# DERIVATIVE CHEATING IN FUSION-BASED PINNS: UNDERSTANDING AND PREVENTING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Physics-Informed Neural Networks (PINNs) increasingly incorporate inter-point fusion mechanisms such as batch normalization, PointNet, and attention. In this paper, we identify a concealed but dangerous failure mode that arises when PDE residuals are derived via Automatic Differentiation (AD) in models that adopt such fusion mechanisms, in which AD may include fusion-induced paths unrelated to spatial derivatives, making the residual loss deceptively small while the absolute error remains large. We formalize this discrepancy, analyze the semantic distinction between AD-computed residual and true spatial derivatives, and introduce a diagnostic metric, Derivative Cheating Intensity (DCI), to quantify this discrepancy. We further propose a gradient-decoupled residual correction strategy that treats the fusion input as auxiliary context when computing spatial derivatives, rather than including it in the computational graph of AD. Experiments on Poisson, Helmholtz, and Heat benchmarks across BatchNorm, neural operator, PointNet, and Transformer backbones show that the proposed correction eliminates the failure and recovers accurate solutions. Our code is available at THIS LINK.

## 1 INTRODUCTION

Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019) are transforming scientific computing by using deep learning to solve Partial Differential Equations (PDEs). With Automatic Differentiation (AD), these models approximate PDE solutions without labeled data, acting as meshless solvers. PINNs enable self-supervised reconstruction of physical fields, producing neural models that generalize well. They show strong performance in fields like linear elasticity (Haghighat et al., 2021; Samaniego et al., 2020), fluid mechanics (Raissi et al., 2020; Jin et al., 2021), heat conduction (Cai et al., 2021; Haddout & Haddout, 2025), and electromagnetism (Chen et al., 2020; Lim & Psaltis, 2022). A well-trained PINN can replace traditional numerical methods, delivering fast solutions to problems that usually require significant computing resources.

As deep learning theory has advanced, a variety of advanced architectures have emerged, including PointNets (Qi et al., 2017) and Transformers (Vaswani et al., 2017). Compared with traditional point-wise feedforward networks, these models enable information exchange between collocation points, thereby enlarging the receptive field and improving the ability to capture spatial dependencies. In AI for PDEs, such architectures have been widely adopted in recent studies (Kashefi & Mukerji, 2022; Hao et al., 2023; Zhao et al., 2024). Moreover, operator learning frameworks based on spatial feature fusion (He et al., 2024; Ding et al., 2026) exhibit similar characteristics and can also integrate physics-driven learning. Importantly, the outputs of these advanced models no longer depend only on a single queried coordinate, but also on spatial features aggregated across the entire batch.

However, training such models under purely physics-based constraints remains challenging. Previous research has identified several common failure modes. For example, Wang et al. (2021a) showed that gradient conflicts between PDE residuals and boundary conditions can impede PINN convergence, while Rathore et al. (2024) further characterized the loss landscape's complexity. Krishnapriyan et al. (2021) found that soft PDE constraints can seriously affect the convergence of PINNs even in some simple cases, and Park et al. (2024) provided a derivative-pathology perspective to explain this issue theoretically. In addition, Daw et al. (2023) identified boundary-to-interior error propagation as another source of convergence difficulty. Collectively, these studies provide theoretical insights that



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

motivate a series of subsequent PINN optimizations (Wang et al., 2022; Liu et al., 2022; Dashtbayaz et al., 2024; Hwang & Lim, 2024; Liu et al., 2025).

Despite recent advances, most failure studies focus on optimization pathologies while neglecting the network's structural correctness. In this work, we identify a potential but dangerous failure mode that emerges specifically in fusion-based architectures or modules. We demonstrate that when spatial fusion is present, AD introduces additional gradient paths both theoretically and in practice that interact with the fusion mechanism. This interaction lets the optimizer minimize the total loss without satisfying the true spatial derivatives, leading to physically inconsistent zero-residual solutions. The key issue is not that fusion is harmful, but that the AD derivative used in the residual loss no longer has the correct physical semantics, a phenomenon we term **derivative cheating**. Our work is the first to elevate this phenomenon into a mechanism that is **derivable, measurable, and interventionable**, and provides clear warnings and guidance for subsequent work. By explicitly decoupling the gradients associated with fusion features from those of the spatial derivatives, we further demonstrate that the AD-based residual loss will revert to the same mode as point-wise PINNs, thereby eliminating this failure mode. Overall, our contributions are as follows:

1. We empirically demonstrate the proposed failure mode across multiple fusion-based backbones, comparing their performance against point-wise MLP under identical settings.
2. We theoretically show this potential issue in fusion-based models via gradient propagation analysis, specifically characterizing how Boundary Value Problems (BVPs) exacerbate this failure mode.
3. We apply a gradient decoupling strategy to avoid this failure and provide practical implementations for several standard fusion architectures to ensure stable backpropagation.
4. We verify the effectiveness of the proposed correction through controlled experiments. We provide an open-source toolbox to help the PINN community diagnose similar failure modes in new backbones.

## 2 PRELIMINARIES

In this section, we specify the PINN loss-function formulation and formalize the mappings of pointwise and fusion-based models from the computational-graph level to distinguish them for subsequent analysis.

### 2.1 MATHEMATICAL FORMULATION OF PINNS

Consider a general PDE of the form

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

where $\mathcal{N}$ is the differential operator defining the PDE, $f$ is the source term, and $\Omega$ is the computational domain. For a complete BVP, boundary conditions are imposed on $\Gamma = \partial\Omega$, which may consist of Dirichlet, Neumann, and Robin types, expressed in a general form:

$$\mathcal{B}[u](x) = \alpha u + \beta \nabla u \cdot \mathbf{n} = g(x), \quad x \in \Gamma, \tag{2}$$

where $\alpha$ and $\beta$ are coefficients of the zeroth- and first-order terms, respectively.

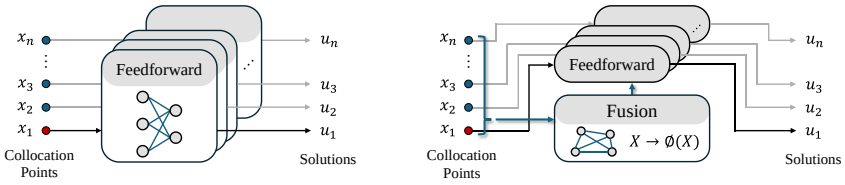
The standard PINN loss is defined as

$$\mathcal{L}(\theta) = w_{\text{res}}\mathcal{L}_{\text{res}} + w_{\text{bc}}\mathcal{L}_{\text{bc}}, \tag{3}$$

where $w_{\text{res}}$ and $w_{\text{bc}}$ balance the PDE residual and boundary condition losses. The PDE residual loss $\mathcal{L}_{\text{res}}$ and boundary loss $\mathcal{L}_{\text{bc}}$ are given by

$$\mathcal{L}_{\text{res}} = \frac{1}{N_{\text{res}}} \sum_{i=1}^{N_{\text{res}}} \left| \mathcal{N}[u_\theta](x_i) - f(x_i) \right|^2, \qquad \mathcal{L}_{\text{bc}} = \frac{1}{N_{\text{bc}}} \sum_{j=1}^{N_{\text{bc}}} \left| \mathcal{B}[u_\theta](x_j) - g(x_j) \right|^2, \tag{4}$$

where $\{x_i\}_{i=1}^{N_{\text{res}}} \subset \Omega$ are interior collocation points and $\{x_j\}_{j=1}^{N_{\text{bc}}} \subset \Gamma$ are boundary collocation points, $u_\theta$ denotes the model approximation parameterized by $\theta$.




(a) Point-wise model $u_\theta^{\mathrm{s}}(x)$ (b) Fusion-based model $u_\theta^{\mathrm{f}}(x, X)$

Figure 1: **Point-wise vs. fusion-based models.** (a) $u_\theta^{\mathrm{s}}$ predicts each collocation point independently; (b) $u_\theta^{\mathrm{f}}$ uses a fused global representation $\phi(X)$ combined with point-wise features.

### 2.2 Point-Wise vs. Fusion-Based Models

Consider a standard PINN training process, with the objective of minimizing the loss $\mathcal{L}(\theta)$. Let $X = \{x_1, \ldots, x_N\} \in \mathbb{R}^{N\times d}$ denote a set with $N$ collocation points in $d$-dimensional domain $\Omega$, the differences between the point-wise model and the fusion-based model are shown in Figure 1.

From the perspective of the computational graph, standard point-wise models map each coordinate independently, which can be abstracted as

$$u_\theta^{\mathrm{s}} : \mathbb{R}^d \to \mathbb{R}, \qquad x \mapsto u_\theta^{\mathrm{s}}(x), \tag{5}$$

where the computational graph from input to output depends only on the queried coordinate $x$, so $\partial u_\theta^{\mathrm{s}}/\partial x$ admits a single gradient path. Fusion-based models, in contrast, receive the entire point set $X$ as an auxiliary input and predict the value at each point:

$$u_\theta^{\mathrm{f}} : \mathbb{R}^d \times \mathbb{R}^{N\times d} \to \mathbb{R}, \qquad (x, X) \mapsto u_\theta^{\mathrm{f}}\big(x, \phi(X)\big), \tag{6}$$

where $\phi$ aggregates features across points in $X$. In PINN training, the queried coordinate $x$ is itself a member of the collocation set, i.e., $x \in X$, so $x$ enters the computational graph through two distinct routes: directly as the query input, and indirectly as an element of $X$ passing through $\phi$. In particular, AD with respect to $x$ accumulates gradients along both routes, so that the fusion feature $\phi$ contributes nontrivially to $\partial u_\theta^{\mathrm{f}}/\partial x$ under AD.

Through random sampling per iteration, the collocation points fitted by the network can theoretically approach infinity over the training process. However, for fusion-based models, the number of points $N$ in each input batch remains finite due to computational constraints.

## 3 Related Work

In this section, we examine mainstream fusion-based models that satisfy Equation 6, broadly categorized as architectural-level fusion and module-level fusion. Architectural-level fusion introduces a dedicated global feature branch that is combined with point-wise features, whereas module-level fusion embeds feature interaction directly within internal operations such as BatchNorm or attention. Some approaches also adopt hybrid fusion mechanisms.

### 3.1 Architectural-Level Fusion

**Neural Operators.** An earlier line of work by Wang et al. (2021b) introduced the integration of the operator learning framework with PINNs, while He et al. (2024) explicitly encodes geometric structures into the network. Furthermore, subsequent methods (Cameron et al., 2023; Zhong & Meidani, 2024; 2025; Boya & Subramani, 2025; Ding et al., 2026) have introduced various improvements on this foundation.

For neural operators that integrate geometric features into the network (He et al., 2024; Zhong & Meidani, 2024), the network mapping can be abstracted as

$$u_i = g\big(t(x_i), \phi(X)\big), \tag{7}$$

where the trunk feature $t(x_i)$ corresponds to the point-wise pathway (blue term), while the branch feature $\phi(X)$ corresponds to the fusion pathway (orange term). Thus, the output jointly depends on the queried coordinate and the fused global representation, consistent with Equation 6.

**PointNets.** Kashefi & Mukerji (2022) pioneered Physics-Informed PointNet (PIPN) for PDE solving on irregular point clouds. Later studies extended it to crack propagation (Tu et al., 2023), multi-physics systems (Wang et al., 2025), and KAN-based PINNs (Kashefi & Mukerji, 2026). These methods use shared point-wise encoders with symmetric aggregation to perceive global features.

Formally, the PointNet mapping can be written as

$$u_i = g\left(x_i, \mathrm{Agg}_{x_j \in X}\left(h(x_j)\right)\right), \tag{8}$$

where $h(\cdot)$ denotes a shared point-wise feature extractor and $g$ denotes a global prediction function. The aggregation function $\mathrm{Agg}(\cdot)$ typically employs symmetric operations, such as max pooling or average pooling. In particular, the output is globally or locally differentiable with respect to both the fusion term (orange term) and the point-wise term (blue term).

### 3.2 Module-Level Fusion

**Batch Normalization.** Several AI4PDE models have incorporated batch normalization (Ioffe & Szegedy, 2015) into their architectures, including FNO (Li et al., 2021), Message Passing Neural PDE Solver (Brandstetter et al., 2022), and the PIPN series (Kashefi & Mukerji, 2022; 2023; 2026). Some of these models are clearly trained in a physics-informed manner.

Batch normalization standardizes the batch data by its mean and variance, expressed as

$$\mathrm{BatchNorm}(z_i) = \frac{z_i - \mu_{\mathcal{B}}}{\sqrt{{\sigma_{\mathcal{B}}}^2 + \epsilon}}, \tag{9}$$

where the input $z_i$ represents the point-wise pathway (blue term), while the batch statistics $\mu_{\mathcal{B}}$ and $\sigma^2_{\mathcal{B}}$ form the fusion pathway (orange term). These statistics are computed over the entire batch $X$, and the gradients explicitly pass through them in the standard implementation of BatchNorm.

**Attention.** Dos Santos et al. (2023) and Zhao et al. (2024) first introduce Transformers to learn temporal sequences in PINNs; Cameron et al. (2023) introduces cross-attention between boundary and internal collocation points; Chen et al. (2025) and Luo et al. (2025) attempt to introduce physics-informed constraints into general Transformer PDE solvers; Song & Yan (2026) applies attention in both spatial and temporal sequences; And Ding et al. (2026) combines Transformer and KAN.

Taking the classic dot-product attention (Vaswani et al., 2017) as an example, the feature fusion in the Transformers can be expressed as

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{10}$$

If $Q$ and $K, V$ originate from the same collocation point sequence $X$, then the derivative computed by AD will simultaneously pass through these three tensors. Under this configuration, query $Q$ represents the point-wise pathway (blue term), as it does not interact with the neighboring points from itself, and key $K$ and value $V$ represent the fusion pathway (orange term).

## 4 Failure Mode

In this section, we run a series of experiments to show when fusion-based models fail on common PINN problems and to compare their behavior with a classical pointwise MLP.

**Experimental setup.** We selected three standard PINN benchmarks, including *(i)* Poisson, a Poisson problem with complex nonlinear boundary conditions, *(ii)* Helmholtz, a Helmholtz-type reaction-diffusion problem with complex geometry and high frequency, and *(iii)* Heat, a heat conduction problem with composite boundary conditions. For comparison, we consider five different types of backbones, including *(i)* vanilla MLP, point-wise model, *(ii)* MLP + BatchNorm (Ioffe & Szegedy, 2015), *(iii)* modified PI-DeepONet (Wang et al., 2021b), neural operator backbone with spatial fusion, *(iv)* PIPN (Kashefi & Mukerji, 2022), PointNet backbone, and *(v)* MMET (Luo et al., 2025), Transformer backbone. To achieve clearer module-wise attribution, we made specific modifications to some models. More experimental setup details are in Appendices B and C.

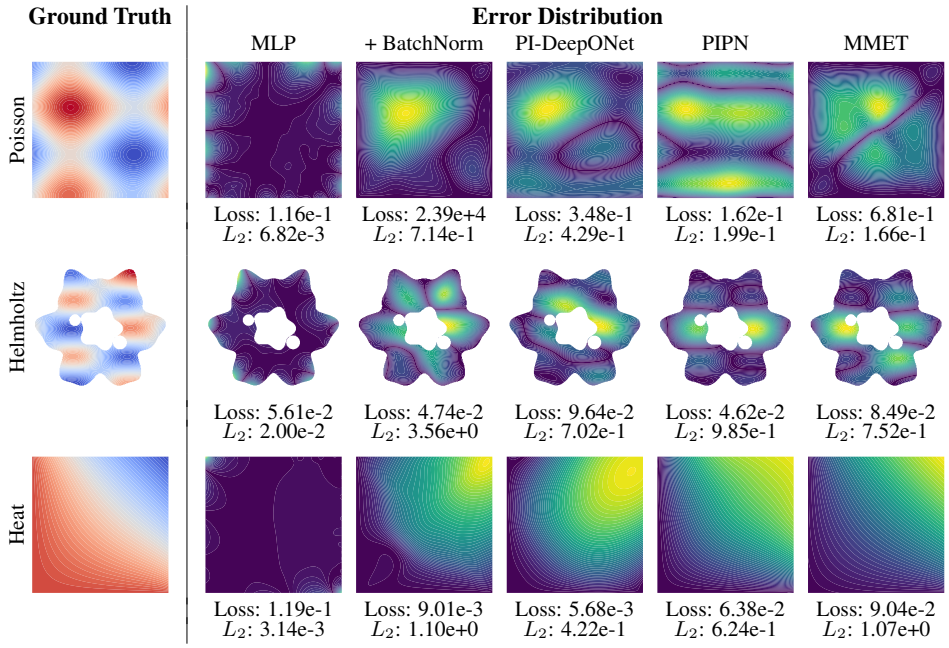



Figure 2: **Error maps of each backbone**, as well as the final loss values ($\mathcal{L}_{\text{res}} + \mathcal{L}_{\text{bc}}$) and relative $L_2$ errors upon convergence. We also reported statistical significance in Appendix D.

**Result analysis.** We present the error maps for each model and dataset in Figure 2, along with the final loss and the relative $L_2$ error at convergence. Among them, the MLP demonstrates excellent prediction accuracy when the loss value is small, and its error distribution is relatively uniform. However, most fusion-based backbone networks, even when achieving the same loss level as the MLP (or even lower), show prediction accuracies that differ by one or more orders of magnitude from the MLP. Most fail completely, with a relative $L_2$ error exceeding $10\%$.

From the error maps, we observe that all fusion-based models exhibit relatively small errors near the Dirichlet boundary. However, in interior regions far from such boundaries, one or more high-error regions emerge, a behavior markedly different from that of the MLP. This phenomenon is most pronounced in the Heat benchmark with composite boundary conditions: the errors of all fusion-based models gradually increase from the lower-left corner (prescribed temperature boundary) to the upper-right corner (heat flux boundary). This reveals a potential failure mode of PINNs, where for fusion-based models, **even when the optimizer reduces the loss to a very small value, the final model output can still deviate significantly from the ground truth**.

## 5 Theoretical Analysis

In this section, we analyze why physics-informed learning in a fusion-based model behaves differently from a traditional point-wise model and discuss the error distribution and its relation to boundary condition types to characterize when the model fails.

### 5.1 Supervised Training

First, consider the classic supervised regression objective

$$\theta^\star = \arg\min_{\theta\in\Theta} \mathcal{L}(u_\theta, u^\star), \tag{11}$$

where $u^\star$ is the ground truth, and the loss function $\mathcal{L}$ is point-wise separable. In this context, a widely recognized consensus within the deep learning community is:

**Proposition 5.1** (Interpolation under Point-wise Supervision)**.** *Under the constraint of supervised regression, the optimal outputs of $u^{\text{s}}_\theta$ and $u^{\text{f}}_\theta$ coincide at every collocation point:*

$$u^{\text{s}}_{\theta^\star_{\text{s}}}(x) = u^{\text{f}}_{\theta^\star_{\text{f}}}\big(x, \phi(X)\big) = u^\star, \quad \forall\, x \in X, \tag{12}$$

*where $\theta_{\mathrm{s}}^{\star}$ and $\theta_{\mathrm{f}}^{\star}$ represent the optimal parameters of the coordinate-based and fusion-based models, respectively. That is, under a point-wise supervised loss, inter-point feature fusion does not alter the optimal output of the model at any individual point.*

### 5.2 Physics-Informed Training

When PDE residuals are incorporated into the loss function, failures may arise. Consider regression objectives involving differential operators of the form $\mathcal{N}^{\mathrm{AD}}[u_\theta] = f$, where $\mathcal{N}^{\mathrm{AD}}[\cdot]$ computed via AD may represent a PDE residual term or a boundary condition term containing derivatives:

$$\theta^{\star} = \arg\min_{\theta\in\Theta} \mathcal{L}\left(\mathcal{N}^{\mathrm{AD}}[u_\theta] - f\right). \tag{13}$$

**Engineering-level issues.** AD tools in modern deep learning frameworks (e.g., `torch.autograd` and `tf.GradientTape`) are not well-suited for explicitly computing the full Jacobian $J$ or Hessian $H$ with respect to spatial variables. Given a PINN with $M$ parameters and $N$ collocation points, constructing the full Jacobian for spatial derivatives incurs a time complexity of $O(MN^2)$ and a space complexity of $O(N^2)$. The cost further grows exponentially for higher-order derivatives, rendering it impractical for densely sampled training. Therefore, they instead compute the vector–Jacobian product:

$$du_j = \sum_{i=1}^{N} \frac{\partial u_i}{\partial x_j}, \tag{14}$$

which reduces the time complexity to $O(MN)$ and the space complexity to $O(N)$, making the computational cost of AD acceptable from an engineering perspective (Griewank & Walther, 2008).

For point-wise models, where each output $u_i$ depends only on its corresponding coordinate $x_i$, the Jacobian $J$ is diagonal, and Equation 14 is equivalent to the spatial derivative $\partial u_j/\partial x_j$. However, for fusion-based models with inter-point interactions, this equivalence no longer holds. In such cases, the Jacobian contains nonzero off-diagonal entries, and the computed quantity aggregates gradients across different points. Consequently, derivatives from neighboring points $\partial u_i/\partial x_j$ act as perturbation terms in the residual, while the optimizer only constrains their aggregated effect. Therefore, under this setting, AD no longer yields the true spatial derivatives, but instead produces a coupled quantity that deviates from the underlying physical operator.

**Theoretical-level issues.** Theoretically, even if the point-wise spatial derivatives $\partial u_j/\partial x_j$ are explicitly constrained in a fusion-based model, the resulting model still does not correspond to a physically meaningful differential operator. For a PINN with a properly configured fusion-based backbone, we assume that the fusion-based model has sufficient capacity and does not degenerate into a point-wise model. The following theorem can be established:

**Theorem 5.2** (Non-Representability of Fusion-Based PINNs)**.** *Suppose that*

$$\mathcal{S} := \{u \in \mathcal{F}(\Omega) \mid \mathcal{N}[u](x) = f(x),\ \forall x \in \Omega\} \tag{15}$$

*is the solution set of the governing PDE. A fusion-based model $u_{\theta^\star}^{\mathrm{f}}$ trained by minimizing the AD residuals may not correspond to any valid solution $u^\star \in \mathcal{S}$, even if the PDE residual is zero over the domain $\Omega$.*

*Proof Sketch.* We let $h(x) := u_\theta^{\mathrm{f}}\big(x, \mathrm{sg}[\phi(X)]\big)$ denote the function obtained by freezing the collocation set, where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator. This operation aligns $h$ with standard point-wise models regarding AD. By the chain rule, the first-order partial derivative from AD satisfies

$$\frac{\partial^{\mathrm{AD}}}{\partial x} u_\theta^{\mathrm{f}}\big(x, \phi(X)\big) = \underbrace{\frac{\partial h}{\partial x}}_{\text{Spatial}} + \underbrace{\frac{\partial u_\theta^{\mathrm{f}}}{\partial \phi}\frac{\partial \phi}{\partial x}}_{\text{Fusion}}, \tag{16}$$

and higher-order derivatives follow a similar expansion by the multivariate Faà di Bruno formula.

When the fusion pathway contributes nontrivially (that is, the fusion term in Equation 16 is non-zero), this AD derivative differs from the point-wise (spatial) term $\partial h/\partial x$, so a parameter $\theta^\star$ may satisfy $\mathcal{N}^{\mathrm{AD}}[u_{\theta^\star}^{\mathrm{f}}](x_0) = f(x_0)$ while $\mathcal{N}[h](x_0) \neq f(x_0)$ at any collocation point $x_0 \in X$ where the fusion contribution is nonzero. Therefore, $h(\cdot) \notin \mathcal{S}$. Since $h$ and $u_{\theta^\star}^{\mathrm{f}}$ coincide pointwise, it follows that $u_{\theta^\star}^{\mathrm{f}}\big(\cdot, \phi(X_{t_0})\big) \notin \mathcal{S}$. □

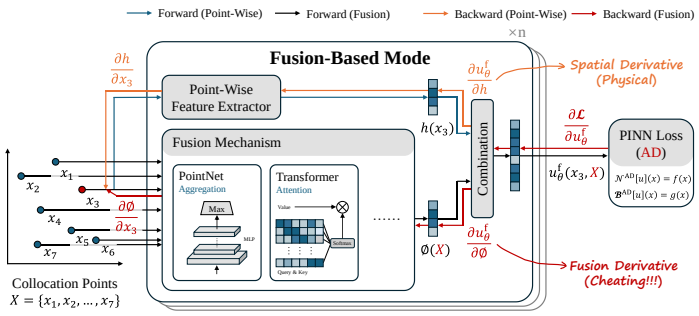



Figure 3: **Schematic illustration** of the derivative cheating phenomenon in fusion-based PINNs.

We provide the strict proof version in Appendix A.1. In this proof, we showed that applying AD to fusion-based models introduces additional fusion derivative terms in Equation 16, which were originally intended to measure "*the sensitivity of $\phi$ to changes in $x$*" rather than "*the rate of change of $u$ within domain $\Omega$.*" As shown in Figure 3, the AD-based PDE residual implicitly couples both contributions, and the optimizer only constrains the overall residual loss. The model can therefore adjust the fusion term to achieve a numerically zero residual, while the spatial derivative term that carries the true physical semantics may remain nonzero, leading to what we term derivative cheating, where the optimization objective does not satisfy the original PDE. To verify that this failure mode is independent of both theoretical and engineering perspectives, we conduct ablation studies on both a toy fusion design and a genuine Transformer in Appendix G.

Now we expand the above analysis to a complete BVP optimization instance, where its objective is

$$\theta^\star = \arg\min_{\theta\in\Theta}\ w_{\mathrm{res}}\mathcal{L}_{\mathrm{res}} + w_{\mathrm{bc}}\mathcal{L}_{\mathrm{bc}}. \tag{17}$$

**Theorem 5.3** (False Solution under BVP)**.** *Suppose that a fusion-based model $u^{\mathrm{f}}_\theta$ is used to solve a PDE subject to a well-posed BVP. There exists an optimal parameter $\theta^\star \in \Theta$ such that the model satisfies both the PDE residual and the boundary conditions by AD, but the resulting function $u^{\mathrm{f}}_{\theta^\star}$ does not coincide with the true solution $u^\star$.*

We provide a proof of this theorem in Appendix A.2, where we show that the coupling term introduced by Equations 14 and 16 is equivalent to adding a pseudo-source term $s_{\mathrm{res}}$ to the PDE, i.e.,

$$\mathcal{N}[h](x) = f(x) - s_{\mathrm{res}}(x), \tag{18}$$

thereby making the resulting optimization problem no longer equivalent to the original BVP.

In particular, under Dirichlet boundary conditions, the optimization objective becomes equivalent to classical supervised learning with the coefficient of the normal derivative term $\beta = 0$ in Equation 2, as characterized by Proposition 5.1. This explains the phenomenon we observed in Section 4, where the error map of the false solution approaches zero only in the neighborhood of the Dirichlet boundary.

## 6 How to Prevent Derivative Cheating?

In this section, we propose a direct remedy for derivative cheating and provide a detailed discussion for mainstream architectures. From Theorem 5.2, the following corollary follows directly:

**Corollary 6.1** (Representability under Gradient-Decoupled Fusion)**.** *Suppose $u^{\mathrm{f}}_\theta$ is a fusion-based model with sufficient capacity. Define $\tilde{u}^{\mathrm{f}}_\theta(x) = u^{\mathrm{f}}_\theta\big(x, \phi(\mathrm{sg}[X])\big)$ by treating $X$ as a stop-gradient input to $\phi$ when computing AD with respect to $x$. Under this formulation, the fusion-induced contribution to $\partial^{\mathrm{AD}}_x u$ is eliminated. If there exists $\theta^\star \in \Theta$ such that*

$$\mathcal{N}[\tilde{u}^{\mathrm{f}}_{\theta^\star}](x) - f(x) = 0, \qquad \forall\, x \in \Omega, \tag{19}$$

*then there exists $u^\star \in \mathcal{S}$ satisfying*

$$\tilde{u}^{\mathrm{f}}_{\theta^\star}(\cdot) = u^\star(\cdot). \tag{20}$$

*That is, after decoupling the fusion-induced gradient path from the AD, any zero-residual solution of the training objective $\tilde{u}^{\mathrm{f}}_{\theta^\star}$ corresponds to a valid PDE solution $u^\star$.*

This corollary requires that $X$ be entirely excluded from the AD computational graph with respect to $u$. However, for some backbones, decoupling gradients at the wrong position can also block backpropagated error, leaving parts of the network without an optimization signal. Therefore, we explore specific solutions for mainstream models at both the architectural and module levels.

### 6.1 Architectural-Level Gradient Decoupling

Architectural-level decoupling is suitable for models such as neural operators (Wang et al., 2021b), PointNets (Kashefi & Mukerji, 2022) and some special Transformers (Zhao et al., 2024; Luo et al., 2025), which only include cross-attention in the decoder. These architectures include an independent feature-fusion component that produces a single global spatial feature $\phi(X)$, which is then provided to the local point-wise component. In particular, no further information exchange occurs within the point-wise part.

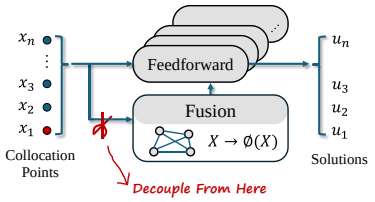



Figure 4: **Architectural-level gradient decouple position.**

For this type of model, it suffices to decouple the gradients at the input of the fusion module, as shown in Figure 4. The decoupled point set $X$ is then treated as an auxiliary input variable, rather than being included in the computational graph of the PDE residual and boundary losses through AD. Backpropagation, however, proceeds normally and therefore does not affect model optimization.

### 6.2 Module-Level Gradient Decoupling

Module-level decoupling suits internal fusion blocks like batch normalization (Ioffe & Szegedy, 2015) and self/cross attention (Vaswani et al., 2017). For models that stack these fusion operations internally, it is not possible to simply decouple the fusion gradients from the input.

**Batch Normalization.** To prevent the gradients of other collocation points from polluting the AD derivative, we directly decouple the mean and variance of batch normalization:

$$\mathrm{BatchNorm}_{\mathcal{D}}(x_i) = \frac{x_i - \mathrm{sg}[\mu_{\mathcal{B}}]}{\sqrt{\mathrm{sg}[\sigma_{\mathcal{B}}]^2 + \epsilon}}. \tag{21}$$

With this operation, the mean $\mu_{\mathcal{B}}$ and variance $\sigma_{\mathcal{B}}$ are regarded as auxiliary normalization constants and are not included in the AD computational graph.

**Attention.** In attention blocks where $Q$, $K$, and $V$ originate from the same input $X$, to prevent derivative cheating from both theoretical and engineering perspectives, we must completely decouple the gradients of $K$ and $V$, defined as

$$\mathrm{Attention}_{\mathcal{D}}(Q, K, V) = \mathrm{softmax}\left(\frac{Q\bar{K}^\top}{\sqrt{d_k}}\right)\bar{V}, \tag{22}$$

where $\bar{K} = \mathrm{sg}[X]W_K$ and $\bar{V} = \mathrm{sg}[X]W_V$. This operation ensures that spatial derivatives are accurately computed via AD, while maintaining gradient propagation through $Q$.

Although it prevents derivative cheating, module-level decoupling is more intrusive than architectural-level alternatives and may introduce potential inductive biases. See Appendix H for a detailed discussion. We therefore encourage future fusion-based PINN architectures to be designed with a clean point-wise/fusion separation that admits architecture-level decoupling.

## 7 Experimental Verification

In this section, we design a controlled experiment to verify the effectiveness of gradient decoupling. All experimental settings are identical to those described in Section 4 for the failure demonstration, with the only modification being the application of the gradient decoupling strategy proposed in Section 6 to all fusion-based models.

In order to quantify the degree of derivative cheating in each backbone, we also introduce the *Derivative Cheating Intensity* (DCI), defined as

$$\mathrm{DCI} = \frac{1}{N}\sum_{i=1}^{N} \frac{|\mathcal{N}^{\mathrm{AD}}[u^{\mathrm{f}}_{\theta^\star}](x_i) - \mathcal{N}[h](x_i)|}{|\mathcal{N}^{\mathrm{AD}}[u^{\mathrm{f}}_{\theta^\star}](x_i)| + |\mathcal{N}[h](x_i)| + \epsilon}, \tag{23}$$

where $h(x) = u^{\mathrm{f}}_{\theta}\big(x, \mathrm{sg}[\phi(X)]\big)$ denotes the model output with the entire fusion pathway frozen, and $\mathcal{N}[h](x)$ is the differential operator evaluated using only spatial derivatives. Unlike the training-time

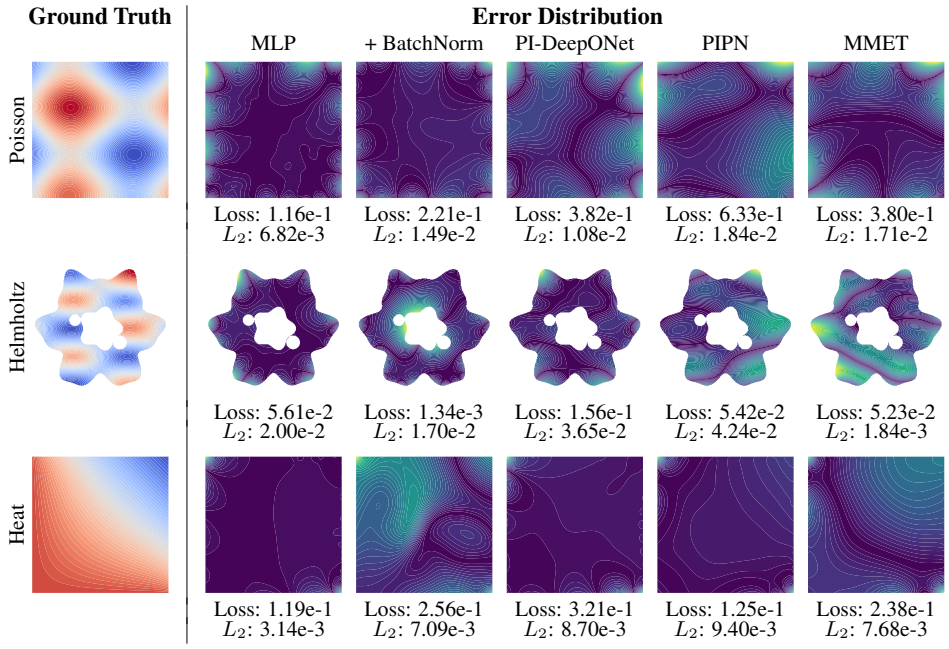



Figure 5: **Error maps of each backbone after decoupling the fusion gradient.**

decoupling in Section 6, which modifies the optimization objective, this construction is used only at inference time to isolate the fusion-induced derivative component of a trained model. Based on this, DCI measures the normalized discrepancy between the AD-evaluated differential operator and the purely spatial operator. A large DCI indicates strong derivative cheating, meaning that the PDE residual is significantly affected by fusion-induced derivatives. As a sanity check of our correction, it can also verify whether fusion-induced derivatives are fully removed. If no derivatives leak from the fusion term, DCI theoretically approaches zero.

Table 1: **DCI before and after gradient decoupling.**

| **Model** | **Decouple** | Pois. | Helm. | Heat |
|---|---|---|---|---|
| MLP+BatchNorm | × | 9.99e-1 | 7.73e-1 | 9.99e-1 |
| | ✓ | 0.00e+0 | 0.00e+0 | 0.00e+0 |
| PI-DeepONet | × | 5.77e-1 | 5.75e-1 | 9.66e-1 |
| | ✓ | 0.00e+0 | 0.00e+0 | 0.00e+0 |
| PIPN | × | 3.21e-1 | 6.62e-1 | 9.00e-1 |
| | ✓ | 0.00e+0 | 0.00e+0 | 0.00e+0 |
| MMET | × | 4.17e-1 | 6.47e-1 | 9.82e-1 |
| | ✓ | 0.00e+0 | 0.00e+0 | 0.00e+0 |

Figure 5 shows the final prediction errors of fusion-based models after decoupling. Compared with Figure 2, all decoupled models succeed across benchmarks, with relative $L_2$ error below $5\%$. Their error distributions move closer to those of the MLP, rather than increasing inward from the Dirichlet boundary. Table 1 compares DCI before and after decoupling. Before decoupling, most models have DCI values above $20\%$, especially in the Heat benchmark with composite boundary conditions, where they exceed $90\%$ for all models. After decoupling, all DCI values approach zero, confirming the effectiveness of the proposed repair strategy. We also ran identical experiments on established complex benchmarks to substantiate our findings; see Appendix E.

## 8 Conclusion

This paper identifies a novel failure mode in fusion-based PINNs, termed derivative cheating, where AD introduces additional gradient paths through feature fusion, enabling models to achieve near-zero residuals without satisfying physical derivatives. Through theoretical and empirical studies, we show that this issue is inherent to fusion-based architectures. To address this, we adopt a gradient decoupling strategy that removes fusion-induced derivatives from the AD computation, restoring consistency between residual minimization and valid PDE solutions. Experiments across multiple backbones confirm the method's effectiveness.

### AI USE STATEMENT

In this work, we used generative AI tools for language editing and writing assistance, including improving grammar, phrasing, clarity, and the manuscript's presentation. We did not use generative AI to formulate the research problem, develop the proposed method, derive the theoretical results, write experimental code, or generate experimental data or results. The authors carefully inspected and revised all AI-assisted text to ensure technical correctness and consistency with the underlying analysis and experimental evidence. The authors take full responsibility for the final content of this work, including any text or artifacts produced with generative AI assistance.

### REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide an anonymous source-code repository containing the implementations and experimental pipelines used in this work. The theoretical assumptions and complete proofs of the main results are provided in Appendix A. Appendix B specifies the governing equations, computational domains, boundary conditions, and reference solutions for the benchmark problems. Appendix C documents the model architectures, gradient-decoupling implementations, training hyperparameters, collocation-point sampling strategy, optimizer settings, training iterations, and computational hardware. Appendix D reports the main experimental results over five independent random seeds using the mean and standard deviation. Together with the released code, these details are intended to enable reproduction of the theoretical and empirical results reported in the paper.

## REFERENCES

Sumanth Kumar Boya and Deepak N. Subramani. Pinto: Physics-informed transformer neural operator for learning generalized solutions of partial differential equations for any initial and boundary condition. *Computer Physics Communications*, 315:109702, 2025.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural PDE solvers. In *10th International Conference on Learning Representations (ICLR)*, 2022.

Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

Scott Cameron, Arnu Pretorius, and S Roberts. Nonparametric boundary geometry in physics informed deep learning. In *37th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Tianyu Chen, Haoyi Zhou, Ying Li, Hao Wang, Chonghan Gao, Rongye Shi, Shanghang Zhang, and Jianxin Li. OmniArch: Building foundation model for scientific computing. In *42nd International Conference on Machine Learning (ICML)*, 2025.

Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics Express*, 28(8):11618–11633, 2020.

Nima Hosseini Dashtbayaz, Ghazal Farhani, Boyu Wang, and Charles X. Ling. Physics-informed neural networks: minimizing residual loss with wide networks and effective activations. In *33rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

Arka Daw, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne. Mitigating propagation failures in physics-informed neural networks using retain-resample-release (R3) sampling. In *40th International Conference on Machine Learning (ICML)*, 2023.

Siqi Ding, Zitong Zhang, Shi Guoyang, Xingyu Li, Xiang Gu, Yanan Xu, Huasheng Xie, Hanyue Zhao, Yuejiang Shi, and Tianyuan Liu. Physics-informed neural operator learning for nonlinear grad-shafranov equation. In *43rd International Conference on Machine Learning (ICML)*, 2026.

Fabricio Dos Santos, Tara Akhound-Sadegh, and Siamak Ravanbakhsh. Physics-Informed Transformer Networks. In *NeurIPS 2023 Workshop on The Symbiosis of Deep Learning and Differential Equations III*, 2023. Workshop paper.

Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Society for Industrial and Applied Mathematics, 2008.

Youssef Haddout and Soufiane Haddout. Deep physics-informed neural networks for stratified forced convection heat transfer in plane couette flow: Toward sustainable climate projections in atmospheric and oceanic boundary layers. *Fluids*, 10(12), 2025.

Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. GNOT: A general neural operator transformer for operator learning. In *40th International Conference on Machine Learning (ICML)*, 2023.

Zhongkai Hao, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, and Jun Zhu. PINNacle: A comprehensive benchmark of physics-informed neural networks for solving PDEs. In *38th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Junyan He, Seid Koric, Diab Abueidda, Ali Najafi, and Iwona Jasiuk. Geom-DeepONet: A point-cloud-based deep operator network for field predictions on 3d parameterized geometries. *Computer Methods in Applied Mechanics and Engineering*, 429:117130, 2024.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Youngsik Hwang and Dong-Young Lim. Dual cone gradient descent for training physics-informed neural networks. In *38th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning (ICML)*, 2015.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

Ali Kashefi and Tapan Mukerji. Physics-informed PointNet: A deep learning solver for steady-state incompressible flows and thermal fields on multiple sets of irregular geometries. *Journal of Computational Physics*, 468:111510, 2022.

Ali Kashefi and Tapan Mukerji. Prediction of fluid flow in porous media by sparse observations and physics-informed pointnet. *Neural Networks*, 167:80–91, 2023.

Ali Kashefi and Tapan Mukerji. Physics-informed kan pointnet: Deep learning for simultaneous solutions to inverse problems in incompressible flow on numerous irregular geometries. *International Journal for Numerical Methods in Fluids*, 2026.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *35th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *9th International Conference on Learning Representations (ICLR)*, 2021.

Joowon Lim and Demetri Psaltis. MaxwellNet: Physics-driven deep neural network training based on maxwell's equations. *APL Photonics*, 7(1):011301, 2022.

Qiang Liu, Mengyu Chu, and Nils Thuerey. ConFIG: Towards conflict-free training of physics informed neural networks. In *13th International Conference on Learning Representations (ICLR)*, 2025.

Songming Liu, Zhongkai Hao, Chengyang Ying, Hang Su, Jun Zhu, and Ze Cheng. A unified hard-constraint framework for solving geometrically complex PDEs. In *36th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Yichen Luo, Jia Wang, Dapeng Lan, Yu Liu, and Zhibo Pang. MMET: A multi-input and multi-scale transformer for efficient PDEs solving. In *34th International Joint Conference on Artificial Intelligence (IJCAI)*, 2025.

Yesom Park, Changhoon Song, and Myungjoo Kang. Beyond derivative pathology of pinns: Variable splitting strategy with convergence analysis. *Journal of Machine Learning Research*, 2024.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. 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, Feb 2019.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training PINNs: a loss landscape perspective. In *41st International Conference on Machine Learning (ICML)*, 2024.

E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, and T. Rabczuk. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362:112790, 2020.

Jin Song and Zhenya Yan. AttnPINNs: Physics-informed neural networks under the self-attention mechanism for solving partial differential equations. *SIAM Journal on Scientific Computing*, 48 (4):C788–C813, 2026.

Jingzhi Tu, Chun Liu, and Pian Qi. Physics-informed neural network integrating pointnet-based adaptive refinement for investigating crack propagation in industrial applications. *IEEE Transactions on Industrial Informatics*, 19(2):2210–2218, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *31st Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Chu Wang, Jinhong Wu, Yanzhi Wang, Zhijian Zha, and Qi Zhou. MPIPN: a multi physics-informed pointnet for solving parametric acoustic-structure systems. *Engineering with Computers*, 41(1): 225–246, 2025.

Chuwei Wang, Shanda Li, Di He, and Liwei Wang. Is L2 physics-informed loss always suitable for training physics-informed neural network? In *36th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40):eabi8605, 2021b.

Chenhui Xu, Dancheng Liu, Yuting Hu, Jiajie Li, Ruiyang Qin, Qingxiao Zheng, and Jinjun Xiong. Sub-sequential physics-informed learning with state space model. In *42nd International Conference on Machine Learning (ICML)*, 2025.

Zhiyuan Zhao, Xueying Ding, and B. Aditya Prakash. PINNsFormer: A transformer-based framework for physics-informed neural networks. In *12th International Conference on Learning Representations (ICLR)*, 2024.

Weiheng Zhong and Hadi Meidani. Physics-informed discretization-independent deep compositional operator network. *Computer Methods in Applied Mechanics and Engineering*, 431:117274, 2024.

Weiheng Zhong and Hadi Meidani. Physics-informed geometry-aware neural operator. *Computer Methods in Applied Mechanics and Engineering*, 434:117540, 2025.

# A Theoretical Proof

## A.1 Proof of Theorem 5.2

To rigorously prove Theorem 5.2, we first clarify the role of random resampling. Let

$$X_t = \{x_{t,1}, \ldots, x_{t,N}\} \subset \Omega, \qquad N < \infty, \tag{24}$$

denote the finite collocation set sampled at iteration $t$, and suppose that

$$\overline{\bigcup_{t=1}^{\infty} X_t} = \Omega. \tag{25}$$

For a point-wise model $u_\theta^{\mathrm{s}}(x)$, all sampled batches constrain the same residual function

$$r_\theta^{\mathrm{s}}(x) := \mathcal{N}[u_\theta^{\mathrm{s}}](x) - f(x). \tag{26}$$

Therefore, if $r_\theta^{\mathrm{s}}$ is continuous and vanishes on every sampled collocation point, then it vanishes on the dense set $\bigcup_{t=1}^{\infty} X_t$, and hence on the entire domain $\Omega$. In contrast, a fusion-based model induces a batch-dependent family of functions

$$h_t(x) := u_\theta^{\mathrm{f}}\big(x, \mathrm{sg}[\phi(X_t)]\big). \tag{27}$$

Although the union of all sampled collocation points may be dense in $\Omega$, the residual constraints associated with different batches are imposed on different functions $h_t$. Consequently, the density of $\bigcup_{t=1}^{\infty} X_t$ does not imply that any fixed $h_t$ satisfies the governing PDE throughout $\Omega$.

For reasonable PINN implementations, the following assumptions can be satisfied either exactly or approximately.

**Assumption A.1** (Sufficient Expressive Capacity of Model)**.** For a model with sufficient capacity, assume that there exists a parameter $\theta^\star \in \Theta$ such that the fusion-based model $u_{\theta^\star}^{\mathrm{f}}$ satisfies

$$\mathcal{N}^{\mathrm{AD}}\big[u_{\theta^\star}^{\mathrm{f}}\big](x) = f(x), \qquad \forall\, x \in X_t, \quad \forall\, t, \tag{28}$$

where $\mathcal{N}^{\mathrm{AD}}$ is calculated by AD through the complete computational graph induced by $X_t$. Equivalently, $u_{\theta^\star}^{\mathrm{f}}$ satisfies the PDE loss everywhere in the domain $\Omega$.

**Assumption A.2** (Non-Degeneracy of the Fusion-Based Model)**.** Assume that the optimal parameter $\theta^\star$ in Assumption A.1 is non-degenerate such that the fusion mechanism contributes nontrivially to the AD residual. Specifically, there exists at least one point $x_0 \in X_t$ and an integer $k \geq 1$ such that

$$\partial_x^{k,\mathrm{AD}}\, u_{\theta^\star}^{\mathrm{f}}(x_0, \phi(X_t)) \neq \partial_x^k u_{\theta^\star}^{\mathrm{f}}\big(x_0, \mathrm{sg}[\phi(X_t)]\big), \tag{29}$$

where the left-hand side is computed by full-graph AD, and the right-hand side is the term that only takes the partial derivative with respect to the point-wise components.

**Assumption A.3** (Local Detectability of the Governing PDE)**.** Let the PDE operator be written in jet form

$$\Psi\big(x, j_x^k u\big) := \mathcal{N}[u](x), \tag{30}$$

where $j_x^k u = \big(u(x), \partial_x u(x), \ldots, \partial_x^k u(x)\big)$ denotes the classical $k$-jet of $u$ at $x$. Let $x_0$ be the point in Assumption A.2, and let $z_0 := j_{x_0}^k u$ and $\delta_0 := j_{x_0}^k u - j_{x_0}^k h$. We assume that the map $j_x^k u \mapsto \Psi(x_0, j_x^k u)$ is locally sensitive along the perturbation direction $\delta_0$, i.e., the perturbation $\delta_0$ caused by decoupled gradients, such that

$$\Psi(x_0, z_0) \neq \Psi(x_0, z_0 - \delta_0). \tag{31}$$

That is, there exists a direction near point $x_0$ such that changing the $k$-th derivative in this direction will cause a change in the value of the differential operator $\mathcal{N}$.

These assumptions are reasonably mild within the analytical framework of this paper. Assumption A.1 is an idealized sufficient-capacity condition corresponding to the zero-residual limit of physics-informed training, which is grounded in results from the Universal Approximation Theorem (Hornik et al., 1989) that underpin the expressive capacity of neural networks. Assumption A.2 holds whenever the fusion pathway contributes nontrivially to the AD derivatives for at least one finite collocation set,

and fails when the model degenerates into a point-wise mapping that ignores the fused feature $\phi(X_t)$. Assumption A.3 is a zero-measure non-cancellation condition requiring the governing operator to detect fusion-induced jet perturbation at at least one collocation point. Our DCI directly quantifies Assumptions A.2 and A.3, which hold whenever the DCI is non-zero. Consequently, our experiments in Section 7 substantiate that these two assumptions almost invariably hold in practice.

Now, we present the strict version of Theorem 5.2:

**Theorem A.4** (Non-Representability of Fusion-Based PINNs)**.** *Suppose that*

$$\mathcal{S} := \{u \in \mathcal{F}(\Omega) \,|\, \mathcal{N}[u](x) = f(x),\ \forall x \in \Omega\} \tag{32}$$

*is the set of all solutions that satisfy the governing PDE. Suppose that there exists a parameter $\theta^\star$ such that the fusion-based model $u^{\mathrm{f}}_{\theta^\star}$ satisfies Assumptions A.1, A.2, and A.3. Then, there exists a finite collocation set $X_{t_0}$ such that*

$$u^{\mathrm{f}}_{\theta^\star}\big(\,\cdot, \phi(X_{t_0})\big) \notin \mathcal{S}. \tag{33}$$

*That is, a fusion-based model $u^{\mathrm{f}}_{\theta^\star}$ trained by minimizing the AD residuals may not correspond to any valid solution $u^\star \in \mathcal{S}$, even if the PDE residual is zero over the domain $\Omega$.*

*Proof.* Define the frozen-fusion function

$$h(x) := u^{\mathrm{f}}_{\theta^\star}\big(x, \mathrm{sg}[\phi(X_{t_0})]\big). \tag{34}$$

Notice that $h(x)$ and $u^{\mathrm{f}}_{\theta^\star}(x, \phi(X_{t_0}))$ coincide pointwise, but their derivatives differ because AD differentiates through the fusion map $\phi(X_{t_0})$ when $x \in X_{t_0}$. For every integer $k \geq 1$, the $k$-th AD derivative admits the decomposition

$$\partial_x^{k,\mathrm{AD}} u^{\mathrm{f}}_{\theta^\star}\big(x, \phi(X_{t_0})\big) = \partial_x^k h(x) + \mathcal{C}_k(x; \theta^\star, X_{t_0}), \tag{35}$$

where $\mathcal{C}_k(x; \theta^\star, X_{t_0})$ is the sum of all terms containing at least one factor involving a derivative of $\phi(X_{t_0})$ with respect to $x$. Equivalently, by the higher-order chain rule (multivariate Faà di Bruno formula),

$$\mathcal{C}_k(x; \theta^\star, X_{t_0}) = \sum_{p=1}^{k} \binom{k}{p} \sum_{q=1}^{p} \big(\partial_x^{k-p} \partial_\phi^q u^{\mathrm{f}}_{\theta^\star}(x, \phi)\big) \times B_{p,q}\big(\partial_x \phi(X_{t_0}), \ldots, \partial_x^{p-q+1} \phi(X_{t_0})\big), \tag{36}$$

where $B_{p,q}$ denotes the partial Bell polynomial.

Consider the parameter $\theta^\star$ such that $u^{\mathrm{f}}_{\theta^\star}$ satisfies Assumption A.1, i.e., $\mathcal{N}^{\mathrm{AD}}[u^{\mathrm{f}}_{\theta^\star}](x) = f(x)$ for all $x \in X_{t_0}$, and also satisfies Assumption A.2 at the point $x_0 \in X_{t_0}$, i.e., $\mathcal{C}_k(x_0; \theta^\star, X_{t_0}) \neq 0$. During training, the residual is evaluated using AD derivatives:

$$\Psi\big(x, j_x^{k,\mathrm{AD}} u^{\mathrm{f}}_{\theta^\star}\big) = f(x), \qquad \forall x \in X_{t_0}, \tag{37}$$

where $j_x^{k,\mathrm{AD}} u^{\mathrm{f}}_{\theta^\star}$ denotes the jet of the model obtained through AD. By Equation 35, the AD jet can be decomposed as

$$j_x^{k,\mathrm{AD}} u^{\mathrm{f}}_{\theta^\star} = j_x^k h + \boldsymbol{\delta}_x^k, \qquad \boldsymbol{\delta}_x^k := \big(0, \mathcal{C}_1(x; \theta^\star, X_{t_0}), \ldots, \mathcal{C}_k(x; \theta^\star, X_{t_0})\big). \tag{38}$$

Therefore, by Assumption A.2, we have $\boldsymbol{\delta}_{x_0}^k \neq 0$ at the point $x_0$, and by Assumption A.3, we have

$$\Psi\big(x_0, j_{x_0}^{k,\mathrm{AD}} u^{\mathrm{f}}_{\theta^\star}\big) \neq \Psi\big(x_0, j_{x_0}^k h\big). \tag{39}$$

Since $\Psi$ is continuous and differentiable with respect to the jet variables, the map $x \mapsto \Psi\big(x, j_x^{k,\mathrm{AD}} u^{\mathrm{f}}_{\theta^\star}\big) - \Psi\big(x, j_x^k h\big)$ is also continuous. Therefore,

$$\mathcal{N}[h](x_0) \neq f(x_0), \qquad x_0 \in X_{t_0}. \tag{40}$$

Hence, $h(\cdot) \notin \mathcal{S}$. Since $h$ and $u^{\mathrm{f}}_{\theta^\star}$ coincide pointwise, it follows that $u^{\mathrm{f}}_{\theta^\star}\big(\,\cdot, \phi(X_{t_0})\big) \notin \mathcal{S}$. □

### A.2 Proof of Theorem 5.3

Consider a complete BVP. We will first strengthen Assumption A.1 to a version that simultaneously satisfies the PDE residuals and the boundary conditions, and assume that the problem is well-posed.

**Assumption A.5** (Uniqueness of the Solution)**.** Assume the governing PDE with the given boundary condition is well-posed and admits a unique solution $u^\star$.

**Assumption A.6** (Sufficient Expressive Capacity of Model Under BVP)**.** Assume that there exists $\theta^\star \in \Theta$ such that the fusion-based model $u^{\mathrm{f}}_{\theta^\star}$ satisfies both the PDE residual and the boundary condition losses:

$$\begin{aligned} \mathcal{N}^{\mathrm{AD}}[u^{\mathrm{f}}_{\theta^\star}](x) &= f(x), \quad x \in X_{\mathrm{res}}, \\ \mathcal{B}^{\mathrm{AD}}[u^{\mathrm{f}}_{\theta^\star}](x) &= g(x), \quad x \in X_{\mathrm{bc}}, \end{aligned} \tag{41}$$

where the operators $\mathcal{N}^{\mathrm{AD}}[u_{\theta^\star}](x)$ and $\mathcal{B}^{\mathrm{AD}}[u_{\theta^\star}](x)$ are calculated by AD, and $X_{\mathrm{res}} \subset \Omega$ is the internal collocation point set, $X_{\mathrm{bc}} \subset \Gamma$ is the boundary collocation point set.

Now, we present the strict version of Theorem 5.3:

**Theorem A.7** (False Solution under BVP)**.** *Suppose that a fusion-based model $u^{\mathrm{f}}_\theta$ is used to solve a PDE subject to a BVP that is well-posed in the sense of Assumption A.5 and admits a unique solution $u^\star$. Under Assumptions A.2, A.3, and A.6, with the non-degeneracy and detectability conditions holding at some $x_0 \in X_{\mathrm{res}}$, there exists a parameter $\theta^\star \in \Theta$ such that*

$$u^{\mathrm{f}}_{\theta^\star}\big(\cdot, \phi(X)\big) \neq u^\star(\cdot). \tag{42}$$

*That is, the fusion-based model $u^{\mathrm{f}}_\theta$ satisfies the AD residual loss, but the resulting function does not coincide with the true solution.*

*Proof.* Let $u^\star$ be the unique classical solution guaranteed by Assumption A.5. In the notation of Appendix A.1, write the PDE operator in jet form as

$$\Psi_{\mathrm{res}}\big(x, j^k_x u\big) := \mathcal{N}[u](x), \tag{43}$$

where $k \geq 1$ is the maximal derivative order appearing in $\mathcal{N}$, and the subscript "res" in $\Psi_{\mathrm{res}}$ is added to distinguish it from the boundary operator. We define the pseudo-source term induced in the PDE residual by the fusion derivatives:

$$s_{\mathrm{res}}(x) := \Psi_{\mathrm{res}}\big(x, j^{k,\mathrm{AD}}_x u^{\mathrm{f}}_{\theta^\star}\big) - \Psi_{\mathrm{res}}\big(x, j^k_x h\big), \tag{44}$$

where $h(x) := u^{\mathrm{f}}_{\theta^\star}\big(x, \mathrm{sg}[\phi(X)]\big)$ follows the definition given earlier.

Then, by the definition of $\Psi_{\mathrm{res}}$, for every $x \in X_{\mathrm{res}}$,

$$\begin{aligned} f(x) &= \Psi_{\mathrm{res}}\big(x, j^{k,\mathrm{AD}}_x u^{\mathrm{f}}_{\theta^\star}\big) \\ &= \Psi_{\mathrm{res}}\big(x, j^k_x h\big) + s_{\mathrm{res}}(x) \\ &= \mathcal{N}[h](x) + s_{\mathrm{res}}(x), \end{aligned} \tag{45}$$

hence

$$\mathcal{N}[h](x) = f(x) - s_{\mathrm{res}}(x), \qquad \forall x \in X_{\mathrm{res}}. \tag{46}$$

For the boundary term, the same decomposition may in general introduce an analogous discrepancy when $\mathcal{B}$ contains derivatives. However, the proof below only requires the interior pseudo-source $s_{\mathrm{res}}$, and does not rely on any nonvanishing property of the boundary discrepancy. In particular, under pure Dirichlet boundary conditions, this boundary discrepancy vanishes identically by Proposition 5.1. Thus, optimizing the AD residual and boundary losses only enforces that the AD-evaluated quantities vanish. It does not require the interior pseudo-source term $s_{\mathrm{res}}$ to vanish.

Following the proof of Theorem A.4, and utilizing Assumptions A.2 and A.3, it can be demonstrated that there exists at least one collocation point $x0 \in X_{\mathrm{res}}$ such that $s_{\mathrm{res}} \neq 0$, and $h$ locally fails to satisfy the original PDE. Consequently, $h(\cdot)$ cannot be the true solution to the original BVP, while $u^{\mathrm{f}}_{\theta^\star}\big(\cdot, \phi(X)\big)$ is also not the true solution. Therefore, we will not elaborate on the remaining proofs here. □

### A.3 A Concrete Counterexample

We provide an explicitly constructed counterexample to illustrate the discrepancy established in Appendix A.1, and demonstrate how the fusion-based model constructs a solution that strictly satisfies the PDE while not belonging to the PDE solution set.

Consider the one-dimensional ODE

$$\mathcal{N}[u](x) := u'(x) = 0, \tag{47}$$

whose solution set consists of all constant functions. For a collocation batch of fixed size $X = \{x_1, \dots, x_N\}$, consider the fusion-based model

$$u_i^{\mathrm{f}} = u^{\mathrm{f}}(x_i, \phi(X)) = ax_i + b\bar{x}, \qquad \phi(X) = \bar{x} = \frac{1}{N}\sum_{j=1}^{N} x_j \tag{48}$$

with parameters $a$ and $b$.

Let $X^{(i\leftarrow t)}$ denote the batch obtained by replacing $x_i$ with $t$. The full-graph perturbation path is

$$\begin{aligned} g_i(t) &:= u^{\mathrm{f}}\big(t, \phi(X^{(i\leftarrow t)})\big) \\ &= at + b\left(\bar{x} + \frac{t - x_i}{N}\right), \end{aligned} \tag{49}$$

whereas the frozen-context spatial path is

$$\begin{aligned} h_i(t) &:= u^{\mathrm{f}}\big(t, \mathrm{sg}[\phi(X)]\big) \\ &= at + b\bar{x}. \end{aligned} \tag{50}$$

At the original collocation point, the two paths have the same forward value:

$$g_i(x_i) = h_i(x_i) = u_i^{\mathrm{f}}. \tag{51}$$

However, their derivatives are

$$g_i'(x_i) = a + \frac{b}{N}, \qquad h_i'(x_i) = a. \tag{52}$$

Choose $a = -\frac{b}{N}$ with $b \neq 0$. Then, for every point in every batch of size $N$, full-graph AD gives

$$\mathcal{N}^{\mathrm{AD}}[u_i^{\mathrm{f}}] = g_i'(x_i) = a + \frac{b}{N} = 0. \tag{53}$$

Thus, the AD residual is identically zero. In contrast, when the fusion context is held fixed,

$$\mathcal{N}[h_i](t) = h_i'(t) = -\frac{b}{N} \neq 0. \tag{54}$$

Meanwhile, the forward function

$$u^{\mathrm{f}}\big(t, \phi(X^{(i\leftarrow t)})\big) = -\frac{b}{N}t + b\bar{x} \tag{55}$$

is nonconstant, and thus does not belong to the solution set of the original ODE.

## B Benchmarks

Unlike papers proposing new PINN methods, we deliberately chose three simple benchmarks follows prior failure analysis literature (Wang et al., 2021a; Krishnapriyan et al., 2021). The selection of these benchmarks aims to isolate other failure modes (such as spectral bias and gradient conflicts), and is intended to more distinctly **discern the proposed failure mode**, rather than to **benchmark model performance**. Consequently, simple benchmarks are more conducive to identifying the problem, whereas the additional confounding factors introduced by complex scenarios instead hinder clearly mechanistic identification.

**Poisson.** We consider a 2D Poisson benchmark on the unit square $\Omega = [0,1] \times [0,1]$. The task is to solve for a scalar field $u(x,y)$ governed by a Poisson equation with a prescribed source term and Dirichlet boundary conditions on the entire boundary $\Gamma$. This benchmark evaluates a model's ability to handle complex, strongly nonlinear boundary conditions.

Inside the domain, $u$ satisfies

$$\Delta u(x,y) = f(x,y), \qquad (x,y) \in \Omega, \tag{56}$$

where the source term is defined as

$$f(x,y) = -(2\pi)^2 A \sin(2\pi x) - (3\pi)^2 B \cos(3\pi y) - 2\pi^2 \sin(\pi x)\sin(\pi y), \tag{57}$$

with constants $A = 3$, $B = 2$.

Dirichlet boundary conditions are imposed on the whole boundary:

$$u(x,y) = g(x,y), \qquad (x,y) \in \Gamma, \tag{58}$$

where

$$g(x,y) = A\sin(2\pi x) + B\cos(3\pi y) + Cx + Dy, \tag{59}$$

and $C = 2$, $D = 1$.

We construct this Poisson problem via the manufactured solutions. The exact solution is given by

$$u^*(x,y) = g(x,y) + \sin(\pi x)\sin(\pi y), \tag{60}$$

which satisfies the prescribed Dirichlet boundary conditions and yields the source term $f(x,y) = \Delta u^*(x,y)$. We use this analytical solution as the ground truth for quantitative error evaluation.

**Helmholtz.** We consider a 2D Helmholtz-type benchmark defined on a bounded, irregular domain $\Omega \subset \mathbb{R}^2$. The task is to solve for a scalar field $u(x,y)$ governed by a variable-coefficient second-order elliptic equation with a reaction term. This benchmark tests models' ability to handle spatially varying anisotropic operators and nontrivial domain geometries.

Inside the domain, $u$ satisfies a generalized Helmholtz equation of the form

$$-\nabla \cdot \big(A(x,y)\nabla u(x,y)\big) + q(x,y)\, u(x,y) = f(x,y), \qquad (x,y) \in \Omega, \tag{61}$$

where the diffusion tensor $A(x,y) \in \mathbb{R}^{2\times 2}$ and the reaction coefficient $q(x,y)$ are given by

$$\begin{aligned} A(x,y) &= \begin{pmatrix} 1 + 0.3x & 0.15\sin(\pi x)\sin(\pi y) \\ 0.15\sin(\pi x)\sin(\pi y) & 1 + 0.3y \end{pmatrix}, \\ q(x,y) &= 2 + \cos(\pi x)\cos(\pi y). \end{aligned} \tag{62}$$

Dirichlet boundary conditions are imposed on the entire boundary:

$$u(x,y) = u^*(x,y), \qquad (x,y) \in \Gamma, \tag{63}$$

where $u^*$ is a prescribed exact solution.

The benchmark is constructed using a manufactured solution,

$$u^*(x,y) = \sin(\pi x)\cos(2\pi y) + 0.2\, e^{x+y}. \tag{64}$$

The source term $f(x,y)$ is obtained analytically by substituting $u^*$ into the governing equation. This exact solution satisfies the Dirichlet boundary conditions and serves as the ground truth for quantitative error evaluation.

**Heat.** We consider a 2D steady-state heat conduction benchmark on a square domain $\Omega = [0,5] \times [0,5]$. The goal is to learn/solve a scalar temperature field $u(x,y)$ that satisfies a homogeneous elliptic PDE in the interior and mixed (Dirichlet/Neumann) boundary conditions on $\Gamma$. This benchmark evaluates a model's ability to handle mixed boundary conditions.

Inside the domain, $u$ satisfies the Laplace equation

$$\Delta u(x,y) \;=\; \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \;=\; 0, \qquad (x,y) \in \Omega. \tag{65}$$

We impose Dirichlet conditions on the left and bottom boundaries,

$$u(0, y) = T_0, \quad u(x, 0) = T_0, \qquad T_0 = 100, \tag{66}$$

and Neumann conditions on the right and top boundaries (outward normals $(1, 0)$ and $(0, 1)$),

$$\frac{\partial u}{\partial x}(L, y) = -q, \quad \frac{\partial u}{\partial y}(x, L) = -q, \qquad q = 15. \tag{67}$$

These correspond to a constant prescribed boundary temperature on $\{x = 0\} \cup \{y = 0\}$ and a constant outward heat flux on $\{x = L\} \cup \{y = L\}$.

The analytical solution is computed via a truncated Fourier-series representation that satisfies the above mixed boundary conditions:

$$u(x, y) = T_0 + \sum_{m=1}^{M} A_n \Big( \sinh(\lambda x) \sin(\lambda y) + \sinh(\lambda y) \sin(\lambda x) \Big), \quad n = 2m - 1, \ \lambda = \frac{n\pi}{2L}, \tag{68}$$

with coefficients

$$A_n = \frac{-8qL}{k\, n^2 \pi^2 \cosh\big(\frac{n\pi}{2}\big)}, \qquad (k = 1), \tag{69}$$

where $k$ is the thermal conductivity, and we use $M = 20$ terms in practice to obtain an accurate numerical approximation of the analytical series solution.

# C Experimental Configuration

## C.1 Model Setting

In recent years, fusion-based PINNs have remained an active and expanding area of research, and **exhaustively evaluating all existing architectures and baselines in one paper is impossible**. Consequently, we focus on a representative subset of architectures to demonstrate the pervasive nature of the proposed derivative cheating phenomenon.

**MLP.** We employ a vanilla MLP configuration. The model settings are shown in Table 2.

Table 2: Configuration of MLP in all the benchmarks.

| Configuration | Value |
|---|---|
| Number of hidden layers | 4 |
| Neurons per hidden layer | 64 |
| Activation function | `Tanh` |

**MLP + BatchNorm.** To verify the failure cases of BatchNorm itself, we specifically designed this set of experiments. We adopt the same configuration as the vanilla MLP, but insert a batch normalization layer only in the second layer. As a control experiment, we adopted the decoupling scheme described in Section 6.2.

**PI-DeepONet (Wang et al., 2021b).** The original PI-DeepONet is point-wise, and thus will not cheat. However, many neural operators incorporating spatial fusion (as discussed in Section 3) raise the possibility of derivative cheating. Therefore, to verify cheating in geometric-aware backbones under neural operator-level fusion, we introduced the simplest fusion mode in the branch network for module-wise attribution: we modified the PI-DeepONet backbone and fed the entire point set $X$ into the branch network. We integrated some design elements from PI-GANO (Zhong & Meidani, 2024), employing mean pooling to aggregate global features across the point set. The model settings are shown in Table 3.

Table 3: Configuration of PI-DeepONet in all the benchmarks.

| Configuration | Value |
|---|---|
| Number of branch network layers | 2 |
| Number of trunk network layers | 2 |
| Neurons per hidden layer | 64 |
| Activation function | `Tanh` |

As a control experiment, we detach the input of the branch network, as described in Section 6.1.

**PIPN (Kashefi & Mukerji, 2022).** We adopted a standard PIPN backbone. The model settings are shown in Table 4.

Table 4: Configuration of PIPN in all the benchmarks.

| Configuration | Value |
|---|---|
| Layer width in MLP-1 | $[2, 32]$ |
| Layer width in MLP-2 | $[32, 64]$ |
| Layer width in MLP-3 | $[96, 64, 32]$ |
| Layer width in MLP-4 | $[32, 1]$ |
| Activation function | `Tanh` |

As a control experiment, we detach the gradients from the input of MLP-2, which corresponds to the decoupling strategy described in Section 6.1.

**MMET (Luo et al., 2025).** We directly use the official implementation of MMET. The model settings are shown in Table 5.

Table 5: Configuration of MMET in all the benchmarks.

| Configuration | Value |
|---|---|
| Number of attention layers in encoder | 2 |
| Number of attention layers in decoder | 2 |
| Number of heads in attention layer | 1 |
| Neurons per hidden layer | 32 |
| Neurons per GCE layer | 32 |
| Patch size | 1 |
| Activation function | `WaveAct` |

As a control experiment, we adopted the decoupling scheme described in Section 6.2 at the input of the attention block to verify the effectiveness of module-level decoupling.

### C.2 Hyperparameters

To ensure experimental consistency, identical training configurations were applied to all models, as detailed in Table 6. Specifically, at each iteration, we generated a new set of randomly sampled collocation points.

Table 6: Training hyperparameters in each benchmark. Where $\eta$ is the learning rate, $w_{\mathrm{res}}$ is the PDE residual loss weight, $w_{\mathrm{bc}}$ is the boundary condition loss weight, $N_{\mathrm{res}}$ is the number of internal collocation points, and $N_{\mathrm{bc}}$ is the number of boundary collocation points.

| **Benchmark** | $\eta$ | $w_{\mathrm{res}}$ | $w_{\mathrm{bc}}$ | $N_{\mathrm{res}}$ | $N_{\mathrm{bc}}$ |
|---|---|---|---|---|---|
| Poisson | 5.0e-4 | 1 | 10 | 2000 | 800 |
| Helmholtz | 5.0e-4 | 1 | 5 | 2000 | 800 |
| Heat | 1.0e-3 | 1 | 1 | 2000 | 800 |

Given the differences in model scale, it is unfair to specify a uniform number of required iterations for all models. Accordingly, Table 7 provides the iteration number for each model. Under this setting, we continued training until the loss no longer decreased.

Table 7: Training iteration number for each model in each benchmark.

| **Benchmark** | MLP | MLP+BatchNorm | PI-DeepONet | PIPN | MMET |
|---|---|---|---|---|---|
| Poisson | 10000 | 12000 | 12000 | 12000 | 10000 |
| Helmholtz | 10000 | 12000 | 12000 | 12000 | 10000 |
| Heat | 8000 | 10000 | 10000 | 10000 | 8000 |

For all the models, we use Adam as the optimizer. We implement all models and training settings in PyTorch and run training on four NVIDIA H200 GPUs.

## D Statistical Significance

To verify that the failure mode in Section 4 and its recovery in Section 7 are not artifacts of seed selection, we run every cell with $K = 5$ independent random seeds. We report the converged total loss $\mathcal{L}_{\mathrm{res}} + \mathcal{L}_{\mathrm{bc}}$ and relative $L_2$ error as mean $\pm$ std. Due to space limitations, the Figures 2 and 5 in our main text only report the means, while all other results are reported in Tables 8 and 9.

Table 8: Loss and relative $L_2$ error (mean$\pm$std over 5 seeds) on the failure-mode setting (Figure 2).

| **Metric** | **Model** | Poisson | Helmholtz | Heat |
|---|---|---|---|---|
| Loss | MLP | 1.16e-1 $\pm$ 1.4e-1 | 5.61e-2 $\pm$ 5.5e-3 | 1.19e-1 $\pm$ 2.4e-4 |
| | MLP+BatchNorm | 2.39e+4 $\pm$ 1.6e+1 | 4.74e-2 $\pm$ 5.9e-2 | 9.01e-3 $\pm$ 6.9e-3 |
| | PI-DeepONet | 3.48e-1 $\pm$ 8.0e-2 | 9.64e-2 $\pm$ 2.6e-2 | 5.68e-3 $\pm$ 1.0e-3 |
| | PIPN | 1.62e-1 $\pm$ 7.2e-2 | 4.62e-2 $\pm$ 2.7e-2 | 6.38e-2 $\pm$ 4.3e-2 |
| | MMET | 6.81e-1 $\pm$ 1.1e-1 | 8.49e-2 $\pm$ 4.3e-3 | 9.04e-2 $\pm$ 1.5e-3 |
| $L_2$ | MLP | 6.82e-3 $\pm$ 7.3e-4 | 2.00e-2 $\pm$ 1.2e-2 | 3.14e-3 $\pm$ 7.8e-7 |
| | MLP+BatchNorm | 7.14e-1 $\pm$ 2.6e-2 | 3.56e+0 $\pm$ 5.5e-2 | 1.10e+0 $\pm$ 7.8e-3 |
| | PI-DeepONet | 4.29e-1 $\pm$ 1.6e-1 | 7.02e-1 $\pm$ 1.7e-1 | 4.22e-1 $\pm$ 6.2e-2 |
| | PIPN | 1.99e-1 $\pm$ 2.9e-2 | 9.85e-1 $\pm$ 1.1e-1 | 6.24e-1 $\pm$ 1.0e-2 |
| | MMET | 1.66e-1 $\pm$ 9.1e-2 | 7.52e-1 $\pm$ 5.1e-2 | 1.07e+0 $\pm$ 6.4e-2 |

## E Additional Experimental Verification

We report results exclusively on standard PINN benchmarks in the main text to isolate confounding failure modes and provide a clearer understanding of the proposed derivative cheating. However, **this does not imply that the phenomenon is confined merely to simplistic benchmarks**. To verify whether derivative cheating also occurs in temporal dimension and complex non-linear benchmark, we conduct experiments on NS2d-LT from PINNacle (Hao et al., 2024), which is a long-horizon nonlinear time-dependent Navier-Stokes problem. We consider two representative temporal-fusion backbones, PINNsFormer (Zhao et al., 2024) and PINNMamba (Xu et al., 2025). For both models,

Table 9: Loss and relative $L_2$ error (mean±std over 5 seeds) after gradient decoupling (Figure 5).

| Metric | Model | Poisson | Helmholtz | Heat |
|---|---|---|---|---|
| Loss | MLP | 1.16e-1 ± 1.4e-1 | 5.61e-2 ± 5.5e-3 | 1.19e-1 ± 2.4e-4 |
| | MLP+BatchNorm | 2.21e-1 ± 2.6e-1 | 1.34e-3 ± 7.2e-4 | 2.56e-1 ± 1.7e-1 |
| | PI-DeepONet | 3.82e-1 ± 3.5e-1 | 1.56e-1 ± 1.7e-2 | 3.21e-1 ± 2.0e-1 |
| | PIPN | 6.33e-1 ± 2.3e-1 | 5.42e-2 ± 5.0e-3 | 1.25e-1 ± 8.5e-3 |
| | MMET | 3.80e-1 ± 3.3e-1 | 5.23e-2 ± 2.3e-2 | 2.38e-1 ± 5.9e-3 |
| $L_2$ | MLP | 6.82e-3 ± 7.3e-4 | 2.00e-2 ± 1.2e-2 | 3.14e-3 ± 7.8e-7 |
| | MLP+BatchNorm | 1.49e-2 ± 6.0e-3 | 1.70e-2 ± 2.6e-2 | 7.09e-3 ± 1.0e-3 |
| | PI-DeepONet | 1.08e-2 ± 1.6e-2 | 3.65e-2 ± 7.4e-2 | 8.70e-3 ± 5.5e-3 |
| | PIPN | 1.84e-2 ± 1.9e-3 | 4.24e-2 ± 1.9e-2 | 9.40e-3 ± 9.6e-4 |
| | MMET | 1.71e-2 ± 5.3e-3 | 1.84e-3 ± 3.6e-3 | 7.68e-3 ± 5.3e-4 |

we preserve their original forward computation and temporal fusion mechanisms, and only decouple the fusion-induced gradient paths when computing the PDE derivatives.

**PINNsFormer (Zhao et al., 2024).** PINNsFormer constructs temporal pseudo-sequences from the spatio-temporal input and applies attention along the temporal dimension. Since the queried temporal coordinate also participates in the construction of the pseudo-sequence, it enters the output through both the point-wise pathway and the temporal fusion pathway. Therefore, the AD-computed temporal derivatives may contain the same fusion-induced components analyzed in Section 5. In the decoupled model, we preserve the complete pseudo-sequence construction and attention computation in the forward pass, while treating the temporal fusion context as auxiliary variables during AD.

**PINNMamba (Xu et al., 2025).** PINNMamba performs sub-sequential feature fusion along the temporal dimension using a state-space backbone. This setting also satisfies the general fusion formulation in Equation 6, except that the fusion is performed over temporal coordinates rather than spatial collocation points. We therefore apply the same gradient-decoupling principle to the temporal fusion pathway while retaining the original forward model.

The results are summarized in Table 10. Under standard AD, both models achieve small residual losses but exhibit large relative $L_2$ errors and high DCI. PINNsFormer obtains a loss of $3.72 \times 10^{-3}$ while its relative $L_2$ error reaches $6.84 \times 10^{-1}$, with a DCI of $8.91 \times 10^{-1}$. Similarly, PINNMamba achieves a loss of $4.16 \times 10^{-3}$ but has a relative $L_2$ error of $6.27 \times 10^{-1}$ and a DCI of $8.74 \times 10^{-1}$. After gradient decoupling, the DCI of both models decreases to zero, while their relative errors are reduced to $4.73 \times 10^{-2}$ and $3.91 \times 10^{-2}$, respectively.

Table 10: Results of temporal fusion backbones on the nonlinear NS2d-LT benchmark.

| Model | Decouple | Loss | $L_2$ Error | DCI |
|---|---|---|---|---|
| PINNsFormer | × | 3.72e-3 | 6.84e-1 | 8.91e-1 |
| | ✓ | 1.26e-2 | 4.73e-2 | 0.00e+0 |
| PINNMamba | × | 4.16e-3 | 6.27e-1 | 8.74e-1 |
| | ✓ | 1.43e-2 | 3.91e-2 | 0.00e+0 |

These results demonstrate that derivative cheating is not restricted to spatial feature fusion or linear BVPs. In particular, the standard models obtain even smaller residual losses than their decoupled counterparts, while their solution errors are more than one order of magnitude larger. This observation is consistent with our analysis that fusion-induced derivatives provide additional paths to minimize the AD residual without satisfying the corresponding physical derivatives. Gradient decoupling removes these paths while preserving the temporal modeling mechanism in the forward computation.

# F CONTROLLED ANALYSIS OF DCI

Our proposed DCI is designed to measure the **semantic correctness of the PINN objective**, rather than claiming to directly predict solution accuracy. The latter is governed by multifaceted optimization factors and cannot be straightforwardly assessed via a single metric. The experiments in this section empirically substantiate the relationship between the DCI and prediction error upon full convergence, while isolating all confounding factors. Specifically, we replace the binary stop-gradient operation with the following continuous formulation:

$$\phi_\lambda(X) = \lambda\phi(X) + (1-\lambda)\,\mathrm{sg}[\phi(X)], \qquad \lambda \in [0,1], \tag{70}$$

where $\phi(X)$ denotes the fusion representation and $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator. Since

$$\phi_\lambda(X) = \phi(X) \tag{71}$$

in the forward pass for any $\lambda$, changing $\lambda$ does not alter the forward model output. Instead, it only scales the contribution of the fusion pathway to the AD derivative. In particular, $\lambda = 0$ corresponds to complete gradient decoupling, while $\lambda = 1$ recovers the original fusion-based formulation.

We conduct this experiment using PIPN on the Heat benchmark. Across all settings, the network architecture, initialization, optimizer, iteration number, random seed, and sampled collocation points are kept identical. Therefore, the only controlled variable is the strength of the fusion-induced derivative pathway.

Table 11: Controlled DCI sweep on PIPN for the Heat benchmark. Only the fusion-gradient strength $\lambda$ is varied, while all other experimental settings are kept identical.

| $\lambda$ | 0 | 0.01 | 0.05 | 0.10 | 1.00 |
|---|---|---|---|---|---|
| DCI | 0.00e+0 | 2.31e-1 | 4.86e-1 | 7.43e-1 | 9.00e-1 |
| AD Loss | 1.25e-1 | 8.72e-2 | 6.45e-2 | 5.92e-2 | 6.38e-2 |
| Relative $L_2$ | 9.40e-3 | 1.82e-1 | 3.22e-1 | 5.98e-1 | 6.24e-1 |

As shown in Table 11, increasing $\lambda$ produces a monotonic increase in DCI, accompanied by a monotonic increase in the relative $L_2$ error. The solution error increases by more than two orders of magnitude from $\lambda = 0$ to $\lambda = 1$, whereas the AD loss remains at a similar scale and even decreases over most of the sweep. This shows that a lower AD residual does not imply a more accurate physical solution when fusion-induced derivatives contribute to the residual.

This experiment isolates the effect measured by DCI from differences in model capacity, forward computation, initialization, data sampling, and optimization. Therefore, among these otherwise identical and successfully optimized models, a larger DCI consistently corresponds to a larger solution error. We emphasize that this result does not imply that DCI alone determines whether a PINN converges. DCI measures the discrepancy introduced by derivative cheating, while other known failure modes may still cause inaccurate solutions even when DCI is zero.

# G ABLATION STUDY

In this section, we conduct two ablation studies, using a toy MLP and a practical Transformer-based backbone, to intuitively illustrate the impact of derivative cheating on PINN optimization.

## G.1 MINIMAL ABLATION ON TOY MLP

We first use the Heat benchmark with a standard MLP architecture, keeping all configurations identical to the main experiment, except for incorporating the coordinate mean of all collocation points into the input. We compare four model variants:

*(a)* Vanilla MLP, identical to that used in the main experiment;

*(b)* MLP that augments the input by concatenating the global mean of all points, defined as

$$x' = \mathrm{concat}\big(x, \mathrm{mean}(X)\big); \tag{72}$$

*(c)* an MLP with the same concatenation, but retaining only the gradient with respect to the current point while removing cross-point gradient interactions, defined as

$$x' = \mathrm{concat}\big(x, \mathrm{mean}(\mathrm{sg}[X]) + \frac{1}{N}(x - \mathrm{sg}[x])\big), \tag{73}$$

which is functionally equivalent to detaching the gradients of all points in $X$ except for point $x$ itself;

*(d)* MLP that concatenates the input with the global mean while fully detaching the fusion points $X$, defined as

$$x' = \mathrm{concat}\big(x, \mathrm{mean}(\mathrm{sg}[X])\big). \tag{74}$$

The input values in settings *(b)*, *(c)*, and *(d)* are identical; their differences arise solely from the gradient propagation mechanisms. Specifically, *(b)* introduces a fusion term with global cross-point gradient interactions, *(c)* retains only the gradients associated with the local coordinates while removing cross-point contributions, and *(d)* fully decouples the fusion term by treating it as a constant input.

Assuming a dense distribution of collocation points, $\mathrm{mean}(X)$ converges to the constant geometric center of the domain $\Omega$. Therefore, this constant theoretically does not influence the model's predictive accuracy. The crucial distinction is that during AD, the computation graph is explicitly routed through both $x$ and $X$ in *(b)* and *(c)*. Among these, setting *(b)* involves fusion across the entire point set and thus introduces cross-point derivatives. Consequently, it leads to derivative cheating from an engineering perspective. In contrast, the derivative in setting *(c)* remains point-wise with respect to the input, yielding a diagonal Jacobian matrix and avoiding such bias at the engineering level. However, according to Theorem 5.2, it still introduces fusion-induced derivative terms $\partial\phi/\partial x$ that lack physical meaning, and therefore still leads to derivative cheating from a theoretical standpoint.

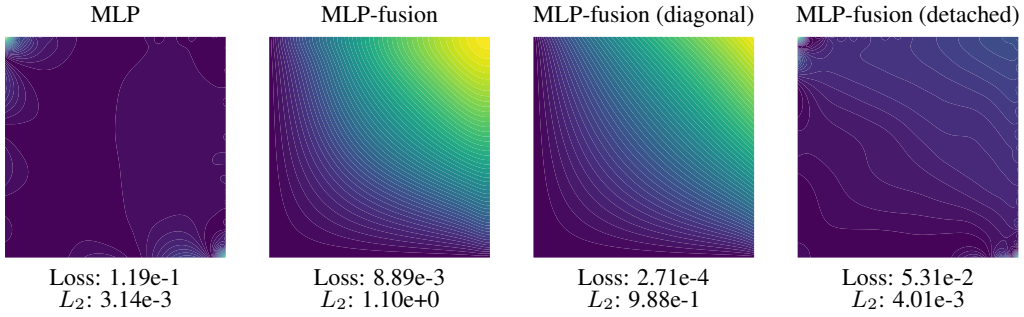



Figure 6: Error maps of the standard MLP, with fusion-based setting and gradient-decoupled fusion-based setting.

Figure 6 presents a comparison of model errors for all settings. As illustrated, settings *(b)* and *(c)* fail completely despite achieving a significantly lower loss than the vanilla MLP, with its $L_2$ error nearly $100\%$. The error distribution exhibits characteristics consistent with those described in Section 4. However, when we decouple the gradient of $X$, model performance recovers immediately. This directly validates the theoretical analysis in Section 5: fusion-based models introduce additional derivative terms at both the engineering and theoretical levels, thereby leading to derivative cheating. Importantly, these two effects are orthogonal.

### G.2 Ablation on Genuine Transformer

The minimal study in Appendix G.1 establishes that engineering-level and theoretical-level cheating are orthogonal contributions in a toy fusion design based on $\mathrm{mean}(X)$. A natural concern is whether this clean decomposition is an artifact of the linear, parameter-free aggregation, or a structural property of fusion mechanisms in general. To address this, we replicate the four-quadrant ablation on a genuine Transformer backbone that is modified from the encoder-only version of MMET, where the fusion operator is nonlinear, learnable, and tightly coupled with the point-wise pathway.

Unlike $\mathrm{mean}(X)$, an attention block does not admit a trivial separation between point-wise and fusion contributions. The same input $X$ simultaneously produces queries, keys, and values, and the

fusion is realized through two successive matrix products $QK^\top$ and $\mathrm{softmax}(\cdot)V$. To separate the engineering and theoretical components, we replace each matrix product with a gradient-masked variant that decouples the diagonal and off-diagonal entries. Specifically, we decompose $MN$ into outer products $M_{:,i}N_{i,:}$ and apply a 3D binary mask $\mathcal{W} \in \{0,1\}^{n\times l\times m}$ to control gradient flow: masked terms use $\mathrm{sg}[\cdot]$, while others remain fully differentiable. The masked product is defined as

$$M \odot_{\mathcal{W}} N \triangleq \sum_{i=1}^{m} \Big( \big(M_{:,i}\mathrm{sg}[N_{i,:}]\big) \circ \mathcal{W}_{:,:,i} + \big(M_{:,i}N_{i,:}\big) \circ (\mathbf{J} - \mathcal{W}_{:,:,i}) \Big), \tag{75}$$

where $\circ$ denotes the Hadamard product and $\mathbf{J}$ is the all-ones tensor. By construction, $M \odot_{\mathcal{W}} N$ yields the same forward value as $MN$, but detaches gradients at the positions indicated by the mask $\mathcal{W}$. We apply this operator to both dot products in attention with separate masks $\mathcal{W}_K$ and $\mathcal{W}_V$:

$$\mathrm{Attention}_{\mathcal{D}}(Q,K,V) = \mathrm{softmax}\left(\frac{Q \odot_{\mathcal{W}_K} K^\top}{\sqrt{d_k}}\right) \odot_{\mathcal{W}_V} V. \tag{76}$$

In the context of attention, the diagonal entries ($p = q$ in $QK^\top$, or $p = i$ in the attention-value product) correspond to a token's interaction with itself, i.e., the point-wise pathway; the off-diagonal entries correspond to inter-point fusion. Therefore, by adjusting these two masks $\mathcal{W}_K$ and $\mathcal{W}_V$, we obtain four variants of MMET:

(*a*) Vanilla encoder-only MMET;

*(b)* MMET with off-diagonal gradients masked, with mask

$$(\mathcal{W}_K)_{p,q,r} = \mathbb{1}[p \neq q], \qquad (\mathcal{W}_V)_{p,j,i} = \mathbb{1}[p \neq i], \tag{77}$$

which eliminates engineering-level cheating only;

*(c)* MMET with diagonal gradients masked, with mask

$$(\mathcal{W}_K)_{p,q,r} = \mathbb{1}[p = q], \qquad (\mathcal{W}_V)_{p,j,i} = \mathbb{1}[p = i], \tag{78}$$

which eliminates theoretical-level cheating only;

*(d)* MMET with both diagonal and off-diagonal fusion gradients masked, corresponding to the module-level decoupling of Equation 22.

The forward values of *(a)*, *(b)*, *(c)*, and *(d)* are identical by construction; differences arise solely from the structure of the gradient graph.

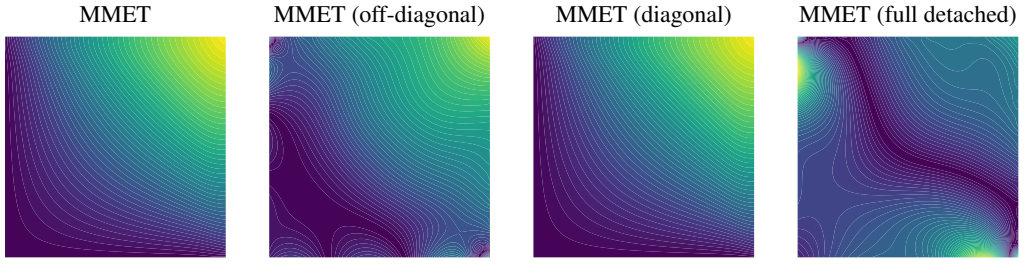



Figure 7: Error maps of the genuine MMET, with fusion-based setting and gradient-decoupled fusion-based setting.

Figure 7 presents the error maps of the four MMET variants. Consistent with the observations in Appendix G.1, both settings *(b)* and *(c)* fail completely despite achieving losses comparable to or lower than the vanilla MMET, with $L_2$ errors exceeding $70\%$ and $100\%$ respectively. The error distributions also exhibit the boundary-to-interior growth pattern characterized in Section 4, indicating that neither eliminating engineering-level cheating nor eliminating theoretical-level cheating alone is sufficient to recover a valid solution. Only after fully decoupling both pathways in setting *(d)* does the model converge to an accurate solution, with the $L_2$ error reduced by two orders of magnitude. This near-identical pattern to the toy study in Appendix G.1 confirms that the orthogonality between engineering- and theoretical-level cheating is intrinsic to the AD chain rule rather than an artifact of any specific fusion operator, thereby validating the generality of Theorem 5.2.

# H Potential Inductive Bias of Module-Level Decoupling

In this section, we investigate whether module-level gradient decoupling, while eliminating derivative cheating, changes the functional bias of fusion-based backbones by weakening their dependence on inter-point feature fusion.

## H.1 Source of the Inductive Bias

Architectural-level and module-level decoupling produce identical forward values but induce different gradient graphs, and the latter determines what function the residual loss actually constrains. For architectural-level decoupling, the point set $X$ enters the model only through a single, isolated fusion branch, and detaching it once at the branch input yields a frozen feature $\phi(\mathrm{sg}[X])$ that is a constant tensor with respect to $x$. The model collapses to a clean point-wise function

$$\tilde{u}_\theta^{\mathrm{arch}}(x) := u_\theta^{\mathrm{f}}\big(x,\, \phi(\mathrm{sg}[X])\big), \tag{79}$$

whose AD spatial derivative coincides exactly with the frozen-fusion function $h(x)$ defined in Section 5. Consequently, the residual loss enforces the PDE on $\tilde{u}_\theta^{\mathrm{arch}}$ directly, and the fusion branch retains its full expressive role in the forward pass while being invisible to differentiation.

For module-level decoupling, in contrast, the fusion and point-wise pathways are interleaved across multiple internal layers (e.g., BatchNorm or self-attention), and stop-gradients are inserted layer-by-layer, as in Equations 21 and 22. Because $\mathrm{sg}[\cdot]$ alters only the backward graph, the function evaluated at forward time and the function differentiated by AD are no longer the same. Concretely, for the decoupled attention block in Equation 22, the AD-effective function at any batch $X$ is

$$\tilde{u}_\theta^{\mathrm{mod}}(x\,;\, \bar{K}, \bar{V}) \;\approx\; \mathrm{softmax}\Big(\tfrac{Q(x)\bar{K}^\top}{\sqrt{d_k}}\Big)\,\bar{V}, \tag{80}$$

where $\bar{K}, \bar{V}$ are computed from $X$ but treated as constants under AD. Hence the residual loss is enforced on a family of point-wise functions $\{\tilde{u}_\theta^{\mathrm{mod}}(\,\cdot\,; \bar{K}_X, \bar{V}_X)\}_X$, parameterized by the collocation batch $X$. Under module-level decoupling, the fusion mechanism reduces to a contextual modulator that selects which point-wise function to use per batch, but it can no longer shape the spatial structure within any individual function. Architectural-level decoupling avoids this restriction because the frozen feature $\phi(\mathrm{sg}[X])$ enters the point-wise pathway directly without further interleaving.

## H.2 Empirical Comparison on MMET

To reveal whether the fusion pathway still influences the forward prediction, we introduce the *Fusion Functional Contribution* (FFC), which measures the functional dependence of the model output on inter-point fusion. For each query point $x_i \in X$, let $u_\theta^{\mathrm{f}}(x_i, \phi(X))$ denote the prediction with the full collocation batch as fusion context, and let $u_\theta^{\mathrm{f}}\big(x_i, \phi(\{x_i\})\big)$ denote the prediction obtained by reducing the fusion context to a singleton containing only $x_i$. We define

$$\mathrm{FFC} = \frac{1}{N}\sum_{i=1}^{N} \frac{\big|u_\theta^{\mathrm{f}}(x_i, \phi(X)) - u_\theta^{\mathrm{f}}(x_i, \phi(\{x_i\}))\big|}{\big|u_\theta^{\mathrm{f}}(x_i, \phi(X))\big| + \big|u_\theta^{\mathrm{f}}(x_i, \phi(\{x_i\}))\big| + \epsilon}, \tag{81}$$

where $\varepsilon$ is a small constant for numerical stability.

The singleton baseline $\phi(\{x_i\})$ removes all inter-point interactions while keeping the model architecture untouched. Unlike the stop-gradient construction used in DCI, this baseline alters the forward value of $\phi$ and therefore yields a non-degenerate diagnostic. A small FFC indicates that the trained model is effectively point-wise, as removing inter-point context leaves the prediction nearly unchanged. A larger FFC indicates that the fusion pathway genuinely contributes to the forward output.

To empirically validate the analysis in Appendix H.1, we compare three configurations of MMET on the Heat benchmark, all sharing identical training hyperparameters from Appendix C:

*(a)* Vanilla MMET, no decoupling;

*(b)* Module-level decoupling, the decoupling scheme of Equation 22 applied to every attention block;

*(c)* Architectural-level decoupling, decoupling applied only at the encoder input by detaching $X$ before it enters the encoder, while cross-attention in decoder retains its standard differentiable form.

Configuration *(c)* is feasible for MMET specifically because its decoder consumes the encoder output as a frozen context, satisfying the structural requirement of Section 6.1.

Table 12: Comparison of vanilla, module-level decoupled, and architectural-level decoupled MMET in Heat benchmarks.

| **Configuration** | Loss | $L_2$ Error | DCI | FFC |
|---|---|---|---|---|
| Vanilla | 9.04e-2 | 1.07e+0 | 9.82e-1 | 5.01e-1 |
| Module-Level | 2.38e-1 | 7.68e-3 | 0.00e+0 | 1.04e-1 |
| Architectural-Level | 1.05e-1 | 3.03e-3 | 0.00e+0 | 4.75e-1 |

For each configuration, we report the converged total loss, relative $L_2$ error, DCI, and FFC in Table 12. Both decoupling schemes reduce DCI to exactly zero and recover an accurate solution with $L_2$ error reduced by two orders of magnitude. The two schemes differ, however, in FFC: architectural-level decoupling preserves an FFC of $0.475$, while module-level decoupling yields only $0.104$, indicating that the latter further suppresses the functional role of fusion in the forward prediction. This gap empirically confirms the inductive bias analyzed in Appendix H.1, namely that module-level stop-gradients restrict the residual loss to a family of batch-conditioned pointwise functions, whereas architectural-level decoupling retains the full expressive role of $\phi$ on the forward pass.

We emphasize that these results offer only a preliminary verification of this inductive bias on a single benchmark, and whether the same issue arises in other architectures or tasks remains uncertain. A more thorough characterization, in particular how the FFC gap interacts with problem structure and backbone design, is left to future work.