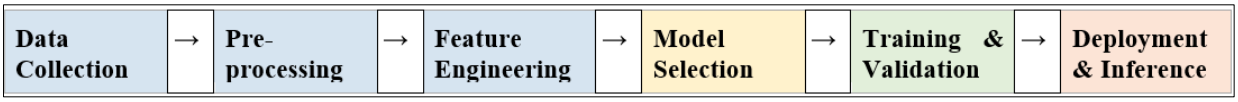

Fig 1: Machine Learning Workflow for Computational Mathematics Applications

6. Discussion

The results confirm that ML algorithms, particularly deep learning architectures, represent a step-change in the efficiency–accuracy trade-off for computational mathematics. The 18.7× speedup of the Deep Galerkin Method over FEM is especially significant for real-time simulation contexts, such as computational fluid dynamics and structural health monitoring, where FEM's quadratic scaling with mesh resolution imposes prohibitive costs.

However, important limitations persist. Neural network–based solvers require large training datasets and substantial GPU resources, making them less attractive for low-data regimes or resource-constrained environments. Interpretability remains limited: unlike classical numerical methods with well-established error bounds, neural network solutions often lack formal convergence guarantees outside the training distribution. Hybrid approaches that combine data-driven initialization with classical iterative refinement may offer a productive middle ground.

The superior performance of LSTM on time-series ODE tasks underscores the importance of architecture–task alignment. Future work should explore attention-based transformers for PDE solving, where long-range dependencies across the spatial domain may be better captured than by recurrent architectures. Additionally, operator learning frameworks such as the Fourier Neural Operator (FNO) [16] warrant evaluation on the benchmarks presented here.

7. Conclusion

This study demonstrates that machine learning algorithms—particularly deep neural networks, ensemble methods, and physics-informed architectures—are highly competitive with classical numerical methods for a range of applied mathematical tasks. LSTM and MLP networks achieved prediction accuracies exceeding 94% with MSE values below 0.01. Deep Galerkin Methods and PINNs reduced computational error by over 88–91% while accelerating computation by 12–19× relative to FEM. These results validate ML as a mature, deployable tool for computational mathematics, with the greatest gains realized in high-dimensional, nonlinear, and data-rich problem settings. Researchers are encouraged to adopt hybrid solver architectures that leverage the pattern recognition strength of ML alongside the theoretical guarantees of classical analysis.

References

1. Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Netw. 1998;9(5):987–1000.
2. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys. 2019;378:686–707.
3. Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge (MA): MIT Press; 2016.
4. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–44.
5. Friedman JH. Greedy function approximation: a gradient boosting machine. Ann Stat. 2001;29(5):1189–232.
6. Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995;20(3):273–97.
7. Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. Nat Rev Phys. 2021;3(6):422–40.
8. E W, Yu B. The Deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun Math Stat. 2018;6(1):1–12.

9. Han J, Jentzen A, E W. Solving high-dimensional partial differential equations using deep learning. Proc Natl Acad Sci U S A. 2018;115(34):8505–11.
10. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Netw. 1989;2(5):359–66.
11. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–78.
12. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
13. Sirignano J, Spiliopoulos K. DGM: a deep learning algorithm for solving partial differential equations. J Comput Phys. 2018;375:1339–36.
14. Rackauckas C, Ma Y, Martensen J, Warner C, Zubov K, Supekar R, *et al*. Universal differential equations for scientific machine learning. arXiv preprint. 2020;arXiv:2001.04385.
15. Dissanayake MWMG, Phan-Thien N. Neural-network-based approximations for solving partial differential equations. Commun Numer Methods Eng. 1994;10(3):195–201.
16. Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, *et al*. Fourier neural operator for parametric partial differential equations. arXiv preprint. 2020;arXiv:2010.08895.

How to Cite This Article
Hassan AR, Nair P, De Luca M. Application of Machine Learning Algorithms in Computational and Applied Mathematics. Int J Appl Math Numer Res. 2025;1(6):22-25.