Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

MixFunn: A Neural Architecture for Differential Equations with Mixed Functions and Second-Order Interactions

Tiago de Souza Farias

tiago.farias@ufscar.br

Universidade Federal de São Carlos

Gubio Gomes de Lima

Universidade Federal de São Carlos

Jonas Maziero

Universidade Federal de Santa Maria

Celso Jorge Villas-Boas

Universidade Federal de São Carlos

Research Article

Keywords:

Posted Date: November 2nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10331669/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

MixFunn: A Neural Architecture for Differential Equations with Mixed Functions and Second-Order Interactions

Tiago de S. Farias,[1,*] Gubio G. de Lima,[1] Jonas Maziero,[2] and Celso J. Villas-Boas[1]

[1] *Physics Department, Federal University of São Carlos, 13565-905, São Carlos, SP, Brazil*
[2] *Physics Department, Federal University of Santa Maria, 97105-900, Santa Maria, RS, Brazil*

We introduce MixFunn, a neural network architecture designed for modeling solutions of differential equations. The proposed architecture combines two main components: (i) mixed-function neurons, which employ a set of predefined activation functions to enrich the functional representation, and (ii) second-order neurons, which explicitly incorporate interactions between different neurons. Together, these elements aim to increase representational flexibility while maintaining a compact model structure. We evaluate MixFunn on a set of differential equation problems, including the damped and forced harmonic oscillator, the one-dimensional viscous Burgers equation, and the quantum infinite square well. The results indicate that the proposed architecture can achieve accurate approximations using fewer parameters compared to standard multilayer perceptrons. Additionally, the structure of the network enables the extraction of analytical expressions that approximate the learned solutions, providing a degree of interpretability. These findings suggest that MixFunn offers a viable alternative for neural modeling of differential equations, particularly in settings where compact representations and interpretability are desirable.

I. INTRODUCTION

Differential equations are central to modeling physical systems across a wide range of disciplines, from classical mechanics to quantum dynamics, as well as engineering applications such as fluid dynamics, structural analysis, and electrical circuits [1, 2]. They provide a fundamental framework for describing system behavior and for enabling predictive modeling in scientific and technological contexts.

Traditional approaches for solving differential equations involve a well-known trade-off. Analytical methods yield exact and interpretable solutions but are typically restricted to simplified systems [3, 4]. In contrast, numerical methods can address more complex and realistic scenarios, but they generally produce discrete approximations that may be computationally expensive and lack the interpretability of closed-form expressions [5, 6].

More recently, neural networks have emerged as a third paradigm for solving differential equations. Owing to their universal approximation capabilities [7], artificial neural networks can represent highly complex solution spaces [8–10]. This has led to the development of Physics-Informed Neural Networks (PINNs), which incorporate physical laws directly into the training process. However, the effectiveness of these approaches depends critically on the architecture of the underlying neural network. In particular, the choice of activation functions and neuron models strongly influences expressivity, optimization, and generalization [11–13]. Standard architectures, such as multilayer perceptrons (MLPs), often act as a bottleneck, limiting performance and requiring substantial over-parameterization to capture complex physical behavior.

In this work, we investigate the role of architectural expressivity in physics-informed learning, and show that limitations of standard neural network backbones can impact the quality of the learned solutions. To address this, we introduce the Mixed-Function Neural Network (MixFunn) and its extension, the Second-Order Mixed-Function Neural Network (Mix2Funn), as alternative neural backbones for differential equation modeling. The proposed architecture combines two key components: (i) mixed-function neurons, which leverage a set of physically motivated activation functions to capture diverse solution behaviors, and (ii) second-order neurons, which explicitly model multiplicative interactions between inputs.

We also show that, when combined with sparsity-inducing regularization and structural pruning, the proposed architecture enables the extraction of compact analytical expressions from trained models, providing an interpretable alternative that bridges numerical solvers and symbolic representations. Together, these components provide a more expressive and structured representation, enabling compact models that better capture the underlying physics.

The goal of this work is not to establish a new state-of-the-art in predictive accuracy, but to propose a neural architecture that improves parameter efficiency and interpretability in physics-informed learning. In particular, the structure of MixFunn enables the extraction of analytical expressions from trained models, bridging the gap between numerical approximation and symbolic representation. As such, the proposed architecture is intended as a general-purpose backbone that can be integrated with existing training methodologies.

The remainder of this article is organized as follows. Section II reviews related work on neural approaches for differential equations. Section III presents the proposed architecture and methodology. Section IV reports experimental results on representative benchmark problems.

* tiago.farias@ufscar.br

Finally, Section V concludes the paper and outlines directions for future research.

II. RELATED WORK

Machine learning has emerged as a flexible paradigm for solving differential equations, complementing traditional numerical methods. Existing approaches can be broadly divided into two categories depending on what the neural network is trained to approximate.

One class of methods, exemplified by Neural Differential Equations [14], learns the underlying dynamics function, which is then integrated using standard numerical solvers. Variants such as Hamiltonian and Lagrangian Neural Networks [15, 16] incorporate physical structure by learning energy-based representations of the system.

In this work, we focus on Physics-Informed Neural Networks [17–19], which directly approximate the solution of a differential equation by embedding the governing equations into the training loss. A detailed formulation of the PINN framework used in this work is provided in Appendix A.

The effectiveness of PINNs depends strongly on the choice of neural network architecture. Most implementations rely on standard multilayer perceptrons [17, 18], which, despite their versatility, present important limitations. In particular, MLPs exhibit spectral bias, favoring low-frequency functions and struggling to represent oscillatory or sharp features [20]. Additionally, the use of a single activation function across the network restricts expressivity, often requiring large models to approximate complex physical behavior [21–23].

To address these limitations, recent work has explored more expressive function approximators. One line of research focuses on improving activation functions. For example, SIREN [24] employs sinusoidal activations to better represent oscillatory phenomena, while methods such as PReLU [25] introduce learnable nonlinearities. Kolmogorov-Arnold Networks (KANs) [26], inspired by representation theorems [27–30], replace fixed activations with learnable spline-based functions, offering a flexible functional representation. In contrast to these approaches, which typically rely on a single functional class, our method combines multiple predefined, physically motivated functions within each neuron, allowing the model to adaptively select suitable representations.

A complementary line of work focuses on improving the ability of neural networks to model interactions between inputs. Higher-order neurons, such as Sigma-Pi units [31, 32], explicitly incorporate multiplicative terms, enabling more efficient representation of correlations compared to standard linear neurons. Similar interaction-based mechanisms also underlie architectures such as Transformers [33]. Our approach builds on this idea by integrating second-order interactions directly into the neuron structure in a manner suitable for physics-informed learning.

Finally, our work is conceptually related to symbolic regression methods such as SINDy [34], which aim to discover governing equations from data. Rather than identifying the differential equation itself, our approach focuses on learning compact, interpretable representations of its solution by combining a predefined functional basis with sparsity-inducing regularization.

Despite these advances, a gap remains for architectures that simultaneously provide high expressivity, parameter efficiency, and interpretability. The MixFunn architecture addresses this by integrating mixed activation functions and higher-order interactions within a unified framework, enabling compact and interpretable solutions in physics-informed settings.

III. METHODS

We introduce MixFunn and its extension, Mix2Funn, as neural network backbones for physics-informed learning. The architecture combines two key elements: (i) mixed-function neurons, which represent solutions using a predefined dictionary of nonlinear functions, and (ii) second-order neurons, which explicitly model multiplicative interactions between inputs. Together, these components provide a structured and expressive function approximator, while additional sparsity mechanisms enable compact and interpretable representations. Implementation details are provided in Appendix B.

A. Neuron models

The design of MixFunn is motivated by two considerations. First, neural networks with mixed activation functions retain universal approximation properties when at least one non-polynomial basis function is present [35]. Second, incorporating periodic and exponential functions introduces an inductive bias aligned with common solution forms of differential equations, while second-order terms allow multiplicative interactions to be represented explicitly.

1. Second-order neuron

Standard neurons compute an affine transformation $s(\mathbf{x}) = \mathbf{W}\mathbf{x} + b$, which limits their ability to represent interactions between input features. To address this, we employ a second-order neuron that augments the linear mapping with quadratic terms. The pre-activation output is defined as

$$s = b + \mathbf{W}\mathbf{x} + \mathbf{x}^T\mathbf{U}\mathbf{x}, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{W} \in \mathbb{R}^{1\times N}$, and $\mathbf{U} \in \mathbb{R}^{N\times N}$ is a symmetric matrix. This formulation enables the neuron to directly model pairwise interactions between input variables.

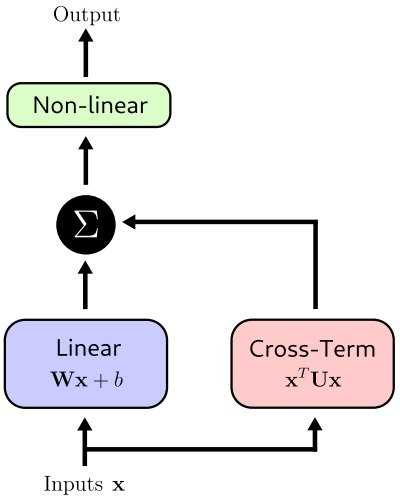


FIG. 1: Schematic of a second-order neuron. The input contributes both linear terms and quadratic interactions, which are summed and passed through a nonlinear activation.

As illustrated in Fig. 1, the neuron combines linear contributions with quadratic cross-terms before applying a nonlinear activation.

Compared to standard MLPs, which must approximate multiplicative interactions through depth, this formulation captures them explicitly within a single layer, yielding a more compact representation for functions involving feature correlations.

2. *Mixed-function neuron*

To enhance representational diversity, we introduce the mixed-function neuron, which replaces a single activation function with a parallel combination of predefined nonlinear functions. Given an input $\mathbf{z}$, the neuron computes

$$s_i = \mathbf{W}_i\mathbf{z} + b_i, \tag{2}$$

for each pathway $i = 1, \ldots, Q$, followed by application of a basis function f_i. The outputs are combined as

$$a = b' + \sum_{i=1}^{Q} w_i' f_i(s_i), \tag{3}$$

where w_i' and b' are trainable parameters.

As shown in Fig. 2, each pathway processes the input through a distinct nonlinear function, and the outputs are linearly combined.

This construction allows the network to represent solutions as combinations of physically motivated functions (e.g., trigonometric or exponential), providing an inductive bias toward realistic solution forms. In contrast to approaches that rely on a single activation type, the model can adaptively select among multiple functional components.

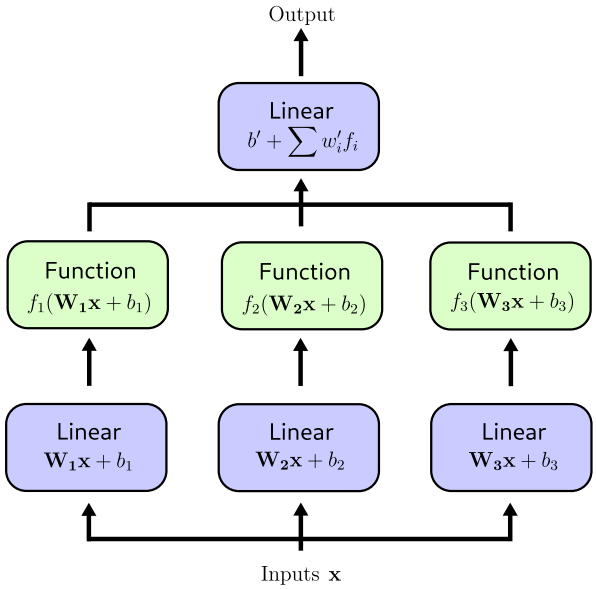


FIG. 2: Mixed-function neuron with Q parallel pathways. Each pathway applies a distinct nonlinear function, and the outputs are linearly combined.

The Mix2Funn architecture extends this by feeding the mixed-function outputs into a second-order combination layer, explicitly capturing interactions between different basis functions. Concretely, the pre-activations are first computed with quadratic input interactions,

$$s_i = b_i + \sum_j w_{ij}x_j + \sum_{m,n} u_{mni}x_m x_n, \tag{4}$$

and the final output combines both first- and second-order terms across the functional pathways:

$$a = b' + \sum_i w_i' f_i(s_i) + \sum_{i,j} u_{ij}' f_i(s_i) f_j(s_j). \tag{5}$$

This combination of functional diversity and multiplicative interactions provides a flexible representation capable of modeling oscillatory, exponential, and coupled behaviors within a compact architecture.

B. Sparsity and interpretability

To promote compact and interpretable solutions, we incorporate sparsity-inducing mechanisms adapted to the mixed-function structure.

First, the combination weights w_i' can be parameterized using a temperature-controlled softmax:

$$w_i' = c\frac{e^{\alpha_i/T}}{\sum_{j=1}^{Q} e^{\alpha_j/T}}, \tag{6}$$

where α_i are trainable logits, T controls the sharpness of the distribution, and c is a learnable scale. Lower values of T encourage sparse selection of basis functions [36, 37].

Second, dropout is applied at the level of functional pathways rather than neuron outputs. During training, individual terms $f_i(s_i)$ are randomly masked before the final combination layer, encouraging robustness and preventing reliance on a single basis function [38].

Finally, we apply magnitude-based structural pruning to the mixing weights w'_i. Setting small coefficients to zero removes the corresponding functional pathways, simplifying the learned representation. The pruned model is then fine-tuned to recover performance. This approach operates at the level of functional components rather than individual weights [39], enabling direct extraction of compact analytical expressions.

IV. RESULTS

We evaluate MixFunn and Mix2Funn on three benchmark problems spanning classical mechanics, fluid dynamics, and quantum mechanics. All models are trained using a standard PINN loss with the Adam optimizer [40], without advanced training schemes such as causal or curriculum learning, so that performance differences can be attributed directly to architectural choices. For all MixFunn/Mix2Funn models, we use a basis set of $Q = 7$ functions: $\{\sin(s), \cos(s), \exp(-|s|), \exp(|s|), \sqrt{|s|+\epsilon}, \log(|s| + \epsilon), s\}$. Each experiment is run five times with different random seeds; shaded regions in plots indicate variance across runs. No dropout or pruning is applied in the architectural comparison experiments; these techniques are used only for symbolic extraction. Full implementation and hyperparameter details are provided in Appendix B.

Performance is evaluated against known analytical or high-fidelity numerical solutions using Mean Squared Error (MSE). Generalization is assessed on a held-out test domain outside the training region.

A. Damped harmonic oscillator

The damped harmonic oscillator is a canonical test for spectral bias, as standard MLPs are known to struggle with periodic solutions over long time horizons [41]. The governing equation is:

$$m\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + kx = 0, \tag{7}$$

with parameters $m = 1$, $k = 1$, $\gamma = 0.1$, and initial conditions $x(0) = 1$, $\dot{x}(0) = 0$. The training domain is $t \in [0, 40]$ (1024 collocation points); the test domain is $t \in (40, 80]$.

Figure 3 compares the three architectures. The baseline MLP correctly learns the solution for $t \lesssim 15$ but then collapses to the trivial solution $x(t) = 0$. This progressive failure is characteristic of spectral bias [20]: the network first captures low-frequency components but cannot sustain the persistent oscillations, eventually finding that the zero function minimizes the average residual over the full domain. MixFunn captures the decaying oscillatory pattern but shows noticeable phase and amplitude errors at later times. Mix2Funn, by contrast, achieves an accurate fit across both training and test domains, correctly modeling the frequency and exponential decay envelope throughout. Quantitative results are given in Table I.

1. Analytical extraction

A distinctive capability of MixFunn is the extraction of closed-form analytical expressions from trained models. Starting from the best Mix2Funn model (77 parameters), we apply the structural pruning procedure described in Section III. As shown in Fig. 4, the physics loss does not degrade with increased sparsity; it reaches its minimum near 100% pruning, indicating that a very simple functional form is sufficient to describe the underlying physics.

Pruning reduces the model from 77 to just 5 parameters, a reduction of four orders of magnitude, and only 0.0075% of the size of the baseline MLP. The resulting analytical expression is:

$$x(t) = 1.085\sin(1.006t + 1.497)\, e^{-|0.085t+0.13|}, \tag{8}$$

which closely matches the true solution $x_{\text{true}}(t) = \sin(t + \pi/2)\, e^{-0.05t}$. The architecture has independently recovered the correct functional form, an exponentially decaying sinusoid, with parameter values close to the ground truth.

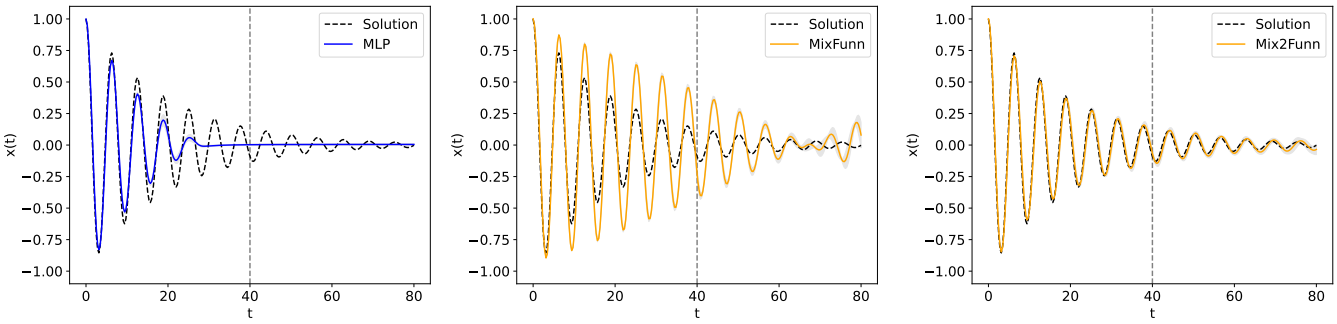
B. Burgers equation

To evaluate performance on a nonlinear PDE, we consider the 1D viscous Burgers' equation [42–44]:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} + k\frac{\partial^2 u}{\partial x^2}, \tag{9}$$

with viscosity $k = 0.01/\pi$, sinusoidal initial condition $u(x, 0) = -\sin(\pi x)$, and periodic boundary conditions $u(-1, t) = u(1, t)$. The training domain is $(x, t) \in [-1, 1] \times [0, 1]$ (2048 collocation points); the test domain is $t \in (1, 2]$. A high-fidelity numerical solution obtained via a fourth-order Runge-Kutta finite-difference scheme serves as ground truth. We compare Mix2Funn against the baseline MLP and a Hybrid model that replaces the first MLP layer with a second-order neuron layer [45].

The Burgers equation results reveal a more nuanced behavior. As shown in Table II, the MLP achieves the lowest test MSE, while Mix2Funn obtains a lower training error with fewer parameters. However, the qualitative comparison in Fig. 6 suggests that the MLP solution is overly smooth and does not accurately reproduce the wave-front steepening characteristic of Burgers dynamics. Therefore, in this case, the average MSE should



(a) (b) (c)

FIG. 3: Predicted solutions for the damped harmonic oscillator using (a) MLP, (b) MixFunn, and (c) Mix2Funn. The vertical dashed line at $t = 40$ marks the boundary between training and test domains. The MLP collapses to the trivial zero solution; Mix2Funn accurately tracks both the frequency and decay envelope throughout.

Model	Parameters	Training error	Test error	Lowest error
MLP	66433	$1.43 \times 10^{-2} \pm 2.08 \times 10^{-3}$	$2.43 \times 10^{-3} \pm 7.74 \times 10^{-5}$	2.35×10^{-3}
MLP	193	$7.18 \times 10^{-2} \pm 1.47 \times 10^{-3}$	$2.23 \times 10^{-3} \pm 1.88 \times 10^{-5}$	2.34×10^{-3}
MixFunn	21	$4.29 \times 10^{-2} \pm 1.90 \times 10^{-3}$	$1.47 \times 10^{-2} \pm 4.94 \times 10^{-4}$	1.42×10^{-2}
Mix2Funn	77	$\mathbf{2.68 \times 10^{-3} \pm 2.57 \times 10^{-3}}$	$\mathbf{1.04 \times 10^{-3} \pm 8.86 \times 10^{-4}}$	$\mathbf{1.37 \times 10^{-4}}$

TABLE I: Training and test errors for the damped harmonic oscillator. Mix2Funn achieves a lower error than both MLP variants despite using far fewer parameters, with the best run reaching an order of magnitude lower error than the best MLP.

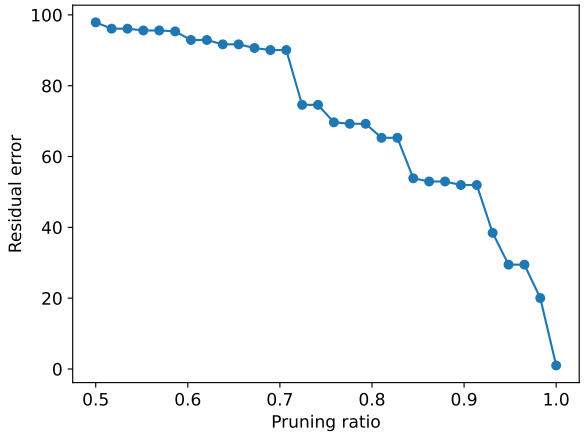


FIG. 4: Physics loss as a function of pruning ratio for the Mix2Funn model on the damped harmonic oscillator. The minimum loss is achieved near complete pruning, confirming that the solution is well-described by a single dominant functional form.

not be interpreted as a complete measure of physical fidelity. Since the test domain contains extended regions where the solution approaches small values, a smooth or near-zero prediction can obtain a low average error while failing to capture localized structures.

The Hybrid model achieves consistently competitive

Model	Parameters	Residual error	Train error	Test error
MLP	353	**0.123**	0.202	**0.089**
Mix2Funn	56	0.155	0.218	0.215
Hybrid	73	0.182	0.171	0.185
Pruned	**4**	0.727	**0.144**	0.410

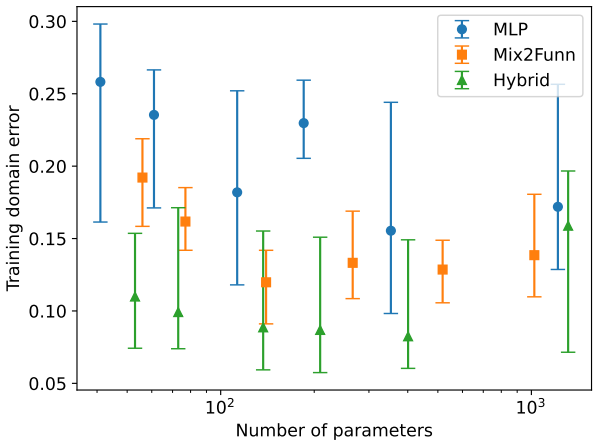
TABLE II: Quantitative comparison for the Burgers equation. The pruned Mix2Funn achieves the lowest training error with only 4 parameters. The Hybrid model offers a strong balance of accuracy and compactness.

performance in both domains, confirming that even a single second-order layer provides a meaningful boost in representational capability. Mix2Funn achieves this with only 56 parameters compared to 353 for the MLP.

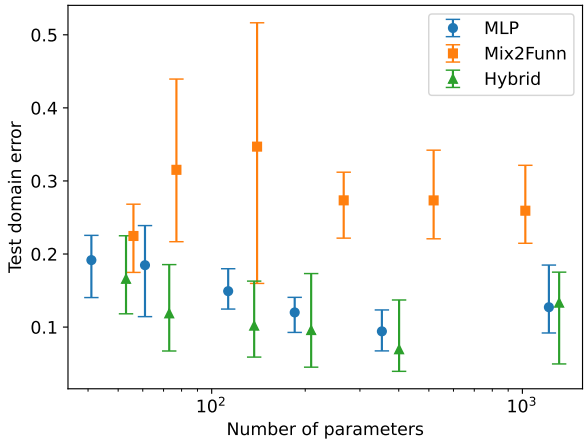
Applying the iterative pruning procedure (detailed in Appendix C) yields a 4-parameter model with the extracted expression:

$$u(x,t) = 0.971 \cos(-1.123 \sin(\pi x)\, t + 3.160x + 1.572)\,. \tag{10}$$

While this cosine-based form differs from the Cole-Hopf series solution, it approximates the solution's shape within the training domain.



(a)



(b)

FIG. 5: Training (a) and test (b) errors for the Burgers equation across varying parameter counts. Mix2Funn achieves lower training error than the MLP; the Hybrid model achieves competitive performance in both domains.

C. Quantum infinite square well

To demonstrate applicability beyond initial-value problems, we apply Mix2Funn to the time-independent Schrödinger equation for the infinite square well [46]:

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} = E\psi, \quad x \in [-1, 1], \tag{11}$$

with boundary conditions $\psi(\pm 1) = 0$. Rather than solving for a single eigenstate, we train a general model that accepts both position x and $\sqrt{E}$ as inputs, enabling prediction across multiple energy levels from a single network.

Eigenvalues are identified by scanning candidate values of $\sqrt{E}$ and selecting those that minimize the PINN residual loss, as shown in Fig. 7. The model is trained on the first two excited states only, and generalization is evaluated on the ground state and third excited state.

Figure 8 shows predicted wavefunctions for all four energy levels. The MLP provides reasonable approximations for the training states but produces phase-shifted predictions for the unseen states. Mix2Funn delivers accurate predictions for all four levels, including those not seen during training. This generalization is further confirmed in Fig. 9: the MLP shows low loss only at the training energies, while Mix2Funn identifies sharp minima at all true eigenvalues across the spectrum.

Applying structural pruning to the trained Mix2Funn yields the compact analytical expression:

$$\psi(x, E) = 1.38 \sin\Big(0.9998\sqrt{E} \cdot x + 8.01\sqrt{E} + 9.359\Big), \tag{12}$$

which closely approximates the true solution $\psi_{\text{true}}(x, E) = \sqrt{2}\sin(\sqrt{E}\,x)$, up to a constant phase shift.

V. CONCLUSIONS

In this work, we introduced MixFunn and Mix2Funn, novel neural network architectures designed to serve as highly expressive and efficient backbones within physics-informed learning frameworks. We argued that the standard Multi-Layer Perceptron used in most Physics-Informed Neural Networks acts as a bottleneck due to its limited representational power and spectral bias. Our architectures address these limitations through two key components: the mixed-function neuron, which uses a parallel structure with a basis of physically-motivated functions, and the second-order neuron, which explicitly models input cross-combinations.

The design of this architecture is inspired by the common forms of solutions to differential equations encountered in physics, where trigonometric and exponential functions, often coupled with cross-combined terms, frequently appear. By embedding this inductive bias through carefully selecting nonlinear functions, the model structure is aligned with the expected solution space, promoting efficiency and accuracy.

We evaluated MixFunn on three benchmark problems: the damped harmonic oscillator, Burgers' equation, and the quantum infinite well. An additional experiment on the forced damped harmonic oscillator is presented in the Appendix. All models were trained in a fully unsupervised manner using only residual, boundary-condition, and initial-condition losses. Despite the challenges associated with training without supervised data, MixFunn achieved consistent performance across all benchmark problems.

To further enhance generalization, we incorporated a suite of regularization techniques, including dropout,

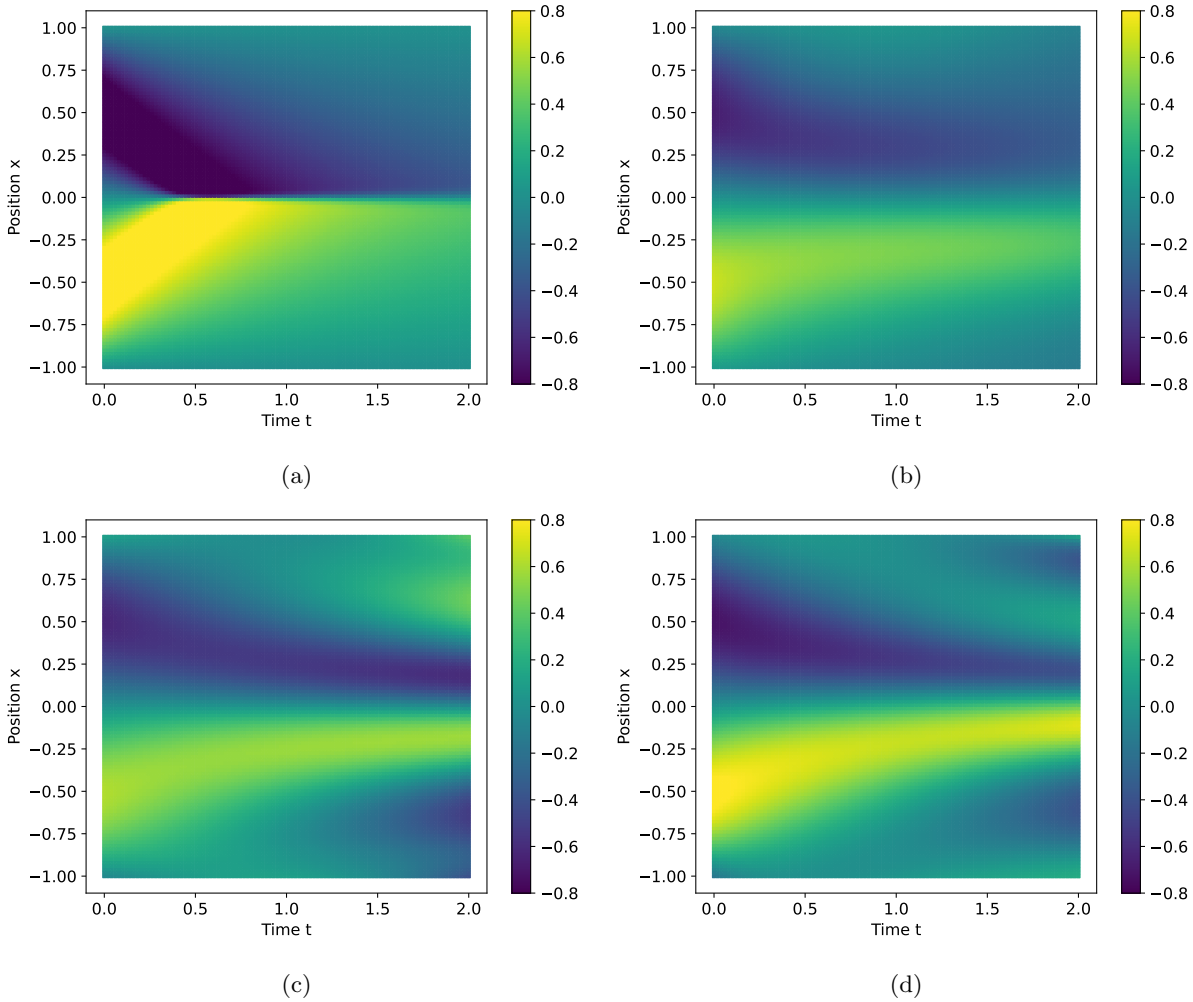


FIG. 6: Solutions for Burgers' equation: (a) numerical reference, (b) MLP, (c) Mix2Funn, (d) Hybrid. The MLP produces an overly smooth solution that misses the wave-front steepening. Mix2Funn and the Hybrid model align more closely with the reference.

pruning, and function normalization. These methods encourage competition among different nonlinear functions and neurons, effectively reducing loss while simplifying the resulting models. Notably, our approach drastically reduced the number of parameters, up to four orders of magnitude fewer than the standard MLP architecture, without compromising performance. This significant reduction in model complexity decreases computational overhead and facilitates the extraction of interpretable analytical expressions from the trained models.

Furthermore, we demonstrated that the interpretability of our architecture is one of its key advantages. By combining the mixed-function basis with targeted, structural regularization techniques, we were able to distill trained networks into concise, human-readable analytical expressions. This unique capability bridges the gap between numerical solvers and symbolic methods, offering deeper insight into the structure of the learned solutions.

Rather than competing directly with specialized state-of-the-art architectures, our contribution is to provide a general-purpose backbone that prioritizes compactness, interpretability, and physical plausibility. This makes MixFunn particularly well-suited as a modular component: it can be integrated into existing frameworks or combined with advanced training strategies to enhance their efficiency and transparency. By focusing on architecture rather than optimization heuristics, we demonstrate that substantial gains in generalization and interpretability can be achieved independently of the training protocol.

DATA AVAILABILITY

The code for the MixFunn architecture, along with the examples shown in this article, can be found at `https://github.com/tiago939/MixFunn`.

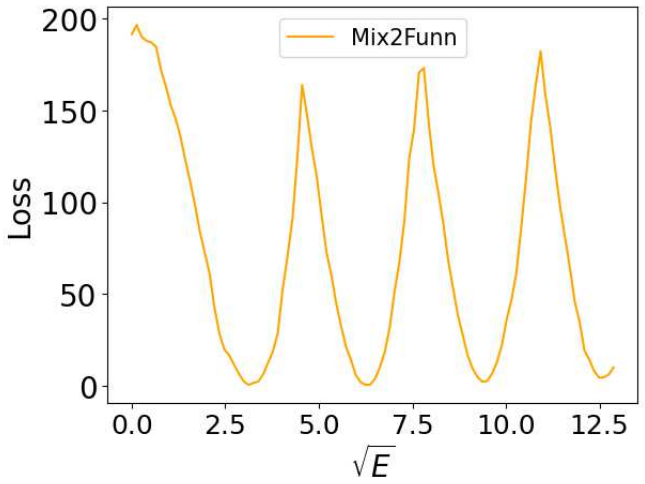


FIG. 7: Residual loss as a function of candidate $\sqrt{E}$ for the infinite quantum well. Loss minima correspond to the true energy eigenvalues, enabling eigenvalue identification without direct supervision.

DECLARATIONS

Competing interests The authors declare that there is no conflict of interest regarding the publication of this article.

ACKNOWLEDGMENTS

This work was supported by the São Paulo Research Foundation (FAPESP), under Grant No. 2023/15739-3 and No. 2022/00209-6, by the National Council for Scientific and Technological Development (CNPq), under Grants No. 140468/2022-6, No. 300083/2025-4, No. 409673/2022-6, No. 311612/2021-0, and No. 421792/2022-1, and by the National Institute for the Science and Technology Applied Quantum Computing (INCT-CQA) under Grant No. 408884/2024-0.

[1] R. Ellahi, C. Fetecau, and M. Sheikholeslami, Recent Advances in the Application of Differential Equations in Mechanical Engineering Problems, Mathematical Problems in Engineering **2018**, 1 (2018).

[2] D. Betounes, *Differential Equations: Theory and Applications* (Springer, New York, NY, 2010).

[3] M. T. Xu, Exact solution of nonlinear ordinary differential equations (2020), arXiv:2004.05329.

[4] V. Ardourel and J. Jebeile, On the presumed superiority of analytical solutions over numerical methods, European Journal for Philosophy of Science **7**, 201 (2017).

[5] S. Fadugba, S. Nathaniel Ogunyebi, K. Adebayo, and J. Okunlola, Review of some numerical methods for solving initial value problems for ordinary differential equations, International Journal of Applied Mathematics and Theoretical Physics **6**, 7 (2020).

[6] F. Sapienza, J. Bolibar, F. Schäfer, B. Groenke, A. Pal, V. Boussange, P. Heimbach, G. Hooker, F. Pérez, P.-O. Persson, and C. Rackauckas, Differentiable programming for differential equations: A review (2024), arXiv:2406.09699.

[7] M. S. Kurt Hornik and H. White, Multilayer feedforward networks are universal approximators, Neural Networks **2**, 359 (1989).

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems*, Vol. 25, edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Curran Associates, Inc., 2012).

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is All you Need, Advances in Neural Information Processing Systems **30** (2017).

[10] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, Highly accurate protein structure prediction with alphafold, Nature **596**, 583 (2021).

[11] V. K. Ithapu, S. N. Ravi, and V. Singh, On architectural choices in deep learning: From network structure to gradient convergence and parameter estimation (2017), arXiv:1702.08670.

[12] G. Philipp, The nonlinearity coefficient - a practical guide to neural architecture design (2021), arXiv:2105.12210.

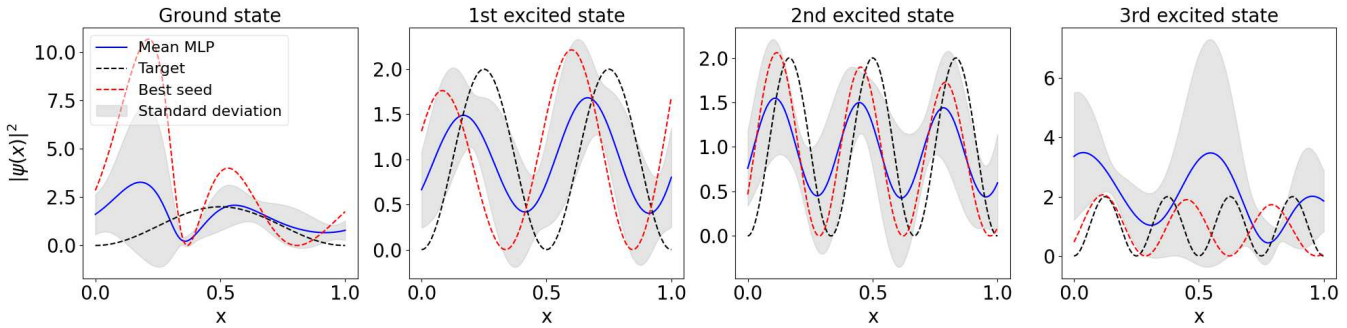
[13] A. Nikanjam and F. Khomh, Design Smells in Deep Learning Programs: An Empirical Study , in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (IEEE Computer Society, Los Alamitos, CA, USA, 2021) pp. 332–342.

[14] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, Neural ordinary differential equations, in *Advances in Neural Information Processing Systems*, Vol. 31, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018).

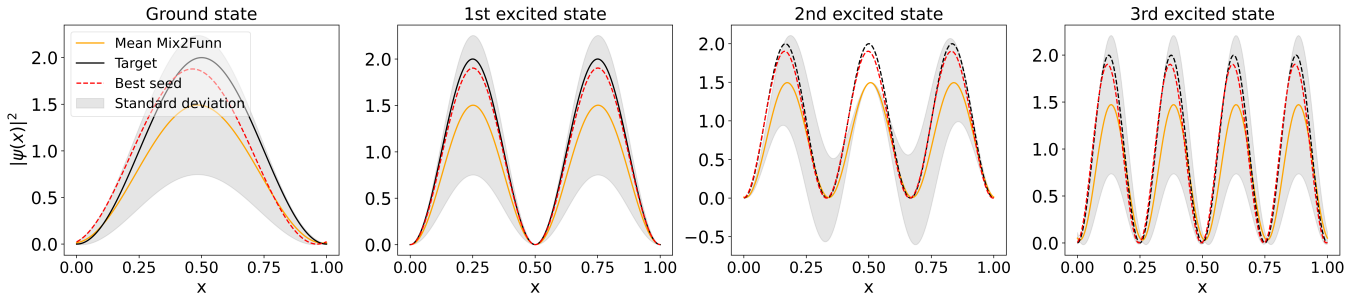
[15] S. Greydanus, M. Dzamba, and J. Yosinski, Hamiltonian neural networks, in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019).

[16] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, Lagrangian neural networks, in *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations* (2020).

[17] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations (2017),



(a)



(b)

FIG. 8: Predicted wavefunctions for (a) MLP and (b) Mix2Funn at four energy levels. The first and second excited states were used for training; the ground state and third excited state test generalization. Mix2Funn generalizes accurately to unseen energy levels; the MLP shows phase errors on the unseen states.

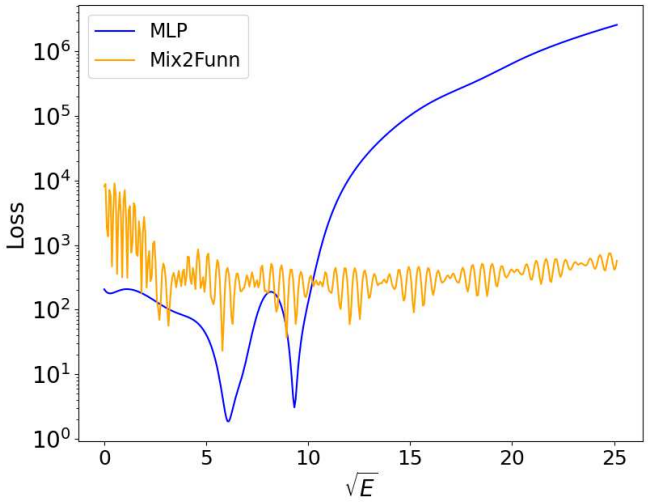


FIG. 9: Residual loss vs. $\sqrt{E}$ for the trained general models. The MLP shows low loss only at training energies. Mix2Funn identifies loss minima at all true eigenvalues, including those outside the training set.

arXiv:1711.10561.

[18] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations (2017), arXiv:1711.10566.

[19] M. Raissi, P. Perdikaris, and G. E. 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 (2019).

[20] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, On the spectral bias of neural networks, in *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, 2019) pp. 5301–5310.

[21] A. Farea and M. S. Celebi, Learnable activation functions in physics-informed neural networks for solving partial differential equations, Computer Physics Communications , 109753 (2025).

[22] H. W. Honghui Wang, L. L. Lu Lu, S. S. Shiji Song, and G. H. Gao Huang, Learning specialized activation functions for physics-informed neural networks, Communications in Computational Physics **34**, 869–906 (2023).

[23] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, Fourier neural operator for parametric partial differential equations, arXiv:2010.08895 (2021).

[24] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, Implicit neural representations with periodic activation functions, in *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Curran Associates, Inc., 2020) pp. 7462–7473.

[25] K. He, X. Zhang, S. Ren, and J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2015).

[26] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, Kan: Kolmogorov-arnold networks (2024), arXiv:2404.19756.

[27] A. N. Kolmogorov, On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition, Doklady Akademii Nauk SSSR **114**, 953 (1957).

[28] V. I. Arnold, On the representation of continuous functions of three variables by superpositions of continuous functions of two variables, Matematicheskii Sbornik **48**, 3 (1959).

[29] J. G. O'Brien and G. D. Pillo, A New Perspective on the Kolmogorov-Arnold Representation Theorem, SSRN Electronic Journal 10.2139/ssrn.5072413 (2024).

[30] J. Schmidt-Hieber, The kolmogorov–arnold representation theorem revisited, Neural Networks **137**, 119 (2021).

[31] Y. Shin and J. Ghosh, The pi-sigma network: an efficient higher-order neural network for pattern classification and function approximation, in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, Vol. I (1991) pp. 13–18 vol.1.

[32] R. M. Pattanayak and H. S. Behera, Higher order neural network and its applications: A comprehensive survey, in *Progress in Computing, Analytics and Networking*, edited by P. K. Pattnaik, S. S. Rautaray, H. Das, and J. Nayak (Springer Singapore, Singapore, 2018) pp. 695–709.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, Attention is all you need, in *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017).

[34] S. L. Brunton, J. L. Proctor, and J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the National Academy of Sciences **113**, 3932 (2016), https://www.pnas.org/doi/pdf/10.1073/pnas.1517384113.

[35] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Networks **6**, 861 (1993).

[36] J. Bridle, Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters, in *Advances in Neural Information Processing Systems*, Vol. 2, edited by D. Touretzky (Morgan-Kaufmann, 1989).

[37] X. Zhang, F. X. Yu, S. Karaman, W. Zhang, and S.-F. Chang, Heated-up softmax embedding (2018), arXiv:1809.04157.

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research **15**, 1929 (2014).

[39] J. Frankle and M. Carbin, The lottery ticket hypothesis: Finding sparse, trainable neural networks, in *International Conference on Learning Representations* (2019).

[40] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization (2017), arXiv:1412.6980.

[41] A. S. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Advances in Neural Information Processing Systems **34** (2021).

[42] J. Burgers, A mathematical model illustrating the theory of turbulence (Elsevier, 1948) pp. 171–199.

[43] M. P. Bonkile, A. Awasthi, C. Lakshmi, V. Mukundan, and V. S. Aswin, A systematic literature review of Burgers' equation with recent advances, Pramana **90**, 69 (2018).

[44] N. K. Kumar, A review on burgers' equations and it's applications, Journal of Institute of Science and Technology **28** (2023).

[45] J. Bu and A. Karpatne, Quadratic residual networks: A new class of neural networks for solving forward and inverse problems in physics involving pdes, in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 675–683, https://epubs.siam.org/doi/pdf/10.1137/1.9781611976700.76.

[46] D. J. Griffiths and D. F. Schroeter, *Introduction to Quantum Mechanics*, 3rd ed. (Cambridge University Press, Cambridge, 2018).

[47] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 9, edited by Y. W. Teh and M. Titterington (PMLR, Chia Laguna Resort, Sardinia, Italy, 2010) pp. 249–256.

Appendix A: Physics-Informed Neural Networks formulation

For completeness, we provide a detailed overview of the PINN framework used throughout this work.

We consider a differential equation written in operator form as:

$$\mathcal{D}[u_{\text{true}}](\mathbf{v}) = 0, \tag{A1}$$

where u_{true} is an unknown function defined on the domain $u_{\text{true}} : \Omega \subset \mathbb{F}^d \to \mathbb{K}^m$, with $\mathbb{F}$ and $\mathbb{K}$ denoting either $\mathbb{R}$ or $\mathbb{C}$, d the number of independent variables, and m the number of dependent variables. Here $\mathbf{v} \in \Omega$ denotes the set of independent variables.

To ensure a unique and physically meaningful solution, the differential equation is supplemented with initial and/or boundary conditions (IC/BCs) imposed on the boundary of the domain $\partial\Omega$:

$$\mathcal{B}[u_{\text{true}}](\mathbf{v}) = g(\mathbf{v}), \quad \text{for } \mathbf{v} \in \partial\Omega, \tag{A2}$$

where $\mathcal{B}$ is a boundary operator and $g(\mathbf{v})$ is a known function.

In the PINN framework, the solution u_{true} is approximated by a neural network u_θ with parameters θ. The residual of the differential equation is defined as

$$r(\mathbf{v}) = \mathcal{D}[u_\theta](\mathbf{v}), \tag{A3}$$

which is generally non-zero for an untrained network. The residual loss is computed as the mean squared error over a set of N_{res} collocation points $\{\mathbf{v}^i_{\text{res}}\}_{i=1}^{N_{\text{res}}}$ sampled from the domain Ω:

$$\mathcal{L}_{\text{res}} = \frac{1}{N_{\text{res}}} \sum_{i=1}^{N_{\text{res}}} \left|\mathcal{D}[u_\theta](\mathbf{v}^i_{\text{res}})\right|^2. \tag{A4}$$

Collocation points are typically sampled randomly from the domain at each training iteration.

Minimizing the residual loss alone is insufficient to guarantee a unique solution. To enforce the initial and boundary conditions, additional loss terms are introduced. The boundary loss is defined as

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{\text{BC}}} \sum_{j=1}^{N_{\text{BC}}} \left|\mathcal{B}[u_\theta](\mathbf{v}^j_{\text{BC}}) - g(\mathbf{v}^j_{\text{BC}})\right|^2, \tag{A5}$$

where $\{\mathbf{v}^j_{\text{BC}}\}_{j=1}^{N_{\text{BC}}}$ are points sampled from the boundary $\partial\Omega$.

Similarly, the initial condition loss is given by

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_{\text{IC}}} \sum_{j=1}^{N_{\text{IC}}} \left|u_\theta(\mathbf{v}^j_{\text{IC}}) - u_{\text{true}}(\mathbf{v}^j_{\text{IC}})\right|^2, \tag{A6}$$

where $\{\mathbf{v}^j_{\text{IC}}\}_{j=1}^{N_{\text{IC}}}$ are the points at which initial conditions are specified.

The total loss function minimized during training is a weighted sum of these components:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{res}}\mathcal{L}_{\text{res}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}} + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}}, \tag{A7}$$

where λ_{res}, λ_{IC}, and λ_{BC} are hyperparameters controlling the relative importance of each term.

By minimizing $\mathcal{L}_{\text{total}}$, the neural network learns an approximate solution that satisfies both the governing differential equation and the associated boundary and initial conditions, leading to a physically consistent approximation of the true solution.

When evaluating generalization outside the training domain (e.g., extrapolation in time), it is typically assumed that the governing equations and boundary/initial conditions remain valid, such that the extended problem remains well-posed.

Appendix B: Implementation details

1. Efficient implementation of the second-order neuron

The quadratic form of the second-order neuron,

$$s = b + \mathbf{W}\mathbf{x} + \mathbf{x}^T\mathbf{U}\mathbf{x}, \tag{B1}$$

can be implemented efficiently by expanding the input into an extended feature vector:

$$\mathbf{x}_{\text{ext}} = \text{concat}\left(\mathbf{x},\ \text{vech}(\mathbf{x}\mathbf{x}^T)\right), \tag{B2}$$

where $\text{vech}(\cdot)$ extracts the lower triangular elements (excluding the diagonal) of $\mathbf{x}\mathbf{x}^T$, yielding a vector of dimension $N + \frac{N(N-1)}{2}$. The neuron output then reduces to a single linear transformation:

$$s(\mathbf{x}) = b + \mathbf{W}_{\text{comp}}\,\mathbf{x}_{\text{ext}}, \tag{B3}$$

where $\mathbf{W}_{\text{comp}} \in \mathbb{R}^{1\times\frac{N(N+1)}{2}}$ contains both linear and quadratic coefficients. This formulation is mathematically equivalent to the quadratic form and is suitable for efficient GPU implementation.

2. Initialization strategy

Training second-order neurons presents a stability challenge: quadratic features can have significantly larger variance than linear features, potentially causing gradient explosion if a single initialization scheme is applied uniformly.

To address this, we adopt a hybrid initialization strategy. The linear weights are initialized using Glorot initialization [47], while the quadratic weights are initialized from a zero-mean normal distribution with small variance. This ensures the network behaves approximately linearly at the start of training, promoting stability during early optimization. We found this strategy to be consistently stable across all experiments reported in this work.

3. Regularization procedures

a. Softmax-based functional selection

To encourage sparse selection of basis functions, the mixing weights w_i' are parameterized through logits α_i and a temperature-controlled softmax:

$$w_i' = c\frac{e^{\alpha_i/T}}{\sum_{j=1}^{Q} e^{\alpha_j/T}}, \tag{B4}$$

where c is a learnable scaling factor and T is the temperature parameter. During training, T can be gradually decreased (annealed), causing the distribution to sharpen. In the limit $T \to 0$, the softmax approaches a one-hot selection, effectively activating a single dominant basis function and promoting interpretability [36, 37].

b. Dropout on functional pathways

We apply dropout directly to the outputs of the functional pathways rather than to neuron outputs. For a given input $\mathbf{z}$:

1. Compute $s_i = \mathbf{W}_i\mathbf{z} + b_i$ and evaluate $f_i(s_i)$ for all i.
2. Sample a binary mask $\mathbf{m} \in \{0,1\}^Q$, where each $m_i \sim \text{Bernoulli}(1-p)$.
3. Apply the mask: $\tilde{f}_i(s_i) = m_i f_i(s_i)$.
4. Compute the neuron output:

$$a = b' + \sum_{i=1}^{Q} w_i' \tilde{f}_i(s_i). \tag{B5}$$

This targeted form of dropout prevents the network from relying on a single functional component, encouraging more robust and distributed representations across the basis.

c. Structural pruning

To obtain compact and interpretable models, we perform magnitude-based pruning on the mixing weights w_i':

1. Train the full model to convergence.
2. Identify mixing weights w_i' with the smallest absolute values.
3. Set these weights to zero, permanently removing the corresponding functional pathways.
4. Fine-tune the pruned model to recover any lost performance.

Unlike standard weight pruning, which removes individual parameters, this approach eliminates entire functional components. As a result, the learned representation is simplified at the level of symbolic terms, enabling direct extraction of concise analytical expressions [39]. When softmax parameterization is used, pruning is equivalently implemented by assigning large negative values to the corresponding logits α_i.

Appendix C: Additional results

1. Forced harmonic oscillator

To assess performance under an external driving force, we consider the forced harmonic oscillator with $F(t) = F_0 \sin(\omega t)$, in the near-resonance regime with $m = 1$, $k = 1$, $\gamma = 0.1$, $F_0 = 1$, and $\omega = 0.9$. The training domain is $t \in [0, 40]$; the test domain is $t \in (40, 80]$.

Figure 10 compares the predicted solutions. The baseline MLP collapses to a constant value immediately upon leaving the training domain. Mix2Funn correctly extrapolates the oscillatory behavior but exhibits a slight unphysical amplitude growth, indicating minor instability. This trade-off is quantified in Fig. 11: Mix2Funn maintains substantially lower cumulative error throughout the evaluated domain, though its error accelerates for $t > 80$ due to the mild instability.

2. Impact of training domain size on generalization

We investigate how the size of the training domain affects generalization for Mix2Funn on the damped harmonic oscillator. The training interval $t \in [0, T_{\max}]$ is varied with $T_{\max} \in \{10, 20, 40, 60, 80\}$, using 1024 collocation points in each case. Generalization is evaluated on a fixed test domain $t \in (80, 150]$.

As shown in Fig. 12, generalization error decreases monotonically with $T_{\max}$. A larger training domain forces the network to find a parameterization valid over a longer time horizon, which naturally improves extrapolation. This supports the interpretation that Mix2Funn learns the underlying dynamics rather than simply interpolating.

3. Basis function ablation

To assess the sensitivity of MixFunn to basis function selection, we compare the full basis $\{\sin, \cos, \exp, \log, \sqrt{\cdot}, s\}$ against a restricted basis with the periodic functions $(\sin, \cos)$ removed, on the damped harmonic oscillator.

Figure 13 shows a clear performance gap. The full-basis model converges rapidly to a low residual error, while the restricted model converges more slowly and

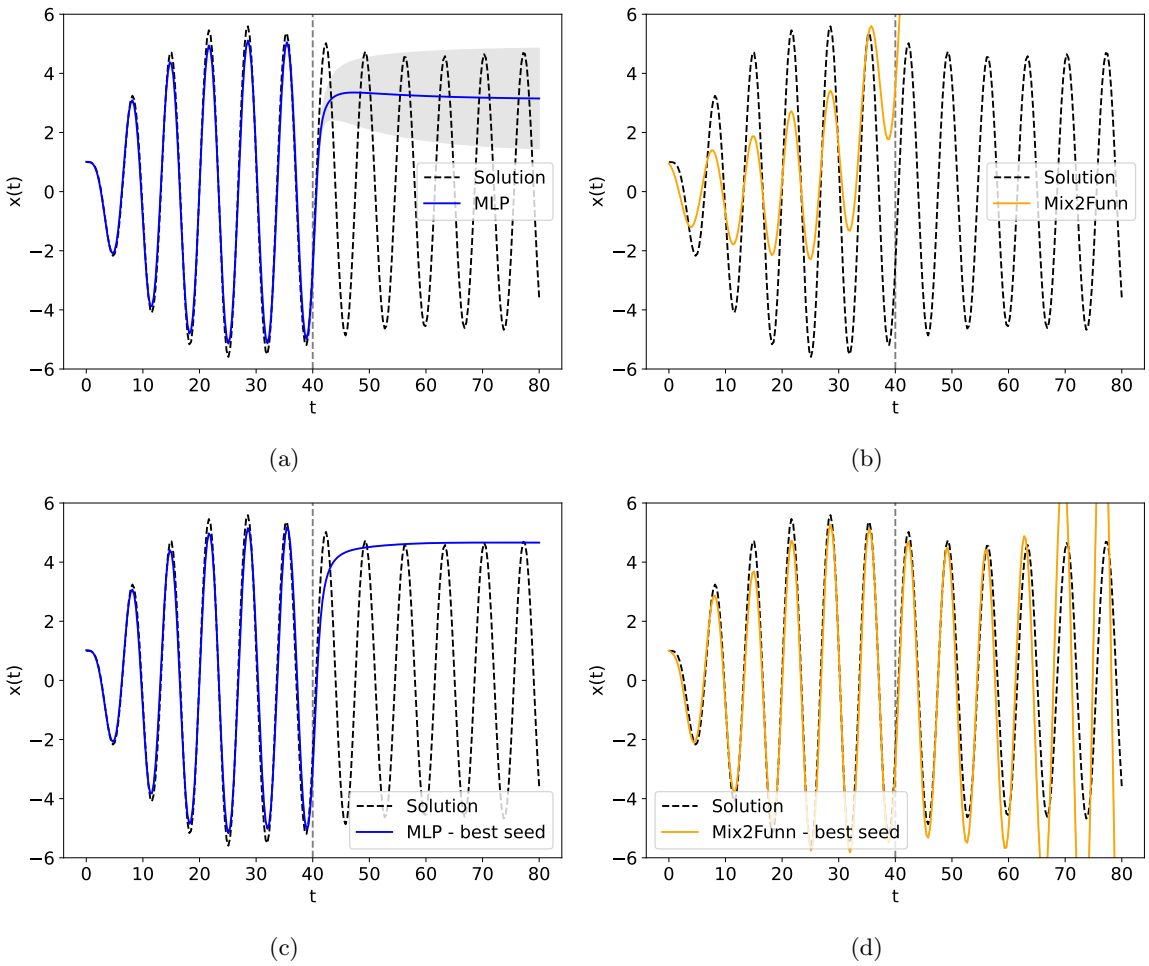


FIG. 10: Predicted solutions for the forced harmonic oscillator. The MLP (a,c) collapses to a constant value in the test domain. The best Mix2Funn run (d) correctly extrapolates frequency and phase but shows mild amplitude growth.

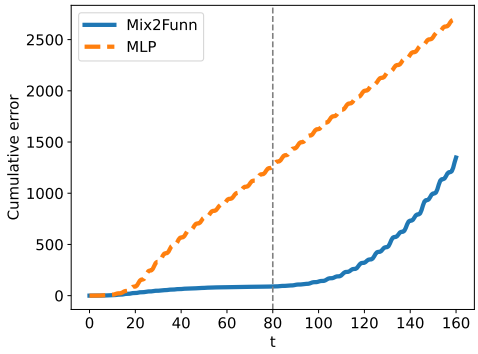


FIG. 11: Cumulative error for the forced harmonic oscillator. Mix2Funn maintains lower total error throughout the evaluated domain; the accelerating growth for $t > 80$ reflects the mild instability visible in Figure 10.

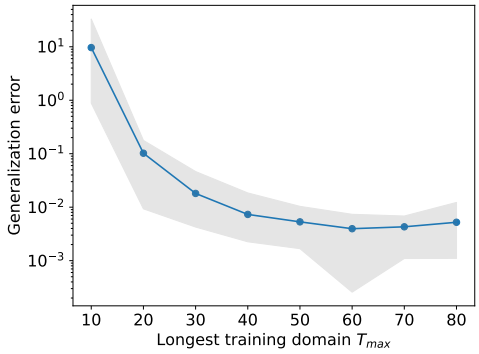


FIG. 12: Generalization error as a function of training domain size $T_{\max}$ for Mix2Funn. Larger training domains consistently yield lower error on the held-out test domain.

plateaus at a higher error, behaving similarly to a standard MLP. This confirms that including basis functions aligned with the expected solution form is important for efficiency, while the architecture remains functional even with a suboptimal basis.

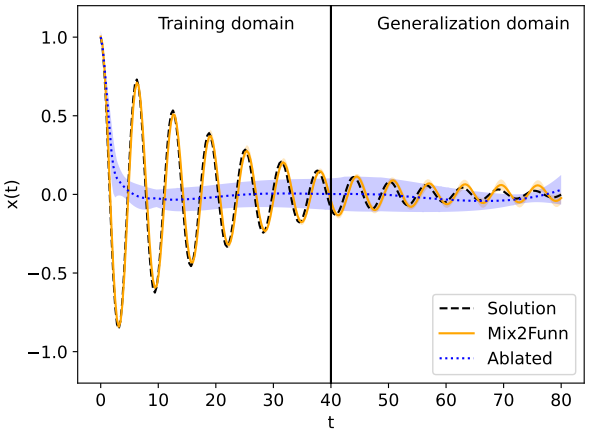


FIG. 13: Convergence comparison between the full-basis MixFunn (including sin, cos) and a restricted-basis variant (monotonic functions only). Shaded regions indicate one standard deviation over five runs. The restricted basis leads to slower convergence and higher final error.

4. Iterative pruning procedure for Burgers equation

The analytical expression for the Burgers equation required an iterative pruning process. Starting from the best-performing Mix2Funn model (0% pruning, 56 parameters), we identified secondary loss minima at 20% and 80% pruning ratios. We selected the 80% pruned model and fine-tuned it for 1000 additional epochs, retaining the checkpoint with the lowest training loss. This was followed by a second fine-tuning phase at 90% pruning under the same protocol, yielding a final model with 4 parameters. The extracted expression is reported in the main text.