Structure-Preserving Physics-Informed Neural Networks With Energy or Lyapunov Structure

Haoyu Chu[1,2,3] , **Yuto Miyatake**[4] , **Wenjun Cui**[5] , **Shikui Wei**[*1,3] and **Daisuke Furihata**[4]
[1]Institute of Information Science, Beijing Jiaotong University
[2]Graduate School of Information Science and Technology, Osaka University
[3]Beijing Key Laboratory of Advanced Information Science and Network Technology
[4]Cybermedia Center, Osaka University
[5]School of Computer and Information Technology, Beijing Jiaotong University
{19112001, 19112048, shkwei}@bjtu.edu.cn, {miyatake, furihata}@cas.cmc.osaka-u.ac.jp

Abstract

Recently, there has been growing interest in using physics-informed neural networks (PINNs) to solve differential equations. However, the preservation of structure, such as energy and stability, in a suitable manner has yet to be established. This limitation could be a potential reason why the learning process for PINNs is not always efficient and the numerical results may suggest nonphysical behavior. Besides, there is little research on their applications on downstream tasks. To address these issues, we propose structure-preserving PINNs to improve their performance and broaden their applications for downstream tasks. Firstly, by leveraging prior knowledge about the physical system, a structure-preserving loss function is designed to assist the PINN in learning the underlying structure. Secondly, a framework that utilizes structure-preserving PINN for robust image recognition is proposed. Here, preserving the Lyapunov structure of the underlying system ensures the stability of the system. Experimental results demonstrate that the proposed method improves the numerical accuracy of PINNs for partial differential equations (PDEs). Furthermore, the robustness of the model against adversarial perturbations in image data is enhanced.

1 Introduction

Physics-informed neural networks (PINNs) [Raissi *et al.*, 2019] aim to embed seamlessly domain knowledge from physical laws into neural networks. Their ability to employ the power of physical principles while leveraging a data-driven approach has shown great potential for addressing complicated tasks in physics, engineering, and beyond [Karniadakis *et al.*, 2021; Lu *et al.*, 2021a; Cuomo *et al.*, 2022]. PINNs offer several advantages over conventional numerical methods, such as significantly reducing the computational cost, being mesh-free, and being capable of solving both forward and inverse problems in a unified framework.

Despite their empirical success, PINNs often suffer from decreased accuracy when dealing with strongly nonlinear and higher-order differential equations [Mattey and Ghosh, 2022]. We consider one potential reason is that PINNs overlook the underlying geometric structure of the physical system, leading to numerical solutions that may suggest nonphysical behavior. Most numerical methods have the same issue and therefore fail to preserve the conservative or dissipative property of the dynamical systems. To overcome this limitation, the development of structure-preserving numerical methods has gained momentum [Hairer *et al.*, 2006; Sharma *et al.*, 2020]. A structure-preserving algorithm can guarantee that qualitative characteristics, such as invariant or energy dissipation, are represented in the simulation, providing accurate numerical results over long periods. Thus, it is essential to investigate the application of the structure-preserving idea in PINNs. Nevertheless, most structure-preserving schemes are inapplicable to conventional neural networks because they need an intricate manual derivation of the system equation [Matsubara *et al.*, 2020]. Thanks to PINNs combining differential equations and neural networks, which provides fundamental support for us to introduce the idea of structure-preserving methods into PINNs.

In addition, the current applications of PINNs are mainly focused on solving differential equations. There is little research on their applications on downstream tasks, such as image recognition tasks. Giga *et al.* [2013] showed the potential power of leveraging the heat equation for binary classification. Another family of differential equations based deep neural networks, namely neural ordinary differential equations (Neural ODEs) [Chen *et al.*, 2018], shows promising results on downstream tasks, which describes the continuous dynamics of hidden units utilizing an ODE parameterized by a neural network. The advantage of utilizing PINNs is that one can directly obtain the solutions through the forward inference of the neural network, avoiding the approximation of the numerical integration in Neural ODEs. Besides, we hypothesize that there exists a structure of stability (in the Lyapunov sense) in image recognition tasks, i.e., minor perturbations on the input image will not influence the classification result. We consider the failure of current neural networks on adversarial samples [Szegedy *et al.*, 2013; Zhang and Li, 2019], crafted by adding

[*] Corresponding author.

minor human-imperceptible perturbations to images, is due to the neglect of preserving the stability structure. Thus, introducing structure-preserving PINNs for image recognition is expected to enable the model to resist initial perturbations. Considering the threat adversarial examples pose to the security of deep learning systems [Eykholt *et al.*, 2018; Lu *et al.*, 2021b], it is meaningful to explore the application of structure-preserving PINNs to robust image recognition.

To address these problems, we propose a new family of PINNs, named structure-preserving PINNs (SP-PINNs), that can be applied to dynamical systems with energy or Lyapunov structure. Our approach is capable of solving PDEs (e.g., the Allen–Cahn equation) and handling downstream tasks (e.g., image recognition). Our main contributions are summarized as follows:

- Based on prior knowledge about the physical system, we introduce a structure-preserving loss function to assist PINN in learning the structure of the underlying system, improving its performance on numerical simulation.
- We propose an SP-PINN for robust image recognition. Here, we treat the input image after downsampling as the initial value of the differential equation and assume a general form of the unknown underlying dynamical system. In this scenario, preserving the Lyapunov structure of the underlying system ensures the stability of the system.
- Experiments on standard benchmarks show that SP-PINN can consistently outperform the baseline model in terms of robustness against adversarial attacks, which supports our hypothesis on the structure of stability regarding image recognition tasks.
- We show that combining the SP-PINN with adversarial training methods further enhances the robustness against adversarial examples.

2 Related Works

2.1 Neural Networks for Differential Equations

The first attempts at utilizing neural networks to solve differential equations began in the 1990s [Lee and Kang, 1990]. Recent progress in automatic differentiation techniques has enabled researchers to design more complicated neural network architectures.

Among different data-driven techniques for solving differential equations, PINNs [Raissi *et al.*, 2019] have shown remarkable promise and versatility. Mattey *et al.* [2022] designed a novel PINN scheme that re-trains the same neural network for solving the PDE over successive time segments while satisfying the already obtained solution for all previous time segments. Krishnapriyan *et al.* [2021] analyzed that possible failure modes in PINNs are attributed to the PINNs' setup making the loss landscape very hard to optimize. Jagtap and Karniadakis [2021] extended PINN by leveraging the generalized space-time domain decomposition for solving arbitrary complex geometry domains. Wang *et al.* [2024] introduced a neural architecture search-guided PINN. Zhao *et al.* [2024] proposed a Transformer-based PINN to capture the crucial temporal dependencies inherent in practical physics systems. Lau *et al.* [2024] presented a algorithm for training PINN that optimally selects collocation and experimental points by jointly considering their interactions and dynamically adjusting proportions during training.

2.2 Structure-Preserving Deep Learning

Greydanus *et al.* [2019] parameterized the Hamiltonian mechanics with a neural network and then learned it via a data-driven approach, which conserved an energy-like quantity. Sosanya *et al.* [2022] proposed a dissipative Hamiltonian neural network that leverages the tools of Hamiltonian mechanics and Helmholtz decomposition to separate conserved quantities from dissipative quantities. Lutter *et al.* [2019] proposed a neural network for learning the mechanic systems of the Euler-Lagrange equation employing end-to-end training while keeping physical plausibility. [Matsubara and Yaguchi, 2023] utilized Neural ODEs for finding and preserving invariant quantities by leveraging the projection method and the discrete gradient method [Matsubara *et al.*, 2020]. Jagtap *et al.* [2020] presented a conservative PINN on discrete sub-domains by using a separate PINN in each sub-domain, and then stitching back all sub-domains through the corresponding conservative quantity, which is different from our approach that is based on the prior knowledge about the underlying dynamics and is theoretically applicable both conservative and dissipative systems.

2.3 Adversarial Defense Methods

Since realizing the instability of deep neural networks, researchers have proposed different kinds of complementary techniques to defend adversarial examples, such as distillation defense [Papernot *et al.*, 2016; Chu *et al.*, 2023] and adversarial training. Maday *et al.* [2017] proposed an adversarial training method by injecting adversarial examples generated by PGD attacks into training data. Ilyas *et al.* [2019] created a robust dataset for adversarial training by removing non-robust features from the dataset. Lamb *et al.* [2022] augmented the adversarial training with interpolation-based training, which aims to tackle the problem that traditional adversarial training aggravates the generalization performance of the networks on clean data.

After knowing the connection between dynamical systems and deep neural networks, designing defense methods based on the Lyapunov stability theory becomes a new trend. Kang *et al.* [2021] proposed stable Neural ODE based on Lyapunov's first method for resisting adversarial examples. Chu *et al.* [2024] leveraged Lyapunov's second method for preventing successful adversarial attacks by inherently constraining deep equilibrium models (DEQ) [Bai *et al.*, 2019] to be stable.

It is worth mentioning that our proposed method is the first attempt at utilizing modified PINNs for adversarial defense.

3 Methodology

This section presents the applications of the proposed SP-PINNs to PDEs (e.g., the Allen–Cahn equation), and robust image recognition.

The strategies are summarized as follows: firstly, a neural network is specified for each of the different tasks to fit the mapping from the input to the numerical solution, and then the derivatives w.r.t. the input is calculated by automatic differentiation; secondly, the core of the proposed method involves incorporating prior knowledge about the energy or Lyapunov structure of the system into the specific neural network; thirdly, the neural network is trained by minimizing the loss function. For different scenarios, we design a corresponding training process and loss function to cater to the unique characteristics of each problem domain.

3.1 SP-PINN for PDEs

We consider the Allen–Cahn equation [Allen and Cahn, 1972], which is a strongly nonlinear reaction-diffusion equation. The Allen–Cahn equation is widely used for modeling some phase separation and domain coarsening phenomena. Currently, PINN's accuracy suffers significantly from strongly nonlinear PDEs [Mattey and Ghosh, 2022]. Therefore, we aim to improve the solutions' accuracy by learning the underlying structure of the equation. The Allen–Cahn equation is formulated as:

$$\frac{\partial u}{\partial t} = pu + ru^3 + q\frac{\partial^2 u}{\partial x^2}, \tag{1}$$

where $p > 0, q > 0, r < 0$, $x \in [0, L]$ and $t \in [0, T]$. The initial condition is $u(x, 0) = u_0(x)$. Here, we employ the Neumann boundary condition $u_x(0, t) = u_x(L, t)$.

To reveal the energy dissipative property of the Allen–Cahn equation, we first introduce a quality, which is called the 'free energy' or 'local energy' of the problem:

$$G(u, u_x) = -\frac{p}{2}u^2 - \frac{r}{4}u^4 + \frac{q}{2}u_x^2. \tag{2}$$

The evolution of the solution is shown to evolve in a direction that the global energy is decreased [Furihata and Matsuo, 2010]:

$$\frac{d}{dt}J(u) = \frac{d}{dt}\int_0^L G(u, u_x)dx \leq 0, \tag{3}$$

where $J(u)$ is the 'global energy', which is a functional of u. Meanwhile, $J(u)$ can be regarded as a function of t.

To solve the Allen–Cahn equation, the structure-preserving PINN is defined as a N-layer fully connected network (FCN) with inputs x, t and output $\hat{u}(x, t)$. The framework is illustrated in Figure 1.

We obtain the derivatives of the output $\hat{u}(x, t)$ w.r.t. position x and time t via automatic differentiation and randomly select N_f in the spatiotemporal region (x, t) to calculate the equation residual:

$$L_{\text{eqn}} = \frac{1}{N_f}\sum_{i=1}^{N_f} \| \frac{\partial \hat{u}_i}{\partial t} - p\hat{u}_i - r\hat{u}_i^3 - q\frac{\partial^2 \hat{u}_i}{\partial x^2} \|_2. \tag{4}$$

Then, we randomly select N_i points in the region $(x, 0)$ to calculate the initial condition residual:

$$L_{\text{ini}} = \frac{1}{N_i}\sum_{i=1}^{N_i} \|\hat{u}_i - u_i\|_2. \tag{5}$$

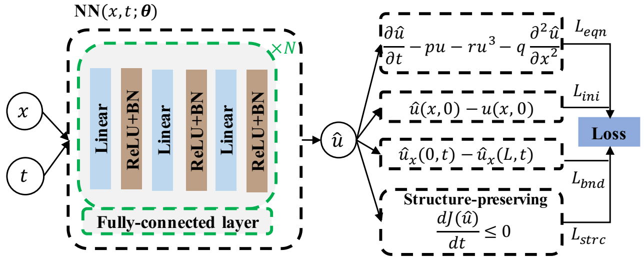
NN$(x, t; \boldsymbol{\theta})$
x
t
Linear
ReLU+BN
Linear
ReLU+BN
Linear
ReLU+BN
$\times N$
Fully-connected layer
$\hat{u}$
$\frac{\partial \hat{u}}{\partial t} - pu - ru^3 - q\frac{\partial^2 \hat{u}}{\partial x^2}$
$\hat{u}(x,0) - u(x,0)$
$\hat{u}_x(0,t) - \hat{u}_x(L,t)$
Structure-preserving
$\frac{dJ(\hat{u})}{dt} \leq 0$
L_{eqn}
L_{ini}
L_{bnd}
L_{strc}
Loss

Figure 1: SP-PINN for Allen–Cahn equation. The SP-PINN is defined as a N-layered FCN with inputs x, t and output $\hat{u}(x, t)$.

Next, we randomly select N_b points in the regions $(0, t)$ and (L, t) to calculate the boundary condition residual:

$$L_{\text{bnd}} = \frac{1}{N_b}\sum_{i=1}^{N_b} \|\hat{u}_x^i(0) - \hat{u}_x^i(L)\|_2. \tag{6}$$

Finally, to specify the structural loss, we define the discrete global energy accordingly by:

$$J(\hat{u}) \triangleq \sum_{k=0}^{M}{}'' G_k(\hat{u}, \hat{u}_x)\Delta x, \tag{7}$$

where $\Delta x = L/M$, M is the number of the spatial grid points, $\sum_{k=0}^{M}{}'' f$ denotes the trapezoidal rule:

$$\sum_{k=0}^{M}{}'' f \triangleq \frac{1}{2}f_0 + f_1 + \cdots + f_{M-1} + \frac{1}{2}f_M. \tag{8}$$

We uniformly select N_e points in the time interval $[0, T]$ to calculate the structural loss:

$$L_{\text{strc}} = \frac{1}{N_e}\sum_{i=1}^{N_e} \| \operatorname{ReLU}(\frac{d}{dt}J(\hat{u}_i))\|_2, \tag{9}$$

where $\operatorname{ReLU} = \max(0, x)$ is the rectified linear unit function.

Therefore, the total loss function for training the proposed model is the summation of the equation residual, the initial condition residual, the boundary condition residual, and the structural loss:

$$L_{\text{total}} = \lambda_1 L_{\text{eqn}} + \lambda_2 L_{\text{bnd}} + \lambda_3 L_{\text{ini}} + \lambda_4 L_{\text{strc}}, \tag{10}$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are hyperparameters.

3.2 SP-PINN for Robust Image Recognition

Currently, PINNs are mostly utilized for solving differential equations. In this section, we delve into the extension of PINNs to downstream tasks and propose an SP-PINN for robust image recognition. First, we define the image recognition task as an initial value problem, where the input image after downsampling is treated as the initial value, and the approximate solution is obtained through the evolution of time. Secondly, we project the learned dynamical system into a space that satisfies the Lyapunov stability condition to preserve the stability structure.

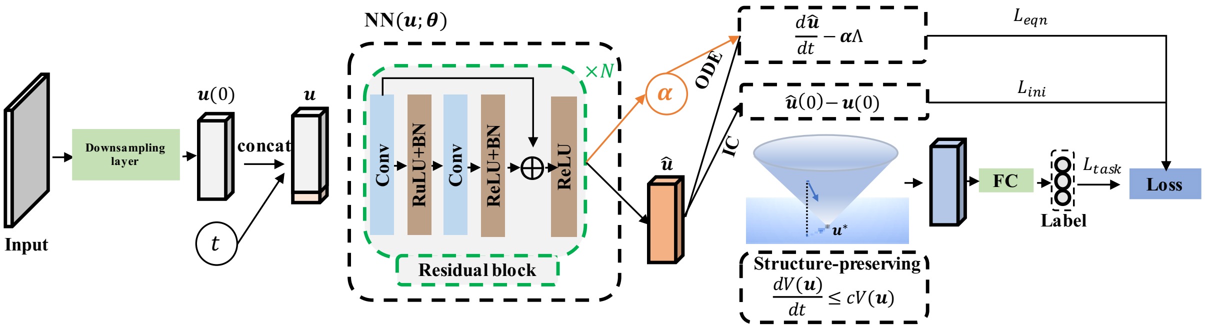

Figure 2: SP-PINN for robust image recognition. The SP-PINN is defined as a N-layer residual network. The orange line denotes the process of solving the inverse problem, in which a data-driven way is resorted to obtain the unknown parameter $\boldsymbol{\alpha}$ of the underlying system. For solving the forward problem, the output is the approximate solution $\hat{\boldsymbol{u}}$. The blue arrow represents a state that satisfies the Lyapunov exponential stability condition. The FC layer predicts the category of the image. When calculating the loss regarding the initial condition, the time t is set to 0. When performing classification, the time t is set to 1.

We hypothesize that there is a structure of stability in the Lyapunov sense in image recognition tasks, that is, small changes put on the input image will not influence the classification result. Thus, introducing the structure-preserving idea into the image recognition tasks is expected to enable the classification model to have resistance against the minor perturbations added on the input image, making the model more robust.

However, the challenge of utilizing PINN for image recognition is that the underlying dynamical system is unknown. Therefore, we specify the following ODE to describe the system:

$$\frac{d\boldsymbol{u}(t)}{dt} = f(\boldsymbol{u}(t), t; \boldsymbol{\theta}), \tag{11}$$

where the initial value is $\boldsymbol{u}(0) = \boldsymbol{u}_0$ and the time interval is $[0, T]$.

It is worth mentioning that Neural ODEs [Chen *et al.*, 2018] also use Eqn. (11) to describe the continuous dynamics of hidden units, which show promising results on downstream tasks. Our proposed method diverges from Neural ODEs in terms of the approach to solving the initial value problem.

Theoretically, the analytic solution can be obtained by integrating Eqn. (11):

$$\boldsymbol{u}(T) = \boldsymbol{u}(0) + \int_0^T f(\boldsymbol{u}(t), t)\, dt. \tag{12}$$

In practice, Neural ODEs utilize the ODE solver, such as Runge–Kutta, Dopri5, to approximate the numerical integration $\int_0^T f(\boldsymbol{u}(t), t)\, dt$. This procedure is formulated by:

$$\boldsymbol{u}(T) = \text{ODESolver}((\boldsymbol{u}(0), f, [0, T], \boldsymbol{\theta})). \tag{13}$$

Unlike Neural ODEs, the advantage of utilizing PINN is that PINN can obtain the numerical solution at time T directly through the forward inference of the neural network, that is, $\boldsymbol{u}(T) = \text{PINN}((\boldsymbol{u}(0), T, \boldsymbol{\theta}))$, avoiding the approximation of the numerical integration. Nevertheless, this also poses a problem in that the form of Eqn. (11) needs to be known in advance. To address this problem, we assume a general form of the underlying dynamical system following Hu *et al.* [2022]:

$$\begin{aligned} \frac{d\boldsymbol{u}(t)}{dt} &= \boldsymbol{\alpha}\boldsymbol{\Lambda}, \\ \boldsymbol{u}(0) &= (u_1(0), u_2(0), \cdots, u_n(0))^T, \end{aligned} \tag{14}$$

where $\boldsymbol{\Lambda} = \{1, u_1, u_2, \cdots, u_n, u_1u_1, u_1u_2, \cdots, u_n^p\}$ is the complete set of p^{th}-order polynomial basis with unknown coefficients $\boldsymbol{\alpha} = (\alpha_{ij})_{n\times M}$, $M = \binom{p+n}{p}$. Here, we take the downsampling result of the input image as the initial value $\boldsymbol{u}(0)$. Like Neural ODEs, the final time for PINN is set to $T = 1$.

To obtain the unknown parameters $\boldsymbol{\alpha}$, we need to first solve the inverse problem. As long as we know the underlying dynamical system, we can calculate the numerical solution via the forward inference, i.e., solving the forward problem. Motivated by Raissi *et al.* [2019] and Kim *et al.* [2023], the learning process of the proposed model is achieved by solving the inverse and forward problems alternately.

The SP-PINN for image recognition is defined as a N-layer residual network, where the input $\boldsymbol{u}$ is a vector concatenated by initial value $\boldsymbol{u}(0)$ with time t (after broadcasting). The scratch of the architecture of the proposed method is shown in Figure 2. The details of the training process are as follows.

How to solve the inverse problem? We solve the inverse problem through a data-driven approach. After initializing the network parameters, we can input the images of the training set and get their corresponding outputs. Then, we minimize the following loss function regarding equation residual to find the $\boldsymbol{\alpha}$:

$$\arg\min_{\boldsymbol{\alpha}} \frac{1}{N_t} \sum_{i=1}^{N_t} \| \frac{d\hat{\boldsymbol{u}}_i}{dt} - \boldsymbol{\alpha}\boldsymbol{\Lambda} \|, \tag{15}$$

where N_t is the number of the training set.

How to solve the forward problem? As long as the underlying system is known, we can calculate the approximate solutions through the forward inference. The new solutions are in turn utilized for solving the inverse problem to update

$\boldsymbol{\alpha}$. Repeated iterations of this process yield more accurate $\boldsymbol{\alpha}$ and $\hat{\boldsymbol{u}}$.

While solving the inverse problem, we fix the trainable parameters $\boldsymbol{\theta}$ of the neural network and learn the unknown parameters $\boldsymbol{\alpha}$ of the underlying dynamical system. While solving the forward problem, we fix the parameters $\boldsymbol{\alpha}$ of the dynamical system and train the parameters $\boldsymbol{\theta}$ of the neural network, in this case, the output of the model is the approximate solution $\hat{\boldsymbol{u}}$.

How to preserve the stability structure? To preserve the stability structure, that is, ensuring the proposed model is robust to minor perturbations on the initial value, we jointly learn a convex positive definite Lyapunov function along with dynamics constrained to be stable and project the learned dynamical system of PINN onto a space where the Lyapunov exponential stability condition holds. Please refer to the [Chu *et al.*, 2024] for the definition of Lyapunov stability and the Lyapunov stability theorem.

We construct the structure-preserving module motivated by Manek and Kolter [2019]. Let $F(\boldsymbol{u}) = d\,\mathrm{NN}(\boldsymbol{u})/dt$ be a basic dynamic model, let $V : \mathbb{R}^n \to \mathbb{R}$ be a positive definite function, and c be a nonnegative constant, the exponentially stable dynamical model is defined as

$$\begin{aligned}\widetilde{F}(\boldsymbol{u}) &= \mathrm{Projection}(F(\boldsymbol{u}), \{F : \nabla V(\boldsymbol{u})^T F \leq -cV(\boldsymbol{u})\}) \\ &= \begin{cases} F(\boldsymbol{u}) & \text{if } \phi(\boldsymbol{u}) \leq 0 \\ F(\boldsymbol{u}) - \nabla V(\boldsymbol{u})\frac{\phi(\boldsymbol{u})}{\|\nabla V(\boldsymbol{u})\|_2^2} & \text{otherwise} \end{cases},\end{aligned} \tag{16}$$

where, $\phi(\boldsymbol{u}) = \nabla V(\boldsymbol{u})^T F(\boldsymbol{u}) + cV(\boldsymbol{u})$. For $\phi(\boldsymbol{u}) > 0$, the output of the base dynamics model is projected onto a halfspace where this condition holds, otherwise, the output is returned unchanged.

The Lyapunov function V is defined as positive definite and continuously differentiable, and has no local minima:

$$V(\boldsymbol{u}) = \sigma(g(\boldsymbol{u}) - g(0)) + \|\boldsymbol{u}\|_2^2, \tag{17}$$

where σ is a positive convex non-decreasing function with $\sigma_k(0) = 0$, and g is represent as a input-convex neural network (ICNN) [Amos *et al.*, 2017]. Since the Lyapunov function is defined as continuously differentiable, we can use automatic differentiation to compute its gradient. This advantageous feature allows us to train our final network in a manner similar to any other network.

Consequently, we ensure that the proposed model satisfies the Lyapunov exponentially stable condition through the procedures described above, thereby preserving the Lyapunov structure of the underlying dynamical system.

The FC layer, that is, a weighted linear transformation, takes part in predicting the category of the image.

The loss function for training the SP-PINN is obtained as follows. First, obtain the derivative of the network's output $\hat{\boldsymbol{u}}$ w.r.t. time t. Second, calculate the equation residual:

$$L_{\text{eqn}} = \frac{1}{N_t}\sum_{i=1}^{N_t} \|\frac{d\hat{\boldsymbol{u}}_i}{dt} - \boldsymbol{\alpha\Lambda}\|. \tag{18}$$

As can be seen, the equation residual in the forward problem is the same form as in the inverse problem, i.e., Eqn. (15). Hence, PINNs can deal with forward and inverse problems in a unified paradigm, which offers a great advantage compared to traditional numerical methods [Chirigati, 2021] that need to design different schemes for different inverse problems.

Then, we calculate the initial condition residual:

$$L_{\text{ini}} = \frac{1}{N_h}\sum_{i=1}^{N_h} \|\hat{\boldsymbol{u}}_i(0) - \boldsymbol{u}_i(0)\|_2, \tag{19}$$

where N_h is the number of elements in $\boldsymbol{u}_i(0)$.

Finally, we utilize the cross-entropy (CE) loss to measure the difference between the FC layer's result $\boldsymbol{y}$ and the true label $\hat{\boldsymbol{y}}$ of the image:

$$L_{\text{task}} = \frac{1}{N_t}\sum_{i=1}^{N_t} \mathrm{CE}(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i), \tag{20}$$

The loss function used to update the network parameters $\boldsymbol{\theta}$ is the summation of the equation residual, the initial condition residual, and the cross-entropy loss. Therefore, the best parameters $\boldsymbol{\theta}$ is obtained by minimizing the loss function:

$$\arg\min_{\boldsymbol{\theta}} \ \lambda_1 L_{\text{eqn}} + \lambda_2 L_{\text{ini}} + \lambda_3 L_{\text{task}}, \tag{21}$$

where $\lambda_1, \lambda_2, \lambda_3$ are hyperparameters.

4 Experiments

In this section, we first present the experimental setup. Then, we evaluate the proposed method on two different scenarios and analyze the experimental results.

4.1 Experimental Setup

We use PyTorch [Paszke *et al.*, 2017] framework for the implementation. The torch version is 1.11.0+cu113. We conducted our experiments on the Ubuntu 20.04.6 LTS operator system. All the experiments are run on a single NVIDIA A100 40GB GPU. In our experiments, we set all the hyperparameters λ_i as 1.

4.2 Experiments on the Allen–Cahn Equation

Regarding the training configurations, we first run the Adam algorithm [Kingma and Ba, 2014] with 10,000 epochs and then employ the L-BFGS algorithm [Liu and Nocedal, 1989].

We show a numerical example in Figure 3 with $p = 1, r = -1, q = 0.0001$ and $x \in [0, 2\pi]$, $t \in [0, 4]$. The initial condition is taken to $u_0(x) = 0.25sin(x)$ and we employ the Neumann boundary condition $u_x(0, t) = u_x(2\pi, t)$. In this experiment, the N_f, N_b, N_i, N_e is set to 8,000, 1,000, 1,000, 2000. Δx is $L/M = 2\pi/2000 \approx 0.00314$. We employ an FCN with 6 hidden layers.

We compare the proposed method with the discrete variational derivative method (DVDM) [Furihata and Matsuo, 2010]. DVDM is a structure-preserving numerical method for PDEs, which improves the qualitative behavior of the PDE solutions and allows for stable computing. Although SP-PINN is slightly inferior to DVDM in terms of accuracy, DVDM needs an intricate derivation to construct a specific numerical scheme. Besides, Figure 5 illustrates that SP-PINN obtains numerical solutions significantly faster than DVDM.

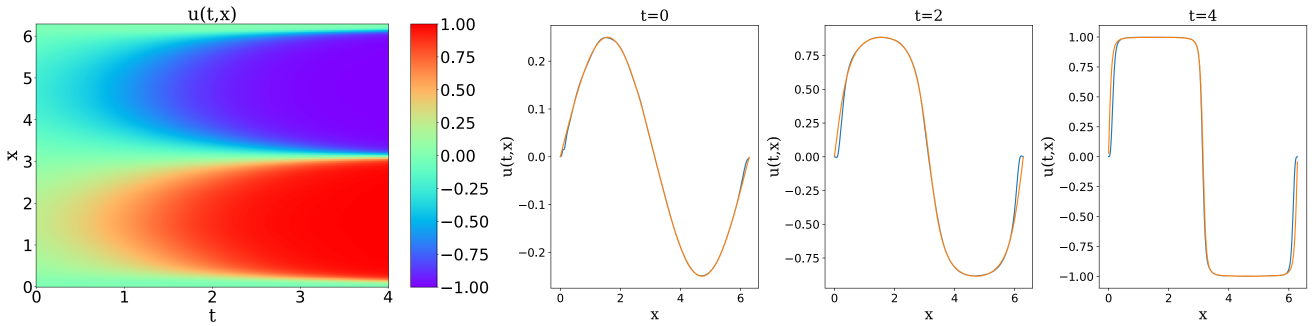

Figure 3: Numerical solutions of the Allen–Cahn equation. The orange line is obtained by DVDM. The blue line is obtained by our method.

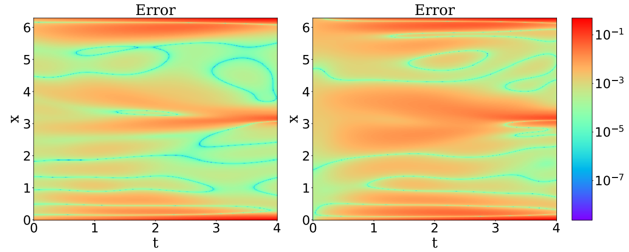

Figure 4: (left) The error between the results obtained by the proposed model and DVDM. (right) The error between the results obtained by the baseline model and DVDM.

Even accounting for the training time of the network, SP-PINN is less time-consuming than DVDM.

As shown in Figure 4, the error between the results obtained by the proposed model and DVDM is smaller than that of the baseline model (i.e., the vanilla PINN) and DVDM, which indicates that the numerical solutions of the proposed model are more accurate than that of the baseline model.

4.3 Experiments on Image Recognition

The Experimental Configurations

We conduct a set of experiments on four datasets, MNIST [LeCun *et al.*, 1998], Street View House Numbers (SVHN) [Yuval, 2011] and CIFAR10/100 [Krizhevsky *et al.*, 2009].

We choose a 2-layer ICNN and set the activation function σ as a smooth ReLU function. For optimization, we use the Adam algorithm with the initial learning rate $= 0.001$ and a cosine annealing schedule. The training epochs for MNIST, SVHN, and CIFAR10/100 are set to 10, 40, and 60/70.

For MNIST, we downsample the input image from 28×28 to 6×6. For SVHN and CIFAR, we downsample the input image from 32×32 to 16×16. We choose an 18-layer residual network [He *et al.*, 2016] as the backbone of the SP-PINN and utilize a Maclaurin series with 5^{th}-order polynomial basis for modeling the underlying systems.

We test the performance of the PINN and SP-PINN on two white-box adversarial attacks: iterative fast gradient sign method (I-FGSM) [Kurakin *et al.*, 2018] and project gradient descent (PGD) [Madry *et al.*, 2017].

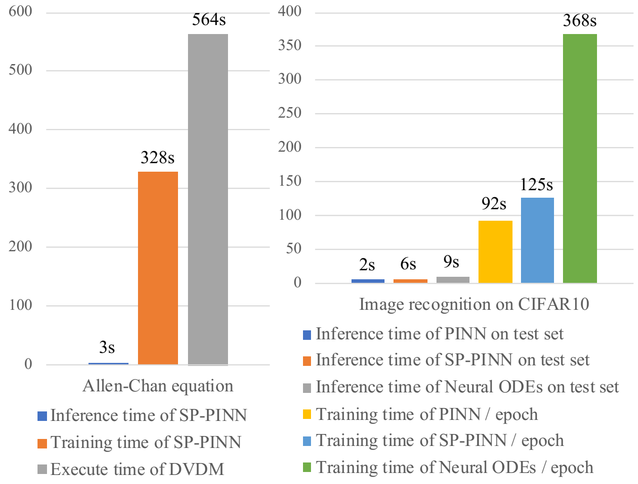

Figure 5: The comparison of the time consuming on same experimental settings. The ODE solver used in Neural ODEs is Dopri5.

The Evaluation of the SP-PINN on Image Data

Table 1 presents the experimental results in terms of classification accuracy and robustness against adversarial examples. On clean data, SP-PINN performs slightly inferior to the baseline model. Figure 5 illustrates that the PINN and SP-PINN are significantly less time-consuming than Neural ODEs. In terms of robustness against adversarial examples, we evaluate the effectiveness of SP-PINN against white-box attacks. The experimental results demonstrate that the SP-PINN outperforms the baseline model on all datasets.

Furthermore, we observe that the accuracy improvement under adversarial attacks increases as the attack radii increase for all datasets. For example, the SP-PINN under I-FGSM with attack radii $\epsilon = 2/255$, $\epsilon = 4/255$, $\epsilon = 6/255$ and $\epsilon = 8/255$ achieves a boost of 8.21%, 15.60%, 18.80%, 21.80% on CIFAR10, respectively.

SP-PINN With Adversarial Training

Our approach is independent of other adversarial defense methods, such as adversarial training. This means that we can combine SP-PINN with adversarial training techniques to further enhance defense performance. We consider three well-known adversarial training methods, namely PAT [Madry *et*

Benchmark	Model	Clean	Attack	$\epsilon = 2/255$	$\epsilon = 4/255$	$\epsilon = 6/255$	$\epsilon = 8/255$
MNIST	PINN (baseline)	99.40	I-FGSM	94.72	92.91	90.55	87.82
			PGD	94.88	92.99	90.81	87.89
	SP-PINN	99.38	I-FGSM	**98.77** (+4.05)	**98.75** (+5.84)	**98.67** (+8.12)	**98.49** (+10.67)
			PGD	**98.76** (+3.88)	**98.75** (+5.76)	**98.72** (+7.91)	**98.56** (+10.67)
SVHN	PINN (baseline)	93.80	I-FGSM	64.47	59.43	56.39	53.47
			PGD	61.67	56.80	54.00	50.71
	SP-PINN	92.69	I-FGSM	**66.54** (+2.07)	**65.44** (+6.01)	**64.26** (+7.87)	**62.74** (+9.27)
			PGD	**65.54** (+3.87)	**63.61** (+6.81)	**61.97** (+7.97)	**60.35** (+9.64)
CIFAR10	PINN (baseline)	88.03	I-FGSM	46.02	37.03	33.02	28.84
			PGD	42.72	35.02	31.14	27.31
	SP-PINN	87.46	I-FGSM	**51.76** (+5.74)	**51.66** (+14.63)	**51.56** (+18.54)	**51.13** (+22.29)
			PGD	**50.93** (+8.21)	**50.62** (+15.60)	**49.94** (+18.80)	**49.11** (+21.80)
CIFAR100	PINN (baseline)	64.47	I-FGSM	23.67	17.03	13.90	11.70
			PGD	21.00	15.51	12.96	10.89
	SP-PINN	62.32	I-FGSM	**26.41** (+2.74)	**26.39** (+9.36)	**26.27** (+12.37)	**26.16** (+14.46)
			PGD	**25.93** (+5.93)	**25.85** (+10.34)	**25.80** (+12.84)	**25.48** (+14.59)

Table 1: Classification accuracy (%) on MNIST, SVHN, and CIFAR. Results that surpass the baseline model are bold. The performance gain in parentheses is compared with the baseline model.

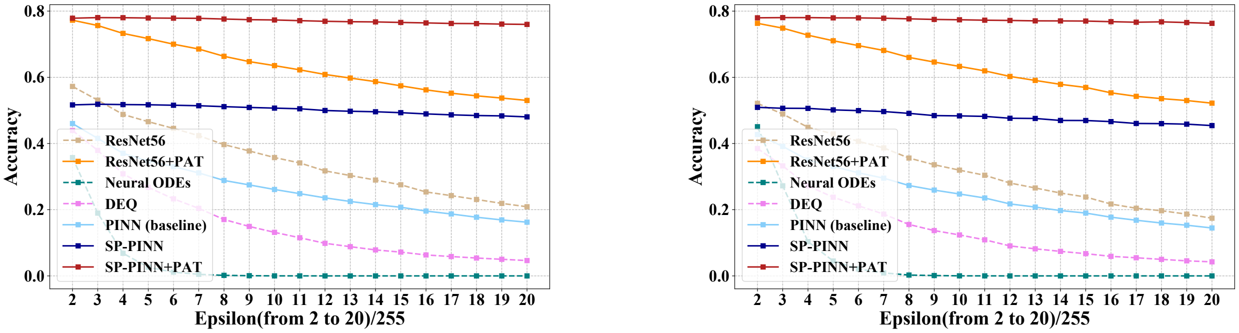

Figure 6: Comparison between different methods on CIFAR10 under I-FGSM attacks (left) and PGD attacks (right).

Radius	Attack	+PAT	+RD	+IAT
$\epsilon = 2/255$	I-FGSM	77.85	61.61	**80.97**
	PGD	78.04	61.58	**80.11**
$\epsilon = 4/255$	I-FGSM	78.00	61.57	**80.80**
	PGD	77.98	61.45	**79.96**
$\epsilon = 6/255$	I-FGSM	77.86	61.54	**80.63**
	PGD	77.95	61.42	**79.93**
$\epsilon = 8/255$	I-FGSM	77.60	61.46	**80.52**
	PGD	77.67	61.35	**79.85**

Table 2: Classification accuracy of the SP-PINN combined with adversarial training method on CIFAR10 under adversarial attacks. The second best result is with the underline.

al., 2017], robust dataset (RD) [Ilyas *et al.*, 2019], and interpolated adversarial training (IAT) [Lamb *et al.*, 2022].

From Table 2, we observe that training SP-PINN with adversarial training methods further improves its robustness against adversarial examples.

Figure 6 provides a comparison between ResNet56 [He *et al.*, 2016], ResNet56 with PAT, DEQ [Bai *et al.*, 2019], Neural ODEs [Chen *et al.*, 2018], PINN, SP-PINN, and SP-PINN with PAT under adversarial attacks ranging from $\epsilon = 2/255$ to $\epsilon = 20/255$. It is apparent that SP-PINN is insensitive to the radius of adversarial attack, which further corroborates the effectiveness of our method.

5 Conclusions

In this paper, we proposed SP-PINNs by introducing the prior knowledge about the properties of the underlying dynamical systems. The applicability of the proposed SP-PINNs ranges from PDEs, and to image recognition. In future work, we will consider employing the proposed method for solving more complex 'conservative' and 'dissipative' equations. Additionally, we consider applying SP-PINN to different natural language processing (NLP) tasks, given adversarial attacks have emerged in the NLP field. When varying the inputs by natural language, through concatenating the vector embeddings with time, SP-PINN can be expected to achieve character/word/sentence level adversarial defense.

Acknowledgments

This work was supported in part by the National Science and Technology Major Project (2018AAA0102100), the National Natural Science Foundation of China (No.62120106009), the China Scholarship Council (No.202207090082), JSPS KAKENHI (20H01822, 20H00581, 21K18301), and JST PRESTP (JPMJPR2129).

References

[Allen and Cahn, 1972] Samuel Miller Allen and John W Cahn. Ground state structures in ordered binary alloys with second neighbor interactions. *Acta Metallurgica*, 20(3):423–433, 1972.

[Amos *et al.*, 2017] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

[Bai *et al.*, 2019] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

[Chen *et al.*, 2018] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[Chirigati, 2021] Fernando Chirigati. Inverse problem via a bayesian approach. *Nature Computational Science*, 1(5):304–304, 2021.

[Chu *et al.*, 2023] Haoyu Chu, Shikui Wei, Qiming Lu, and Yao Zhao. Improving neural ordinary differential equations via knowledge distillation. *IET Computer Vision*, pages 1–11, 2023.

[Chu *et al.*, 2024] Haoyu Chu, Shikui Wei, Ting Liu, Yao Zhao, and Yuto Miyatake. Lyapunov-stable deep equilibrium models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11615–11623, 2024.

[Cuomo *et al.*, 2022] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[Eykholt *et al.*, 2018] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

[Furihata and Matsuo, 2010] Daisuke Furihata and Takayasu Matsuo. *Discrete variational derivative method: a structure-preserving numerical method for partial differential equations*. CRC Press, 2010.

[Giga *et al.*, 2013] Mi-Ho Giga, Yoshikazu Giga, Takeshi Ohtsuka, and Noriaki Umeda. On behavior of signs for the heat equation and a diffusion method for data separation. *Communications on Pure & Applied Analysis*, 12(5), 2013.

[Greydanus *et al.*, 2019] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.

[Hairer *et al.*, 2006] Ernst Hairer, Christian Lubich, and Gerhard Wanner. Structure-preserving algorithms for ordinary differential equations. *Geometric numerical integration*, 31, 2006.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Hu *et al.*, 2022] Pipi Hu, Wuyue Yang, Yi Zhu, and Liu Hong. Revealing hidden dynamics from time-series data by odenet. *Journal of Computational Physics*, 461:111203, 2022.

[Ilyas *et al.*, 2019] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

[Jagtap and Karniadakis, 2021] Ameya D Jagtap and George E Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. In *AAAI spring symposium: MLPS*, volume 10, 2021.

[Jagtap *et al.*, 2020] Ameya D Jagtap, Ehsan Kharazmi, and George Em 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*, 365:113028, 2020.

[Kang *et al.*, 2021] Qiyu Kang, Yang Song, Qinxu Ding, and Wee Peng Tay. Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. *Advances in Neural Information Processing Systems*, 34:14925–14937, 2021.

[Karniadakis *et al.*, 2021] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[Kim *et al.*, 2023] Jungeun Kim, Seunghyun Hwang, Jeehyun Hwang, Kookjin Lee, Dongeun Lee, and Noseong Park. Partial differential equation-regularized neural networks: An application to image classification, 2023.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kolter and Manek, 2019] J Zico Kolter and Gaurav Manek. Learning stable deep dynamics models. *Advances in neural information processing systems*, 32, 2019.

[Krishnapriyan *et al.*, 2021] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[Kurakin *et al.*, 2018] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[Lamb *et al.*, 2022] Alex Lamb, Vikas Verma, Kenji Kawaguchi, Alexander Matyasko, Savya Khosla, Juho Kannala, and Yoshua Bengio. Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy. *Neural Networks*, 154:218–233, 2022.

[Lau *et al.*, 2024] Gregory Kang Ruey Lau, Apivich Hemachandra, See-Kiong Ng, and Bryan Kian Hsiang Low. PINNACLE: PINN adaptive collocation and experimental points selection. In *The Twelfth International Conference on Learning Representations*, 2024.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lee and Kang, 1990] Hyuk Lee and In Seok Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110–131, 1990.

[Liu and Nocedal, 1989] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[Lu *et al.*, 2021a] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[Lu *et al.*, 2021b] Qiming Lu, Shikui Wei, Haoyu Chu, and Yao Zhao. Towards transferable 3d adversarial attack. In *ACM Multimedia Asia*, pages 1–5. 2021.

[Lutter *et al.*, 2019] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *7th International Conference on Learning Representations (ICLR)*. ICLR, May 2019.

[Madry *et al.*, 2017] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[Matsubara and Yaguchi, 2023] Takashi Matsubara and Takaharu Yaguchi. FINDE: Neural differential equations for finding and preserving invariant quantities. In *The Eleventh International Conference on Learning Representations*, 2023.

[Matsubara *et al.*, 2020] Takashi Matsubara, Ai Ishikawa, and Takaharu Yaguchi. Deep energy-based modeling of discrete-time physics. *Advances in Neural Information Processing Systems*, 33:13100–13111, 2020.

[Mattey and Ghosh, 2022] Revanth Mattey and Susanta Ghosh. A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390:114474, 2022.

[Papernot *et al.*, 2016] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[Raissi *et al.*, 2019] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framewojagtapjagtaprk for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[Sharma *et al.*, 2020] Harsh Sharma, Mayuresh Patil, and Craig Woolsey. A review of structure-preserving numerical methods for engineering applications. *Computer Methods in Applied Mechanics and Engineering*, 366:113067, 2020.

[Sosanya and Greydanus, 2022] Andrew Sosanya and Sam Greydanus. Dissipative hamiltonian neural networks: Learning dissipative and conservative dynamics separately. *arXiv preprint arXiv:2201.10085*, 2022.

[Szegedy *et al.*, 2013] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[Wang and Zhong, 2024] Yifan Wang and Linlin Zhong. Nas-pinn: Neural architecture search-guided physics-informed neural network for solving pdes. *Journal of Computational Physics*, 496:112603, 2024.

[Yuval, 2011] Netzer Yuval. Reading digits in natural images with unsupervised feature learning. In *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[Zhang and Li, 2019] Jiliang Zhang and Chen Li. Adversarial examples: Opportunities and challenges. *IEEE transactions on neural networks and learning systems*, 31(7):2578–2593, 2019.

[Zhao *et al.*, 2024] Zhiyuan Zhao, Xueying Ding, and B. Aditya Prakash. PINNsformer: A transformer-based framework for physics-informed neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.