PINNfluence: Interpreting PINNs through Influence Functions

Aleksander Krasowski [*1] **Jonas Naujoks** [*1] **Moritz Weckbecker** [1] **Galip Ü. Yolcu** [1] **Thomas Wiegand** [1 2 3] **Sebastian Lapuschkin** [1 4] **Wojciech Samek** [1 2 3] **René P. Klausen** [1]

Abstract

Physics-informed neural networks (PINNs) have emerged as a powerful deep learning approach for solving partial differential equations (PDEs) in the physical sciences, yet their behavior remains largely opaque and is typically understood through failure mode analyses rather than explicit interpretability. To address this issue, we introduce PINNFLUENCE, a training data attribution framework for interpreting PINNs based on influence functions. By extending influence functions to composite physics-informed training objectives, we enable fine-grained attribution between predictions, loss components, and training data points. Through benchmark experiments across various PDEs, we demonstrate that influence patterns provide granular diagnostics that distinguish structural characteristics across well-trained and poorly-trained PINNs. PINNFLUENCE thus opens a new avenue for understanding and improving the reliability of PINNs through the lens of their data.

1. Introduction

Partial differential equations (PDEs) are the central modeling language in the sciences, describing how quantities evolve across space and time under governing laws. Solving these equations accurately and efficiently is essential for scientific prediction and control, yet often computationally demanding. In recent years, machine learning (ML) approaches have proven themselves exceptionally adept at approximating solutions to such problems, bundling fast inference and strong empirical performance once trained. Physics-informed neural networks (PINNs) (Raissi et al., 2019) integrate physical knowledge by embedding PDEs into neural network training objectives (Karniadakis et al., 2021; Kim et al., 2021; Cuomo et al., 2022), enabling applications across fluid mechanics, electromagnetics, disease modeling, and optics, inter alia (Cai et al., 2021; Beltrán-Pulido et al., 2022; Berkhahn & Ehrhardt, 2022; Medvedev et al., 2023). Despite their flexibility, understanding and improving PINN behavior when training fails remains challenging (Wang et al., 2021; Krishnapriyan et al., 2021; Wang et al., 2022; Rathore et al., 2024; Liu et al., 2025).

The existing understanding of PINNs is largely based on analyses of training dynamics, failure modes, and ill-conditioned loss landscapes. Prior work attempts to alleviate these issues through improvements in optimization, architecture, objective function design, and training data selection (Jagtap et al., 2020; McClenny & Braga-Neto, 2023; Wang et al., 2023; Wu et al., 2025; Jeong et al., 2025). Further, rigorous error guarantees and uncertainty quantification for PINNs are active fields of research and often difficult to obtain in practice (Eiras et al., 2024; Hillebrecht & Unger, 2025). Both of these strands of research aim to make PINNs reliable and performant, qualities that are crucial for wide-scale adoption. In parallel, a substantial body of work on explainable artificial intelligence (XAI) has developed tools to interpret and diagnose complex machine learning models, improving trust, scientific insights, and model validation (Ross et al., 2017; Rieger et al., 2020) in varied domains such as medical imaging (Hense et al., 2024), computer vision (Bach et al., 2015; Bender et al., 2023) and natural language processing (Bricken et al., 2023).

While interpretability[1] has become a central theme in deep learning (Samek et al., 2021; Rudin et al., 2021) and scientific machine learning in particular (Wetzel et al., 2025), PINNs still lack dedicated explainability methods that support transparency, systematic diagnosis and validation – a precondition for their widespread application. In particular, although existing analyses of PINN failure modes provide

[*]Equal contribution [1]Department of Artificial Intelligence, Fraunhofer HHI, Berlin, Germany [2]Department of Electrical Engineering, Technische Universität Berlin, Berlin, Germany [3]BIFOLD – Berlin Institute for the Foundations of Learning and Data, Berlin, Germany [4]Centre of eXplainable Artificial Intelligence, Technological University Dublin, Dublin, Ireland. Correspondence to: Aleksander Krasowski <aleksander.krasowski@hhi.fraunhofer.de>, Jonas Naujoks <jonas.naujoks@hhi.fraunhofer.de>, René P. Klausen <rene.pascal.klausen@hhi.fraunhofer.de>.

Proceedings of the AI4Physics Workshop at the 43rd International Conference on Machine Learning (AI4Physics@ICML 2026), Seoul, South Korea. 2026. Copyright 2026 by the author(s).

[1]We use this term interchangeably with explainability.

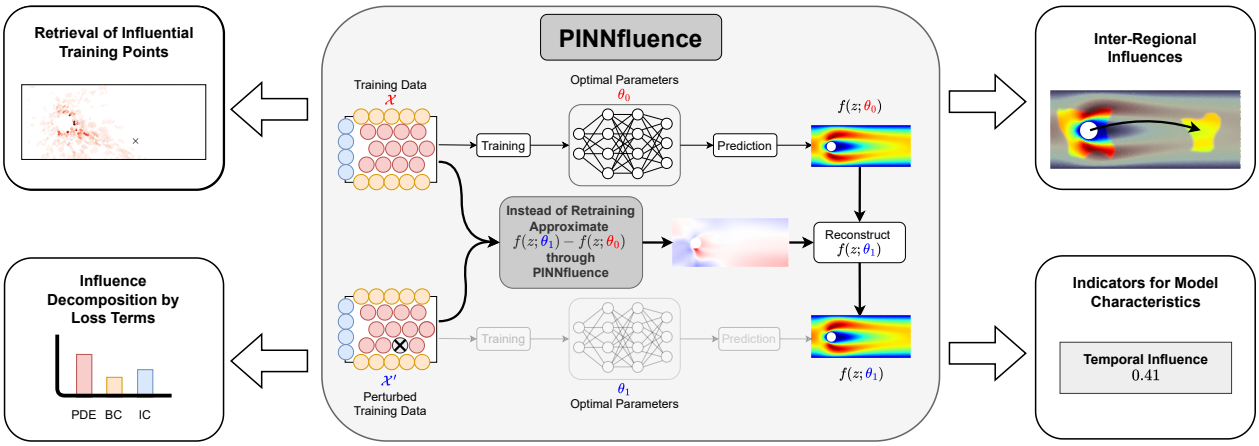


Figure 1. PINNFLUENCE approximates the effect of perturbing training data by estimating the difference $f(z, \theta_1) - f(z, \theta_0)$ *without retraining*. Here, θ_0 denotes the parameters obtained from the original training set $\mathcal{X}$, while θ_1 corresponds to a perturbed training set $\mathcal{X}'$ that differs by a single training point. This enables attribution of model predictions at test points z to individual training samples and loss components, supporting **(i)** retrieval of influential training points for the given prediction, **(ii)** inspection of how one region influences another, **(iii)** decomposition by individual loss terms, and **(iv)** construction of diagnostic indicators that characterize model behavior.

valuable insights at the level of training dynamics and model classes, they fall short of providing diagnostic or attribution-based explanations that reveal how individual training points and physics constraints shape model predictions. This leaves a gap for training-data-centric interpretability methods for PINNs, which we address in this work. This gap is of particular importance, as PINNs face a verification problem distinct from classical supervised learning: a low training loss does not imply a correct solution to the underlying PDE, since PINNs can converge to trivial solutions with low residuals yet high errors (Daw et al., 2023; Rohrhofer et al., 2023). Trained models can thus exhibit pathological structure invisible to residual inspection alone. Such post-hoc methods have the potential to improve transparency and robustness by enabling fine-grained diagnosis and control of PINN behavior, thereby complementing existing approaches.

Specifically, we adapt influence functions, a training data attribution (TDA) method, to PINNs. TDA quantifies the influence of individual training points on model behavior and has proven effective for debugging models, identifying memorization and harmful data, and understanding model behavior through the lens of the training set in domains such as computer vision and large language models (Koh & Liang, 2017; Pruthi et al., 2020; Braun et al., 2022; Hammoudeh & Lowd, 2024). An overview of our proposed PINNFLUENCE framework and the types of influence-based analyses it provides is shown in Figure 1.

Our contributions can be summarized as follows:

- We generalize influence functions to PINNs, resulting in PINNFLUENCE, an interpretability framework that attributes predictions to individual training points and decomposes their influence by loss components.
- We provide a toolbox of diagnostic indicators, including temporal, directional, and region-to-region measures, that enable systematic analysis of trained PINNs.
- Across five time-dependent PDEs, we identify consistent structural differences between well-trained and poorly-trained models. Well-trained models show diminishing influence of the initial condition (IC), while poorly-trained models exhibit persistent IC dependence over time.

2. Theoretical Background

2.1. Physics-Informed Neural Networks (PINNs)

PDEs are ubiquitous modeling tools in the computational sciences: For example, they describe the motion of galaxies or the spread of infectious diseases. A PDE is typically posed together with initial and boundary conditions, such as the starting position of galaxies or the initial amount of infected specimens. A PDE with these constraints is called an initial-boundary value problem (IBVP). PINNs (Raissi et al., 2019) provide a widely-used ML-based framework for approximating solutions of IBVPs. Let $\Omega \subset \mathbb{R}^n$ be a connected, bounded, open set with piecewise smooth boundary $\partial\Omega$, and define its closure by $\overline{\Omega} := \Omega \cup \partial\Omega$. We study PDE constraints in the domain Ω together with K boundary and/or initial constraints posed on subsets $\Gamma_k \subseteq \partial\Omega$:

$$\mathcal{N}[u](x) = 0, \quad x \in \Omega \tag{1}$$

$$\mathcal{B}_k[u](x) = 0, \quad x \in \Gamma_k \text{ for } k = 1, \ldots, K \quad , \tag{2}$$

where $\mathcal{N}$ and $\mathcal{B}_k$ are differential operators acting on the solution $u : \overline{\Omega} \to \mathbb{R}^d$ of (1) and (2) with K initial and/or

boundary conditions. Here, x denotes the spatio-temporal coordinates, which may be purely spatial, purely temporal, both, or more generally any physical quantity on which the PDE depends. The above formulation (2) encompasses both Dirichlet and Neumann boundary conditions, among others. The aim of the PINN approach is to approximate a solution $u(x)$ of (1) and (2) with a neural network $\phi(x;\theta)$ by optimizing its trainable parameters $\theta \in \Theta \subseteq \mathbb{R}^p$. Training proceeds by minimizing an empirical risk defined through the composite loss function

$$\mathcal{L}(\theta) := \frac{\lambda_{\text{pde}}}{|\mathcal{X}_{\text{pde}}|} \sum_{x \in \mathcal{X}_{\text{pde}}} L_{\text{pde}}(x;\theta) + \sum_{k=1}^{K} \frac{\lambda_{\text{bc},k}}{|\mathcal{X}_{\text{bc,k}}|} \sum_{x \in \mathcal{X}_{\text{bc,k}}} L_{\text{bc},k}(x;\theta) \,, \tag{3}$$

with per-sample losses $L_{\text{pde}}(x;\theta) := \|\mathcal{N}[\phi(x;\theta)]\|_2^2$, $L_{\text{bc,k}}(x;\theta) := \|\mathcal{B}_k[\phi(x;\theta)]\|_2^2$ and weights $\lambda_{\text{pde}}, \lambda_{bc,k} > 0$. We consider training points $\mathcal{X}_{\text{pde}} \subset \Omega$ and $\mathcal{X}_{\text{bc,k}} \subset \Gamma_k$, which construct $\mathcal{X}_{\text{train}} = \mathcal{X}_{\text{pde}} \cup \bigcup_k \mathcal{X}_{\text{bc,k}}$. Data-driven regression terms for incorporating measurements can be included but are omitted here for the sake of clarity.

2.2. Influence Functions (IFs)

Influence functions (IFs) (Hampel, 1974; Cook & Weisberg, 1982; Koh & Liang, 2017) address the question of how a model's behavior would change if specific training points were removed from or added to the training set. Through a linear approximation of leave-one-out retraining, IFs enable the analysis of a model in terms of the contributions of individual training points.

Compared to the original formulation of influence functions (Koh & Liang, 2017), we generalize the framework in several important ways: First, we extend the analysis from strict minima to non-degenerate stationary points, which is critical in neural network training, where optimization typically converges to saddle points rather than global optima (Dauphin et al., 2014). This relaxation is especially relevant in PINNs, whose optimization landscapes are often ill-conditioned (Rathore et al., 2024; Ryck et al., 2024). Second, we refine the required assumptions on the loss landscape, noting that strong convexity, while necessary, is not sufficient. Finally, by allowing f to be an arbitrary differentiable function, we generalize IFs from loss-to-loss attributions to influence *from any quantity to any component of a composite objective*, which is crucial for analyzing PINN objectives, where quantities of interest extend beyond the loss to physical observables and individual constraint terms.

Definition 2.1 (Influence Function)**.** Let $L : \overline{\Omega} \times \Theta \to \mathbb{R}_{\geq 0}$ be a twice differentiable, single-sample loss function that induces an empirical risk $\mathcal{L}(\theta) = \frac{1}{N}\sum_{x \in \mathcal{X}} L(x;\theta)$ over a training set $\mathcal{X} \subset \overline{\Omega}$ of size N, and $f : A \times \Theta \to \mathbb{R}^m$ be an arbitrary differentiable function. For two points $x \in \overline{\Omega}$, $z \in A$, we call

$$\text{Inf}_{\theta_0}^{L \to f}(x,z) := -\nabla_\theta f(z;\theta_0)^\top \, \mathcal{H}_{\theta_0}^{-1} \, \nabla_\theta L(x;\theta_0) \tag{4}$$

the *influence function* from L at the point x towards f at the point z with respect to θ_0, where $\mathcal{H}_{\theta_0} = \nabla_\theta^2 \mathcal{L}(\theta_0)$ denotes the Hessian at θ_0, which is assumed to be invertible.

Theorem 2.2 (Influence Functions Approximate Retraining)**.** *Assume the setting of Definition 2.1 with L and f being analytic. Further, assume that θ_0 is a non-degenerate stationary point of $\mathcal{L}$ with respect to a training set $\mathcal{X}$ (i.e., $\nabla_\theta \mathcal{L}(\theta_0) = 0$, $\det \mathcal{H}_{\theta_0} \neq 0$). Let θ_1 denote the stationary point of $\mathcal{L}(\theta)$ with respect to $\mathcal{X}'$, where we added or removed a training point $x^\pm$. For sufficiently large N, the resulting effect on the function f is given by*

$$f(z;\theta_1) - f(z;\theta_0) = \pm \, \text{Inf}_{\theta_0}^{L \to f}(x^\pm, z) \frac{1}{N} + \mathcal{O}(1/N^2) \quad .$$

We provide the proof in Appendix A.1. Thus, the IF computes the effect of infinitesimally upweighting the contribution of a data point in the training objective. Downweighting a training point yields a sign-flipped influence expression.

In practice, IFs are most often evaluated for the influence on the test loss. This corresponds to the special case $f = L$, which reads $\text{Inf}_{\theta_0}^{L \to L}(x,z)$. It is important to note that IFs constitute only a first-order approximation. In practical application, discrepancies may arise when the number of training points N is not sufficiently large or when θ_0 is only close to being a stationary point. Further, note that the accuracy of the approximation depends on the local geometry of the loss landscape.

Corollary 2.3 (Properties of Influence Functions)**.** *Assume the setting of Definition 2.1 and Theorem 2.2.*

(Linearity) *The influence function is bilinear in the loss L and the quantity of interest f. For scalars $\alpha, \beta \in \mathbb{R}$, and suppressing the dependence on (x,z) for brevity, we have*

$$\begin{aligned} \text{Inf}_{\theta_0}^{\alpha L_1 + \beta L_2 \to f} &= \alpha \, \text{Inf}_{\theta_0}^{L_1 \to f} + \beta \, \text{Inf}_{\theta_0}^{L_2 \to f}, \\ \text{Inf}_{\theta_0}^{L \to \alpha f_1 + \beta f_2} &= \alpha \, \text{Inf}_{\theta_0}^{L \to f_1} + \beta \, \text{Inf}_{\theta_0}^{L \to f_2} \quad . \end{aligned}$$

(Additivity) *In addition, we can remove or add several points. Let $\mathcal{X}^+$ be a set of points added to the training set and $\mathcal{X}^- \subset \mathcal{X}$ a set of points removed. Then, for sufficiently large N, the resulting change in f satisfies*

$$N\big(f(z;\theta_1) - f(z;\theta_0)\big) \approx \sum_{x \in \mathcal{X}^+} \text{Inf}_{\theta_0}^{L \to f}(x,z) - \sum_{x \in \mathcal{X}^-} \text{Inf}_{\theta_0}^{L \to f}(x,z) \quad .$$

This corollary enables the analysis of influence between arbitrary regions of the domain and individual components of the PINN objective, forming the basis of the PINNFLUENCE framework developed in Section 4. Notably, the linearity in the loss argument implies that the loss weights directly scale corresponding influences, giving our approach a principled way to study loss-weighting effects.

Beyond point-wise effects, it is often practical to consider aggregated measures such as the total influence $\sum_{z\in R} \mathrm{Inf}_{\theta_0}^{L\to f}(x,z)$ or its absolute variant $\sum_{z\in R} \left|\mathrm{Inf}_{\theta_0}^{L\to f}(x,z)\right|$ over regions $R \subseteq \overline{\Omega}$. Together with Corollary 2.3, this permits the study of influences from points to regions, regions to points, and region-to-region interactions, while simultaneously allowing for the analysis of individual loss components. For a discussion on the practical considerations of the computation of influences, such as circumventing the costly explicit construction of the Hessian, we refer to Appendix A.2.

3. Related Work

Interpretability and explainability have emerged as central themes in the machine learning community, driven by the increasing deployment of complex models in high-stakes applications. Naturally, this trend has also extended to scientific machine learning (Schnake et al., 2022; Kitouni et al., 2024; Wetzel et al., 2025), where model transparency and verification are closely tied to scientific validity. Still, dedicated interpretability methods for PINNs remain scarce.

Analyses and Improvements of PINN Training In the absence of dedicated interpretability frameworks, much of the existing understanding of PINNs has been developed through the systematic analysis of their training behavior. There is a significant line of work investigating the mechanisms underlying observed pathologies. This includes gradient flow pathologies and ill-conditioned loss landscapes (Wang et al., 2022; Ryck et al., 2024), information propagation failure (Krishnapriyan et al., 2021; Daw et al., 2023), failure of initial condition learning (Steger et al., 2022), and different phases of learning in PINNs (Anagnostopoulos et al., 2024). Substantial efforts have focused on mitigating these failures, including loss reweighting (Wang et al., 2023; 2024), adaptive sampling (Lu et al., 2021; Liu et al., 2022; Wu et al., 2023; Lau et al., 2024), refined network architectures (Jagtap et al., 2020; McClenny & Braga-Neto, 2023), and curvature-aware optimizers tailored to PINNs (Rathore et al., 2024; Vyas et al., 2025), including energy natural gradients (Müller & Zeinhofer, 2023), second-order approaches with gradient-alignment analysis (Wang et al., 2025), and self-scaled quasi-Newton schemes (Kiyani et al., 2025). Furthermore, the neural tangent kernel (NTK) framework is a standard investigatory setting (Jacot et al., 2018; Wang et al., 2022), although recent work has shown that this linearized regime breaks down for nonlinear PDEs (Bonfanti et al., 2024). These studies provide valuable diagnostic insights into training dynamics and optimization pathologies but explicit post-hoc interpretability and data attribution mechanisms are missing. We provide a detailed methodological comparison of PINNFLUENCE with related training-time approaches in Appendix A.3.

Interpretability in PINNs and Learning of the Solutions of PDEs Another line of research aims to increase interpretability in scientific machine learning for PDEs more explicitly. Here, some approaches include constraining the architecture to increase interpretability (Ramabathiran & Ramachandran, 2021) or developing novel architectures to be interpretable by design, such as Kolmogorov-Arnold Networks (Liu et al., 2024) and GINN-KANs (Ranasinghe et al., 2024), but they do not address post-hoc analysis of already trained PINNs.

Although not aimed at PINNs, another emerging area of interpretability research is concerned with the post-hoc analysis of the learned representation of data-driven PDE solvers. Recently, methods from the field of mechanistic interpretability (Bricken et al., 2023) such as sparse autoencoders (Cunningham et al., 2023) and linear probing (Alain & Bengio, 2018) have been used to investigate neural operators trained to solve PDEs and predict weather dynamics (MacMillan & Ouellette, 2025; Fear et al., 2025), discovering concept activation vectors and latent features that correspond to weather and flow phenomena such as vorticity and diffusivity. Although promising, these methods do not extend to PINNs.

Despite these advances, explicit attribution and diagnostic tools for PINNs remain largely unexplored. In particular, to the best of our knowledge, no training-data-centric framework for PINNs exists that enables attributing model behavior to individual collocation points or physics constraints, which motivates the approach proposed in this work.

4. Methodology of PINNFLUENCE

In this section, we introduce our toolbox of PINNFLUENCE-based analysis primitives and indicators for diagnosing PINN behavior. Conceptually, PINNFLUENCE decomposes influence along two axes: *what* contributes and *where* contributions arise. Building on the theoretical results of Section 2, we show how PINNFLUENCE can be utilized at different levels of granularity. Starting from point-wise influences, we construct decompositions of loss components and regions, and define normalized indicators that summarize influence effects for systematic comparison across models.

These tools are designed to test concrete diagnostic hy-

Table 1. Overview of PINNFLUENCE-based diagnostics used in this work and the insights they provide.

Diagnostic Question	PINNFLUENCE tool	Insight
Which training points shape a given prediction?	Point-wise influence maps	Localizes sensitivity to individual collocation points.
How do individual physics constraints shape model behavior?	Loss-component influence decomposition (6)	Quantifies the relative contribution of PDE, boundary, and IC. Exposes IC-dominated failure patterns.
Do learned solutions reflect temporally oriented influence?	Temporal influence indicator (9)	Shows that well-trained PINNs exhibit near-symmetric temporal influence, while asymmetry signals failure.
How does influence distribute across domain subsets?	Regional influence indicators	Probes problem-specific structure: downstream propagation, spatial symmetry, shock locality, see Appendix C.

potheses about the behavior of an optimized model, such as *"Which constraints dominate a prediction?"* and *"How does influence align with temporal or directional structure of the underlying PDE?"* In Table 1, we give a summary of the concrete questions together with the insights they yield.

Point-wise Influences As fundamental building blocks of our analysis, point-wise influences (4) quantify how an individual training point affects a function f, such as the prediction, the loss or a physical observable, evaluated at a specific test location. Influence values can be positive and negative, indicating whether a training point increases or decreases f at the test location. Empirically, we observe that the signs of influence effects can be noisy. Since we are primarily interested in the strength of the effect, we mainly consider absolute influence values $|\operatorname{Inf}_{\theta_0}^{L\to f}(x,z)|$, which provide a more robust estimate of sensitivity. Further, points with large absolute influences can be interpreted as regions of high complexity for the model (Zhang & Zhang, 2022). Thus, point-wise influences are particularly suited for attributing model behavior at individual points in the domain Ω to specific training points. This enables diagnosis beyond localization: one can identify not only where a model fails, but also which training points dominate predictions across different regions, yielding a structural characterization invisible to residual analysis.

Point-to-Region Influences Building on the linearity and additivity properties of influence functions (see Corollary 2.3), we aggregate point-wise influences to study region-level effects. Throughout the section, we denote $R_{\text{tr}}, R_{\text{te}} \subset \Omega$ as subsets of training and test samples, respectively. Fixing a training point $x \in \mathcal{X}$ and summing its influence over test locations yields point-to-region influence maps, revealing where and how strongly this datum affects a quantity of interest across the domain.

$$\operatorname{Inf}_{\theta_0}^{L\to f}(x, R_{\text{te}}) = \sum_{z\in R_{\text{te}}} \operatorname{Inf}_{\theta_0}^{L\to f}(x,z) \quad . \tag{5}$$

Notably, this provides insights into how the quantity of interest f would change at a macroscopic scale if the training point were perturbed or removed. Analogously, one can fix a test point and aggregate the estimated effects of a set of training points R_{tr}, which reveals how each training point contributes to model behavior at that location.

Decomposition by Individual Loss Components The linearity of the influence in the loss argument (Corollary 2.3) allows us to decompose it with respect to every individual loss term. This allows for fine-grained analysis of the effect of the PDE residual, or any of the boundary and initial conditions on a given test point's prediction. Notably, residual inspection alone does not offer the same level of granularity, as it does not permit the attribution of predictions or quantities of interest to specific loss components or constraints. We obtain the *fractional contribution* of each loss term by taking its ratio with respect to the sum of all absolute contributions, which yields

$$r_{L_i}(x,z) = \frac{\left|\operatorname{Inf}_{\theta_0}^{L_i\to f}(x,z)\right|}{\sum_j \left|\operatorname{Inf}_{\theta_0}^{L_j\to f}(x,z)\right|} \in [0,1] \quad , \tag{6}$$

satisfying $\sum_i r_{L_i}(x,z) = 1$. Since individual influences may have opposing signs, we additionally report a cancellation score

$$\begin{aligned}\kappa(x,z) &= 1 - \frac{\left|\sum_j \operatorname{Inf}_{\theta_0}^{L_j\to f}(x,z)\right|}{\sum_j \left|\operatorname{Inf}_{\theta_0}^{L_j\to f}(x,z)\right|} \\ &= 1 - \frac{\left|\operatorname{Inf}_{\theta_0}^{L\to f}(x,z)\right|}{\sum_j \left|\operatorname{Inf}_{\theta_0}^{L_j\to f}(x,z)\right|} \in [0,1]\end{aligned} \tag{7}$$

that quantifies the degree to which loss terms reinforce ($\kappa \approx 0$) or counteract ($\kappa \approx 1$) each other. When cancellation is low, the fractions reliably indicate each term's contribution to the total influence. Conversely, high cancellation corresponds to competing constraints whose individual effects largely negate each other at the prediction level.

Region-based Indicators In analogous fashion to the loss components decomposition, we can decompose influence with respect to training and test regions. We first define relative regional influences, which capture the estimated effect on model behavior of certain regions relative to the whole domain. The *relative absolute regional influence* of

R_{tr} on R_{te} is given by

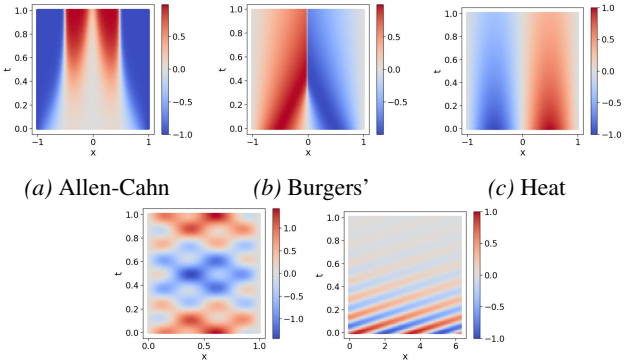
$$\rho_{\theta_0}^{L\to f}(R_{\text{tr}}, R_{\text{te}}) = \frac{1}{|R_{\text{te}}|} \sum_{z\in R_{\text{te}}} \frac{\sum_{x\in R_{\text{tr}}} \left|\text{Inf}_{\theta_0}^{L\to f}(x,z)\right|}{\sum_{x\in \mathcal{X}_{\text{train}}} \left|\text{Inf}_{\theta_0}^{L\to f}(x,z)\right|}, \tag{8}$$

with $\rho_{\theta_0}^{L\to f} \in [0,1]$. It allows quantifying how strongly a given training region contributes to model behavior over a specific test region. For region-level analysis, point-wise loss decompositions (6) can also be aggregated over subsets of training and test points $\bar{r}_{L_i}(R_{\text{tr}}, R_{\text{te}}) = \frac{1}{|R_{\text{te}}|}\sum_{z\in R_{\text{te}}}\sum_{x\in R_{\text{tr}}} r_{L_i}(x,z)$. An important variation of the aforementioned regional indicator arises in time-dependent problems. For such settings, we define a *temporal influence indicator* that quantifies how strongly the model behavior at test points is influenced by the training points in the past

$$\eta_{\theta_0}^{L\to f}(R_{\text{tr}}, R_{\text{te}}) = 1 - \frac{1}{|R_{\text{te}}|} \sum_{z\in R_{\text{te}}} \frac{\sum_{x\in R_{\text{tr}}: x_t \le z_t} \left|\text{Inf}_{\theta_0}^{L\to f}(x,z)\right|}{\sum_{x\in \mathcal{X}_{\text{train}}} \left|\text{Inf}_{\theta_0}^{L\to f}(x,z)\right|}. \tag{9}$$

Again, we have $\eta_{\theta_0}^{L\to f} \in [0,1]$. The subtraction from 1 ensures that smaller values of η correspond to stronger past-directed influence; larger values imply the opposite. Assuming that the distribution of training points is uniform, values of ≈ 0.5 are expected if influence is temporally symmetric. Notably, this notion cannot be probed through residual analysis alone. These indicators can be specialized to various other diagnostic hypotheses, e.g., boundary dominance, spatial locality, or symmetry, through appropriate choices of the loss component L_i, the function f, and the sets of training and test regions. In analogy to (9), directional indicators can be constructed for time-independent problems by defining ordering relations over spatial directions. For this and further instances, we refer to Appendix C.

Indicators summarize influence pattern analysis and provide first-order, Hessian-based sensitivity diagnostics under the trained parameters of a PINN. Notably, PINNFLUENCE does not provide formal error guarantees or interventional causal claims. It characterizes the converged model's sensitivity to how predictions would respond to perturbations in the training data, rather than the training dynamics that produced the structure. In this sense, PINNFLUENCE supports the comparative and hypothesis-driven diagnosis of trained PINNs, as demonstrated in the following section.



(a) Allen-Cahn *(b)* Burgers' *(c)* Heat

(d) Wave *(e)* Drift-Diffusion

Figure 2. Investigated problems.

5. Experiments

We demonstrate how PINNFLUENCE helps to understand the behavior of PINNs across five time-dependent PDEs spanning diverse physics: the heat, Allen-Cahn, Burgers', wave, and drift-diffusion equations, shown in Figure 2. Additionally, we provide an analysis of two steady-state problems, namely the Poisson and Navier-Stokes equations in Appendix C.1. For every problem, we include two different settings: a *well-trained* model that accurately solves the IBVP, and a *poorly-trained* model that fails to accurately solve the IBVP due to insufficient data or suboptimal optimization. Comparing influence patterns between these two configurations allows us to identify differences in how the two classes of models depend on their training data at convergence. This enables us to provide diagnostic indicators that distinguish well-performing from pathological model states in a controlled way. To assess consistency, we include 10 random seeds for each configuration. The complete problem specifications and training setup are provided in Appendix B.

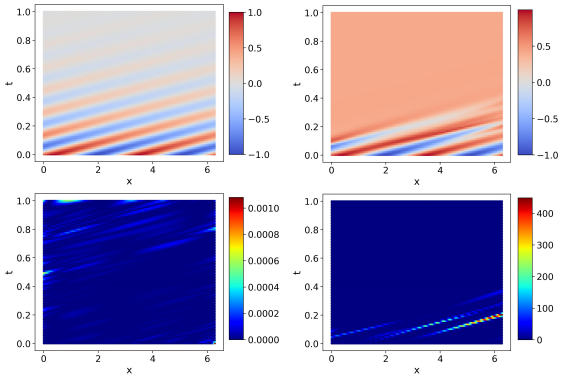
Across experiments, we focus on three recurring research questions that exemplify how PINNFLUENCE yields insights into PINN behavior:

(i) Can PINNFLUENCE give insights into differences between well-trained and poorly-trained models, beyond localizing where errors occur?

(ii) Can PINNFLUENCE attribute model sensitivity to specific loss components L_i?

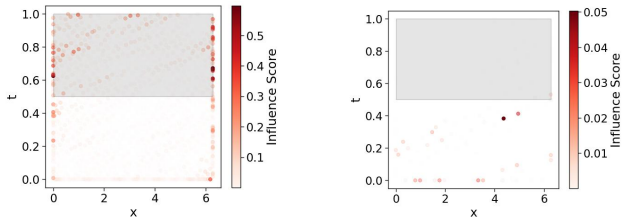
(iii) Can we reveal structural properties of trained PINNs using PINNFLUENCE, such as temporal influence patterns that are not evident from residuals alone?

Point-wise Attribution To demonstrate the capabilities of PINNFLUENCE, we focus our analysis on the drift-diffusion equation, a convection problem where PINNs are known



(a) Well-trained *(b)* Poorly-trained

Figure 3. Prediction and losses for the drift-diffusion problem for a representative seed.



(a) Well-trained *(b)* Poorly-trained

Figure 4. Point-to-region influence heatmaps showcasing the distribution of influences across the training set affecting the network *prediction* at the top half ($t > 0.5$, marked in gray) of the domain.

to suffer from *propagation failure*, i.e., collapsing to near-constant solutions forward in time (Krishnapriyan et al., 2021). This exact failure mode is present in our poorly-trained configuration and is straightforward to localize. Predictions show the collapse to a constant output emerging at around $t = 0.2$ and the losses indicate a large PDE violation at the transition region, as shown in Figure 3.

PINNFLUENCE aids us in understanding how the prediction was shaped, through the lens of the PINN's training data. Figure 4 shows point-to-region influence attribution maps for model predictions at times $t > 0.5$, identifying training points to which these predictions are most sensitive. For the well-trained model, the prediction in that region is primarily shaped by information from the data points on the periodic boundaries ($x = 0$ and $x = 2\pi$). For the poorly-trained model, the influence patterns differ significantly: alongside the region of failure, the prediction is highly sensitive to training points along the initial condition ($t = 0$), while influence from the boundary appears absent. Thus, PINNFLUENCE reveals a key distinction invisible to residuals: which training data shapes predictions differs drastically between configurations.

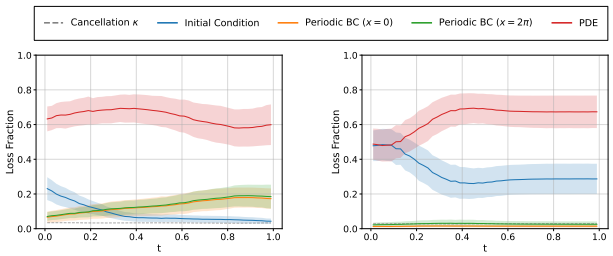
Decomposition by Individual Loss Components Pointwise attribution reveals the training points to which the models are most sensitive. Using the linearity of influence functions in the loss argument, PINNFLUENCE also allows us to directly decompose and quantify influence contributions into individual loss terms (6). This enables studying the contributions of loss terms on a more global scale.

We aggregate the fractional influence contribution $\bar{r}_{L_i}$ of each loss component (the PDE residual, the initial and the two periodic boundary conditions), averaged over time bins in the test data, as shown in Figure 5.

For the well-trained models, the IC contribution starts at around $\bar{r}_{L_{\text{ic}}} \approx 0.25$ near $t = 0$ and decreases asymptotically towards zero with increasing time. Simultaneously, the influence contributions of both periodic boundary conditions increase evenly over time, indicating that the model's predictions reflect the problem's periodic structure. The PDE term dominates throughout.

For the poorly-trained model, the IC displays a much greater share of influence. While it also decreases with time, the term's relative influence $\bar{r}_{L_{\text{ic}}}$ remains elevated in comparison to the well-trained model. After $t \approx 0.4$, the fractions plateau, coinciding with pathological, near-constant outputs and stagnating gradients. Notably, the boundary conditions exhibit negligible contributions, i.e., the model's predictions are effectively insensitive to its boundary training data. The model has converged to a state where perturbations to boundary points have no effect, a clear structural misalignment for a problem driven by boundary propagation.

We further show that the elevated relative IC importance $\bar{r}_{L_{\text{ic}}}$ can be identified across all investigated problems, as depicted in Figure 6. Throughout all well-trained models, the IC influence share at $t = 0$ is lower and decays more rapidly compared to their poorly-trained counterparts. For the latter, the IC generally displays greater influence over time. Note that while the rate of decay varies by problem, the general diagnostic insight holds: well-trained models show diminishing IC sensitivity with time, poorly-trained



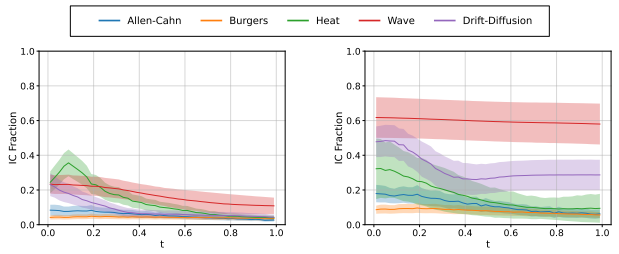
(a) Well-trained *(b)* Poorly-trained

Figure 5. Fractions of influences $\bar{r}_{L_i}(\mathcal{X}, R_{\text{te}})$ for 50 time bins, averaged over seeds. Shaded regions indicate the standard deviation across seeds.

models do not. This is striking, as not all suboptimally trained models perform equally *poorly*. We refer the interested reader to Appendix B.3, where all respective model predictions and targets are visualized. PINNFLUENCE can thus attribute model sensitivity to individual loss components L_i, thereby reliably distinguishing well-trained from poorly-trained models. We further present a complementary analysis in Appendix C.3, varying the number of training points on the time-dependent PDEs. There, we show that PINNFLUENCE indicators respond to the gradual transition between failure and success in PDE-specific ways, revealing each problem's data requirements.

Indicator-based Analysis The loss decomposition reveals that poorly-trained models are overly sensitive to changes in the initial condition. We now examine this pattern from a complementary perspective: the temporal distribution of influences across the domain. Time-dependent PDEs possess an inherent causal structure in which initial conditions determine future states. One might therefore hypothesize that a well-trained PINN that yields an accurate approximation of the solution would reflect this causality. Under this proposition, predictions at time t would be predominantly influenced by training points at times $\leq t$. We probe this hypothesis using the temporal influence indicator (9) and compare it against the baseline temporal average $\bar{t} = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} x_t$ of the training set. Note that sampling on the initial boundary introduces a distribution skewed toward earlier times.

Table 2 reports temporal influence indicators across all time-dependent problems, computed with respect to the prediction. Contrary to the aforementioned hypothesis, well-trained models do not exhibit a temporal influence aligned with the causal structure of the underlying PDEs. Rather, they show an influence distribution that is close to the data distribution baseline $\bar{t}$, only deviating slightly from it. This shows that well-trained models are, on average, equally sensitive to changes in training data at both earlier and later times. This is sensible given that PINNs are trained to solve



(a) Well-trained *(b)* Poorly-trained

Figure 6. Fractions of IC influences $\bar{r}_{L_{\text{ic}}}(\mathcal{X}, R_{\text{te}})$ for 50 time bins, across all problems, averaged over seeds. Shaded regions indicate the standard deviation across seeds.

Table 2. Mean temporal influence indicator for each problem, computed with respect to predictions $f = \hat{u}$. We report mean $\pm$ standard deviation over 10 seeds.

Problem	$\bar{t}$	Well-trained	Poorly-trained
Heat	0.46	0.33 ± 0.02	0.26 ± 0.06 (*)
Allen-Cahn	0.43	0.50 ± 0.02	0.32 ± 0.05
Burgers'	0.43	0.41 ± 0.02	0.28 ± 0.02
Drift-Diffusion	0.46	0.46 ± 0.04	0.21 ± 0.06
Wave	0.43	0.41 ± 0.03	0.11 ± 0.02

(*) The baseline is $\bar{t} = 0.41$ for the poorly-trained model due to different sampling.

a given problem globally, instead of sequentially. Hence, this illustrates that a well-trained PINN has learned to leverage information from across the spatio-temporal domain. Poorly-trained models, on the other hand, display skewed temporal influences. This finding aligns directly with the elevated IC influence ratio observed earlier: these models show disproportionate sensitivity to early-time training points as exemplified in Figure 6. Hence, strong causal alignment in terms of influences is a signal of training failure rather than of success.

The previous results constitute only a selection of potential questions that one may study using PINNFLUENCE. In Appendix C, we present complementary use-cases, such as additional point-wise attribution heatmaps for other problems, the decomposition of loss terms for each setting. We also introduce further indicators, including a demonstration of region-to-region influence analysis. Notably, we also study steady-state problems with appropriately constructed influence-based indicators. Beyond these extensions to new problems, we also verify in Appendix C.4 on two optimizers, NNCG (Rathore et al., 2024) and SOAP (Vyas et al., 2025), that PINNFLUENCE is applicable independently of optimizer choice.

To summarize, PINNFLUENCE offers a complementary perspective on PINNs by attributing predictions directly to the training data and losses. It reliably distinguishes between well-trained and poorly-trained PINNs across diverse problems, enhancing the understanding of PINNs.

Limitations PINNFLUENCE inherits limitations from influence functions. PINN loss landscapes are often severely ill-conditioned, leading to near-singular Hessians, a setting under which raw leave-one-out retraining is known to be an unreliable validation target in practice (Basu et al., 2020a). In Appendix A.2, we examine the impact of this fragility for our benchmark PDEs. The low-rank Arnoldi approximation recovers projected inverse Hessians faithfully and influence estimates align with local fine-tuning behavior that IFs actually approximate (PBRF, Bae et al., 2022), out-

performing a gradient-only baseline (Charpiat et al., 2019). While these findings support reliability in our setting, the approach still remains a first-order approximation and may exhibit discrepancies under larger perturbations or more ill-conditioned settings. While the explicit computation of the inverse Hessian can be avoided through inverse-Hessian vector products, computational cost is non-negligible. The pairwise evaluation of influences scales linearly in both training and test set size, though it is typically faster than PINN training itself. In Appendix A.2.3, we provide an overview of the practical scaling of our approach.

Our methodological choices introduce further caveats: Aggregating absolute influences discards sign information, which is partially addressed by the cancellation score κ. Furthermore, indicator design is problem-dependent and requires domain knowledge for appropriate construction; we provide a practitioner's guide in Appendix A.4. Interpretation also requires care: influences quantify sensitivity to perturbations in training data, not causal importance. High attribution does not imply necessity or sufficiency for correct behavior. In particular, the temporal indicator should be read as a diagnostic signal rather than a statement about physical causality.

Future Work Beyond diagnosis, influence scores naturally rank training points by importance, suggesting avenues for actively improving PINNs. For example, targeted resampling of collocation points from influential regions and informed adjustment of loss weights that rescales components according to their attributed influence both present themselves as sensible extensions of the PINNFLUENCE framework. We outline both future directions schematically in Appendix A.5.

6. Conclusion

Existing studies of PINNs focus primarily on training dynamics and failure modes, but do not provide attribution-based explanations of how training data and physics constraints shape predictions. To address this, we propose PINNFLUENCE, a novel, training-data-centric interpretability method that adapts influence functions to PINNs and their composite losses. By enabling attribution between training data, loss components, and model predictions, PINNFLUENCE supports fine-grained diagnostics of PINN behavior that go beyond residual-based analysis. Our experiments show that loss-component and temporal influence patterns can distinguish well-trained from poorly-trained PINNs, localize pathological patterns in the trained model's data dependence, and reveal structural properties of learned solutions. Overall, while with limitations, our method presents a principled diagnostic lens for studying PINNs with the aim of imbuing scientific machine learning with interpretability methods. In future work, exploring the use of these methods for intervening in training and informed model design emerges as a promising next step.

Acknowledgments

This work was supported by the Fraunhofer Internal Programs (Fraunhofer) as grant ESPINN (PREPARE 40-08394).

Code Availability

The implementation of PINNFLUENCE is available at https://github.com/aleks-krasowski/pinnfluence.

Impact Statement

This work introduces a training-data-centric interpretability framework for physics-informed neural networks. By enabling practitioners to attribute model behavior to specific training points and loss components, PINNFLUENCE provides diagnostic capabilities that complement existing methods. We hope this opens up new directions for understanding PINN behavior and, in future work, for developing principled interventions based on this method.

As an interpretability method for scientific machine learning, we do not anticipate negative societal impacts from this work.

References

Agarwal, N., Bullins, B., and Hazan, E. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(1): 4148–4187, 2017.

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes, 2018. arXiv:1610.01644.

Anagnostopoulos, S. J., Toscano, J. D., Stergiopulos, N., and Karniadakis, G. E. Learning in PINNs: Phase transition, total diffusion, and generalization, 2024. arXiv:2403.18494.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7), 2015.

Bae, J., Ng, N., Lo, A., Ghassemi, M., and Grosse, R. If Influence Functions are the Answer, Then What is the Question? In *Proceedings of the 36th international conference on neural information processing systems*, Nips

'22, New Orleans, LA, USA, 2022. Curran Associates Inc.

Basu, S., Pope, P., and Feizi, S. Influence Functions in Deep Learning Are Fragile. In *ICLR 2021*, 2020a.

Basu, S., You, X., and Feizi, S. On Second-Order Group Influence Functions for Black-Box Predictions. In *Proceedings of the 37th international conference on machine learning*, ICML'20, 2020b.

Beltrán-Pulido, A., Bilionis, I., and Aliprantis, D. Physics-informed neural networks for solving parametric magnetostatic problems. *IEEE Transactions on Energy Conversion*, 37(4):2678–2689, 2022. arXiv:2202.04041.

Bender, S., Anders, C. J., Chormai, P., Marxfeld, H., Herrmann, J., and Montavon, G. Towards Fixing Clever-Hans Predictors with Counterfactual Knowledge Distillation. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 2599–2607. IEEE, 2023.

Berkhahn, S. and Ehrhardt, M. A physics-informed neural network to model COVID-19 infection and hospitalization scenarios. *Advances in Continuous and Discrete Models*, 2022(1):61, 2022.

Bonfanti, A., Bruno, G., and Cipriani, C. The Challenges of the Nonlinear Regime for Physics-Informed Neural Networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Braun, J., Kornreich, M., Park, J., Pawar, J., Browning, J., Herzog, R., Odry, B., and Zhang, L. Influence Based Re-Weighing for Labeling Noise in Medical Imaging. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pp. 1–5, 2022.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. Physics-informed neural networks (PINNs) for fluid mechanics: A review, 2021. arXiv:2105.09506.

Charpiat, G., Girard, N., Felardos, L., and Tarabalka, Y. Input Similarity from the Neural Network Perspective. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Cook, R. D. and Weisberg, S. *Residuals and Influence in Regression*. New York: Chapman and Hall, 1982.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models, 2023. arXiv:2309.08600.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, 92(3):88, 2022.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 2933–2941. MIT Press, 2014.

Daw, A., Bu, J., Wang, S., Perdikaris, P., and Karpatne, A. Mitigating Propagation Failures in Physics-informed Neural Networks using Retain-Resample-Release (R3) Sampling. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 7264–7302. PMLR, 2023.

Eiras, F., Bibi, A., Bunel, R., Dvijotham, K. D., Torr, P. H., and Kumar, M. P. Efficient error certification for physics-informed neural networks. In *Proceedings of the 41st international conference on machine learning*, ICML'24, 2024.

Fear, R. A., Mukhopadhyay, P., McCabe, M., Bietti, A., and Cranmer, M. Physics Steering: Causal Control of Cross-Domain Concepts in a Physics Foundation Model. In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025.

Fritzsche, K. and Grauert, H. *From Holomorphic Functions to Complex Manifolds*. Springer, 2002. ISBN 978-1-4684-9273-6.

Ghorbani, A. and Zou, J. Data Shapley: Equitable Valuation of Data for Machine Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2242–2251. PMLR, 2019.

Grosse, R., Bae, J., Anil, C., Elhage, N., Tamkin, A., Tajdini, A., Steiner, B., Li, D., Durmus, E., Perez, E., Hubinger, E., Lukošiūtė, K., Nguyen, K., Joseph, N., McCandlish, S., Kaplan, J., and Bowman, S. R. Studying Large Language Model Generalization with Influence Functions, 2023. arXiv:2308.03296.

Gupta, V., Koren, T., and Singer, Y. Shampoo: Preconditioned Stochastic Tensor Optimization. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Hammoudeh, Z. and Lowd, D. Training data influence analysis and estimation: a survey. *Machine Learning*, 113(5):2351–2403, 2024.

Hampel, F. R. The Influence Curve and its Role in Robust Estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974.

Hense, J., Jamshidi Idaji, M., Eberle, O., Schnake, T., Dippel, J., Ciernik, L., Buchstab, O., Mock, A., Klauschen, F., and Müller, K.-R. xMIL: Insightful explanations for multiple instance learning in histopathology. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in neural information processing systems*, volume 37, pp. 8300–8328, 2024.

Hillebrecht, B. and Unger, B. Rigorous a Posteriori Error Bounds for PDE-Defined PINNs. *IEEE Transactions on Neural Networks and Learning Systems*, 36(1):1583–1593, 2025.

Jacot, A., Gabriel, F., and Hongler, C. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.

Jeong, M., Seo, G., and Hwang, E. Mitigating Instability in High Residual Adaptive Sampling for PINNs via Langevin Dynamics. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kim, S. W., Kim, I., Lee, J., and Lee, S. Knowledge Integration into deep learning in dynamical systems: an overview and taxonomy. *Journal of Mechanical Science and Technology*, 35(4):1331–1342, 2021.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization, 2017. arXiv:1412.6980.

Kitouni, O., Nolte, N., Pérez-Díaz, V. S., Trifinopoulos, S., and Williams, M. From Neurons to Neutrons: A Case Study in Interpretability. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 24726–24748. PMLR, 2024.

Kiyani, E., Shukla, K., Urbán, J. F., Darbon, J., and Karniadakis, G. E. Optimizing the optimizer for physics-informed neural networks and Kolmogorov-Arnold networks. *Computer Methods in Applied Mechanics and Engineering*, 446:118308, 2025.

Koh, P. W. and Liang, P. Understanding Black-box Predictions via Influence Functions. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1885–1894. PMLR, 2017.

Koh, P. W. W., Ang, K.-S., Teo, H., and Liang, P. S. On the Accuracy of Influence Functions for Measuring Group Effects. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., and Reblitz-Richardson, O. Captum: A unified and generic model interpretability library for PyTorch, 2020.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26548–26560, 2021.

Lau, G. K. R., Hemachandra, A., Ng, S.-K., and Low, B. K. H. PINNACLE: PINN adaptive ColLocation and experimental points selection. In *The Twelfth International Conference on Learning Representations*, 2024.

Liu, D. C. and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

Liu, Q., Chu, M., and Thuerey, N. ConFIG: Towards Conflict-free Training of Physics Informed Neural Networks, 2025. arXiv:2408.11104.

Liu, Y., Chen, L., and Ding, J. Grad-RAR: An Adaptive Sampling Method Based on Residual Gradient for Physical-Informed Neural Networks. In *2022 International Conference on Automation, Robotics and Computer Engineering (ICARCE)*, pp. 1–5, 2022.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljacic, M., Hou, T. Y., and Tegmark, M. KAN: Kolmogorov–Arnold Networks. In *The Thirteenth International Conference on Learning Representations*, 2024.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. DeepXDE: A Deep Learning Library for Solving Differential Equations. *SIAM Review*, 63(1):208–228, 2021.

MacMillan, T. and Ouellette, N. T. Towards mechanistic understanding in a data-driven weather model: internal activations reveal interpretable physical features, 2025.

McClenny, L. D. and Braga-Neto, U. M. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722, 2023.

Medvedev, V., Erdmann, A., and Rosskopf, A. Modeling of Near-and Far-Field Diffraction from EUV Absorbers Using Physics-Informed Neural Networks. In *2023 Photonics & Electromagnetics Research Symposium (PIERS)*, pp. 297–305, 2023.

Müller, J. and Zeinhofer, M. Achieving High Accuracy with PINNs via Energy Natural Gradient Descent. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 25471–25485. PMLR, 2023.

Oliveira, O. R. B. d. The Implicit and the Inverse Function theorems: easy proofs, 2012. arXiv:1212.2066.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019.

Pezeshkpour, P., Jain, S., Wallace, B., and Singh, S. An Empirical Comparison of Instance Attribution Methods for NLP. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 967–975. Association for Computational Linguistics, 2021.

Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. Estimating Training Data Influence by Tracing Gradient Descent. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19920–19930, 2020.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Ramabathiran, A. A. and Ramachandran, P. SPINN: Sparse, Physics-based, and partially Interpretable Neural Networks for PDEs. *Journal of Computational Physics*, 445: 110600, 2021.

Ranasinghe, N., Xia, Y., Seneviratne, S., and Halgamuge, S. GINN-KAN: Interpretability pipelining with applications in Physics Informed Neural Networks, 2024. arXiv:2408.14780.

Rathore, P., Lei, W., Frangella, Z., Lu, L., and Udell, M. Challenges in Training PINNs: A Loss Landscape Perspective. In *Proceedings of the 41st international conference on machine learning*, ICML'24, 2024.

Rieger, L., Singh, C., Murdoch, W., and Yu, B. Interpretations are Useful: Penalizing Explanations to Align Neural Networks with Prior Knowledge. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 8116–8126. PMLR, 2020.

Rohrhofer, F. M., Posch, S., Gößnitzer, C., and Geiger, B. C. On the Role of Fixed Points of Dynamical Systems in Training Physics-Informed Neural Networks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Ross, A. S., Hughes, M. C., and Doshi-Velez, F. Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 2662–2670, 2017.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., and Zhong, C. Interpretable Machine Learning: Fundamental Principles and 10 Grand Challenges, 2021. arXiv:2103.11251.

Ryck, T. D., Bonnet, F., Mishra, S., and Bézenac, E. d. An operator preconditioning perspective on training in physics-informed machine learning, 2024. arXiv:2310.05801.

Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., and Müller, K.-R. Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications. *Proceedings of the IEEE*, 109(3):247–278, 2021. arXiv:2003.07631.

Schioppa, A., Zablotskaia, P., Vilar, D., and Sokolov, A. Scaling Up Influence Functions, 2021. arXiv:2112.03052.

Schioppa, A., Filippova, K., Titov, I., and Zablotskaia, P. Theoretical and Practical Perspectives on what Influence Functions Do. *Advances in Neural Information Processing Systems*, 36:27560–27581, 2023.

Schnake, T., Eberle, O., Lederer, J., Nakajima, S., Schutt, K. T., Muller, K.-R., and Montavon, G. Higher-Order Explanations of Graph Neural Networks via Relevant Walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7581–7596, 2022.

Schäfer, M., Turek, S., Durst, F., Krause, E., and Rannacher, R. Benchmark Computations of Laminar Flow Around a Cylinder. In Hirschel, E. H. (ed.), *Flow Simulation with High-Performance Computers II*, volume 48, pp. 547–566. 1996. ISBN 978-3-322-89851-7. Series Title: Notes on Numerical Fluid Mechanics (NNFM).

Steger, S., Rohrhofer, F. M., and Geiger, B. C. How PINNs cheat: Predicting chaotic motion of a double pendulum. In *The Symbiosis of Deep Learning and Differential Equations II @ NeurIPS 2022*, 2022.

Vyas, N., Morwani, D., Zhao, R., Shapira, I., Brandfonbrener, D., Janson, L., and Kakade, S. SOAP: Improving and Stabilizing Shampoo using Adam for Language Modeling. In *International Conference on Learning Representations, 2025*, pp. 93423–93444, 2025.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 2021.

Wang, S., Yu, X., and Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Wang, S., Sankaran, S., Wang, H., and Perdikaris, P. An Expert's Guide to Training Physics-informed Neural Networks, 2023. arXiv:2308.08468.

Wang, S., Sankaran, S., and Perdikaris, P. Respecting causality for training physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 421:116813, 2024.

Wang, S., Bhartari, A., Li, B., and Perdikaris, P. Gradient Alignment in Physics-informed Neural Networks: A Second-Order Optimization Perspective. In *Advances in Neural Information Processing Systems 38*, pp. 168482–168532, 2025.

Wei, D., Padhi, I., Ghosh, S., Dhurandhar, A., Ramamurthy, K. N., and Chang, M. Final-Model-Only Data Attribution with a Unifying View of Gradient-Based Methods, 2024. arXiv:2412.03906.

Wetzel, S. J., Ha, S., Iten, R., Klopotek, M., and Liu, Z. Interpretable Machine Learning in Physics: A Review, 2025. arXiv:2503.23616.

Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

Wu, H., Ma, Y., Zhou, H., Weng, H., Wang, J., and Long, M. ProPINN: Demystifying Propagation Failures in Physics-Informed Neural Networks, 2025. arXiv:2502.00803.

Yeh, C.-K., Kim, J., Yen, I. E.-H., and Ravikumar, P. K. Representer Point Selection for Explaining Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Yolcu, G. U., Weckbecker, M., Wiegand, T., Samek, W., and Lapuschkin, S. Sparse, Efficient and Explainable Data Attribution with DualXDA. *Transactions on Machine Learning Research*, 2025.

Zhang, R. and Zhang, S. Rethinking Influence Functions of Neural Networks in the Over-parameterized Regime. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):9082–9090, 2022.

Appendix

The Appendix is structured as follows: Appendix A focuses on influence functions and provides an extended theoretical background in Appendix A.1 and detailed practical considerations in Appendix A.2, including validation of the Hessian approximation quality, empirical confirmation of influence estimates, computational cost analysis, and a discussion of alternative training data attribution methods. Appendix A.3 contrasts PINNFLUENCE with training-time diagnostics, Appendix A.4 gives a short recipe for constructing diagnostic indicators, and Appendix A.5 outlines resampling and loss-weighting extensions.

Appendix B contains descriptions of the investigated problems, including two steady-state PDEs (Appendix B.1). Appendix B.2 specifies the training environment and hyperparameters for model configurations across problems and showcases predictions in Appendix B.3.

Finally, Appendix C features supplemental experiments utilizing PINNFLUENCE. Concretely, we adapt the temporal indicator for steady-state problems in Appendix C.1, and showcase additional region-to-region indicators in Appendix C.2. We further sweep the training-data quantity continuously in Appendix C.3 to confirm that the regional indicators respond to gradual variation in training quality, and verify that the diagnostics depend on the converged model rather than the optimizer used to reach it by evaluating curvature-aware optimizers in Appendix C.4. Finally, we provide loss-term decompositions for the remaining problems in Appendix C.5 and additional point-wise heatmaps in Appendix C.6.

A. Supplementary Information for Influence Functions

A.1. Theoretical Background

In this part, we formulate the key steps for deriving influence functions, initially introduced in (Hampel, 1974) to study robust estimators, and which were recently applied to neural networks (Koh & Liang, 2017; Koh et al., 2019; Hammoudeh & Lowd, 2024). However, as already observed in (Schioppa et al., 2023), the functions one typically encounters in neural networks require a more exhaustive analysis. Therefore, we carefully redraw the results proposed in (Koh & Liang, 2017) and generalize them slightly for the application to PINNs. In particular, we formulate influence functions in a manner that enables the study of the effect of various modifications of the problem (i.e., changes in the training set, in the loss weighting, or in the PDE) on the prediction and any derived quantity. Furthermore, we provide rigorous proofs for the statements made in Section 2.2.

We start with a crucial lemma for influence functions, which was already noted in (Cook & Weisberg, 1982). As remarked in (Schioppa et al., 2023), influence functions are usually stated for strictly convex functions[2] in the vicinity of global minima. However, in practice, we typically only reach saddle points (Dauphin et al., 2014), which is the reason why the following lemma is extended to stationary points.

Lemma A.1 (argmin Trick)**.** *Consider an open set of parameters $\Theta \subseteq \mathbb{R}^p$ and an open set $U \subseteq \mathbb{R}$. Let $g : \Theta \times U \to \mathbb{R}$ be a twice continuously differentiable function and $(\theta_0, \epsilon_0) \in \Theta \times U$, such that θ_0 is a non-degenerate stationary point of $g(\,\cdot\,, \epsilon_0)$, i.e., $\nabla_\theta g(\theta_0, \epsilon_0) = 0$ and the Hessian $\mathcal{H}_\theta(\theta_0, \epsilon_0) := \nabla^2_\theta g(\theta_0, \epsilon_0) \in \mathbb{R}^{p\times p}$ is invertible.*

Then, there exists a non-empty, open neighborhood $U_0 \subseteq U$ around ϵ_0, such that $g(\,\cdot\,, \epsilon)$ has a unique stationary point in the neighborhood of θ_0 for all $\epsilon \in U_0$. Denote by $h : U_0 \to \Theta$ the function which describes such stationary points. Then h is continuously differentiable and

$$\frac{\partial h(\epsilon)}{\partial \epsilon} = -\mathcal{H}_{\theta_0}(h(\epsilon), \epsilon)^{-1} \cdot \nabla_\theta \frac{\partial g(\theta, \epsilon)}{\partial \epsilon}\bigg|_{\theta = h(\epsilon)} \quad . \tag{10}$$

Furthermore, if g is analytic in a neighborhood of (θ_0, ϵ_0), then we can choose U_0 such that h is analytic as well. Moreover, if $B \subseteq U$ is an open, connected set, such that for every $\epsilon \in B$, there exists a non-degenerate stationary point of $g(\,\cdot\,, \epsilon)$, then h is extendable to B and $h : B \to \Theta$ is also continuously differentiable with (10).

Proof. Let us denote the gradient of g as $s(\theta, \epsilon) := \nabla_\theta g(\theta, \epsilon)$. By our initial assumption, g is twice continuously differentiable, implying that s is continuously differentiable. Since (θ_0, ϵ_0) is a stationary point, $s(\theta_0, \epsilon_0) = 0$, the implicit

[2]Note that for the application of influence functions with global minima (e.g. Koh & Liang, 2017), strict convexity is a necessary, but not a sufficient criterion.

function theorem (see Oliveira, 2012, Theorem 2) guarantees the existence of an open set $U_0 \subseteq U$ around ϵ_0 and a unique, continuously differentiable function $h : U_0 \to \Theta$ such that $h(\epsilon_0) = \theta_0$ and $s(h(\epsilon), \epsilon) = 0$ for all $\epsilon \in U_0$. This establishes that for any $\epsilon \in U_0$, a unique stationary point exists in the neighborhood of θ_0. And h is continuously differentiable with (10).

Now if g is analytic, then so is its gradient s. Therefore also the function h is analytic, following the holomorphic implicit function theorem (see Fritzsche & Grauert, 2002, Section 7.6).

If for all $\epsilon \in B$ the above assumptions are fulfilled, then we can apply Lemma A.1 for all of these points ϵ. The function $h : B \to \Theta$ is continuously differentiable, since it is continuously differentiable in every point and (10) extends to all points in B. Moreover, $h(B)$ is connected, since h is continuous and B is connected by assumption. □

By means of this theorem, we can now prove Theorem 2.2 from Section 2.2. For better readability, we restate that theorem.

Theorem A.2 (Influence Functions Approximate Retraining, analogous to Theorem 2.2)**.** *Assume the setting of Definition 2.1 with L and f being analytic, and that θ_0 is a non-degenerate stationary point of $\mathcal{L}$ with respect to a training set $\mathcal{X}$. Let θ_1 denote the stationary point of $\mathcal{L}(\theta)$ (i.e., $\nabla_\theta \mathcal{L}(\theta_0) = 0$, $\det \nabla^2_\theta \mathcal{L}(\theta_0) \neq 0$) with respect to $\mathcal{X}'$, where we added or removed a training point $x^\pm$ (with $x^- \in \mathcal{X}$). For sufficiently large N, the resulting effect on the function f is given by*

$$f(z, \theta_1) - f(z, \theta_0) = \pm \operatorname{Inf}^{L \to f}_{\theta_0}(x^\pm, z) \frac{1}{N} + \mathcal{O}(1/N^2) \quad . \tag{11}$$

Proof. Let $\tilde{\mathcal{L}}(\theta, \epsilon) := \mathcal{L}(\theta) + \epsilon\, \kappa(\theta)$ be a small perturbation of the original loss with some differentiable function $\kappa : \Theta \to \mathbb{R}$ and $\epsilon \in \mathbb{R}$. Since θ_0 is a non-degenerate stationary point of $\mathcal{L}(\theta)$, $(\theta_0, 0)$ is a non-degenerate stationary point of $\tilde{\mathcal{L}}(\theta, \epsilon)$ in the sense of Lemma A.1. Hence, there is a neighborhood U_0 around the origin $\epsilon_0 = 0$, such that the stationary points θ_ϵ of $\tilde{\mathcal{L}}(\theta, \epsilon)$ are uniquely defined for all $\epsilon \in U_0$ and the derivative of θ_ϵ is given by (10).

Since θ_ϵ is also analytic, due to the assumptions and Lemma A.1, we can approximate it with a Taylor expansion as

$$\theta_\epsilon - \theta_0 = \left.\frac{\partial \theta_\epsilon}{\partial \epsilon}\right|_{\epsilon=0} \epsilon + \mathcal{O}(\epsilon^2) = -\mathcal{H}^{-1}_{\theta_0} \cdot \nabla_\theta \kappa(\theta)\big|_{\theta=\theta_0} \cdot \epsilon + \mathcal{O}(\epsilon^2) \tag{12}$$

with $\mathcal{H}_{\theta_0} = \nabla^2_\theta \mathcal{L}(\theta)\big|_{\theta=\theta_0}$. When setting $\kappa(\theta) = L(x^+; \theta)$, we have $\tilde{\mathcal{L}}(\theta, 1/N) = \frac{1}{N} \sum_{x \in \mathcal{X} \cup \{x^+\}} L(x; \theta)$, which has the same stationary points as the empirical risk over $\mathcal{X} \cup \{x^+\}$, which is $\frac{1}{N+1} \sum_{x \in \mathcal{X} \cup \{x^+\}} L(x; \theta)$. In other words, we have $\theta_{1/N} = \theta_1$ providing that $1/N \in U_0$, which is always satisfied for large N. We obtain by chain rule

$$f(z, \theta_1) - f(z, \theta_0) = \nabla_\theta f(z, \theta_0) \cdot \left.\frac{\partial \theta_\epsilon}{\partial \epsilon}\right|_{\epsilon=0} \cdot \frac{1}{N} + \mathcal{O}(1/N^2) \tag{13}$$

which concludes the proof for x^+ by application of Lemma A.1. Similarly, we obtain the result for x^- by considering $\kappa(\theta) = -L(x^-; \theta)$ with an additional sign.

The Corollary 2.3 follows immediately, by considering the linearity in equations (12) and (13). □

From the details of the proof we also immediately see the effect of adding several or removing several points. Hence, we find that a change from a training set $\mathcal{X}$ to a set $\mathcal{X}' = \mathcal{X} \cup \mathcal{X}^+ \setminus \mathcal{X}^-$, where we assume $\mathcal{X}^- \subset \mathcal{X}$ will affect a function $f : A \times \Theta$ according to

$$f(z; \theta_1) - f(z; \theta_0) = \frac{1}{N} \left(\sum_{x \in \mathcal{X}^+} \operatorname{Inf}^{L \to f}_{\theta_0}(x, z) - \sum_{x \in \mathcal{X}^-} \operatorname{Inf}^{L \to f}_{\theta_0}(x, z) \right) + \mathcal{O}(1/N^2) \tag{14}$$

for large $N > N_{\text{min}}$. There is always a value N_{min}, where (14) is valid, but the precise number depends on the set U_0, and for large changes of the training set $\mathcal{X} \to \mathcal{X}'$ this number might increase.

One can easily extend Theorem A.2 to study also the influence of other parts in the optimization task. E.g. one could study the influence of certain parameters in the PDE, such as the diffusivity constant α in the heat equation. By computing the Taylor expansion with respect to that parameter α, one approximates the effect of increasing or lowering that parameter. The proof is completely analogous to the proof of Theorem A.2 and is left to the reader.

A.2. Practical Considerations On The Computation Of Influences

Influence functions are typically derived under the assumption of a well-conditioned Hessian at a strict local minimum (Koh & Liang, 2017), which in practice is very unlikely to hold for neural networks in general. For PINNs, in particular, the loss landscape is known to be non-convex and ill-conditioned (Rathore et al., 2024), and converged models reach non-degenerate stationary points rather than strict minima. Whether the resulting influence estimates remain numerically meaningful is therefore a legitimate concern.

We address this in two ways. First, we check whether the low-rank Arnoldi approximations that we utilize faithfully invert the Hessian on the gradients that enter the influence computation. Second, we evaluate whether the resulting influence scores predict actual data-removal effects, validated using the Proximal Bregman Response Function (Bae et al., 2022), which captures the local fine-tuning behavior that IFs approximate. We further quantify computational costs and situate PINNFLUENCE among alternative training data attribution methods.

A.2.1. Approximating the inverse Hessian

Explicit computation of the inverse Hessian $\mathcal{H}_{\theta_0}^{-1}$ is costly, but can be circumvented by utilizing inverse-Hessian vector products (IHVPs). Several efficient approximations of IHVPs have been proposed in the literature, including the linear time stochastic second-order algorithm (LiSSA) (Agarwal et al., 2017; Koh & Liang, 2017), Arnoldi iterations (Schioppa et al., 2021) and Eigenvalue-corrected Kronecker-Factored Approximate Curvature (EK-FAC) (Grosse et al., 2023). We utilize a modified version of the `ArnoldiInfluenceFunction` from the Captum library (Kokhlikyan et al., 2020) with the following hyperparameters: 200 Arnoldi iterations, top-50 retained eigenvectors, and damping $\mathcal{H}_{\theta_0} + \lambda I$ with $\lambda = 10^{-3}$.

Concretely, the approximation recovers top eigenvectors and eigenvalues of the damped Hessian using only Hessian-vector products, from which a low-rank surrogate of $\mathcal{H}_{\theta_0}^{-1}$ is assembled as follows. Let $V \in \mathbb{R}^{p\times k}$ contain the top-k eigenvectors of the damped Hessian with the corresponding eigenvalues on the diagonal of $S \in \mathbb{R}^{k\times k}$, and let $R = VS^{-\frac{1}{2}}$, so that $RR^\top = VS^{-1}V^\top \approx \mathcal{H}_{\theta_0}^{-1}$. With R at hand, the influence calculation in (4) becomes

$$\mathrm{Inf}_{\theta_0}^{L\to f}(x,z) = -\big(\nabla_\theta f(z;\theta_0)^\top R\big)\big(R^\top \nabla_\theta L(x;\theta_0)\big) \quad . \tag{15}$$

Hessian approximation quality To validate the IHVP approximation, we compute the full dense Hessian for each problem configuration. The exact spectra confirm severe ill-conditioning with condition numbers from $\kappa \approx 10^{20}$ to 10^{33} across all problems (Table 4). Damping $\mathcal{H}_{\theta_0} + \lambda I$ with $\lambda = 10^{-3}$ reduces κ to between 10^8 and 10^{15}. Across all configurations, the smallest retained eigenvalue exceeds λ by at least three orders of magnitude, so in practice, damping has negligible effect on the top-k subspace and primarily acts as a safeguard against the ill-conditioned tail. We note that small or even negative eigenvalues are admissible in this tail since θ_0 is a non-degenerate stationary point rather than a strict minimum.

What remains is to confirm that the discarded tail is in fact irrelevant for influence computation. We probe how well $RR^\top$ acts as an inverse to $\mathcal{H}_{\theta_0}$ on the gradients that enter the influence estimation by evaluating the cosine similarity between $\mathcal{H}_{\theta_0} RR^\top g$ and g, where g is either a per-sample loss gradient $\nabla_\theta L(x)$ or a prediction gradient $\nabla_\theta f(x)$ at a training point $x \in \mathcal{X}_{\text{train}}$. Table 3 reports the mean cosine similarities over 10 seeds. Recovery is near-perfect for Allen-Cahn, Burgers', Heat, and the well-trained Drift-Diffusion, and remains strong for the remaining configurations. The gradients therefore concentrate in the top-k subspace and the truncation to top-k eigendirections acts as a regularization of a numerically ill-conditioned operator rather than purely as a computational shortcut (Table 4). In fact, it has been observed that low-rank Arnoldi approximations can outperform exact inversion of the Hessian in practice (Schioppa et al., 2021).

A.2.2. Validation of Influence Functions

The first-order approximation underlying influence functions assumes convexity and is found to be fragile as a true leave-one-out estimator (Basu et al., 2020a), particularly for very deep neural networks. While our model configurations are fairly shallow, especially in comparison to modern computer vision and LLM architectures, PINNs exhibit non-convex and ill-conditioned loss landscapes (Rathore et al., 2024), so the question of whether influence estimates correspond to actual data-removal effects warrants empirical validation. Despite the theoretical limitations, influence functions have been found to reliably recover influential samples in practice (Koh et al., 2019; Basu et al., 2020b). Recent work suggests that they are better understood as approximators of local fine-tuning behavior rather than exact leave-one-out retraining (Bae et al., 2022; Wei et al., 2024). We confirm this distinction directly: correlation with exact leave-one-out retraining is near zero (Table 5), while correlation with the PBRF, *the object influence functions are designed to approximate*, is substantially higher.

Table 3. Cosine similarities between reconstructed $\mathcal{H}_{\theta_0} RR^\top g$ and true gradients g averaged over the whole training set $\mathcal{X}_{\text{train}}$ and over 10 runs (mean $\pm$ std).

	$\nabla_\theta L$		$\nabla_\theta f$	
Problem	Well-trained	Poorly-trained	Well-trained	Poorly-trained
Heat	$1.00 \pm 0.00^{(*)}$	$1.00 \pm 0.00^{(*)}$	$1.00 \pm 0.00^{(*)}$	$1.00 \pm 0.00^{(*)}$
Allen-Cahn	$1.00 \pm 0.00^{(*)}$	$1.00 \pm 0.00^{(*)}$	0.99 ± 0.03	0.99 ± 0.02
Burgers'	0.99 ± 0.01	0.99 ± 0.01	1.00 ± 0.01	0.99 ± 0.00
Drift-Diffusion	1.00 ± 0.01	0.90 ± 0.10	0.99 ± 0.01	0.85 ± 0.08
Wave	0.99 ± 0.02	0.99 ± 0.02	0.88 ± 0.05	0.92 ± 0.04
Poisson	0.96 ± 0.04	0.98 ± 0.02	0.93 ± 0.01	0.93 ± 0.02
Navier-Stokes$^{(**)}$	0.87 ± 0.14	0.80 ± 0.16	0.87 ± 0.06	0.94 ± 0.02

$^{(*)}$ For Heat and for Allen-Cahn, the stds are below $7 \cdot 10^{-4}$ and 10^{-3}, respectively.
$^{(**)}$ For Navier-Stokes f is u, i.e., the predicted velocity in x direction.

Table 4. Mean condition numbers of the Hessian $\mathcal{H}_{\theta_0}$ at three stages of the inverse approximation: the exact (dense) Hessian, the damped Hessian $\mathcal{H}_{\theta_0} + \lambda I$ with $\lambda = 10^{-3}$, and the top-k subspace retained by the Arnoldi iteration.

	Exact $\mathcal{H}_{\theta_0}$		Damped $\mathcal{H}_{\theta_0} + \lambda I$		Arnoldi top-k	
Problem	Poorly-tr.	Well-tr.	Poorly-tr.	Well-tr.	Poorly-tr.	Well-tr.
Heat	3.0×10^{21}	8.5×10^{20}	9.2×10^{8}	2.4×10^{11}	4.5×10^{6}	8.4×10^{5}
Allen-Cahn	9.7×10^{25}	9.9×10^{24}	1.8×10^{12}	9.2×10^{12}	1.4×10^{4}	2.1×10^{4}
Burgers'	3.5×10^{21}	4.7×10^{21}	1.3×10^{12}	8.5×10^{11}	1.0×10^{4}	5.1×10^{3}
Drift-Diffusion	2.6×10^{23}	8.0×10^{23}	3.4×10^{12}	2.1×10^{12}	3.1×10^{3}	1.2×10^{3}
Wave	1.3×10^{32}	1.3×10^{33}	2.0×10^{14}	8.6×10^{15}	1.8×10^{3}	8.2×10^{2}
Navier-Stokes	5.0×10^{24}	6.4×10^{27}	3.7×10^{11}	1.2×10^{13}	5.2×10^{2}	4.0×10^{2}
Poisson	2.2×10^{22}	1.3×10^{22}	4.9×10^{12}	3.5×10^{13}	1.9×10^{4}	3.1×10^{2}

Influence estimates do not approximate leave-one-out retraining We first verify that influence functions do not recover exact leave-one-out effects in our setting. For each problem, we retrain from scratch with 30 individual training points removed and correlate the observed prediction change with the IF estimate. Table 5 confirms near-zero Spearman correlations across all PDEs and training regimes, consistent with findings in IF literature (Basu et al., 2020a).

Table 5. Leave-one-out validation. Spearman rank correlation between IF-predicted and LOO-observed prediction differences for 30 randomly removed training points, retrained from scratch, averaged over 10 seeds (mean $\pm$ std).

	Influence Function		Grad-Dot	
Problem	Poorly-trained	Well-trained	Poorly-trained	Well-trained
Heat	-0.02 ± 0.13	0.00 ± 0.03	0.01 ± 0.06	0.00 ± 0.02
Allen-Cahn	-0.01 ± 0.04	0.00 ± 0.02	-0.05 ± 0.05	0.00 ± 0.01
Burgers'	0.00 ± 0.02	-0.00 ± 0.01	0.00 ± 0.02	-0.00 ± 0.02
Drift-Diffusion	0.01 ± 0.03	-0.00 ± 0.02	0.01 ± 0.02	-0.00 ± 0.01
Wave	0.02 ± 0.07	0.00 ± 0.03	0.01 ± 0.02	-0.00 ± 0.03
Navier-Stokes	0.02 ± 0.14	-0.05 ± 0.09	0.03 ± 0.07	-0.01 ± 0.05
Poisson	-0.01 ± 0.04	-0.01 ± 0.04	-0.00 ± 0.02	0.00 ± 0.02

Proximal Bregman Response Function Recent work shows that influence functions are better understood as approximators of the Proximal Bregman Response Function (PBRF) (Bae et al., 2022), which captures the regularized local effect of data removal via fine-tuning rather than retraining from scratch.

For each problem, we sample 30 training points randomly. For each sampled point individually, we fine-tune the converged model for 1 000 L-BFGS steps with that point removed and compute the Spearman rank correlation between IF-predicted

and PBRF-observed prediction differences across test points (averaged over 10 seeds). As a baseline, we additionally report Grad-Dot (Charpiat et al., 2019), a gradient-based attribution method which corresponds to setting $\mathcal{H}_{\theta_0}^{-1} = I$, to confirm the utility of the inverse Hessian. Table 6 reports the resulting Spearman rank correlations for both well-trained and poorly-trained configurations.

Influence functions achieve Spearman rank correlations of 0.08 to 0.77 across the evaluated PDEs and both training regimes, substantially exceeding the gradient-only baseline. The inverse-Hessian factor therefore carries meaningful information about data-removal effects beyond what is available from raw gradients.

Table 6. PBRF validation. Spearman rank correlation between IF-predicted and PBRF-observed prediction differences for 30 randomly removed training points, averaged over 10 seeds.

	Influence Function		Grad-Dot	
Problem	Poorly-trained	Well-trained	Poorly-trained	Well-trained
Heat	0.45 ± 0.26	0.28 ± 0.06	0.09 ± 0.06	0.12 ± 0.05
Allen-Cahn$^{(*)}$	0.69 ± 0.12	0.45 ± 0.06	-0.01 ± 0.06	0.00 ± 0.03
Burgers'	0.17 ± 0.21	0.77 ± 0.04	0.09 ± 0.07	-0.08 ± 0.09
Drift-Diffusion	0.08 ± 0.23	0.54 ± 0.20	0.07 ± 0.13	0.15 ± 0.06
Wave$^{(*)}$	0.15 ± 0.25	0.47 ± 0.11	-0.00 ± 0.04	0.02 ± 0.05
Navier-Stokes$^{(\dagger)}$	0.36 ± 0.14	0.37 ± 0.15	-0.04 ± 0.15	-0.01 ± 0.06
Poisson	0.54 ± 0.16	0.47 ± 0.12	0.02 ± 0.09	0.12 ± 0.05

$^{(*)}$ For the poorly-trained Allen-Cahn and Wave configurations, training uses Adam (cf. Table 11); PBRF is computed with 1 000 L-BFGS iterations, matching the protocol used for the other PDEs.
$^{(\dagger)}$ For Navier-Stokes f is u, i.e., the predicted velocity in x direction.

A.2.3. Computational Costs

We characterize the cost of PINNfluence both asymptotically and empirically. Let $n \in \mathbb{N}$ denote the number of training points, $m \in \mathbb{N}$ the number of test points, $p \in \mathbb{N}$ the number of network parameters, $k \in \mathbb{N}$ the number of retained top eigenvectors, and $d \in \mathbb{N}$ the number of Arnoldi iterations.

One-time Costs (Approximation of the IHVP) The Arnoldi approximation produces a low-rank surrogate $R \in \mathbb{R}^{p \times k}$ with $RR^\top \approx \mathcal{H}_{\theta_0}^{-1}$, where $\theta_0 \in \mathbb{R}^p$ are the parameters of the converged neural network. We compute Hessian-vector products exactly by accumulating per-batch contributions across all mini-batches. This stage requires $\mathcal{O}(n \cdot p \cdot d)$ time and $\mathcal{O}(d \cdot p)$ peak space. The subsequent distillation to the top-k eigenpairs reduces the stored representation to $\mathcal{O}(k \cdot p)$, with $k < d \ll p$ in practice.

Per-pair Costs (Influence Scoring) For a training-test pair (x, z) using Arnoldi iterations, the influence score decomposes as in (15), yielding a per-pair time complexity of $\mathcal{O}(p \cdot k)$. Note that both projected gradients $\nabla_\theta f(z; \theta_0)^\top R$ and $R^\top \nabla_\theta L(x; \theta_0)$ can be cached, such that the full $n \times m$ influence matrix requires $\mathcal{O}\big((n+m) \cdot p \cdot k\big)$ time for projection and $\mathcal{O}(n \cdot m \cdot k)$ for the pairwise dot products, with an additional $\mathcal{O}\big((n+m) \cdot k\big)$ space for the cached projections and $\mathcal{O}(n \cdot m)$ for storing all results.

Empirical Costs In Tables 7 to 9, we report the wall-clock time and peak memory footprints across model size, training set size and test set size, all measured on the Drift-Diffusion PDE as a representative configuration. Training points n are split into 1 000 domain points, 100 initial and 100 boundary points. The training-size sweep scales these ratios proportionally. All measurements use 6 cores of Intel Xeon Gold 6448H CPUs and are reported as mean $\pm$ standard deviation over 10 seeds. We note that GPU acceleration and implementation-level improvements may reduce wall-clock time significantly.

In practice, an $\sim 18\times$ increase in model size (from 2 241 to 40 801 parameters) only roughly doubles wall-clock time, and the empirical scaling in n and m matches the asymptotic behavior outlined above. For reference, training a PINN for the same problem with 3×64 hidden neurons and 1 200 training points for 15 000 Adam and 5 000 L-BFGS iterations takes 200.7s on average on the same hardware (across 10 runs).

Table 7. Computational costs by model size for $n = 1\,200$ training and $m = 1\,000$ test points. We report mean $\pm$ standard deviation across 10 runs.

Layers	Parameters p	Hessian Approx. (s)	Influence Scoring (s)	Total (s)	Memory (MB)
3×16	609	12.6 ± 1.6	4.1 ± 0.44	16.7 ± 2.0	260 ± 48
3×32	2 241	14.2 ± 1.5	5.5 ± 0.56	19.7 ± 2.1	534 ± 56
3×64	8 577	17.4 ± 2.5	7.6 ± 1.8	25.0 ± 4.1	775 ± 164
3×100	20 601	19.9 ± 2.5	11.1 ± 1.9	31.0 ± 4.4	$1\,236 \pm 259$
5×100	40 801	28.0 ± 3.8	17.2 ± 2.8	45.2 ± 6.5	$1\,959 \pm 66$

Table 8. Computational costs by training set size for $m = 1\,000$ test points and an MLP with 3×64 hidden layers. We report mean $\pm$ standard deviation across 10 runs.

Number of Train points n	Hessian Approx. (s)	Influence Scoring (s)	Total (s)	Memory (MB)
100	6.2 ± 0.31	3.6 ± 0.29	9.8 ± 0.54	846 ± 244
1 000	12.3 ± 1.6	7.1 ± 1.3	19.4 ± 2.9	662 ± 240
10 000	99.7 ± 11.5	37.7 ± 7.6	137 ± 18.2	739 ± 167
100 000	978 ± 105	332 ± 61.7	1310 ± 160	$1\,131 \pm 134$

A.2.4. Alternative Data Attribution Methods

Beyond influence functions, alternative training data attribution methods exist with different trade-offs. Retraining-based approaches such as naive leave-one-out retraining and Data Shapley (Ghorbani & Zou, 2019) provide gold-standard estimates but are computationally prohibitive. Simpler approximations include similarity-based methods that compare loss gradients (e.g., Grad-Dot (Charpiat et al., 2019)), or penultimate-layer representations (Pezeshkpour et al., 2021). Alternative approaches that do not rely on estimating leave-one-out retraining effects include the Representer Points method (Yeh et al., 2018), which avoids Hessian computation via L^2 regularization and application of the representer theorem. TracIn (Pruthi et al., 2020) accumulates gradient dot products throughout training to trace influence over the optimization trajectory. DualXDA (Yolcu et al., 2025) performs attribution in the penultimate layer using SVM dual coefficients for sparse and efficient estimates, further employing relevance back-propagation (Bach et al., 2015) to pinpoint influential features in the given train and test samples, explaining the interaction of these data points.

A.3. Comparison to Training-Time Diagnostics

To highlight the core methodological distinction, we compare against three representative works that too examine how loss components, collocation points, and temporal structure shape PINN behavior: Wang et al. (2025) diagnose gradient conflicts between loss terms via cosine-similarity metrics, PINNACLE (Lau et al., 2024) selects collocation points based on alignment with NTK eigenfunctions, and Wang et al. (2024) enforce temporal causality during training. These approaches operate during training at batch or trajectory granularity. PINNfluence instead operates post-hoc – that is, after training – and per-point: it attributes individual predictions to individual training points and individual loss components on a converged model, addressing complementary questions rather than competing with training-time interventions. Table 10 summarizes the methodological distinctions between PINNfluence and the closest training-time approaches discussed in Section 3.

Table 9. Computational costs by test set size for $n = 1\,200$ training points and an MLP with 3×64 hidden layers. We report mean $\pm$ standard deviation across 10 runs.

Number of Test points m	Hessian Approx. (s)	Influence Scoring (s)	Total (s)	Memory (MB)
100	18.3 ± 2.5	4.3 ± 0.64	22.6 ± 3.1	684 ± 341
1 000	18.2 ± 2.2	7.9 ± 1.1	26.1 ± 3.3	758 ± 188
10 000	18.0 ± 2.1	43.3 ± 8.7	61.3 ± 10.7	925 ± 112
100 000	18.7 ± 2.0	406 ± 60.0	425 ± 61.8	$1\,271 \pm 178$

Table 10. Comparison of related training-time diagnostic methods and PINNFLUENCE.

	Gradient Alignment (Wang et al., 2025)	PINNACLE (Lau et al., 2024)	Causal PINN (Wang et al., 2024)	PINNFLUENCE
When	Training	Training	Training	Post-hoc
Goal	Fix gradient conflicts	Select collocation pts	Enforce temporal causality	Attribute predictions to training data
Granularity	1 Scalar/loss-pair/step	Batch-level	Temporal slice	Per point × per prediction × per loss
Per-prediction attribution?	No	No	No	Yes

Beyond the aforementioned differences, the methods differ in the kind of question they answer. Gradient alignment analysis (Wang et al., 2025) reveals *that* gradients of different loss terms conflict during training and proposes a condensed scalar cosine similarity per loss pair. PINNFLUENCE on the other hand identifies *where* in the domain and *through which* loss component this interplay shapes a given model prediction. As a result, it provides a spatiotemporal resolution that aggregate gradient statistics do not provide. PINNACLE (Lau et al., 2024) ranks candidate collocation points *prospectively* based on their alignment with eNTK eigenfunctions, whereas our approach operates *retrospectively* by quantifying an already converged model's sensitivity to training data perturbations. Therefore, these methods capture different notions of influence: contribution to convergence versus counterfactual perturbation. A consequence of the per-point post-hoc design is that PINNFLUENCE can surface phenomena inaccessible to training-time analyses. Causality-enforcing training schemes (Wang et al., 2024) impose temporal ordering during training; our experiments reveal that well-trained PINNs exhibit temporally symmetric rather than causally ordered influence (Table 2), a finding inaccessible to training-dynamics analyses alone.

A.4. Practitioner's Guide

In order to exemplify the application of PINNFLUENCE, we give a short recipe on how to use and develop diagnostic indicators in this subsection. The general-purpose influence heatmaps and loss-component decompositions are always computable and already constitute the principal diagnostic signal throughout this work (e.g., Figures 4 and 5). Generating new indicators for hypothesis-driven testing does require some domain knowledge, but enables domain-specific questions beyond generic diagnostics. This reliance on human judgment is inherent to most interpretability methods, where practitioners must decide which patterns are meaningful. This is the case for example with image saliency maps in computer vision or token attributions in natural language processing.

We suggest the following recipe. First, (a) compute and inspect the generic influence patterns to identify anomalies. This step does not require domain knowledge. Second, (b) examine whether the observed patterns align with some expected physical behavior of the problem. This can include notions of symmetries, spatial locality, or temporal causality. Third, (c) formalize these hypotheses into indicators for quantitative verification, using constructions as shown in (8) and (9). The prominent example in our work is the temporal influence indicator (9). Further instantiations are provided in Appendix C and can be adapted as templates to new PDEs. In summary, step (a) is problem-agnostic and applicable out of the box. Steps (b) and (c) presuppose some domain knowledge to know what physical behavior to expect and to formalize the hypothesis into an indicator. In the main text, Table 1 summarizes which tool answers which diagnostic question.

A.5. Resampling and Loss-Weighting Based on PINNFLUENCE

In this subsection, we succinctly outline two possible extensions of our framework for active training interventions in PINNs. The diagnostics developed in this work are post-hoc, but the influences they are based upon can in principle also be used to rank training points and loss components by their effect on model predictions or physical observables. This suggests two possible directions for utilizing PINNFLUENCE diagnostics as interventions: resampling and loss-reweighting.

Resampling Adaptive collocation sampling resamples or adds points based on some scoring function of potential training points, conventionally the magnitude of the PDE residual (Lu et al., 2021; Liu et al., 2022; Wu et al., 2023; Lau et al., 2024). The influence scores of PINNFLUENCE constitute another possible scoring criterion. Using (4), one can score current as well as hypothetical training points at any domain location, since it is defined on the whole domain. The most straightforward such criterion is a variation of an absolute point-to-region indicator (see (5) and (8)). Such an influence-guided train–score–resample scheme would complement residual-based sampling.

Loss-weighting The imbalance of loss components is a known PINN failure mode (Wang et al., 2022; 2025). As stated in the main text, the influence is linear in the loss argument (see Corollary 2.3). Scaling a loss weight λ_i thus scales the corresponding influence contribution by the same factor. The fractional loss contributions r_{L_i} (6) can therefore act as both a diagnostic and a control signal. For example, the persistent IC dominance observed in poorly-trained models (see Figure 6) could be addressed by iteratively rebalancing the loss weights during training: downweighting the overly attributed loss term (or equivalently upweighting the remaining terms), turning the diagnostic signal into an actionable reweighting criterion.

B. Problem Specifications

B.1. Investigated Partial Differential Equations

Allen-Cahn Equation: A nonlinear reaction-diffusion equation modeling phase separation in materials which is given by

$$\frac{\partial u}{\partial t} - \epsilon^2 \frac{\partial^2 u}{\partial x^2} + f(u) = 0 \quad . \tag{16}$$

We take $\Omega = (-1, 1) \times (0, 1)$, select $f(u) = -5(u^3 - u)$ and $\epsilon = \sqrt{10^{-4}}$, with the following boundary conditions $u(x, 0) = x^2 \cos(\pi x)$, $u(-1, t) = -1$ and $u(1, t) = -1$.

In the specified setting the reaction term $-5(u^3 - u) = -5u(u^2 - 1)$ has two stable wells at $u = \pm 1$ for regions with negligible diffusion. The setup drives negative u to -1 and positive to 1, respectively. Note however that the reaction term may also collapse to 0 for $u = 0$, allowing for low PDE residuals with an incorrect solution. We obtained reference values from (Wu et al., 2023) shown in Figure 7.

Burgers' Equation: A second-order nonlinear PDE that models phenomena combining convection and diffusion, often used as a simplified model for fluid dynamics or traffic flow. It is given by

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} = 0 \quad . \tag{17}$$

We take $\Omega = (-1, 1) \times (0, 1)$ and set the viscosity to $\nu = \frac{10^{-3}}{\pi}$, with the following boundary conditions $u(x, 0) = -\sin(\pi x)$, $u(-1, t) = 0$ and $u(1, t) = 0$. In this setup it develops a sharp shock around $x = 0$. We obtained reference values from (Wu et al., 2023) shown in Figure 8.

Drift-Diffusion Equation: A second-order linear PDE describing diffusion with coefficient α with given drift β

$$\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} + \beta \frac{\partial u}{\partial x} = 0 \tag{18}$$

We take $\Omega = (0, 2\pi) \times (0, 1)$ and set $\alpha = 1, \beta = 20$, with the following initial and boundary conditions $u(x, 0) = \sin\left(2x + \frac{\pi}{4}\right)$, $u(0, t) = \sin\left(\frac{\pi}{4} - 2 \cdot \beta t\right)\ e^{-4\alpha t}$ and $u(2\pi, t) = \sin\left(\frac{17\pi}{4} - 2\beta t\right)\ e^{-4\alpha t}$. This yields the closed form solution

$$u(x, t) = \sin\left(2x - 2\beta t + \frac{\pi}{4}\right)\ e^{-4\alpha t} \ . \tag{19}$$

A similar problem was found to cause propagation failures in PINNs (Krishnapriyan et al., 2021), where the PINN fails to generalize over the domain, which motivated the inclusion of this example as a prominent problem in the main text. We provide a visualization in Figure 9.

Heat Equation: A second-order linear PDE for modeling diffusion

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - s(x) = 0 \quad , \tag{20}$$

where $s : \Omega \to \mathbb{R}$ denotes a source term, which we set to $s(x, t) := (\pi^2 - 1) \sin(\pi x) e^{-t}$.

We chose $\Omega = (-1, 1) \times (0, 1)$, with the following boundary conditions $u(x, 0) = \sin(\pi x)$, $u(-1, t) = 0$ and $u(1, t) = 0$. This yields the closed form solution (Wu et al., 2023).

$$u(x, t) = \sin(\pi x) e^{-t} \tag{21}$$

The heat equation is a considerably simple PDE for basic benchmarks. See Figure 10 for visualization.

Wave Equation: A second-order linear PDE describing the propagation of waves (e.g., vibrations, acoustics), given by

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = 0 \tag{22}$$

We take $\Omega = (0,1) \times (0,1)$, set $c = 2$, with the following boundary conditions $u(x,0) = \sin(\pi x) + \frac{1}{2}\sin(4\pi x)$, $\frac{\partial u}{\partial t}(x,0) = 0$, $u(0,t) = 0$ and $u(1,t) = 0$. This yields the following closed form solution

$$u(x,t) = \sin(\pi x)\cos(\pi c t) + \frac{1}{2}\sin(4\pi x)\cos(4\pi c t). \tag{23}$$

This IBVP yields antisymmetric patterns as well as high and low frequency changes in the solution shown in Figure 11.

Navier-Stokes Equations: A system of second-order nonlinear PDEs describing the motion of viscous fluids, such as water or air. We follow the formulation from the DFG flow around cylinder benchmark in the laminar case with the Reynolds number being set to 20 (Schäfer et al., 1996). This results in an incompressible and steady instance of the Navier-Stokes equations, which are given in this case as

- x-Velocity: $u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{1}{\rho}\frac{\partial p}{\partial x} - \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0$
- y-Velocity: $u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{1}{\rho}\frac{\partial p}{\partial y} - \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0$
- Continuity: $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$

where $(u,v) \in \mathbb{R}^2$ denotes the velocity field, $p \in \mathbb{R}$ the pressure, ρ the fluid density and ν the kinematic viscosity. A Reynolds number of 20 induces $\rho = 1$ and $\nu = \frac{1}{1000}$. We solve the non-dimensionalized form of these equations, scaling coordinates by characteristic length $L^* = 0.1$ and velocities by $U^* = 0.2$. The domain is set to $\Omega = (0, 2.2) \times (0, 0.41) \setminus C_r(0.2, 0.2)$, where C_r denotes a cylinder with center at $(0.2, 0.2)$ and radius $r = 0.05$. The boundary conditions are given as follows, the inflow is characterized by a parabola $u(0,y) = \left(\frac{4Uy(0.41-y)}{0.41^2}, 0\right)$ where $U = 0.3$ denotes the maximum inflow velocity. As boundary conditions for the lower and upper walls $\Gamma_{\text{bottom}} = (0, 2.2) \times \{0\}$ and $\Gamma_{\text{top}} = (0, 2.2) \times \{0.41\}$ as well as on the cylinder boundary $\Gamma_{\text{cylinder}} = \partial C_r(0.2, 0.2)$, the predicted x and y-velocity should be 0.

Furthermore, for the outflow on the right edge, i.e., $\Gamma_{\text{outflow}} = \{2.2\} \times (0, 0.41)$ do-nothing boundary conditions are defined, i.e., $\nu\frac{\partial u}{\partial x} - p$ and $\nu\frac{\partial v}{\partial y}$ should be 0.

Target values are obtained from (Wang et al., 2023) and are visualized in Figure 12.

Poisson Equation with a point source over a unit disk: The Poisson equation models the electrostatic potential induced by an electric point charge and is given by

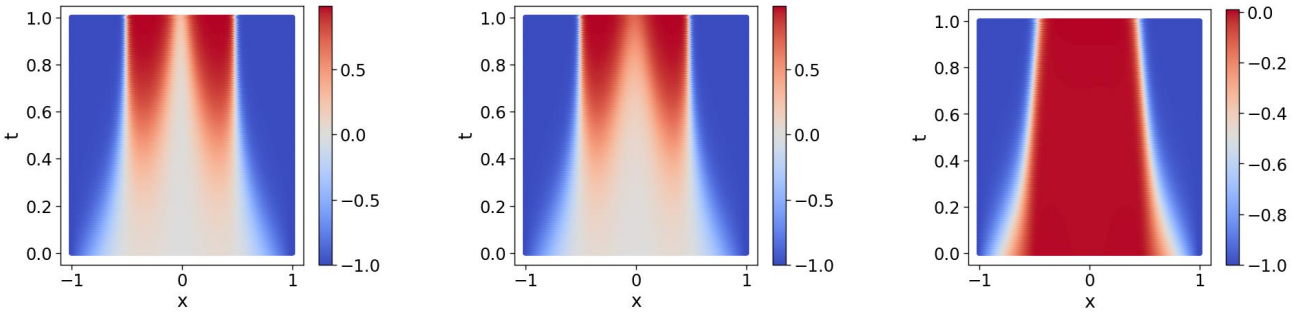
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \delta_\sigma(x) = 0 \quad , \tag{24}$$

where δ_σ denotes a Gaussian approximation of the Dirac delta with bandwidth $\sigma = 0.02$. The domain we consider is the unit disk $\Omega = \{x \in \mathbb{R}^2 : \|x\|_2 < 1\}$ with a unit point source at the origin $c = (0,0)$. The analytical solution corresponding to the Dirac source (recovered in the limit $\sigma \to 0$) is radially symmetric and is given as

$$u(r) = -\frac{1}{2\pi}\log r \quad , \tag{25}$$

where $r = \|x\|_2$ with $0 < r \leq 1$. We specify the following boundary condition $u(x) = 0$ for $\|x\| = 1$.

This problem yields a difficult to learn singularity and exhibits radial symmetries, which result in an interesting benchmark problem. We provide a visualization in Figure 13.



(a) Targets *(b)* Well-trained *(c)* Poorly-trained

Figure 7. Allen-Cahn equation: target values and predictions for well and poorly-trained configurations averaged over 10 seeds.

B.2. Training Setup

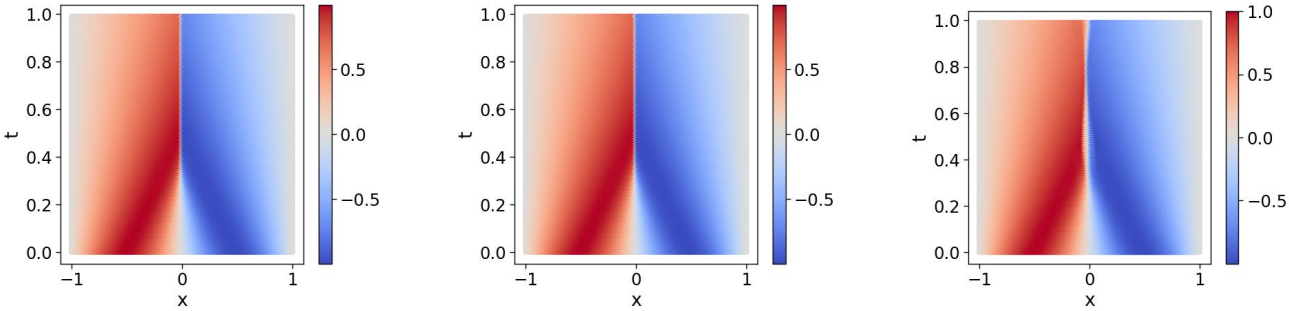
All models have been trained with the DeepXDE library (Lu et al., 2021), including the geometry generation and sampling, with PyTorch (Paszke et al., 2019) used as its backend. Optimization was carried out in two stages, first with the Adam optimizer (Kingma & Ba, 2017) followed by fine-tuning with the second-order L-BFGS optimizer (Liu & Nocedal, 1989). Note that to construct some of the poorly-trained models, we deliberately skipped the L-BFGS optimization step. Further, we note that for the Allen-Cahn well-trained configuration a minority of seeds collapse to the spurious constant solution $u \equiv 0$ in the domain interior despite the well-trained protocol (see the reaction term discussion in Appendix B.1). We therefore report the well-trained Allen-Cahn statistics over seeds that converged to the correct solution.

The hyperparameters for each configuration are presented in Table 11. All models were trained as simple feed-forward neural networks with equal number of neurons per hidden layer and $\tanh$ activations.

Table 11. Hyperparameters for well-trained and poorly-trained configurations. Layers and neurons specify depth and width of the feed-forward neural networks; N_{domain}, N_{bc}, N_{ic} specify numbers of sampled points inside the domain, on boundaries and on the initial time slice; $N_{\text{iter}}^{(\text{Adam})}$ and $N_{\text{iter}}^{(\text{L-BFGS})}$ specify the number of optimization steps using the respective optimizer.

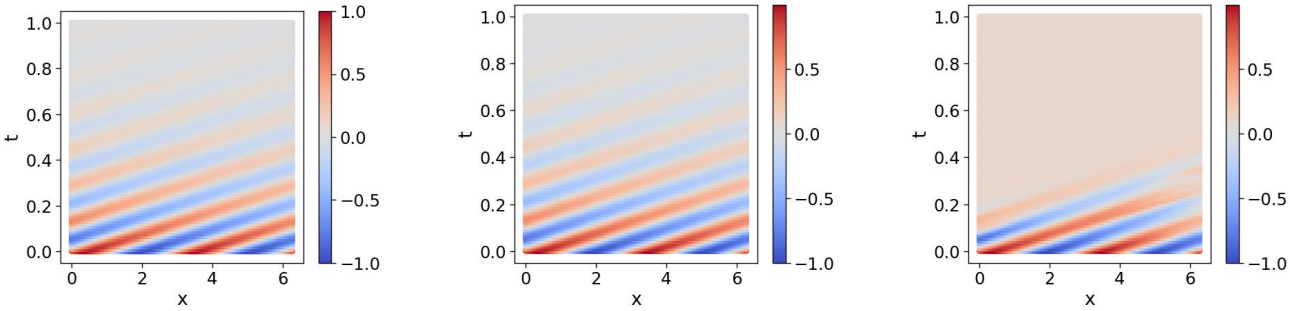
Problem	Config	Layers	Neurons	N_{domain}	N_{bc}	N_{ic}	$N_{\text{iter}}^{(\text{Adam})}$	$N_{\text{iter}}^{(\text{L-BFGS})}$
Allen-Cahn	Well-trained	3	64	2 500	500	500	100 000	25 000
	Poorly-trained	3	64	2 500	500	500	100 000	—
Burgers'	Well-trained	3	32	2 500	500	500	50 000	12 000
	Poorly-trained	3	32	500	100	100	50 000	12 000
Drift-Diffusion	Well-trained	3	64	1 000	100	100	15 000	5 000
	Poorly-trained	3	64	200	20	20	15 000	5 000
Heat	Well-trained	3	32	1 000	100	100	15 000	5 000
	Poorly-trained	3	32	10	2	2	15 000	5 000
Wave	Well-trained	5	100	2 500	500	500	100 000	25 000
	Poorly-trained	5	100	2 500	500	500	100 000	—
Navier-Stokes	Well-trained	3	64	7 500	2 500	—	100 000	25 000
	Poorly-trained	3	64	1 500	500	—	100 000	25 000
Poisson	Well-trained	3	32	2 500	500	—	50 000	12 000
	Poorly-trained	3	32	100	20	—	15 000	5 000

B.3. Model Predictions and Targets



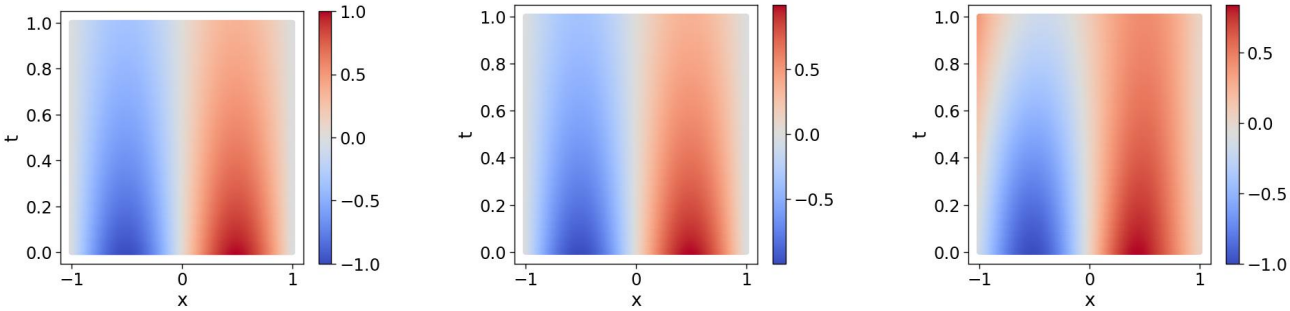
(a) Targets *(b)* Well-trained *(c)* Poorly-trained

Figure 8. Burgers' equation: target values and predictions for well and poorly-trained configurations averaged over 10 seeds.



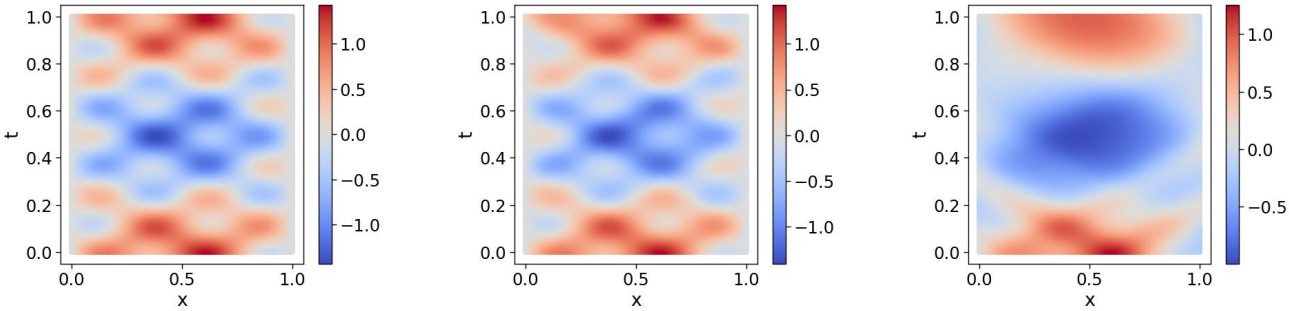
(a) Targets *(b)* Well-trained *(c)* Poorly-trained

Figure 9. Drift-diffusion equation: target values and predictions for well and poorly-trained configurations averaged over 10 seeds.



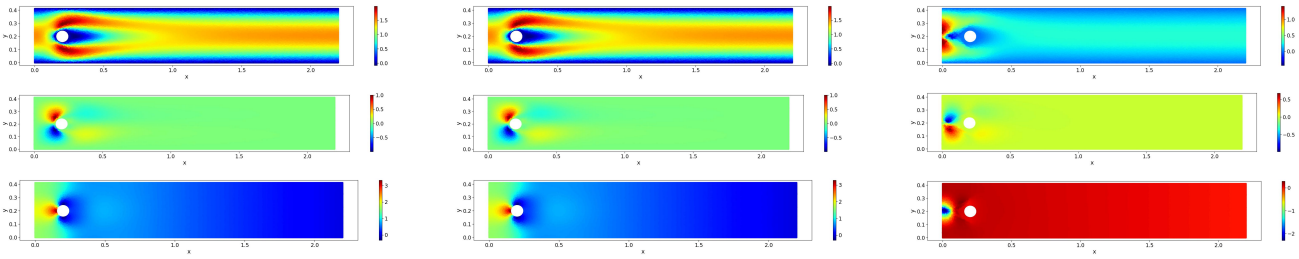
(a) Targets *(b)* Well-trained *(c)* Poorly-trained

Figure 10. Heat equation: target values and predictions for well and poorly-trained configurations averaged over 10 seeds.



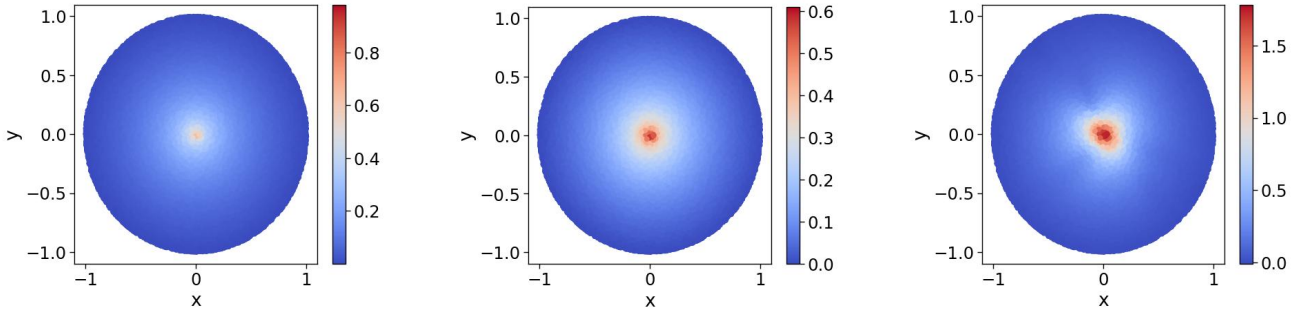
(a) Targets *(b)* Well-trained *(c)* Poorly-trained

Figure 11. Wave equation: target values and predictions for well and poorly-trained configurations averaged over 10 seeds.



(a) Target *(b)* Well-trained *(c)* Poorly-trained

Figure 12. Navier-Stokes equations: target values and predictions for x-velocity u (top row), y-velocity v (middle row), and pressure p (bottom row). Predictions averaged over 10 seeds.



(a) Targets *(b)* Well-trained *(c)* Poorly-trained

Figure 13. Poisson equation: target values and predictions for well and poorly-trained configurations averaged over 10 seeds.

Table 12. L_2 relative errors across all configurations. Reported as mean $\pm$ standard deviation $\times$ order of magnitude across all 10 seeds.

Problem	Well-trained	Poorly-trained
Heat	$(2.8 \pm 1.3) \times 10^{-4}$	$(2.0 \pm 0.4) \times 10^{-1}$
Allen-Cahn	$(6.3 \pm 3.4) \times 10^{-2}$	$(5.3 \pm 0.01) \times 10^{-1}$
Burgers'	$(5.6 \pm 3.4) \times 10^{-3}$	$(1.8 \pm 0.7) \times 10^{-1}$
Drift-Diffusion	$(8.6 \pm 9.3) \times 10^{-3}$	$(8.6 \pm 4.3) \times 10^{-1}$
Wave	$(6.8 \pm 4.7) \times 10^{-2}$	$(3.8 \pm 0.1) \times 10^{-1}$
Poisson	$(2.9 \pm 0.2) \times 10^{-2}$	$(1.9 \pm 0.7) \times 10^{0}$
Navier-Stokes (x-Velocity)	$(3.4 \pm 0.8) \times 10^{-3}$	$(7.8 \pm 1.2) \times 10^{-1}$
Navier-Stokes (y-Velocity)	$(1.2 \pm 0.2) \times 10^{-2}$	$(1.4 \pm 0.3) \times 10^{0}$
Navier-Stokes (Pressure)	$(8.3 \pm 1.3) \times 10^{-3}$	$(9.7 \pm 1.7) \times 10^{-1}$

C. Additional Investigations

C.1. Reframing the Temporal Indicator for Steady-State Problems as a Measure of Directionality

The causality indicator can be extended to the featured steady-state problems, replacing temporal influence with directional influence, appropriate to the problem structure.

For the Navier-Stokes equation, we consider the inflow boundary condition analogous to an initial condition. Here, the flow propagates along the x-axis instead. This results in the analogous indicator to (9) with a different spatial coordinate

$$\eta_{\theta_0}^{L\to f}(R_{\text{tr}}, R_{\text{te}}) = 1 - \frac{1}{|R_{\text{te}}|} \sum_{z\in R_{\text{te}}} \frac{\sum_{x\in R_{\text{tr}}: x_x \le z_x} \left|\text{Inf}_{\theta_0}^{L\to f}(x,z)\right|}{\sum_{x\in\mathcal{X}} \left|\text{Inf}_{\theta_0}^{L\to f}(x,z)\right|}. \tag{26}$$

For the Poisson equation, we consider the problem stated at a disk domain, featuring a singularity in the center acting as a source. The directional indicator measures influence from training points closer to the center, relative to the total influence

$$\eta_{\theta_0}^{L\to f}(R_{\text{tr}}, R_{\text{te}}) = 1 - \frac{1}{|R_{\text{te}}|} \sum_{z\in R_{\text{te}}} \frac{\sum_{x\in R_{\text{tr}}: \|x\| \le \|z\|} \left|\text{Inf}_{\theta_0}^{L\to f}(x,z)\right|}{\sum_{x\in\mathcal{X}} \left|\text{Inf}_{\theta_0}^{L\to f}(x,z)\right|}. \tag{27}$$

Table 13. Spatial directionality indicator for steady-state problems, evaluated for predictions $f = \hat{u}$. We report mean $\pm$ standard deviation over 10 seeds.

Problem	f	$\bar{x}$	Well-trained	Poorly-trained
Poisson	$\hat{u}$	0.28	0.29 ± 0.03	0.69 ± 0.06
Navier-Stokes	x-velocity	0.48	0.15 ± 0.01	0.15 ± 0.03
	y-velocity	0.48	0.25 ± 0.01	0.2 ± 0.05
	Pressure	0.48	0.14 ± 0.02	0.11 ± 0.02

Table 13 reports directionality indicators. For the well-trained Poisson model, the indicator matches the expected baseline. For the poorly-trained model, the indicator is far above the baseline. Contrary to the time-dependent problems, the model is heavily influenced by points near the boundary. This represents the inverse of the time-dependent pathology: rather than over-relying on the source term the poorly-trained model completely neglects it. Note that it does not feature a separate loss term and is instead modeled solely through the PDE residual.

For the Navier-Stokes problem, we consider steady flow around an obstacle close to the inflow. We find that the directionality indicator shows no separation between the model configurations. This is because influences are concentrated around the obstacle, which is the defining structure of the flow, rather than distributed along the flow direction. The measured indicator values confirm this, as they closely match the ratio of training points downstream versus upstream of the obstacle.

C.2. Additional Indicators

In this section we demonstrate how one may construct problem-specific indicators using PINNFLUENCE to specifically probe individual model characteristics.

Navier-Stokes: Downstream Influence on Individual Regions The region-based influence indicator $\rho_{\theta_0}^{L\to f}$ (8) allows us to quantify the influence of one region in the training set R_{tr} onto another region in the test set R_{test}. We will exemplify this on the Navier-Stokes problem.

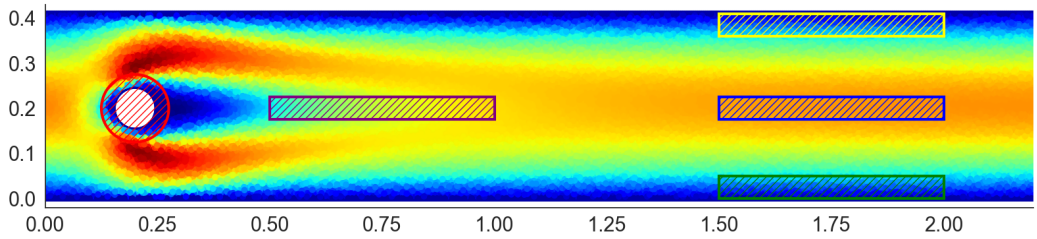


Figure 14. Regions of interest to quantify the influence of the region surrounding the cylinder on downstream locations: (i) closely behind the cylinder; (ii) close to the outflow, centered; (iii) close to the outflow, top; (iv) close to the outflow, bottom.

Table 14. Influence scores from cylinder region for different loss terms and spatial regions. We report mean $\pm$ standard deviation over 10 seeds.

f	Configuration	Center-Rectangle	Outflow-Rectangle	Top-Rectangle	Bottom-Rectangle
x-Velocity	Well-trained	0.43 ± 0.05	0.38 ± 0.05	0.33 ± 0.05	0.36 ± 0.05
	Poorly-trained	0.31 ± 0.09	0.27 ± 0.09	0.26 ± 0.09	0.25 ± 0.09
y-Velocity	Well-trained	0.27 ± 0.04	0.25 ± 0.03	0.22 ± 0.03	0.23 ± 0.02
	Poorly-trained	0.24 ± 0.08	0.23 ± 0.08	0.23 ± 0.09	0.21 ± 0.09
Pressure	Well-trained	0.48 ± 0.07	0.47 ± 0.07	0.47 ± 0.07	0.46 ± 0.07
	Poorly-trained	0.25 ± 0.06	0.25 ± 0.06	0.24 ± 0.06	0.26 ± 0.06
Loss	Well-trained	0.32 ± 0.03	0.28 ± 0.04	0.24 ± 0.03	0.25 ± 0.02
	Poorly-trained	0.26 ± 0.08	0.23 ± 0.08	0.23 ± 0.09	0.22 ± 0.06

The results presented in Table 14 confirm physical expectations: excluding the predictions for pressure, the influence of the cylinder is strongest in the rectangle directly downstream and diminishes toward the outflow. Top and bottom regions show similar values, consistent with the flow's approximate symmetry. Note, though, that the cylinder is not perfectly centered, hence slight asymmetries are expected. Well-trained models exhibit consistently higher influences than poorly-trained ones, most strikingly for pressure, and lower variance across seeds.

Wave Equation: Symmetry Indicator We test for symmetry in influences for the wave equation by comparing the influence of the left spatial half ($x < 0.5$) onto the right ($x \geq 0.5$) and vice versa.

Table 15. Spatial symmetry indicators for the Wave equation. Values represent the fraction of influence from one spatial half to the opposite half. The ratio quantifies symmetry (values near 1.0 indicate symmetric influence patterns). We report mean ± standard deviation over 10 seeds.

f	Configuration	$\rho_{\theta_0}^{L\to f}(R_{\text{left}}, R_{\text{right}})$	$\rho_{\theta_0}^{L\to f}(R_{\text{right}}, R_{\text{left}})$	$\frac{\rho_{\theta_0}^{L\to f}(R_{\text{right}}, R_{\text{left}})}{\rho_{\theta_0}^{L\to f}(R_{\text{left}}, R_{\text{right}})}$
$\hat{u}$	Well-trained	0.48 ± 0.04	0.49 ± 0.05	1.01 ± 0.17
	Poorly-trained	0.48 ± 0.04	0.51 ± 0.04	0.96 ± 0.14
Loss	Well-trained	0.35 ± 0.03	0.36 ± 0.04	0.98 ± 0.18
	Poorly-trained	0.38 ± 0.02	0.40 ± 0.02	0.95 ± 0.08

As shown in Table 15, both configurations display symmetric influence patterns with ratios close to 1. Notably, the influence fractions of 0.48 to 0.51 indicate that predictions in each half are nearly equally shaped by training points from the opposite half, consistent with perfect symmetry. Although the poorly-trained model fails to capture high-frequency features in the solution, these results suggest that it still accurately learns the problem's spatial symmetry. The loss decomposition, however, clearly separates well-trained from poorly-trained models, see Appendix C.5.

Burgers' Equation: Locality of the Shock Burgers' equation develops a sharp shock around $x = 0$. We test how locally distributed influences are around the shock region by computing the region-to-region influence from training points within $x \in [-0.2, 0.2]$ onto test points within the same region.

Table 16. Shock locality indicator for Burgers' equation. Values represent the fraction of influence on shock region predictions (x coordinates in $[-0.2, 0.2]$) originating from training points within the same region. We report mean ± standard deviation over 10 seeds.

f	Configuration	$\rho_{\theta_0}^{L\to f}(R_{\text{shock}}, R_{\text{shock}})$
$\hat{u}$	Well-trained	0.66 ± 0.03
	Poorly-trained	0.38 ± 0.06
Loss	Well-trained	0.71 ± 0.02
	Poorly-trained	0.47 ± 0.05

Table 16 indicates that well-trained models show strong locality, while poorly-trained models exhibit weaker locality, indicating over-sensitivity to regions irrelevant to the shock.

C.3. Continuous Sweep over Training-Data Quantity

The evaluation of PINNFLUENCE on controlled well- and poorly-trained models shows that the method distinguishes accurately between them. To show that PINNFLUENCE's regional indicators respond to *gradual* variation in training quality rather than as a binary artifact of two hand-picked configurations, we additionally sweep the fraction of domain training points continuously, holding the well-trained architecture and optimizer fixed (Table 11). IC/BC points scale proportionally with the domain count to preserve the 1:5 or 1:10 ratio used throughout each configuration. We aggregate indicator values by the mean $\pm$ standard deviation over 10 seeds.

Shock proximity on Burgers' The shock-proximity indicator $\rho^{L\to\hat{u}}(R_s, R_s)$ with $R_s = [-0.2, 0.2]$ (Appendix C.2) rises near-monotonically with domain points (Table 17), recovering the gap between the well- and poorly-trained configurations of Table 16 without reference to a held-out ground truth.

Table 17. Burgers': Shock-proximity indicator across domain-point counts (mean $\pm$ std over 10 seeds).

Domain pts	250	750	1250	1750	2500
$\rho^{L\to\hat{u}}(R_s, R_s)$	0.24 ± 0.04	0.50 ± 0.03	0.60 ± 0.03	0.65 ± 0.03	0.66 ± 0.03

Temporal indicator across PDEs We additionally sweep over the temporal influence indicator (see (9)) for cross-PDE comparison over all time-dependent problems, with the results given in Table 18. This indicator quantifies the extent to which predictions are influenced by training points at earlier times; low values flag over-reliance on early-time data. The pattern shared by Burgers', Allen-Cahn, Heat, and Drift-Diffusion is a recovery of the indicator values from depressed low-fraction values toward the well-trained reference (Table 2), with the transition occurring at a PDE-specific data scale that is itself diagnostic of each problem's data requirements. Wave is the exception and is discussed separately below.

Table 18. Temporal influence indicator across PDEs and training-data fractions (mean $\pm$ std over 10 seeds). Full sets contain 2500 domain points (Allen-Cahn, Burgers', Wave) or 1000 (Heat, Drift-Diffusion) and IC/BC points sampled at respective ratios matching Table 11.

Fraction	Allen-Cahn	Burgers'	Heat	Drift-Diff.	Wave
0.1	0.41 ± 0.05	0.25 ± 0.02	0.29 ± 0.03	0.28 ± 0.09	0.40 ± 0.05
0.3	0.47 ± 0.02	0.32 ± 0.02	0.31 ± 0.02	0.45 ± 0.08	0.41 ± 0.06
0.5	0.48 ± 0.03	0.35 ± 0.01	0.31 ± 0.02	0.52 ± 0.03	0.41 ± 0.07
0.7	0.48 ± 0.04	0.38 ± 0.03	0.31 ± 0.03	0.50 ± 0.02	0.40 ± 0.07
1.0	0.50 ± 0.02	0.40 ± 0.02	0.32 ± 0.03	0.47 ± 0.02	0.40 ± 0.07

Burgers' mirrors the shock-proximity sweep (0.25–0.40). **Allen-Cahn** saturates already near fraction 0.3, and even the lowest fraction exceeds the poorly-trained value of 0.32 (Table 2), since that configuration differs in optimizer choice rather than data quantity (Table 11). **Heat** varies only mildly across the swept range (0.29–0.32): the equation needs substantially fewer points to be learned and only breaks down at very low data. The poorly-trained Heat row in Table 11 corresponds to fraction 0.01 with an indicator value of 0.26 (Table 2), matching the depressed regime seen at low fractions in the other PDEs. **Drift-Diffusion** is non-monotone, peaking near fraction 0.5 before mildly declining at full data: low values at low fractions are consistent with the information-propagation failure discussed in Section 3, while at high fractions influence distributes uniformly across the temporal domain and saturates the indicator. **Wave** departs from this pattern: the indicator stays essentially flat (≈ 0.40) across the swept range and shows none of the low-fraction depression seen elsewhere. It therefore does not track training quality monotonically, and the monotone recovery reading should not be carried over to it from the other PDEs.

Taken together, the two sweeps show that PINNFLUENCE's regional indicators respond to training quality in a PDE-dependent fashion: monotonically for Burgers', with early saturation, non-monotonicity, or (for Wave) no monotone trend in the others. Additionally, we observe that the form of this response itself encodes PDE-specific information about each problem's data requirements.

C.4. Alternative Optimizers

A natural concern is whether the diagnostics produced by PINNFLUENCE reflect genuine properties of the converged model or are an artifact of the optimizer used to reach it. Because PINNFLUENCE operates post-hoc on a stationary point, its outputs should depend on where training converged rather than on how it got there. To show this, we additionally provide evaluations of models trained with curvature-aware optimizers.

We consider NysNewton-CG (NNCG) (Rathore et al., 2024) and SOAP (Vyas et al., 2025). NNCG is a second-order method tailored to ill-conditioned PINN landscapes. SOAP instead utilizes Adam (Kingma & Ba, 2017) with the Shampoo preconditioner (Gupta et al., 2018). Recent work (Wang et al., 2025) has shown that this optimizer excels at resolving gradient-conflicts between individual loss terms (PDE, IC, BCs) present in PINN training, hence achieving more stable training in comparison to Adam.

Setup For the alternative optimizers, we adopt the well-trained configurations' training points and network setups of Table 11. The SOAP experiments replace Adam+L-BFGS completely and are trained from scratch. We use $N_{\text{iter}}^{(\text{SOAP})} = N_{\text{iter}}^{(\text{Adam})}$. For the NNCG experiments we take the well-trained models and fine-tune them additionally with $1\,000$ steps of NNCG[3].

Results Figure 15 shows prediction plots across all problems for both SOAP and NNCG trainings. Almost all configurations achieve good results, the one exception being the Wave equation with the SOAP optimizer, where similar patterns arise as in the poorly-trained configuration.

Table 19 reports the resulting temporal (and spatial) influence indicator values (see Tables 2 and 13 for well- and poorly-trained values). NNCG closely reproduces the well-trained Adam+L-BFGS reference across every evaluated problem. This is expected as NNCG fine-tunes already trained models. SOAP, however, is trained from scratch and thus, we assume, converges to meaningfully different optima. It matches the well-trained configuration only for Allen-Cahn and Burgers' equations. For the Heat, Drift-Diffusion, and Poisson equations, it produces accurate solutions whose indicator values nonetheless fall well below the well-trained reference. For the Wave equation, SOAP fails to solve the problem, exhibiting both high error and low indicator values, mirroring the poorly-trained configuration.

These accurate-but-low-indicator SOAP solutions isolate what the indicator captures: the data dependence of the converged model, not the accuracy of its solution. The Drift-Diffusion case illustrates this. SOAP reaches an accurate solution whose influence profile nonetheless leans more heavily on the initial condition than the Adam+L-BFGS well-trained model (Figure 16d), plausibly reflecting SOAP's IC-weighted resolution of inter-component gradient conflicts (Wang et al., 2025). This underscores how PINNFLUENCE characterizes the converged model: the indicator describes the relationship of model behavior with respect to its data and does not constitute an optimizer-independent measure of solution quality.

Finally, for Navier-Stokes the loss-component cancellations are large under both optimizers (Figure 16g), so the corresponding fractions should be read with care.

The Hessian condition numbers (Table 20) and the inverse-Hessian gradient cosine similarities (Table 21) are comparable to those of the Adam+L-BFGS runs, indicating that the approximation underlying PINNFLUENCE is equally reliable for these optimizers.

[3] As NNCG runs extremely slowly with default hyperparameters, Wave and Navier-Stokes use lower tolerances for the Hessian-approximation in NNCG optimization.

Table 19. Temporal (and spatial, for the steady-state problems) influence indicator for models trained with the SOAP and NNCG optimizers, computed with respect to predictions $f = \hat{u}$, alongside the relative L_2 error of each configuration. For the steady-state Poisson and Navier-Stokes problems the indicator and baseline are spatial rather than temporal (cf. Table 13); the baseline column reports $\bar{t}$ for the time-dependent problems and $\bar{x}$ for the steady-state ones. Well- and poorly-trained reference values are given in Tables 2 and 13, and the corresponding reference L_2 errors in Table 12. We report mean $\pm$ standard deviation over 10 seeds.

		SOAP		NNCG	
Problem	Baseline	Indicator	L_2 Rel. Error	Indicator	L_2 Rel. Error
Heat	0.46	0.25 ± 0.02	$(1.4 \pm 0.7) \times 10^{-3}$	0.32 ± 0.02	$(2.6 \pm 1.0) \times 10^{-4}$
Allen-Cahn	0.43	0.52 ± 0.05	$(3.3 \pm 0.3) \times 10^{-2}$	0.50 ± 0.02	$(6.2 \pm 3.3) \times 10^{-2}$
Burgers'	0.43	0.45 ± 0.04	$(1.7 \pm 1.1) \times 10^{-2}$	0.40 ± 0.02	$(5.5 \pm 3.3) \times 10^{-3}$
Drift-Diffusion	0.46	0.25 ± 0.04	$(2.5 \pm 1.1) \times 10^{-3}$	0.47 ± 0.04	$(5.6 \pm 6.8) \times 10^{-3}$
Wave	0.43	0.16 ± 0.02	$(3.8 \pm 0.1) \times 10^{-1}$	0.45 ± 0.03	$(1.3 \pm 0.5) \times 10^{-2}$
Poisson	0.28	0.16 ± 0.02	$(3.1 \pm 0.4) \times 10^{-2}$	0.29 ± 0.03	$(3.0 \pm 0.2) \times 10^{-2}$
Navier-Stokes (x-Velocity)	0.48	0.16 ± 0.03	$(6.2 \pm 0.6) \times 10^{-2}$	0.15 ± 0.01	$(2.7 \pm 1.3) \times 10^{-3}$

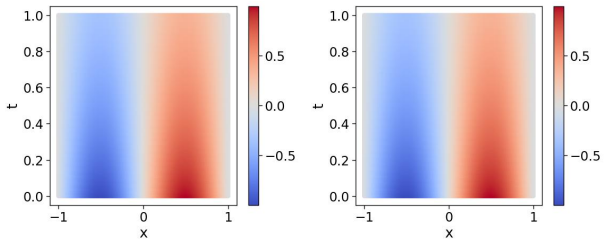
Table 20. Mean condition numbers of the Hessian $\mathcal{H}_{\theta_0}$ at three stages of the inverse approximation: the exact (dense) Hessian, the damped Hessian $\mathcal{H}_{\theta_0} + \lambda I$ with $\lambda = 10^{-3}$, and the top-k subspace retained by the Arnoldi iteration. Wave is omitted due to computational costs.

	Exact $\mathcal{H}_{\theta_0}$		Damped $\mathcal{H}_{\theta_0} + \lambda I$		Arnoldi top-k	
Problem	SOAP	NNCG	SOAP	NNCG	SOAP	NNCG
Heat	3.1×10^{21}	6.4×10^{20}	9.8×10^{10}	1.4×10^{11}	8.3×10^{4}	8.8×10^{5}
Allen-Cahn	2.5×10^{24}	1.7×10^{25}	1.4×10^{13}	2.8×10^{13}	4.6×10^{3}	2.1×10^{4}
Burgers'	1.4×10^{22}	4.4×10^{21}	5.6×10^{11}	8.0×10^{11}	2.2×10^{3}	5.1×10^{3}
Drift-Diffusion	8.8×10^{23}	2.8×10^{23}	3.0×10^{11}	4.8×10^{12}	1.2×10^{3}	1.2×10^{3}
Poisson	7.7×10^{20}	1.9×10^{21}	4.5×10^{12}	4.2×10^{13}	2.0×10^{2}	3.1×10^{2}
Navier-Stokes	3.3×10^{22}	3.0×10^{27}	4.4×10^{13}	6.5×10^{13}	3.5×10^{2}	5.7×10^{2}

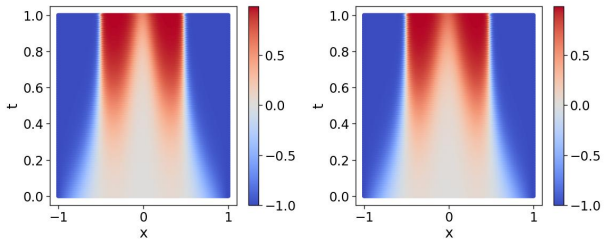
Table 21. Cosine similarities between reconstructed $\mathcal{H}_{\theta_0} R R^\top g$ and true gradients g averaged over the whole training set $\mathcal{X}_{\text{train}}$ and over 10 runs (mean $\pm$ std). NNCG Wave is omitted due to computational costs.

	$\nabla_\theta L$		$\nabla_\theta f$	
	SOAP	NNCG	SOAP	NNCG
Problem				
Heat	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
Allen-Cahn	1.00 ± 0.00	1.00 ± 0.00	0.97 ± 0.05	0.99 ± 0.03
Burgers'	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.00	1.00 ± 0.01
Drift-Diffusion	0.99 ± 0.01	1.00 ± 0.01	0.98 ± 0.02	0.99 ± 0.01
Wave	1.00 ± 0.00	—	0.97 ± 0.02	—
Poisson	0.95 ± 0.05	0.96 ± 0.04	0.92 ± 0.01	0.93 ± 0.01
Navier-Stokes$^{(*)}$	0.92 ± 0.08	0.86 ± 0.14	0.87 ± 0.06	0.88 ± 0.06

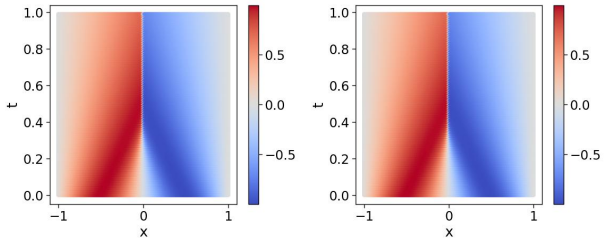
$^{(*)}$ For Navier-Stokes f is u, i.e., the predicted velocity in x direction.



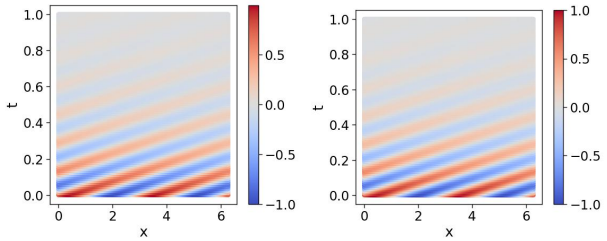
(a) Heat



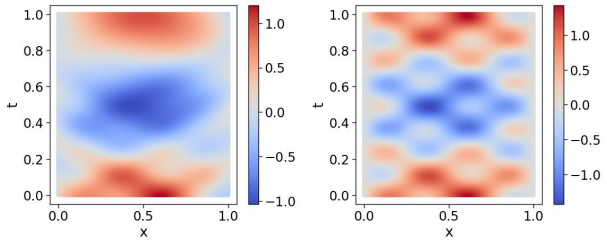
(b) Allen-Cahn



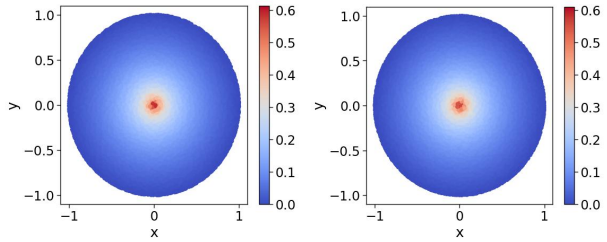
(c) Burgers'



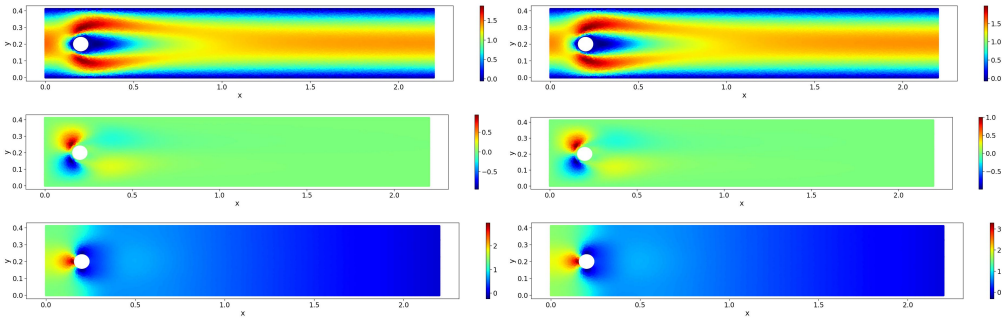
(d) Drift-Diffusion



(e) Wave

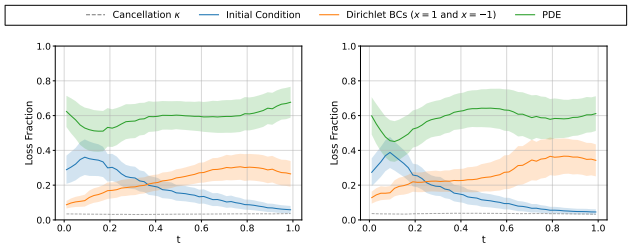


(f) Poisson

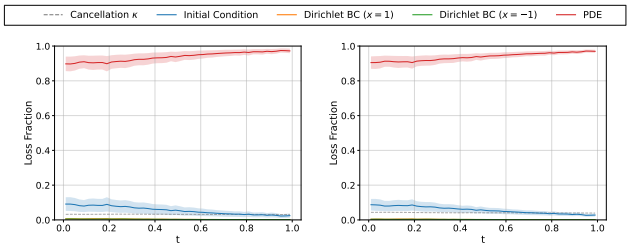


(g) Navier-Stokes (u, v, p top to bottom)

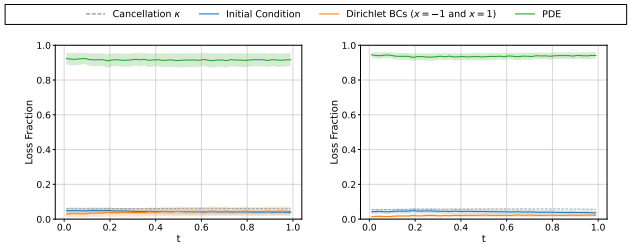
Figure 15. Predictions for SOAP (left) and NNCG (right) across all PDEs.



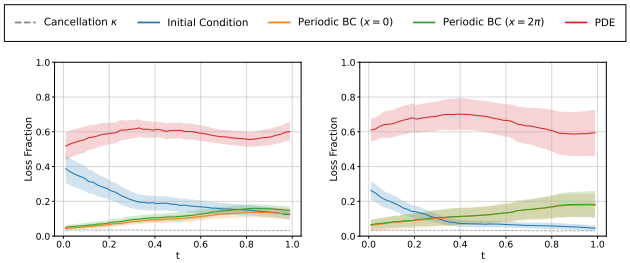
(a) Heat



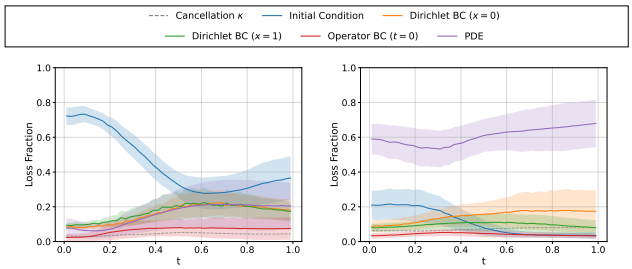
(b) Allen-Cahn



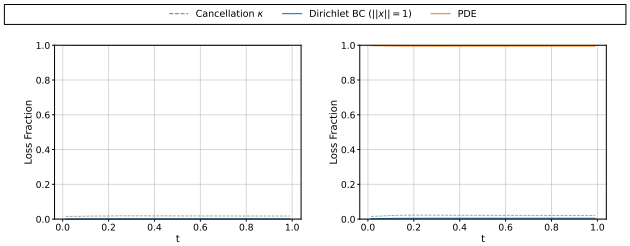
(c) Burgers'



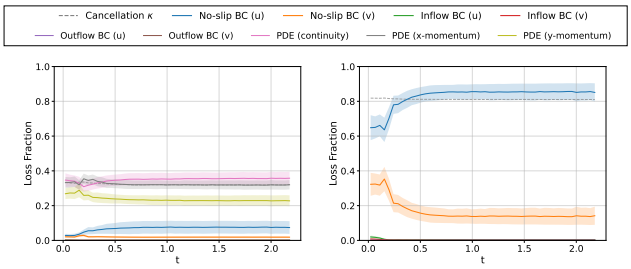
(d) Drift-Diffusion



(e) Wave



(f) Poisson

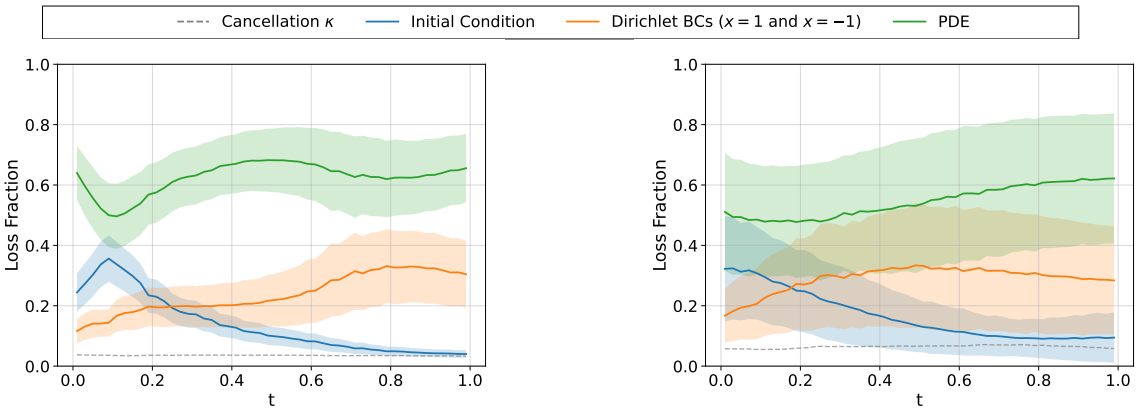


(g) Navier-Stokes

Figure 16. Loss fractions $\bar{r}_{L_i}(\mathcal{X}, R_{\text{test}})$ for the curvature-aware optimizers SOAP (left) and NNCG (right) across all PDEs. Influences are binned along the relevant coordinate (time for the time-dependent PDEs, distance from the center for Poisson, and the x-coordinate for Navier-Stokes) and averaged over seeds; shaded regions indicate the standard deviation.

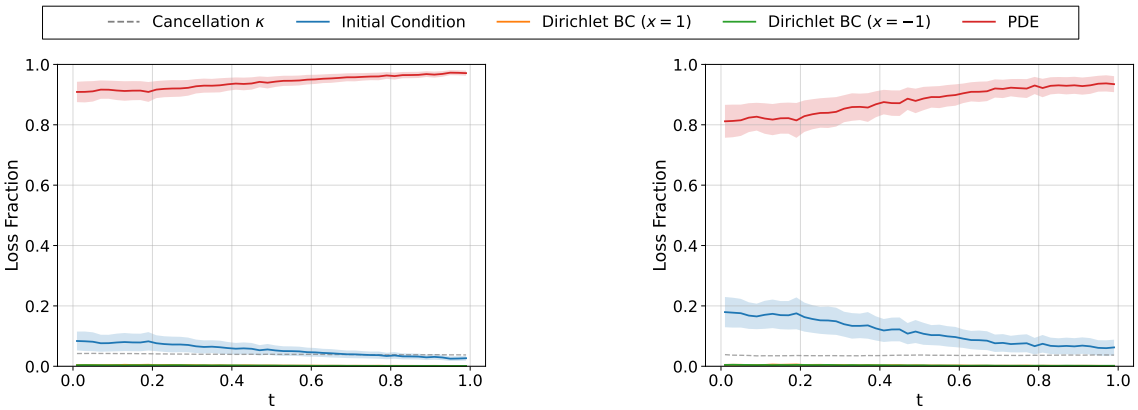
C.5. Decompositions by Loss-Components for other Problems

Here we present the loss decompositions for all problems not featured in the main text.



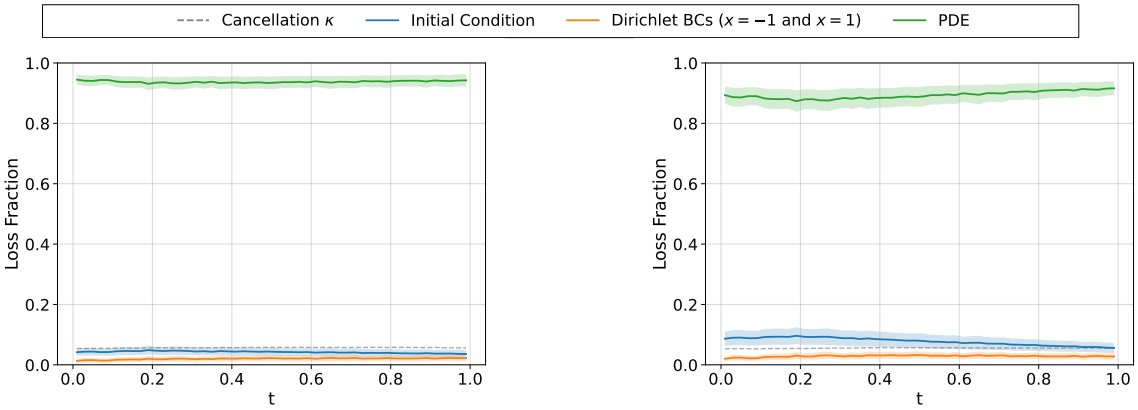
(a) Well-trained *(b)* Poorly-trained

Figure 17. Heat equation: Fractions of influences $\bar{r}_{L_i}(\mathcal{X}, R_{\text{test}})$ for 50 time bins, averaged over seeds. Shaded regions indicate the standard deviation.



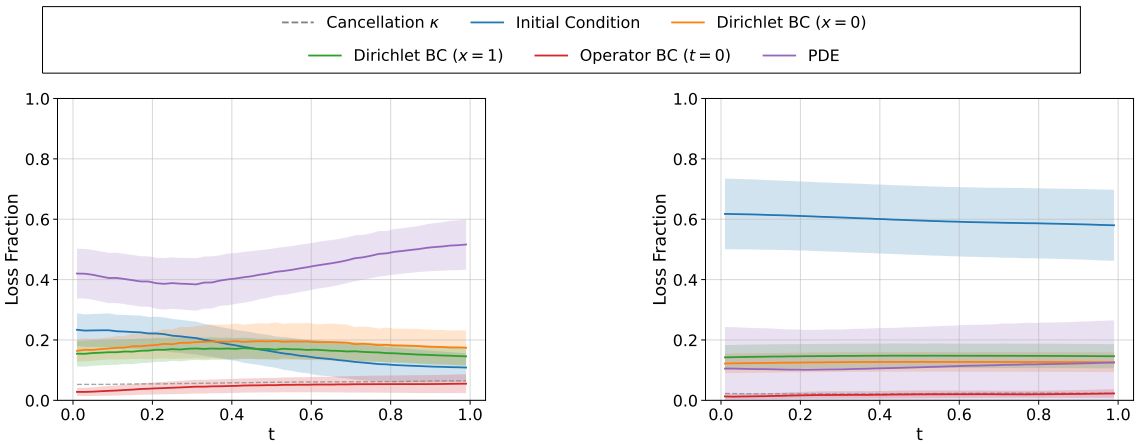
(a) Well-trained *(b)* Poorly-trained

Figure 18. Allen-Cahn equation: Fractions of influences $\bar{r}_{L_i}(\mathcal{X}, R_{\text{test}})$ for 50 time bins, averaged over seeds. Shaded regions indicate the standard deviation.



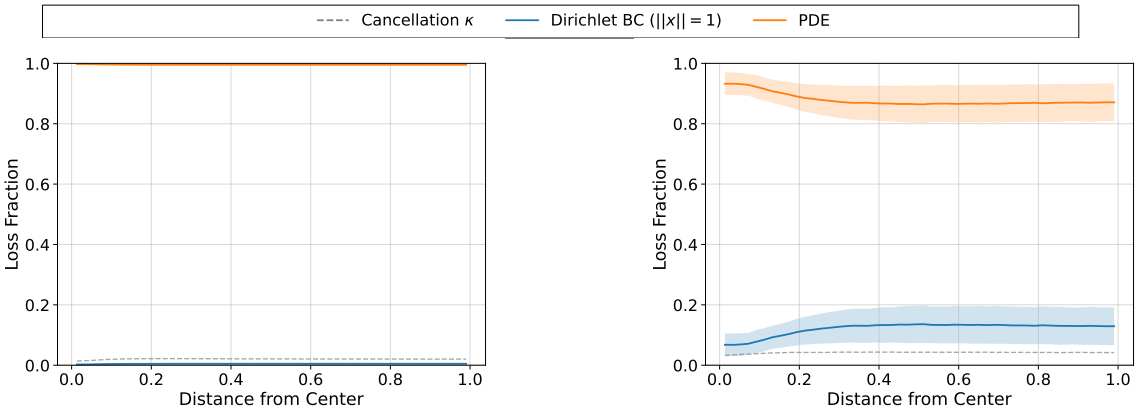
(a) Well-trained *(b)* Poorly-trained

Figure 19. Burgers' equation: Fractions of influences $\bar{r}_{L_i}(\mathcal{X}, R_{\text{test}})$ for 50 time bins, averaged over seeds. Shaded regions indicate the standard deviation.



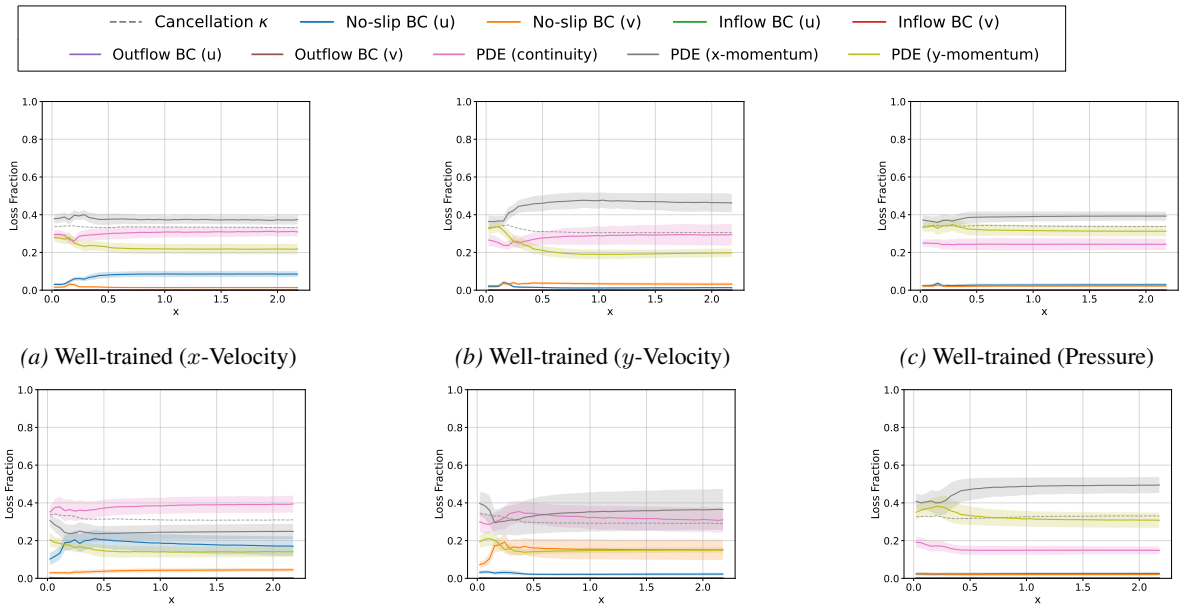
(a) Well-trained *(b)* Poorly-trained

Figure 20. Wave equation: Fractions of influences $\bar{r}_{L_i}(\mathcal{X}, R_{\text{test}})$ for 50 time bins, averaged over seeds. Shaded regions indicate the standard deviation.



(a) Well-trained *(b)* Poorly-trained

Figure 21. Poisson equation: Fractions of influences $\bar{r}_{L_i}(\mathcal{X}, R_{\text{test}})$ for 50 bins based on the distance from the center, averaged over seeds. Shaded regions indicate the standard deviation.



(a) Well-trained (x-Velocity) *(b)* Well-trained (y-Velocity) *(c)* Well-trained (Pressure)

(d) Poorly-trained (x-Velocity) *(e)* Poorly-trained (y-Velocity) *(f)* Poorly-trained (Pressure)

Figure 22. Navier-Stokes equations: Fractions of influences $\bar{r}_{L_i}(\mathcal{X}, R_{\text{test}})$ for 50 bins of x coordinates, averaged over seeds. Shaded regions indicate the standard deviation. Evaluated for all three outputs respectively.

C.6. Supplementary Point-wise Influence Plots

We visualize point-wise influence scores for each benchmark problem. For two representative points per problem, we compare well-trained and poorly-trained models side by side. Red heatmaps show which training points influence a fixed test point, whereas blue heatmaps show how a fixed training point influences the whole domain.

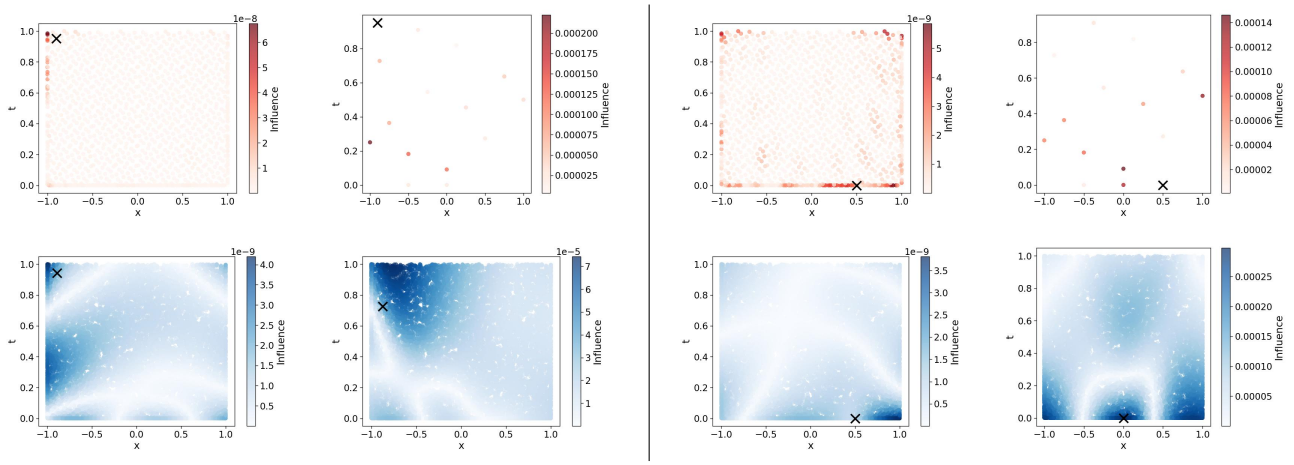


Figure 23. Heat equation: Influence scores for well-trained (left) and poorly-trained (right) models. Top: training point influences for fixed test point. Bottom: influences for a fixed training point onto the domain.

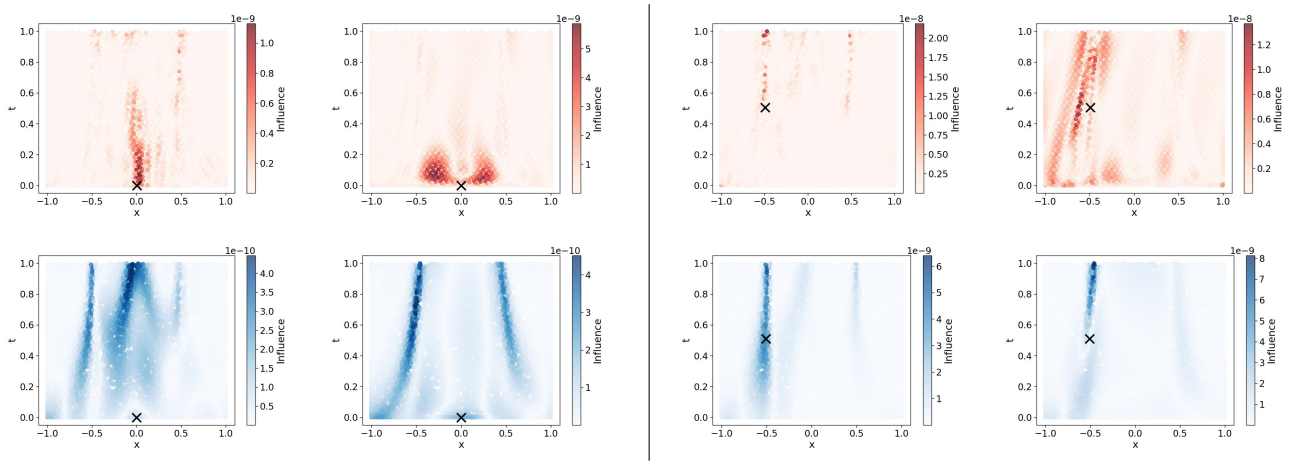


Figure 24. Allen-Cahn equation: Influence scores for well-trained (left) and poorly-trained (right) models. Top: training point influences for fixed test point. Bottom: influences for a fixed training point onto the domain.

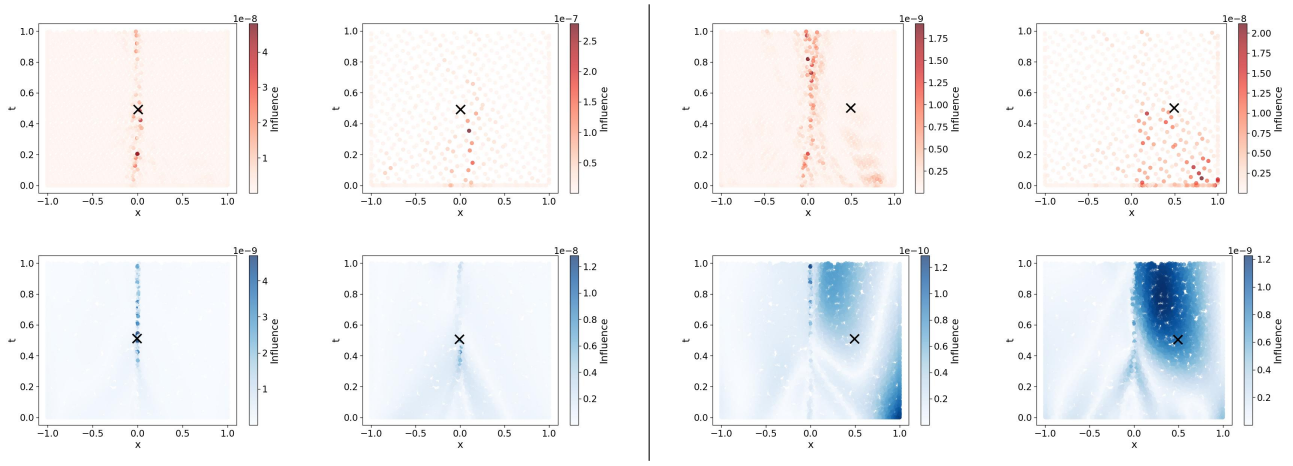


Figure 25. Burgers equation: Influence scores for well-trained (left) and poorly-trained (right) models. Top: training point influences for fixed test point. Bottom: influences for a fixed training point onto the domain.

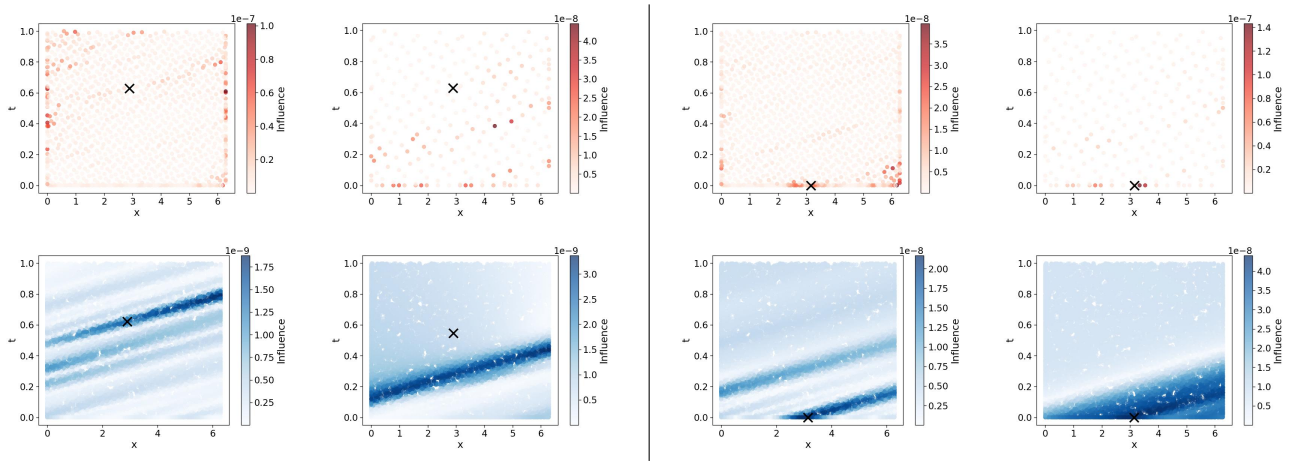


Figure 26. Drift-diffusion equation: Influence scores for well-trained (left) and poorly-trained (right) models. Top: training point influences for fixed test point. Bottom: influences for a fixed training point onto the domain.

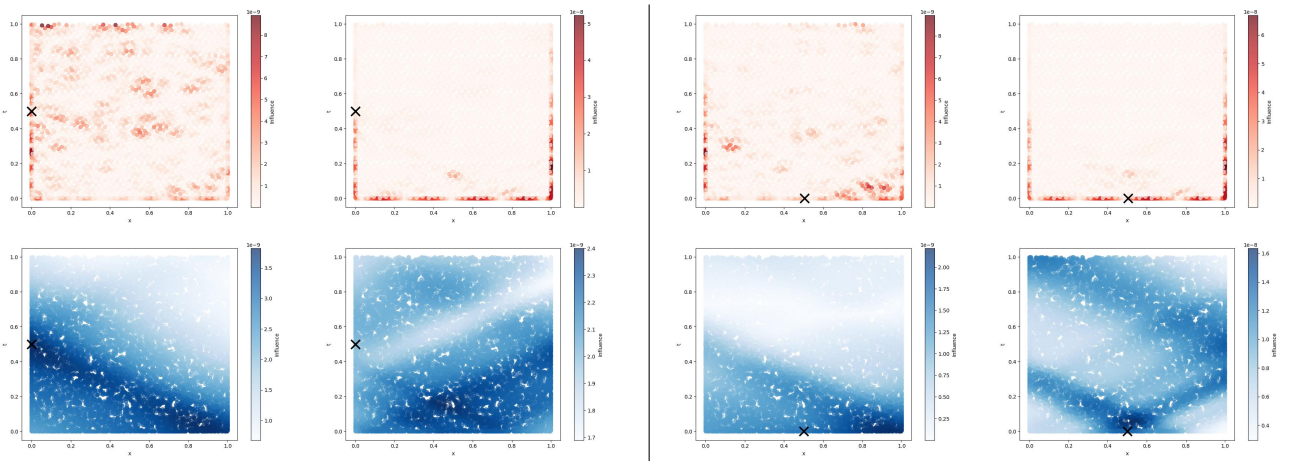


Figure 27. Wave equation: Influence scores for well-trained (left) and poorly-trained (right) models. Top: training point influences for fixed test point. Bottom: influences for a fixed training point onto the domain.

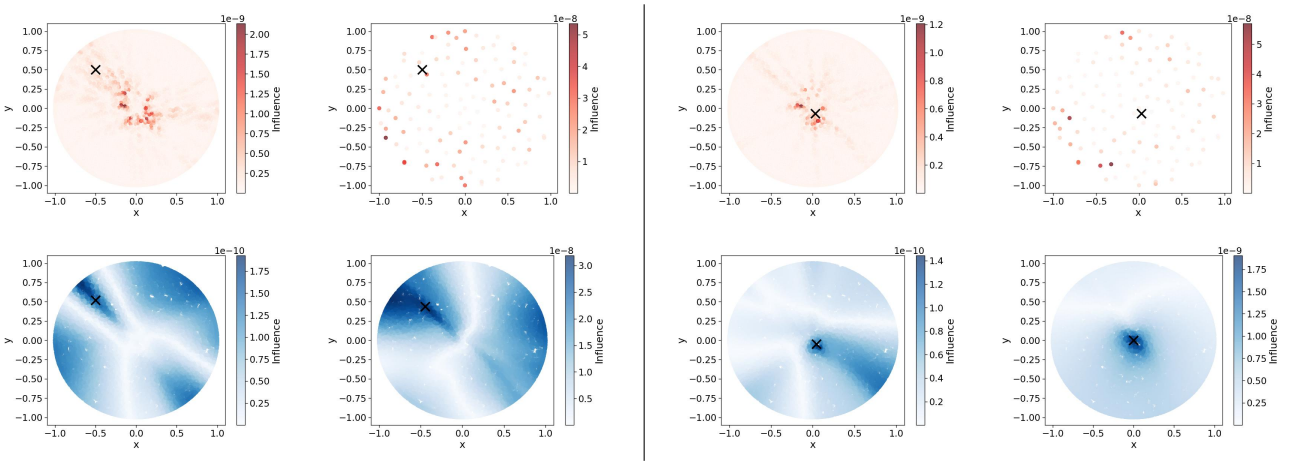


Figure 28. Poisson equation: Influence scores for well-trained (left) and poorly-trained (right) models. Top: training point influences for fixed test point. Bottom: influences for a fixed training point onto the domain.

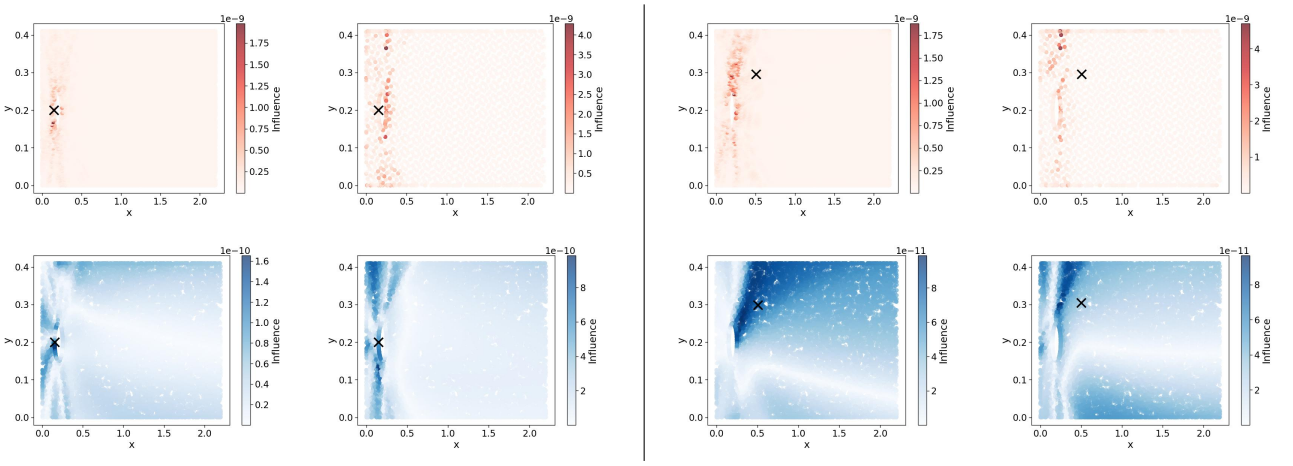


Figure 29. Navier-Stokes equations (x-Velocity): Influence scores for well-trained (left) and poorly-trained (right) models. Top: training point influences for fixed test point. Bottom: influences for a fixed training point onto the domain

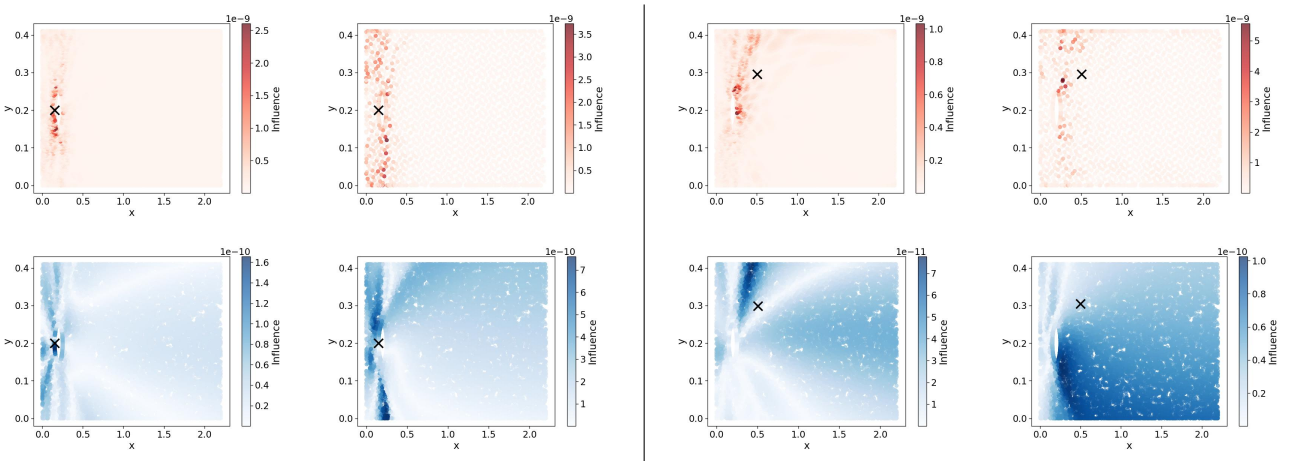


Figure 30. Navier-Stokes equations (y-Velocity): Influence scores for well-trained (left) and poorly-trained (right) models. Top: training point influences for fixed test point. Bottom: influences for a fixed training point onto the domain.

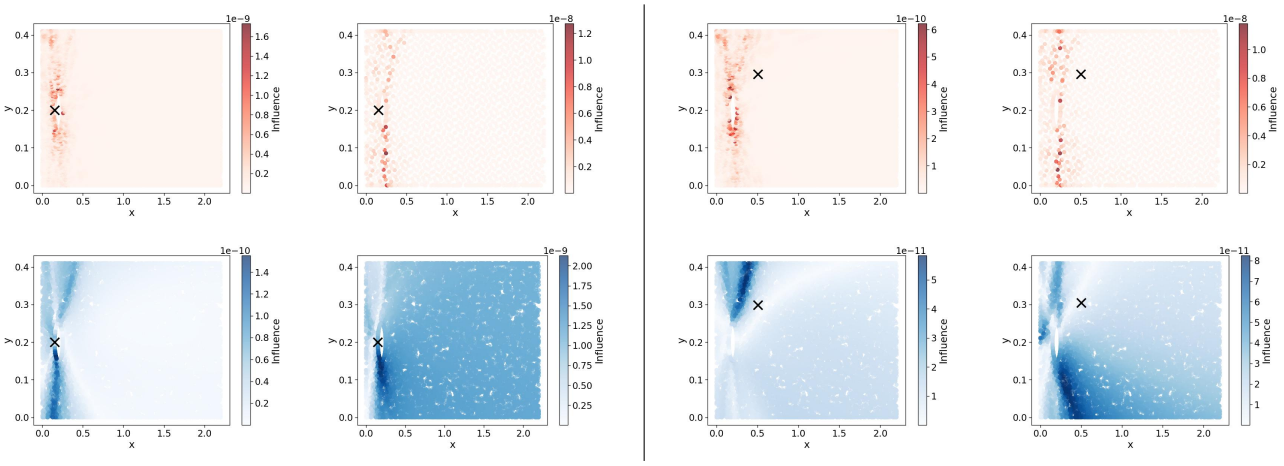


Figure 31. Navier-Stokes equations (Pressure). Influence scores for well-trained (left) and poorly-trained (right) models. Top: training point influences for fixed test point. Bottom: influences for a fixed training point onto the domain.