What is a PINN?

PINNs: Universal Function Approximators

- Physics-Informed Neural Networks (PINNs) are a type of **universal function approximator**.
- They embed the knowledge of physical laws, often described by Partial Differential Equations (PDEs), directly into the learning process.
- PINNs are trained to solve supervised learning tasks while simultaneously respecting these physical laws.
- They represent a new family of **data-efficient spatio-temporal function approximators**.

Why are PINNs Necessary?

Addressing Data Scarcity in Scientific Domains

- Neural Networks traditionally require a **large amount of data** to train successfully.
- However, in many scientific domains (like biological and engineering systems), collecting large-scale data is often **not possible** or requires immense effort.
- PINNs overcome this challenge by leveraging **physical laws** (conservation of mass, momentum, energy, etc.) as an important source of data.
- Incorporating these physical invariances can help **improve generalization** or regularize training, especially when observed data is scarce.

History of Physics-Informed ML

- The concept of physics-informed machine learning debuted in the **1990s**.
- These ideas appeared in scattered papers throughout the decade.
- Around **2010**, a resurgence in machine learning breathed new life into this promising field.
- The publication count for research related to "Physics-Informed Neural Networks" has increased dramatically since 2017.

The PINN Milestone (2019)

- The specific **Physics-Informed Neural Networks (PINN) approach** was formally introduced by Raissi et al. in their seminal 2019 paper:
  - Title: "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations".
- Raissi et al. proposed incorporating known physical laws, represented by the PDE, directly into the neural network architecture.
- This framework allows the network to learn from **both observed data and the underlying physics** of the system being modeled.
- Since 2019, the approach has been widely applied across fields such as fluid mechanics, heat transfer, and materials science.

Key Advantages of PINNs

- **Physics-based Constraints:** Physical principles are incorporated as constraints during training, ensuring predictions are **physically meaningful** and accurate.
- **Yields Quick Results:** Once trained, PINNs yield results quickly, often in just a **fraction of a second**.
- **Improved Generalization:** Physical invariances can **improve generalization** or regularize training, especially when data is scarce.
- **Versatility:** PINNs are applicable to a wide range of physical problems, including fluid dynamics, materials science, and quantum mechanics.

Data Robustness and Efficiency

- **Data Robustness:** PINNs can handle **noisy or incomplete data** by learning the underlying physical relationships to make accurate predictions.
- **Faster Computation:** PINNs can often solve physical problems much faster than traditional methods once the network has been trained.
- **Automation:** They reduce the need for manual intervention required by traditional methods, such as setting up boundary conditions or adjusting parameters, thereby **speeding up the modeling process**.
- PINNs are popular because they benefit from late AI research and are easily applicable to any topic.

Disadvantages and Implementation Challenges

- **Limited Interpretability:** The complexity of the neural network can make it difficult to understand how the model arrived at its predictions, which is a disadvantage when **physical principles need to be understood** and validated.
- **Data Requirements:** PINN typically requires a large amount of data to be effective, especially for **highly complex problems**.
- **Computational Intensity:** Training can be computationally intensive, leading to **longer training times** and a need for powerful computing resources.
- **Hyperparameter Sensitivity:** Performance is highly dependent on hyperparameters (layers, neurons, learning rate), and selection can be **challenging and time-consuming**.
- **Implementation Difficulty:** Implementing PINNs requires expertise in **both neural networks and physics**.

Solving Boundary Value Problems (BVPs)

- PINNs are used to approximate the latent (hidden) solution $u(t, x)$ of parameterized and nonlinear PDEs, such as those defined by the general form:

$$\frac{\partial u}{\partial t} + N[u; \lambda] = 0$$

  where $N[u; \lambda]$ is a nonlinear operator.
- BVPs require a governing equation $F(\hat{u}, \dots) = 0$, Boundary Conditions (BC), and Initial Conditions (IC).
- The PINN uses an **Artificial Neural Network** ($\hat{u}$) as the trial/basis function space to approximate the solution $u(t, x)$.

PINN vs. Traditional FEM

Table: Comparison of key components in FEM and PINN.

Component	**Traditional FEM**	**PINN**
Discretization	Mesh	Neural network architecture
Trial/basis function	Piece-wise polynomials	Artificial neural network
Parameters	Mesh nodal values	Network weights and biases
Resolution	Matrix inversion	Stochastic optimization

- For a well-posed problem, the solution found via FEM is unique; the PINN solution is **non-unique** due to optimization and generalization errors.

The PINN Loss Function Structure

- The overall objective is to minimize a total loss function, which combines three primary residual components:

$$L_{total} = \mathcal{L}_U + \mathcal{L}_{BC/IC} + \mathcal{L}_{PDE}$$

- The loss function measures compliance to the PDE, the Boundary/Initial Conditions (BCs/ICs), and the available data.

Components of the Total Loss

- **Data Loss ($\mathcal{L}_U$):** Measures the residual between the PINN approximation $\mathcal{N}(X)$ and ground truth measurements u at points $X \in \Omega$:

$$\mathcal{L}_U = \sum_{X\in\Omega} (\mathcal{N}(X) - u)^2 \quad \text{(Often using Mean Squared Error)}$$

- **Boundary/Initial Condition Loss ($\mathcal{L}_{BC/IC}$):** Measures the residual between the PINN approximation $\mathcal{N}(X)$ and prescribed BC/IC values $u_{BC}(X)$ at points $X \in \partial\Omega$.
- **Physics Loss ($\mathcal{L}_{PDE}$):** Measures compliance to the PDE governing equation $F(\hat{u}, \frac{\partial \hat{u}}{\partial X}, \dots) = 0$:

$$\mathcal{L}_{PDE} = \sum_{X\in\Omega} F(\hat{u}, \frac{\partial \hat{u}}{\partial X}, \frac{\partial \hat{u}}{\partial t}, \dots)^2$$

Automatic Differentiation for Physics Loss

- The key to the PINN framework is the ability to compute the derivatives needed for the $\mathcal{L}_{PDE}$ term (like $\frac{\partial \hat{u}}{\partial X}$ and $\frac{\partial \hat{u}}{\partial t}$).
- This is performed using **Automatic Differentiation (AD)**.
- AD allows the computation of derivatives by applying the chain rule through the neural network architecture.
- The training itself involves finding the network parameters (weights and biases) that minimize the total loss function via **stochastic optimization**.

Imposing Boundary Conditions: Soft Constraints

- **Soft Constraints** penalize the non-respect of BCs by adding $\mathcal{L}_{BC/IC}$ as a loss term.
- This is generally the simplest method for incorporating BCs.
- **Pros:**
  - The approach is **general** and seamless to implement.
  - It is considered a relaxed constraint.
- **Cons:**
  - It leads to a **multi-term optimization problem**, which can make convergence harder.

Imposing Boundary Conditions: Hard Constraints

- **Hard Constraints** directly enforce BCs by applying a mask function on the neural network output.
- The BC-compliant output u is given by:

$$u = F_{mask}[\mathcal{N}(X)]$$

  where F_{mask} ensures the output matches U_{BC} on the boundary $\partial\Omega$.
- **Pros:**
  - Provides **exact imposition** of the boundary conditions.
  - Often leads to **better convergence**.
- **Cons:**
  - This method is **specific to every problem**, although generalization is possible.

Solving Forward and Inverse Problems

- **Forward Problems:**
  - The goal is to solve the PDE (find the solution $\hat{u}$) when all parameters and boundary conditions are known.
  - If the PDE and BCs/ICs are sufficient, the PINN requires **no labeled data** ($\mathcal{L}_U = 0$).
- **Inverse Problems:**
  - The goal is to determine **unknown PDE parameters** (p_i) given sparse data ($\mathcal{L}_U > 0$).
  - The unknown parameters p_i are treated as **trainable parameters** alongside the network weights during optimization.
  - Inverse quantification is used for finding material parameters.
  - PINNs, particularly variants like SPINN, have been shown to be 10x more accurate than regression models in benchmark inverse quantification problems dealing with noisy data.

Advanced Applications

- **Continuum Mechanics:** PINNs can model complex physics problems such as solid mechanics, utilizing equations like momentum balance and material laws.
- **Uncertainty Propagation (PINN-PC):**
  - The PINN-Polynomial Chaos (PINN-PC) framework is used to quantify total uncertainty in forward and inverse stochastic problems.
  - This involves using a neural network to approximate the space-dependent Polynomial Chaos Expansion (PCE) coefficients $y_\alpha(x)$.
  - The total output $Y(\xi, x)$ is a sum of these coefficients multiplied by PCE basis functions $\Psi_\alpha(\xi)$.

Core Challenges in Training PINNs

- **Optimization Difficulties:** While the soft constraint formulation (using penalty methods) is easy to implement, it makes the loss landscape complex and **difficult to optimize**.
- **Soft Constraint Issues:** Using the soft-regularization method introduces subtle issues, and PINNs **do not always work well**, even for simple problems.
- **Need for Full Batch Optimization:** Unlike classical ML tasks, PINNs often **require LBFGS with full batch size**, making training very slow and hard to optimize.
- These challenges necessitate a rethinking of the NN architecture, training strategy, and input data design for scientific problems.

Failure Mode I: Sensitivity to Loss Weighting

- PINN performance is highly sensitive to the chosen **multiplier (λ_F)** for the PDE loss term.
- Adding the PDE constraint makes the loss landscape difficult to optimize.
- If the multiplier λ_F is reduced, the optimization becomes easier, but the PINN's solution error can be severe, potentially reaching **100% error**.
- Analyzing the loss landscape shows that the complexity increases dramatically when the physics loss is included.

Changing the multiplier (λ_F) for the PDE loss drastically alters the loss landscape and optimization difficulty.

Failure Mode II: Collocation Point Sampling

- An important consideration is **where** to enforce the physical constraints (collocation points).
- Sampling collocation points **uniformly is suboptimal** and can result in large errors.
- For example, the testing error for a Poisson equation solved with uniform sampling can be as high as **60%**.
- Sampling bias in the physics loss can be detrimental to the solution accuracy.

Demonstrated PINN Failures

- PINNs can fail to converge to a reasonable solution for equations with non-trivial coefficients.
- PINNs have been observed to fail to learn the relevant physics for three families of PDEs:
  1. **Advection Equation** (aka wave equation).
  2. **Reaction Equation**.
  3. **Reaction-Diffusion Equation**.
- In these failure cases, analysis suggests that while the NN has enough capacity, the optimization problem created by the soft-regularization is too difficult to solve.

Training Strategy I: Curriculum Learning

- The goal of **Curriculum Learning (Causal Training)** is to introduce complexity iteratively throughout the learning process.
- The network starts by learning simple physical constraints before being penalized for the exact, complex PDE.
- **Example (Advection Equation):** Training starts with very small velocities, and the velocity is slowly increased to the target value.
- This approach has been shown to work quite well for reaction problems, alleviating the difficulty seen in regular training.

Training Strategy II: Seq2Seq Learning

- Standard PINN formulation attempts to predict the entire space-time simultaneously, which is a very difficult function to approximate.
- **Sequence-to-Sequence (Seq2Seq) Learning** reframes the problem.
- The PINN learns to predict the solution in a **finite time horizon**.
- It then iteratively predicts the next time steps.
- The Seq2Seq approach can achieve **significantly lower error** for problems like Reaction-Diffusion compared to regular training.

Training Strategy III: Adaptive Sampling

- This technique addresses the suboptimality of uniform collocation point sampling.
- **Adaptive Sampling** involves resampling the points during training based on a **proxy function** (e.g., regions where the residual is high).
- This is typically combined with a regularization step that includes uniform sampling.
- Adaptive sampling has been shown to improve the testing error significantly, for instance, reducing the error for the Poisson equation from 60% to 10%.

Implementation and Architectural Solutions

- **Maximizing Hard Constraints:** Utilize hard constraints wherever possible (e.g., for displacement BCs) to limit the order of derivatives required, thus simplifying the optimization.
- **Mixed-PINN Formulation:** Incorporate different fields (e.g., displacement u and stress σ) as network outputs, ensuring both material laws and momentum balance are satisfied.
- **Separable-PINN (SPINN):** Uses a different network for each dimension, resulting in a low-rank tensor approximation. SPINN can outperform standard PINNs in both **speed and accuracy**.
- Other techniques include Non-dimensionalization, Fourier features, and combining **Adam + LBFGS** optimization.

Open Problems and Future Directions

- **Optimization Reliance:** The heavy reliance on **LBFGS with full batch size** makes training PINNs slow, presenting a major open problem. Overcoming this barrier is crucial for scalability.
- **Architecture Specialization:** Classical NN architectures may not be optimal for PINNs.
- **PDE-Specific Architectures:** Research is needed to investigate how architectures should change depending on the underlying dynamics, such as:
  - Elliptical PDEs.
  - Hyperbolic PDEs.
  - Parabolic PDEs.
- PINNs remain a powerful and maturing tool, but research continues to focus on improving the robustness of the soft-regularization method.