Structural Variations in PINNs: The Role of Activation Functions and Noise

Kushal Gurrapu
Department of Mathematical Sciences
University of Delaware
Newark, DE, USA
kushalg@udel.edu

Owen He
Department of Mathematical Sciences
University of Delaware
Newark, DE, USA
heowen@udel.edu

***Abstract*—Physics-Informed Neural Networks (PINNs) solve partial differential equations (PDEs) by embedding physical laws directly into the loss function of neural networks. This study investigates how the choice of activation function and the presence of additive Gaussian noise affect PINN performance on the nonlinear Schrödinger equation. We compare six activation functions, ReLU, Tanh, Sigmoid, Swish, HardSwish, and Mish, across zero, low, and medium noise levels. Our results show that smooth and non-monotonic activations such as Swish and Mish yield the best generalization and robustness, while piecewise functions like ReLU and HardSwish suffer from poor accuracy and noise sensitivity. These findings underscore the importance of activation smoothness and highlight architectural considerations for deploying PINNs in noisy or uncertain environments.**

***Index Terms*—Physics-Informed Neural Networks, Activation Functions, Schrödinger Equation, Partial Differential Equations, Gaussian Noise, Scientific Machine Learning.**

I. Introduction

Physics-Informed Neural Networks (PINNs) are a class of deep learning models that integrate physical laws, typically expressed as partial differential equations (PDEs), into the loss function [1]. While PINNs have demonstrated success across various scientific domains, their performance can be sensitive to architectural choices and data conditions. In particular, the selection of activation functions and the presence of noise may significantly affect training dynamics and solution accuracy.

This paper aims to study the impact of activation function choice and additive Gaussian noise on the training and performance of PINNs. Our experiments focus on the one-dimensional time-dependent nonlinear Schrödinger equation. To simulate realistic measurement conditions, we introduce controlled levels of Gaussian noise directly into the training data. Our work seeks to address two key research questions:

- RQ1: How do different activation functions affect the stability and accuracy of PINNs when solving the nonlinear Schrödinger equation?
- RQ2: How robust are PINNs to varying levels of additive Gaussian noise in the training data, and how does this robustness vary across different activation functions?

II. Related Work

Recent advancements in Physics-Informed Neural Networks (PINNs) have demonstrated their effectiveness in solving forward and inverse PDE problems, especially when conventional numerical methods become infeasible due to sparse data or high computational cost.

Zhang [2] analyzed deep networks under various activation functions, showing that approximation guarantees traditionally limited to ReLU networks can be extended to a broad class of activations with only modest increases in width and depth requirements. Pilar and Wahlström [3] extended PINNs to handle non-Gaussian noise by co-training an energy-based model to learn the noise distribution. To address instability during training, Cheng and Zhang [4] incorporated ResNet blocks into the PINN framework and showed that deeper architectures with enhanced activation functions improved accuracy under noisy conditions. Finally, Mishra and Molinaro [5] provided theoretical bounds on PINN generalization errors, emphasizing the importance of architecture and regularization when data is limited or uncertain.

Building on this prior work, our study focuses specifically on how different activation functions influence PINN performance across varying levels of Gaussian noise.

A. Activation Function Variations

Activation functions play a pivotal role in determining the expressiveness, smoothness, and convergence behavior of Physics-Informed Neural Networks (PINNs). In this work, we evaluate the performance of several standard and advanced activation functions within the PINN framework:

1) Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$. Maps real values into the range $(0, 1)$. It is a smooth, bounded, and differentiable function commonly used in early neural networks and binary classification tasks.
2) Tanh: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$. Maps inputs to the range $(-1, 1)$ and centers outputs around zero. Also smooth and differentiable.
3) Rectified Linear Unit: $\text{ReLU}(x) = \max(0, x)$. Non-linear but still vanishing gradient problem for $x > 0$. Not differentiable at $x = 0$.
4) Swish: $\text{Swish}(x) = x \cdot \sigma(\beta x) = \frac{x}{1+e^{-\beta x}}$, where σ is the Sigmoid function and β is either constant or trainable. We have $\beta = 1$. Smooth and non-monotonic.
5) HardSwish: $\text{HardSwish}(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6}$, where $\text{ReLU6}(x) = \min(\max(0, x), 6)$. An efficient approxi-

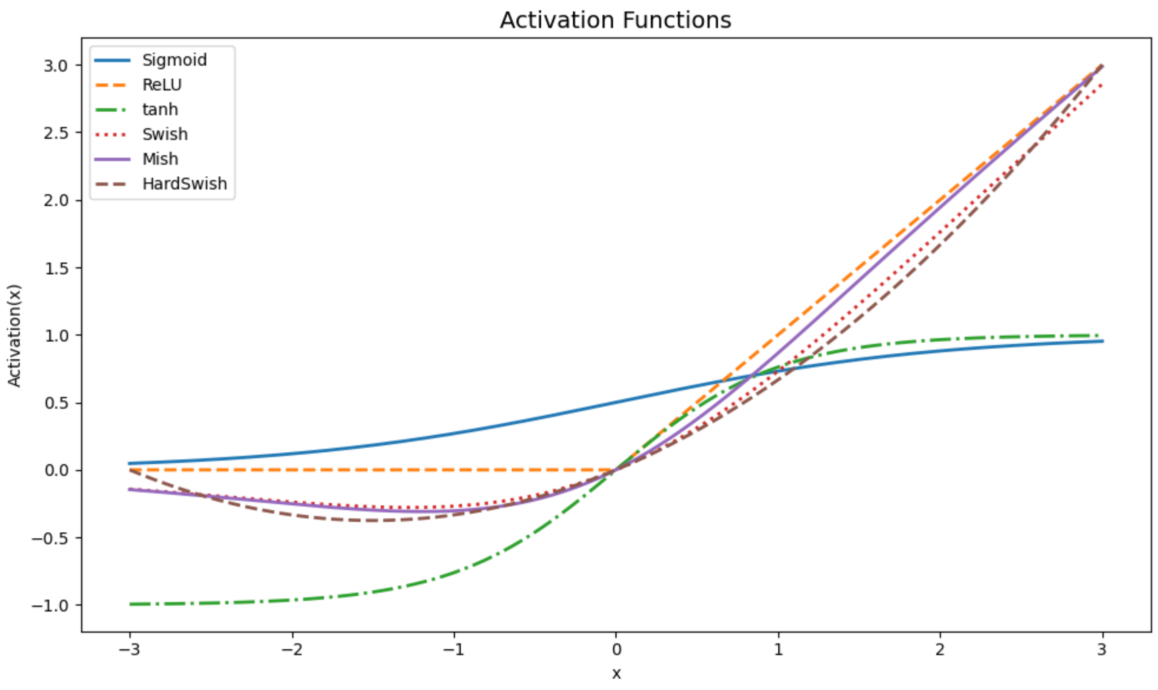


Fig. 1: Visualization of Activation Functions.

mation of Swish, commonly used for devices with limited processing power.

6) Mish: $\text{Mish}(x) = x \cdot \tanh(\ln(1 + e^x))$. Smooth and self-regularized, providing strong empirical performance in some deep learning tasks.

These activation functions offer a range of properties in terms of smoothness, non-linearity, and computational cost. In PINNs, where accurate gradients and stable training are important, the activation function can strongly influence performance. In our experiments, we compare these activations function under different noise levels to analyze their effect on training.

III. Methods

A. Time-Dependent 1-D Schrödinger Equation

In quantum mechanics, it governs the behavior of particles like electrons and photons by treating them like waves. Instead of describing a particle's exact position or momentum, the equation yields a wave function, denoted $h(t,x)$, whose magnitude squared, $|h(t,x)|^2$, gives the probability of finding the particle at a certain location and time.

In this work, we study the nonlinear Schrödinger equation in one spatial dimension:

$$ih_t + \frac{1}{2}h_{xx} + |h|^2 h = 0,$$

with initial condition

$$h(0,x) = 2\,\text{sech}(x),$$

and periodic boundary conditions

$$h(t,-5) = h(t,5), \quad h_x(t,-5) = h_x(t,5),$$

defined over the domain $x \in [-5,5]$, $t \in [0,\pi/2]$. Here, the solution, $h(t,x)$, is a complex-valued wave function, with h_t and h_{xx} denoting partial derivatives with respect to time and space.

B. Dataset Description

The dataset used in this study represents a numerical solution to the one-dimensional nonlinear Schrödinger equation under periodic boundary conditions. It is provided in MATLAB `.mat` format and contains three key variables:

- x: A vector of 256 spatial grid points uniformly distributed over the domain $[-5,5]$.
- tt: A vector of 201 time steps spanning the interval $[0,\pi/2]$.
- uu: A complex-valued matrix of shape 256×201 representing the wave function $h(t,x)$ at each point in space and time.

C. PINN Architecture

Our PINN model maps coordinates (t,x) to the real and imaginary components of the solution, $u(t,x)$ and $v(t,x)$. The PINN architecture used is a fully connected feedforward neural network with 4 hidden layers of 100 neurons each and a total structure of $[2] + 4 \times [100] + [2]$, where the input is the pair (t,x) and the output is (u,v).

Each hidden layer contains 100 neurons and applies a fixed activation function, while the output layer is linear to allow unbounded outputs. The network is trained using mean squared error loss over three main types of residuals:

- Initial condition: enforcing $h(0,x) = 2\,\text{sech}(x)$,

- Periodic boundary: enforcing $h(t,-5) = h(t,5)$ and $h_x(t,-5) = h_x(t,5)$,
- PDE: enforcing the Schrödinger equation itself.

We define the mesh as a cartesian product grid over our space and time domains with 256 spatial points and 201 time points. Training data is generated from three sources:

- $N_f = 20000$ collocation points within the interior of the domain,
- $N_0 = 50$ points from the initial condition,
- $N_b = 50$ points from the periodic boundary conditions.

The network is trained for up to 60,000 epochs and the final model returns predictions for the full solution $h(t,x) = u(t,x) + iv(t,x)$ over the domain $[0, \frac{\pi}{2}] \times [-5,5]$ (time $\times$ space).

The loss function combines the mean squared errors from all three components:

$$\text{Loss} = \text{MSE}_0 + \text{MSE}_b + \text{MSE}_c,$$

where MSE_0 enforces the initial condition, MSE_b enforces the boundary conditions, and MSE_c enforces the PDE residuals at the collocation points. The model is trained using the Adam optimizer for gradient descent.

D. Implementation

For all experiments in this study, we used the `pinns-torch` framework introduced by Bafghi et al. [6], which is a PyTorch-based library designed to streamline the implementation of physics-informed neural networks. As the nonlinear Schrödinger equation is already included among the examples provided by the library, we retained its formulation and modified the training pipeline to inject configurable levels of Gaussian noise into the data.

IV. Experiments

A. Experimental Setup

To approximate the nonlinear Schrödinger equation using a Physics-Informed Neural Network (PINN), we model the complex-valued solution $h(t,x) = u(t,x) + iv(t,x)$ by training a neural network that outputs both the real and imaginary parts: $[u(t,x), v(t,x)]$. The governing PDE is reformulated into a system of two real-valued equations:

$$f_u = u_t + 0.5 v_{xx} + v(u^2+v^2), \quad f_v = v_t + 0.5 u_{xx} + u(u^2+v^2).$$

The domain for the solution is set to $x \in [-5,5]$ and $t \in [0, \pi/2]$, with the following conditions:

- Initial Condition:

$$u(0,x) = 2\,\text{sech}(x), \quad v(0,x) = 0.$$

- Periodic Boundary Conditions:

$$u(t,-5) = u(t,5), \quad v(t,-5) = v(t,5),$$

$$u_x(t,-5) = u_x(t,5), \quad v_x(t,-5) = v_x(t,5).$$

B. Activation Functions

We evaluate six activation functions in our experiments: Sigmoid, Tanh, ReLU, Swish, HardSwish, and Mish. In each experiment, the activation function is applied after every hidden layer in the PINN, acting element-wise on the output of each neuron. All other aspects of the architecture are kept identical across experiments to make our comparisons fair.

C. Noise Settings

To determine appropriate noise levels, we first examined the dataset and computed the mean magnitude of the wave function as $\bar{h} \approx 0.5828$. Based on this value, we selected standard deviations of 0.01 and 0.05 to represent approximately 1.7% and 8.6% of the signal magnitude, respectively. This ensures that the introduced noise remains proportional to the scale of the underlying solution. To evaluate the robustness of each activation function, we conduct experiments under three levels of noise:

- Zero Noise: The dataset remains unaltered, using the exact real and imaginary components of the wave function without any perturbation. This serves as a baseline for performance comparison.
- Low Noise: Additive Gaussian noise with standard deviation 0.01.
- Medium Noise: A higher level of Gaussian noise is introduced, with standard deviation 0.05.

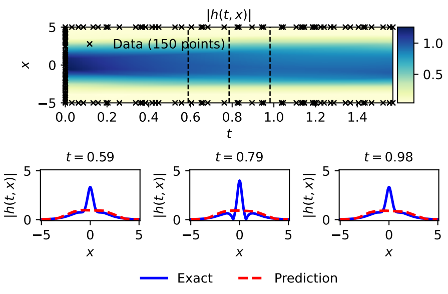
Noise is injected directly into the real and imaginary parts of the wave function by sampling from a normal distribution with zero mean and the specified standard deviation. After noise is added, the modulus $|h|$ is recomputed to maintain consistency in the target outputs. The same noise levels are applied uniformly across all activation function experiments.

D. Results

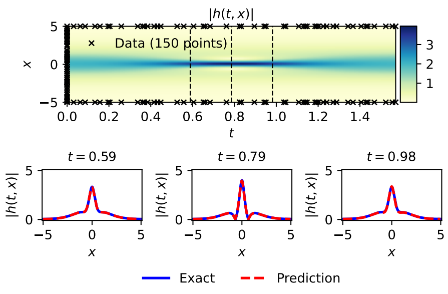
Activation Function		Noise Level		
		Zero	Low	Medium
Sigmoid	val_error_h	0.1263	0.1270	0.1655
	train_loss	0.0030	0.0029	0.0094
Tanh	val_error_h	0.0203	0.0207	0.0802
	train_loss	0.0047	0.0027	0.0032
ReLU	val_error_h	0.5129	0.5045	0.5359
	train_loss	0.1033	0.1052	0.1281
Swish	val_error_h	0.0079	0.0200	0.0781
	train_loss	0.0014	0.0003	0.0030
HardSwish	val_error_h	0.4749	0.4667	0.4855
	train_loss	8.833e-5	8.368e-5	0.0012
Mish	val_error_h	0.0108	0.0211	0.0793
	train_loss	0.0012	0.0029	0.0042

TABLE I: Validation error on h and training loss across different noise levels.

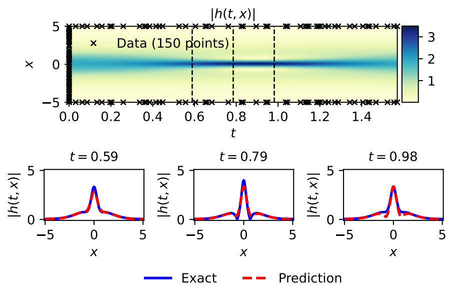
Under zero noise, the ordering of activation functions from best to worst validation performance is as follows: Swish, Mish, tanh, Sigmoid, HardSwish, and ReLU.



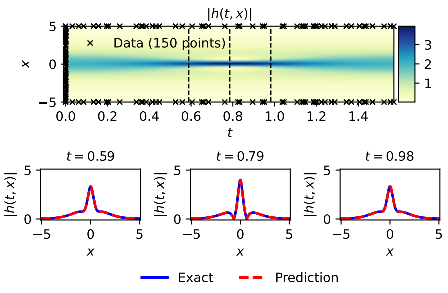
(a) ReLU



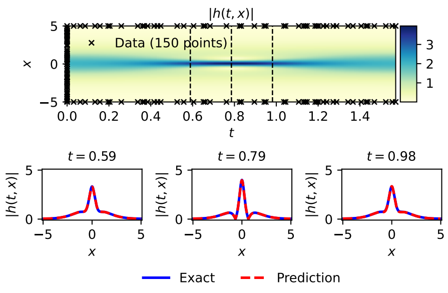
(b) tanh



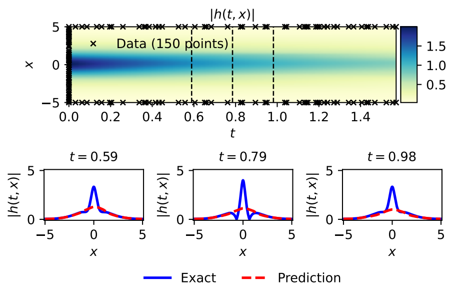
(c) Sigmoid



(d) Swish



(e) Mish



(f) HardSwish

Fig. 2: Comparison of activation functions under Zero Noise.

As noise levels increase, all activation functions suffer in performance. However, the relative ranking remains mostly consistent. At low noise, Swish, Mish, and Tanh continue to perform well, while Sigmoid, HardSwish, and ReLU get left behind. The pattern continues under medium noise as well.

Training losses were generally low for all functions, as expected. In particular, HardSwish achieved near-zero training losses (8.833×10^{-5} under zero noise), yet this did not correlate with improved validation performance. In contrast, Swish and Mish demonstrated both low training loss and minimal validation error, meaning better generalization and robustness.

V. Discussion and Conclusion

The observed performance patterns immediately show the difference between smooth, differentiable functions, and non-smooth, piecewise functions. ReLU and HardSwish, both of which are piecewise and non-smooth, consistently underperform. This follows from the fact these types of functions can't approximate the wave functions of the Schrödinger equation as well as smooth ones. Additionally, the difference in training and validation loss suggests significant overfitting.

In contrast, smoother and differentiable functions like Tanh, Mish, and Swish are more naturally aligned with the structure of our PDE. Tanh, with its symmetry and bounded range, remains stable across noise settings and performs reliably. Swish and Mish which aren't strictly increasing or decreasing, allow better gradients during training, which likely accounts for their superior performance.

Although Sigmoid is also smooth, its tendency to suffer from vanishing gradients limit its effectiveness in deeper networks, placing it below Tanh, Mish, and Swish in performance.

Smooth and expressive activation functions are well-suited for solving PDEs with PINNs, especially under varying noise levels. On the other hand, piecewise activations like ReLU and HardSwish show poor generalization and are more sensitive to noise. These results reinforce the importance of activation function choice in scientific machine learning, particularly in settings with noisy training data.

We only analyzed performance for only one PDE, the Schrödinger equation, and only one PINN model. Future work could explore whether these findings generalize to other PDEs with different dynamics as well as deeper or more complex neural networks. Additionally, none of our activation functions contained learnable parameters, which could add a new level of depth to the problem. These extensions hold potential for further advancing the design of PINNs in real-world applications.

References

[1] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999118307125

[2] S. Zhang, J. Lu, and H. Zhao, "Deep network approximation: Beyond relu to diverse activation functions," 2024. [Online]. Available: https://arxiv.org/abs/2307.06555

[3] P. Pilar and N. Wahlström, "Physics-informed neural networks with unknown measurement noise," in *Proceedings of the 6th Annual Learning for Dynamics amp; Control Conference*, ser. Proceedings of Machine Learning Research, A. Abate, M. Cannon, K. Margellos, and A. Papachristodoulou, Eds., vol. 242. PMLR, 15–17 Jul 2024, pp. 235–247. [Online]. Available: https://proceedings.mlr.press/v242/pilar24a.html

[4] C. Cheng and G.-T. Zhang, "Deep learning method based on physics informed neural network with resnet block for solving fluid flow problems," *Water*, vol. 13, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2073-4441/13/4/423

[5] S. Mishra and R. Molinaro, “Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes,” *IMA Journal of Numerical Analysis*, vol. 42, no. 2, pp. 981–1022, 06 2021. [Online]. Available: https://doi.org/10.1093/imanum/drab032

[6] R. A. Bafghi and M. Raissi, “PINNs-torch: Enhancing speed and usability of physics-informed neural networks with pytorch,” in *The Symbiosis of Deep Learning and Differential Equations III*, 2023. [Online]. Available: https://openreview.net/forum?id=nl1ZzdHpab