Topologically-Infused, Physics-Informed Neural Networks (TIPINN) using Sub-Level Set Persistence

Collin J. Kovacs
University of Tennessee - Knoxville

***Abstract*—The physics-informed neural network (PINN) is a type of neural network that uses the shape and curvature of a curve produced by a neural network instead of relying on the actual training data of the solution like a standard neural network model. In this paper, we propose a topological version of the PINN. Instead of encoding the recovered differential equation as a guidance system for our loss, we additionally implement sub-level set persistence on the recovered equation to encode topological information rather than physical. We use the Cooling Law as our case study by comparing the model's training using the different losses and how the losses compare when an increasing noise level is introduced to the training data. TIPINN's performance was similar to a basic differential model, even when additional noise was introduced.**

***Index Terms*—persistent homology, PINNs, sub-level persistence**

I. Introduction

Since the inception of physics-informed neural networks (PINNs) [4], many papers have studied and analyzed the efficacy of accurately predicting the solution to the given differential equation. PINNs have sparked significant creative behavior for neural networks beyond their initial construction. Different authors have looked into ways of tuning a PINN to their specific problem either through abstracting the domain [5], changing inter-layer activation functions [6], or employing dropout between layers for effective noise prediction [9]. Some papers have analyzed where PINNs fail and provided solutions to allow PINNs to overcome these intricate subtleties [8], [11]. Due to this large influx of creative abstractions of PINNs, review papers [10] detailing all of the different PINN models to the date of the paper's publication. Overall, PINNs provide interesting physical reinforcement, whether the data represents a noisy environment or its differential equation may not have an analytical solution.

In this paper, we propose a topological version of the PINN. Instead of encoding the recovered differential equation as a guidance system for an additional loss constraint, we sequentially implement sub-level set persistence on the recovered differential equation to encode topological information rather than physical. It is necessary to note that the new method, Topologically Infused Physics Informed Neural Networks (TIPINN), does not simply replace the physical loss but enhances it further using persistent homology. The subsequent sections provide a comprehensive description of both the sub-level set persistence and the exact loss function. We will interchangeably reference the PINN as the physical model and TIPINN as the topological model. The method utilizes the shape and curvature of the differential curve produced by a neural network instead of relying on the solution's actual training data like a standard neural network model.

To our knowledge, no other method combines persistence homology and physics-informed neural networks. This paper aims to provide a novel interpretation of curves by not necessarily guiding prediction by physics information but by nudging the prediction in the way of recovered physical information topologically.

II. Background

A. Physics Informed Neural Networks

PINNs are neural network models, often developed through linear layers, that predict solutions to differential equations [4]. In general, PINNs solve an abstraction of a differential equation $\mathcal{F}(u(x,t)) = 0$ for $t \in [0,T]$ and $x \in \Omega \subset \mathbb{R}^d$. Here, $\mathcal{F}$ is the differential operator, $u(x,t)$ is the state variable, Ω is the spatial domain, with x being the space and t being time.

A generalized data-driven approach from a neural network model can be expressed as $u_t + \mathcal{F}[u(t,x)] = 0$ for all $x \in \Omega$ and $t \in [0,T]$. Here, $u(t,x)$ represents the latent solution obtained from the neural network. Furthermore, in the case of continuous time models, we can let $f(t,x)$ as

$$f(t,x) \coloneqq u_t + \mathcal{F}[u(t,x)]$$

where we approximate $u(t,x)$ through the neural network. Therefore, we can define the crux of the physics-informed neural networks using this previous definition. Since neural networks' basis is optimization by minimization of loss, we need to add an extra condition of loss to ensure physical minimization. Since this problem is generally a regression problem, we aim to use mean square error (MSE) and minimize this as much as possible. A typical loss minimization problem of a PINN would be minimizing ε_u while incorporating the physical information. We add the physical loss ε_f to get $\varepsilon = \varepsilon_u + \omega\varepsilon_f$ for a weight $\omega \in \mathbb{R}_+$ where

$$\varepsilon_u = \frac{1}{N_u}\sum_{i=1}^{N_u} \left|u(t_u^i, x_u^i) - u^i\right|^2$$

and

$$\varepsilon_f = \frac{1}{N_f}\sum_{i=1}^{N_f} \left|f(t_f^i, x_f^i)\right|^2.$$

Physical loss minimization is a natural process in that we are essentially training a neural network to its given solution and adding a pseudo-regularization term by adding physical information.

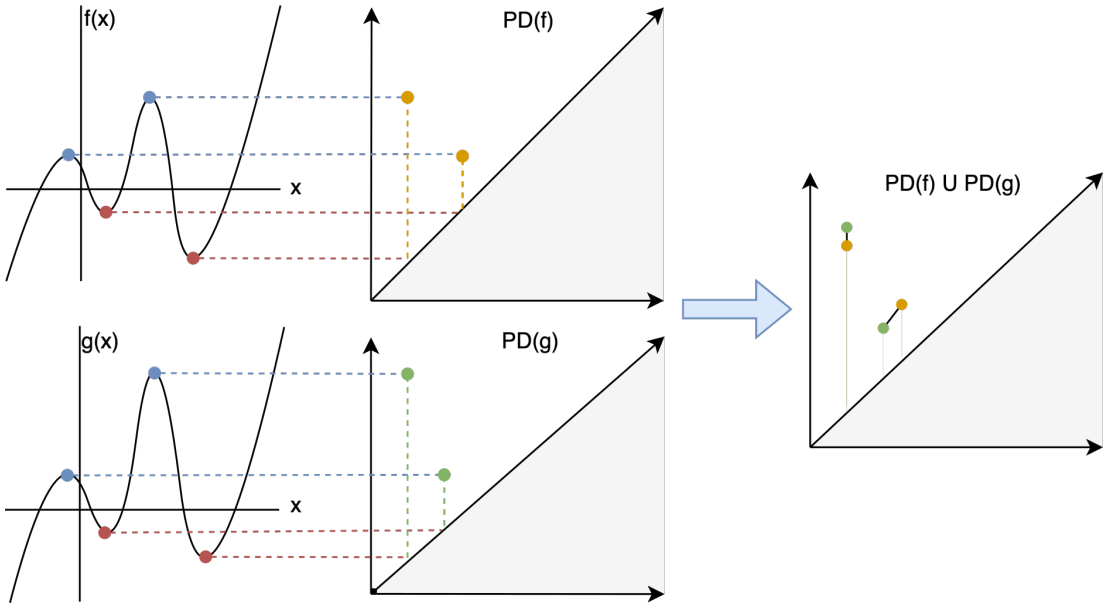


Fig. 1: Visualization of obtaining persistence diagrams from sub-level set persistence for calculating Wasserstein distance.

B. Sub-level Set Persistence

A recent construction [7] outlines sub-level set persistence through an abstract topological framework for constructing a filtration over sub-level sets. Let X be some topological space where our data resides and $f : X \to \mathbb{R}$ be continuous. The sub-level set of f to some $s \in \mathbb{R}$ is defined as

$$F(s) = f^{-1}((-\infty, s]) = \{x \in X | f(x) \leq s\}.$$

Thus, by changing the value of s, we can evaluate the values of x that create the sub-level sets. Furthermore, we can relate this to persistent homology by considering $s \leq s'$, which provides a functional inclusion by $F(s) \subseteq F(s')$. The filtration allows us to follow the homological features of the inclusion of sets as the value of s increases indefinitely. Sub-level set persistence is excellent in categorizing the local extrema within the function.

We can evaluate these structures through building, for the most straightforward cases, simplicial complexes or a collection of simplices that contain multi-dimensional faces like vertices and edges. Commonly, the 0-simplex is a point, the 1-simplex is a line segment, the 2-simplex is a triangle, and the 3-simplex is a tetrahedron. The basis of persistent homology depends on creating a filtration, an increasing sequence of sub-complexes over our given sub-complex. Depending on our problem, we can order this sequence in a specific way to define different filtration complexes like the Vietoris-Rips and Čech Complex. By increasing the filtration value and analyzing how the homological features are created and disappear, we can understand the persistent homological features of the dataset.

But, in the context of structured data like time series, curves, images, or molecules, sub-level set persistence is computed using a Cubical or an Alpha complex [2]. In the case of this paper, we use the Cubical Complex to perform a filtration on this and describe the process of its creation. The boundary of elementary non-degenerate and degenerate intervals of the form $[n, n+1]$ and $[n, n]$ respectively for $n \in \mathbb{Z}$ can be found as

$$\partial[n, n+1] = [n+1, n+1] \setminus [n, n]$$

and $\partial[n, n] = 0$ by [1]. Given these forms of intervals, we can define an elementary cube $\mathcal{C} = I_1 \times \ldots \times I_k$ with its boundary as

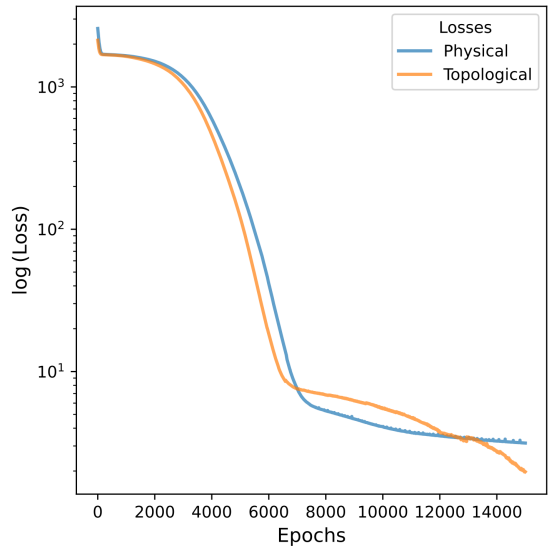
$$\partial\mathcal{C} = \sum_{j=1}^{k} (I_1 \times \cdots \times \partial I_j \times \cdots \times I_k).$$

The cubical complex $\mathcal{K}$ is the collection of cubes $\mathcal{C}$ closed under taking the boundary. In other words, $\mathcal{K}$ is a cubical complex if and only if $\mathcal{C} \in \mathcal{K}$, then $\partial\mathcal{C} \in \mathcal{K}$. Thus, to get the filtration of the cubical complex, we follow the steps in [1] to compute the *lower star filtration*, or the persistence cubical complex. The sub-level sets of f form cubical complexes, and the computation of persistent homology is based on the increasing sequence of these complexes or sub-level sets of f.

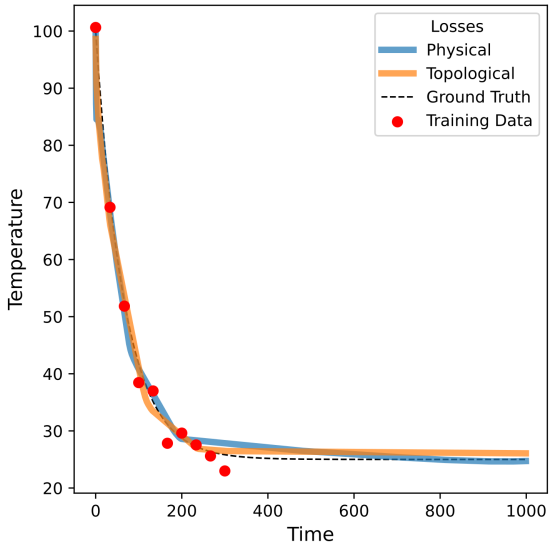
We can visualize how the creation of features is destroyed by computing the filtration through a persistence diagram. A persistence diagram describes when each k-Homology feature was born and died on the diagram's respective x and y axes. We can compare persistence diagrams through different distance metrics like the Wasserstein distance. For two persistence diagrams $D_1 = \{(b_i, d_i)\}_{i=1}^{N_{D_1}}$ and $D_2 = \{(b_i, d_i)\}_{i=1}^{N_{D_2}}$, we can compute the Wasserstein distance as

$$W_p[L_q](D_1, D_2) \coloneqq \inf_{\gamma : D_1 \to D_2} \left(\sum_{x \in D_1} \|x - \gamma(x)\|_q \right)^{1/p}$$

where $1 \leq p, q \leq \infty$ and γ ranges over all bijections between D_1 and D_2. Figure 1 shows the process of obtaining sub-level set persistence to calculate the previously described Wasserstein distance.



(a)



(b)

Fig. 2: The logarithm of the physical and topological loss and extrapolated prediction beyond times are given. The topological model can predict within reason compared to the physical model. (2a) Log loss over many epochs compared using the different types of losses. Notice that the topological loss continues to decrease, and the training could still be beyond the current number of trained epochs. (2b) Comparison of predictions for the cooling law for the different losses are shown.

III. Methodology

A simple, shallow, fully connected network implements Mean Squared Error as its criterion and Adam optimization with a learning rate of 0.007. Each hidden layer consists of 200 hidden units and does not apply any dropout—the only difference between the models when training is in the different additional loss implementations. As mentioned, the PINN garners additional physical loss structure from recovering the differential equation through numerical derivation. The topological loss is constructed by applying a cubical complex to the physical loss aspect of the PINN. Recall that a simple data-driven approach with physics-informed loss is built by reducing

$$\varepsilon_{\text{PINN}} = \frac{1}{N_u}\sum_{i=1}^{N_u}\left|u(x_u^i,t_u^i)-u^i\right|^2 + \frac{\omega}{N_f}\sum_{i=1}^{N_f}\left|f(x_f^i,t_f^i)\right|^2$$

In the case of TIPINN, we apply a cubical complex to $f(x_f^i,t_f^i)$ for each $i = 1,\ldots,N_f$ and the recovering of the partial differential equation through our approximation $u(x_u^i,t_u^i)$ denoted as $U(x_u^i,t_u^i) = \frac{\partial}{\partial t}u(x_u^i,t_u^i)$. We compute the persistence diagram of $U(x_u^i,t_u^i)$ and a persistence diagram of $f(x_f^i,t_f^i)$ to which we compare them using Wasserstein distance. Therefore, we measure the loss as a mixture of mean square error and the Wasserstein distance of the cubical complexes of the original and recovered differential equation. Formally, we can find the TIPINN loss as

$$\varepsilon_{\text{TIPINN}} = \frac{1}{N_u}\sum_{i=1}^{N_u}\left|u(x_u^i,t_u^i)-u^i\right|^2 + \frac{\omega}{P_f}\sum_{i=1}^{P_f} W_q(D_f^i, D_U^i)$$

where P_f is the number of persistence points from the cubical complex, D_f^i is the persistence diagram of $f(x_f^i,t_f^i)$, and D_U^i is the persistence diagram of the recovered differential equation as $U(x_u^i,t_u^i)$. In this case, we set $q = 1$ and $\omega = 1$. Notice that we compare the points in order of prediction rather than finding the optimal matching points to compute the smallest distance. The comparison in prediction order ensures that backpropagation utilizes loss from a similar structure to the ground truth differential equation rather than an optimization problem for minimizing the loss between the two diagrams. The comparison process is identical to the method shown in Figure 1, which discovers optimal matching between points.

IV. Results

In this section, we use the Cooling Law as our case study by comparing the model's training using the different losses and how the losses compare when an increasing noise level is introduced. By extrapolating the cooling law solution, we show how the TIPINN method works similarly, if not better than PINNs.

A. Cooling Law

The cooling law, or Newton's Law of Cooling, describes the rate of change of the temperature of an object as its temperature is affected by its surrounding environmental temperature. Let $T(t)$ be the temperature of the object at time t and T_{env} be the temperature of the surrounding environment. Note that the environmental temperature could change with respect to time, but in this case, it is left as a constant for all t. Thus, Newton's Law of Cooling states that the rate of change of the object's temperature is proportional to the difference in temperature

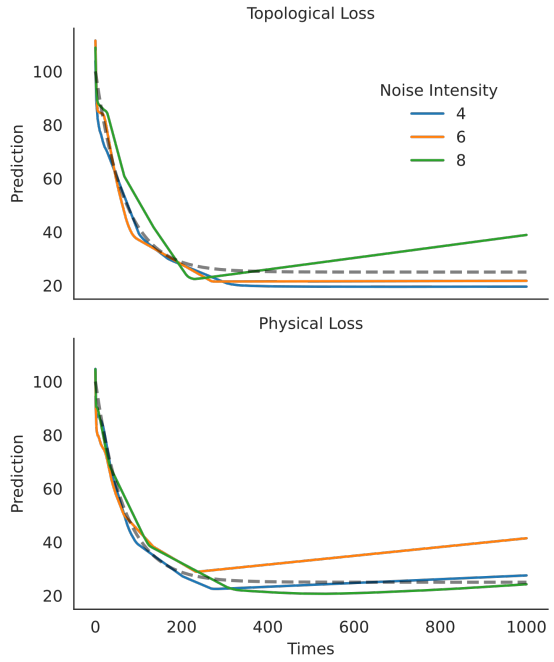


Fig. 3: prediction with varying noise intensity. The different colored curves represent the prediction from the training data at a larger noise intensity.

between the object and the surrounding environment. The law of cooling is stated mathematically as

$$\frac{dT}{dt} = -R(T - T_{\text{env}})$$

where R is the cooling constant. We can solve this differential equation to obtain the solution

$$T(t) = T_{\text{env}} + (T_0 - T_{\text{env}})e^{-Rt}$$

where T_0 is the initial temperature of the object.

B. Data Generation

For the training of the model, we consider the ground truth time to be the set $I_G = \{0, 1, \ldots, 1000\}$ seconds and the training times to be the interval $I_T = \{0, 10, 20, \ldots, 300\}$ seconds. We set the cooling constant to be $R = 0.015$, the environmental temperature as $T_{\text{env}} = 25$, and the initial temperature of the object as $T_0 = 100$. We can obtain a new differential equation and solution to generate data by these parameters. The ground truth differential equation is found by applying it to the set I_G, and the training data is obtained by using the differential equation to the set I_T with some scaled Gaussian noise. By default, we have set the noise intensity to be two as this would produce variability representative of accurate world testing but not too much to obfuscate the equation.

C. Initial Comparison

Both models being trained for 15000 epochs over the same data (the red dots in Figure 2b) are compared against each other in Figure 2. As the models are trained, we see that losses are relatively the same from Figure 2a, and the loss for the topological model could be optimized further. Training is stopped here as the physical loss starts to flatten, and we would like to avoid overfitting the model. This produces an interesting notion as the topological model could continue to train. Still, it is unnecessary because it produces comparable results to the physical model, shown in Figure 2b. Another interesting result from Figure 2b is that the Topological model does not produce a kink in the initial times compared to the physical model. This bend can be discovered using a PINN discovery model to find parameters for optimal convergence. Still, the topological model does not need parameter discovery to bypass this prediction bend.

D. Varying noise intensities

As stated, the training data is the ground truth differential equation with additional scaled Gaussian noise where the initial noise intensity is set to 2. We increment the noise intensity from 4 to 6 to 8 and train the model using each loss per each noise intensity. Naturally, we would expect as the noise intensity increases, the prediction should get worse and worse, which is what happens with the topological model in Figure 3. Interestingly, this is not the case for the physical model as the noise intensities 4 and 8 predictions are comparable—however, the prediction for the noise intensity six trails away from the ground truth. As expected, the more the noise intensity increased, the worse the topological predictions. This could be due to the training data size where instead of taking every 10th point from 0 to 300, we take every 5th to obtain more persistence points and thus a more accurate reflection of how the curve moves given time.

V. Conclusion and Future Work

The performance of TIPINN, when compared to a PINN, was similar to a basic differential model, even when additional noise was introduced to the training data. While TIPINN was developed for an ordinary differential equation, further research could be carried out to apply and compare this model to a single or system of partial differential equations, which includes boundary conditions and collocation points. To apply TIPINN to a partial differential equation, it may be necessary to use the cubical complex to recover both the boundary and differential solutions. This is a starting point; more testing may be required to understand how it should be applied. Additionally, the initial neural network size can be adjusted by adding or removing layers, changing activation functions, switching layer types from Linear to one-dimensional Convolutional layers, and changing the number of hidden units within each hidden layer. Instead of a Cubical complex, one could use an Alpha complex, which works well with structure data. But, this would change the minimization objective and would need to switch distance metrics using Bottleneck or a clustering type distance metric like the one described in [3].

REFERENCES

[1] H. Wagner, C. Chen, and E. Vuçini, "Efficient computation of persistent homology for cubical data," in *Topological methods in data analysis and visualization II: theory, algorithms, and applications*, Springer, 2011, pp. 91–106.

[2] J.-D. Boissonnat, F. Chazal, and M. Yvinec, *Geometric and topological inference*. Cambridge University Press, 2018, vol. 57.

[3] A. Marchese and V. Maroulas, "Signal classification with a point process distance on the space of persistence diagrams," *Advances in Data Analysis and Classification*, vol. 12, pp. 657–682, 2018.

[4] 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*, vol. 378, pp. 686–707, 2019.

[5] A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," *Communications in Computational Physics*, vol. 28, no. 5, 2020.

[6] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113 028, 2020.

[7] E. Carlsson and J. Carlsson, "A new construction for sublevel set persistence," *arXiv preprint arXiv:2106.04020*, 2021.

[8] A. 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*, vol. 34, pp. 26 548–26 560, 2021.

[9] L. Yang, X. Meng, and G. E. Karniadakis, "B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data," *Journal of Computational Physics*, vol. 425, p. 109 913, 2021.

[10] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics–informed neural networks: Where we are and what's next," *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, 2022.

[11] S. Wang, X. Yu, and P. Perdikaris, "When and why pinns fail to train: A neural tangent kernel perspective," *Journal of Computational Physics*, vol. 449, p. 110 768, 2022.