Lightweight Explainable Physics-Informed Neural Networks by Learnable Activation Function and Contextual Modulation

Wenqi Gu and Carlo Vittorio Cannistraci *

Posted Date: 9 May 2026

doi: 10.20944/preprints202605.0613.v1

Keywords: physics-informed neural networks; generalized logistic-logit function; activation function; transponder

 doi:10.20944/preprints202605.0613.v1

Article

Lightweight Explainable Physics-Informed Neural Networks by Learnable Activation Function and Contextual Modulation

Wenqi Gu [1,2] and Carlo Vittorio Cannistraci [1,2,3,*]

[1] Center for Complex Network Intelligence (CCNI), Tsinghua Laboratory of Brain and Intelligence (THBI), Department of Psychological and Cognitive Sciences, Tsinghua University, Beijing, China
[2] School of Biomedical Engineering, Tsinghua University, Beijing, China
[3] Department of Computer Science and Technology, Tsinghua University, Beijing, China
* Correspondence: kalokagathos.agon@gmail.com

Abstract

Physics-informed neural networks (PINNs) provide a data-efficient framework for solving partial differential equations, but improving their accuracy often requires enlarging multilayer perceptron backbones, which increases parameter count and computational cost. This study investigates whether PINN performance can be improved while keeping the underlying MLP lightweight. We introduce the Cannistraci-Muscoloni-Gu Generalized Logistic-Logit Function (CMG-GLLF) as a learnable activation function for compact PINNs. To make CMG practical for PINN training, we reformulate its implicit logit-phase approximation into an explicit differentiable form using a one-step Newton approximation, reducing numerical instability and computational overhead. Empirical validation on Burgers' equation shows that the explicit CMG formulation substantially outperforms both the implicit CMG implementation and fixed tanh activation. We further show that a layer-wise CMG design achieves a favorable accuracy-parameter trade-off, adding only two trainable parameters per hidden layer while improving over vanilla MLPs in most settings. In addition, we evaluate transponder-based contextual modulation, which adaptively modulates hidden-layer representations according to the network input. Across Burgers, Allen-Cahn, and diffusion-reaction benchmarks, Transponder-NS consistently improves over parameter-matched vanilla MLPs and achieves the best overall ranking, with approximately order-of-magnitude error reductions on Burgers and Allen-Cahn. Combining CMG with transponder modulation further improves performance on Allen-Cahn and remains competitive across tasks. Finally, parameter-level analysis on Allen-Cahn shows that learned CMG parameters differ from the fixed Tahn and that transponder modulation varies across both layers and nodes, providing explainability on why CMG and transponder could outperform vanilla networks through depth-dependent modulation behavior. These results suggest that learnable activation functions and contextual modulation offer a practical route toward lightweight, accurate, and explainable PINNs.

Keywords: physics-informed neural networks; generalized logistic-logit function; activation function; transponder

1. Introduction

Partial differential equations (PDEs) are fundamental for describing physical phenomena across science and engineering. Physics-informed neural networks (PINNs) have emerged as a scientific machine learning framework for solving forward and inverse problems involving nonlinear PDEs [1–3]. By embedding governing physical laws directly into the loss function, PINNs can approximate PDE solutions and assimilate sparse observations without relying entirely on traditional grid-based numerical solvers.

Despite their success, improving PINN accuracy often relies on scaling the multilayer perceptron (MLP) backbone by increasing depth or width. This can introduce substantial computational overhead

 doi:10.20944/preprints202605.0613.v1

and optimization pathologies, including stiff gradients, numerical instability, and convergence failures [4]. Recent studies also show that compact or shallow PINN architectures can maintain favorable training efficiency and avoid some optimization plateaus that affect deeper networks [5,6]. This motivates lightweight PINN designs that improve expressivity without aggressive architectural scaling.

The goal of this study is to investigate how to improve performance while keeping the size of the PINN MLP small. Instead of pursuing deeper networks, we explore whether performance and explainability can be improved by providing fine-grained, dynamic modulation to compact networks.

We introduce two mechanisms for lightweight PINNs. First, we use the Cannistraci-Muscoloni-Gu Generalized Logistic-Logit Function (CMG-GLLF) [7] as a layer-wise learnable activation function. To make CMG practical in the PINN context, we reformulate the implicit logit-phase approximation into a fully explicit and differentiable expression through a one-step Newton approximation. Second, we integrate transponder contextual modulation [8], which modulates hidden-layer outputs with respect to the network inputs. Together, these mechanisms allow compact PINNs to learn task-adaptive activation and modulation patterns with a small parameter overhead.

2. Methodology and Training Setups

To evaluate whether compact PINNs can be improved without aggressive depth or width scaling, we compare standard multilayer perceptrons (MLPs) with variants using the CMG-GLLF learnable activation function [7], transponder contextual modulation [8], and the PirateNet baseline [4]. Experiments use three standard PDE benchmarks: Burgers' equation, the Allen-Cahn equation, and a diffusion-reaction equation [1]. These tasks cover transport with sharp gradients, phase-field interface dynamics, and coupled diffusion with local reaction kinetics.

2.1. Network Architectures

The eight network structures evaluated in this study are:

1. **Vanilla MLP:** The original MLP with no additional structure.
2. **Vanilla MLP (match):** A widened MLP matching the parameter count of Transponder-NS, used as a parameter-matched baseline.
3. **CMG layer-wise:** An MLP where the standard `tanh` activation is replaced by explicit CMG-GLLF. All neurons in the same hidden layer share one CMG curve, adding only two trainable parameters per hidden layer: the inflection rate μ and the deviate inflection point I.
4. **Transponder-NS:** An MLP with transponder modulation using both node-wise and scalar-wise modulators. The modulator hidden dimension is 8. Scalar-wise modulation provides layer-level control, while node-wise modulation provides channel-level adjustment of hidden representations.
5. **Transponder-S:** An MLP with transponder modulation using scalar-wise modulation only.
6. **CMG-TNS:** An MLP combining layer-wise CMG activation with Transponder-NS modulation.
7. **CMG-TS:** An MLP combining layer-wise CMG activation with Transponder-S modulation.
8. **PirateNet:** A physics-informed residual adaptive network using global U and V modulators and adaptive residual connections controlled by a trainable parameter α initialized to zero [4].

2.2. Optimization Process

PINNs are commonly trained with full-batch optimization, and MLPs are the standard neural function approximators in many PINN implementations. Following this convention, we use a two-stage training process. The first stage uses Adam [9] to reach a favorable loss basin, and the second stage uses a fixed-budget L-BFGS optimizer [10] to refine the solution. Learning rates for standard network weights and special CMG parameters are searched independently over the same grid. The training configurations are based on DeepXDE PINN tutorials [11]. Each experiment is run with five random seeds, and the reported metric is the mean relative L2 error.

 doi:10.20944/preprints202605.0613.v1

Experiments are conducted on a platform with an Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10GHz and four NVIDIA A6000 GPUs.

Table 1. Hyperparameter configurations for PINN training. Diffusion-Reaction uses the updated three-hidden-layer network from the write guide.

Task	Vanilla MLP structure	Adam iters	L-BFGS iters	Learning rates
Burgers	[2, 20, 20, 20, 1]	15,000	15,000	2e-3, 1e-3, 5e-4
Allen-Cahn	[2, 20, 20, 20, 1]	40,000	15,000	2e-3, 1e-3, 5e-4
Diffusion-Reaction	[2, 30, 30, 30, 1]	20,000	None	2e-3, 1e-3, 5e-4

3. Results

3.1. Derivation of the Explicit CMG-Newton Formulation

The Cannistraci-Muscoloni-Gu Generalized Logistic-Logit Function (CMG-GLLF) provides a unified mathematical framework for trainable activation functions, allowing control over steepness, asymmetry, and input/output boundaries [7]. The original logit-phase formulation lacked a compact explicit expression and relied on implicit approximation to invert the logistic curve. In PINNs, where PDE residuals require repeated differentiation with respect to spatiotemporal inputs, implicit inversion can increase computational cost and introduce instability in gradient computation.

To address this bottleneck, we use a one-step Newton approximation to obtain an explicit, differentiable CMG formulation. Let $x \in [x_{\min}, x_{\max}]$ be the input and $y \in [y_L, y_R]$ be the mapped output limits. Define the normalized input $t \in [0,1]$ as

$$t = \frac{x - x_{\min}}{x_{\max} - x_{\min}}. \tag{1}$$

The unified explicit CMG expression, with inflection rate $\mu \in [0,1]$ and deviate inflection point $I \in [0,1]$, is

$$\mathrm{CMG}(x) = \begin{cases} y_L + (y_R - y_L)\dfrac{1}{1 + \frac{1-t}{t}\exp\left[-\left(\frac{1}{\mu} - 2\right)(t - I)\right]}, & 0 < \mu \le 0.5, \\ y_L + (y_R - y_L)\dfrac{t + \left(\frac{1}{1-\mu} - 2\right) It(1-t)}{1 + \left(\frac{1}{1-\mu} - 2\right) t(1-t)}, & 0.5 < \mu < 1. \end{cases} \tag{2}$$

This formulation preserves the flexible steepness and asymmetry control of the implicit CMG framework while allowing gradients to be computed by standard automatic differentiation. The derivation is provided in Appendix A.1.

3.2. Empirical Validation of CMG-Explicit

We first validate CMG-Explicit as a node-wise hidden-layer activation function on the Burgers' equation benchmark. Table 2 shows that CMG-Implicit degrades performance relative to `tanh`, while CMG-Explicit substantially decreases mean relative L2 error. This supports using CMG-Explicit as the default CMG implementation in the remaining experiments.

Table 2. Empirical validation of CMG-Explicit on Burgers' equation.

Method	Mean relative L2 error
CMG-Explicit	6.34E-03
CMG-Implicit	1.54E-01
Tanh	2.10E-02

 doi:10.20944/preprints202605.0613.v1

We then compare node-wise and layer-wise CMG-Explicit configurations. In the layer-wise configuration, all neurons in a hidden layer share the same trainable CMG parameters.

Table 3. Performance comparison of CMG node-wise and layer-wise configurations on Burgers' equation.

Method	Mean relative L2 error
CMG Node-wise	6.34E-03
CMG Layer-wise	8.83E-03
Tanh	2.10E-02

Both CMG configurations outperform `tanh`. Since the layer-wise configuration adds only two trainable parameters per hidden layer, it provides the better accuracy-parameter trade-off and is used in the following experiments.

3.3. Evaluation of Transponder and CMG on Different PINN Tasks

Table 4 reports relative L2 errors on Burgers, Allen-Cahn, and Diffusion-Reaction, along with mean rank, worst rank, and parameter counts. Ranks are computed within each task according to relative L2 error, where lower error receives a better rank.

Table 4. Evaluation of transponder and CMG on different PINN tasks. Best performance in each non-parameter column is highlighted in bold.

Method	Burgers	Allen-Cahn	Diff-Reaction	Mean Rank	Worst Rank	#Param (B/AC)	#Param (DR)
Vanilla MLP	2.10E-02	1.68E-02	3.90E-03	6	8	921	1981
Vanilla MLP (match)	1.48E-02	1.18E-02	7.66E-03	6.67	8	2241	3781
CMG layer-wise	8.83E-03	1.36E-02	6.69E-03	6	7	927	1987
Transponder-NS	**1.43E-03**	2.41E-03	1.66E-03	**1.67**	**2**	2211	3861
Transponder-S	4.12E-03	8.83E-03	5.12E-03	4.67	6	1311	2531
CMG-TNS	1.46E-03	**1.82E-03**	3.90E-03	2.33	4	2217	3867
CMG-TS	1.14E-02	6.04E-03	4.78E-03	4.67	5	1317	2537
PirateNet	3.46E-02	5.21E-03	**7.83E-04**	4	8	2202	3912

Compared with Vanilla MLP, CMG layer-wise improves Burgers and Allen-Cahn while adding only six parameters to the three-hidden-layer network. Transponder-NS consistently improves over the parameter-matched Vanilla MLP across all three tasks, achieves the best Burgers error, and obtains the best mean and worst ranks. Transponder-S also improves over the matched Vanilla MLP on Burgers and Allen-Cahn, but its gains are smaller than those of Transponder-NS, consistent with the removal of node-wise modulation.

Combining CMG with Transponder-NS gives the best Allen-Cahn result and the second-best ranking profile overall. PirateNet obtains the best Diffusion-Reaction error, but it does not outperform the matched Vanilla MLP on Burgers and has one of the lowest worst ranks. These results indicate that CMG can reduce error at extremely low parameter cost, while transponder modulation provides consistent gains for lightweight PINNs.

3.4. Explainability of CMG and Transponder Design

To examine why CMG and transponder modulation can outperform Vanilla MLP and PirateNet, we analyze learned CMG activation parameters and transponder gate values in the CMG-TNS model on Allen-Cahn. This claim is restricted to Allen-Cahn and should not be read as proving that the same parameter distributions occur for all PDE systems.

Figure 1 shows that the learned CMG parameters differ across layers, indicating that different depths require different activation-value modulation. A fixed `tanh` activation cannot provide this layer-specific functional adaptation. The transponder gate values also vary across layers and nodes: scalar-wise modulation separates layers, while channel-wise modulation varies within each layer. This fine-grained, local modulation explains why CMG and transponder designs can improve over Vanilla MLP and over global modulation mechanisms such as PirateNet in the Allen-Cahn setting.

 doi:10.20944/preprints202605.0613.v1

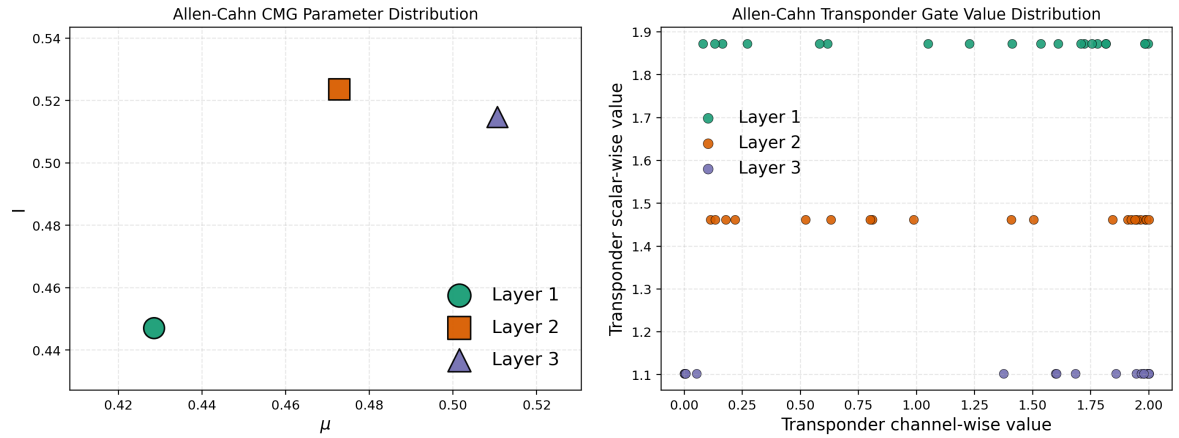

Figure 1. Parameter distributions of the CMG-TNS model trained on Allen-Cahn. Left: learned CMG-Explicit parameters, inflection rate μ and deviate inflection point I, for each hidden layer. Right: transponder gate modulation values, showing channel-wise and scalar-wise values for hidden-layer nodes.

4. Conclusion

This study investigated a route for improving PINN accuracy and robustness without relying on aggressive MLP scaling. We introduced an explicit, differentiable CMG-GLLF formulation as a parameter-efficient layer-wise learnable activation function, and combined it with transponder contextual modulation for input-dependent hidden-state modulation.

Across Burgers, Allen-Cahn, and Diffusion-Reaction benchmarks, these mechanisms improved lightweight PINNs at low parameter cost. CMG layer-wise improved over Vanilla MLP on Burgers and Allen-Cahn while adding only two trainable parameters per hidden layer. Transponder-NS consistently improved over the parameter-matched Vanilla MLP and achieved the best overall ranking profile, while CMG-TNS achieved the best Allen-Cahn performance. PirateNet obtained the best Diffusion-Reaction error but was less stable across tasks.

The Allen-Cahn parameter analysis further supports the explainability of the design. Learned CMG parameters and transponder gate values differed across layers and nodes, indicating localized activation and modulation requirements that fixed activations and global modulation cannot provide. These results suggest that learnable activation functions and contextual modulation offer a practical path toward PINNs that are lightweight, accurate, and explainable.

The main limitation of this study is that the empirical evaluation covers only three PINN benchmark tasks. Although Burgers, Allen-Cahn, and Diffusion-Reaction equations represent different PDE behaviors, broader validation across more PINN tasks, higher-dimensional systems, inverse problems, and real-world scientific computing settings is needed before making stronger claims about generality. The broader impact of this work is primarily methodological: more accurate lightweight PINNs could reduce the computational cost of scientific machine learning and make PDE-based modeling more accessible.

A. Technical Appendices

A.1. Detailed Derivation of the CMG-Explicit Formulation

When $0 < \mu \leq 0.5$, CMG-GLLF is in the logistic phase and admits an explicit analytical expression. When $0.5 < \mu < 1$, CMG enters the logit phase, which is defined as the inverse of the corresponding logistic-phase mapping. Because an exact inverse in a simple closed form is not available, we derive an explicit approximation by applying one Newton step [12].

 doi:10.20944/preprints202605.0613.v1

Normalization.

Let $x \in [x_{\min}, x_{\max}]$ be the input and $y \in [y_L, y_R]$ be the mapped output limits. We normalize both axes to $[0,1]$:

$$t = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \qquad s = \frac{y - y_L}{y_R - y_L}. \tag{3}$$

Logistic-phase expression.

In normalized coordinates, the logistic-phase CMG can be written as

$$s = \sigma(\mathrm{logit}(t) + \alpha(t - I)), \tag{4}$$

where $\sigma(z) = 1/(1+\exp(-z))$, $\mathrm{logit}(t) = \log(t/(1-t))$, and $I \in [0,1]$ is the deviate inflection point. In the logit phase,

$$\alpha = \frac{1}{1-\mu} - 2, \qquad 0.5 < \mu < 1. \tag{5}$$

Inverse problem.

To derive the logit-phase expression, we invert the normalized function:

$$t = \sigma(\mathrm{logit}(s) + \alpha(s - I)). \tag{6}$$

Applying the logit transform gives

$$\mathrm{logit}(t) = \mathrm{logit}(s) + \alpha(s - I). \tag{7}$$

Define

$$F(s) = \mathrm{logit}(s) + \alpha(s - I) - \mathrm{logit}(t). \tag{8}$$

Solving $F(s) = 0$ gives the logit-phase output.

One-step Newton approximation.

Using $s_0 = t$ as the initial guess, one Newton step gives

$$s_1 = s_0 - \frac{F(s_0)}{F'(s_0)}. \tag{9}$$

At $s_0 = t$,

$$F(t) = \alpha(t - I), \qquad F'(t) = \frac{1}{t(1-t)} + \alpha. \tag{10}$$

Therefore,

$$s_1 = t - \frac{\alpha(t-I)}{\frac{1}{t(1-t)} + \alpha} = \frac{t + \alpha I t(1-t)}{1 + \alpha t(1-t)}. \tag{11}$$

Final expression.

Returning to the original coordinates yields the explicit logit-phase approximation:

$$\mathrm{CMG}_{\mathrm{logit}}(x) \approx y_L + (y_R - y_L)\frac{t + \left(\frac{1}{1-\mu} - 2\right) I t(1-t)}{1 + \left(\frac{1}{1-\mu} - 2\right) t(1-t)}. \tag{12}$$

This expression avoids grid-based implicit inversion and supports direct automatic differentiation in PINN training.

 doi:10.20944/preprints202605.0613.v1

References

1. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *Journal of Computational Physics* **2019**, *378*, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045.
2. Cuomo, S.; Di Cola, V.S.; Giampaolo, F.; Rozza, G.; Raissi, M.; Piccialli, F. Scientific Machine Learning Through Physics-Informed Neural Networks: Where We Are and What's Next. *Journal of Scientific Computing* **2022**, *92*, 88. https://doi.org/10.1007/s10915-022-01939-z.
3. Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-Informed Machine Learning. *Nature Reviews Physics* **2021**, *3*, 422–440. https://doi.org/10.1038/s42254-021-00314-5.
4. Wang, S.; Li, B.; Chen, Y.; Perdikaris, P. PirateNets: Physics-Informed Deep Learning with Residual Adaptive Networks. *Journal of Machine Learning Research* **2024**, *25*, 1–51.
5. Rishi, J.; Gafoor, A.; Kumar, S.; Subramani, D. On the Training Efficiency of Shallow Architectures for Physics Informed Neural Networks. In Proceedings of the Computational Science – ICCS 2024. Springer, 2024, Vol. 14834, *Lecture Notes in Computer Science*, pp. 363–377. https://doi.org/10.1007/978-3-031-63759-9_39.
6. Hu, W.F.; Shih, Y.J.; Lin, T.S.; Lai, M.C. A Shallow Physics-Informed Neural Network for Solving Partial Differential Equations on Static and Evolving Surfaces. *Computer Methods in Applied Mechanics and Engineering* **2024**, *418*, 116486. https://doi.org/10.1016/j.cma.2023.116486.
7. Gu, W.; Zhang, Y.; Muscoloni, A.; Cannistraci, C.V. A Generalized Logistic-Logit Function and Its Application to Multi-Layer Perceptron and Neuron Segmentation, 2025. Preprint, https://doi.org/10.20944/preprints202510.2439.v2.
8. Zhang, Y.; Gu, W.; Hu, W.; Li, J.; Cannistraci, C.V. From Transformer to TRANSPONDER: Introducing Contextual Modulation Training for Residual Learning in LLMs, 2025. Preprint, version 2 posted 30 September 2025, https://doi.org/10.20944/preprints202506.0120.v2.
9. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations, 2015, [1412.6980]. https://doi.org/10.48550/arXiv.1412.6980.
10. Nocedal, J. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation* **1980**, *35*, 773–782. https://doi.org/10.1090/S0025-5718-1980-0572855-7.
11. Lu, L.; Meng, X.; Mao, Z.; Karniadakis, G.E. DeepXDE: A Deep Learning Library for Solving Differential Equations. *SIAM Review* **2021**, *63*, 208–228. https://doi.org/10.1137/19M1274067.
12. Kelley, C.T. *Solving Nonlinear Equations with Newton's Method*; Society for Industrial and Applied Mathematics: Philadelphia, PA, 2003. https://doi.org/10.1137/1.9780898718898.