PINN-GAN: An architecture for inferring physical system with unknown parameters from partial observations

Haotian Hang, Jingyi Liu, Chenchen Huang
USC ID: 3993483291, 5179170292, 1748875110
Email address: haotianh@usc.edu, liu027@usc.edu, chencheh@usc.edu
Instructor: Dr. Assad Oberai

October 25, 2021

1 Objective

For a problem governed by known differential equation(s) $\mathcal{L}[\mathbf{u}(\mathbf{x}, t); \alpha] = 0$ with some unknown parameter(s) α with experimental/numerical measurement at some location and time, we want to deduce the solution of the equation in the whole field and solve the unknown parameter simultaneously by combination of a Physics Informed Neural Network (PINN) which solve the equation based on a guess of the parameter and a Neural Network which updates the parameter based on the solution given by the PINN and measurement data. The two neural networks works together like a Generative Adversarial Network (GAN). For a better understanding of what we are trying to do, we show the objective as following schematic 1.(In appendix, we show a comparison between standard PINN problem and our sample problem in Figure 4)

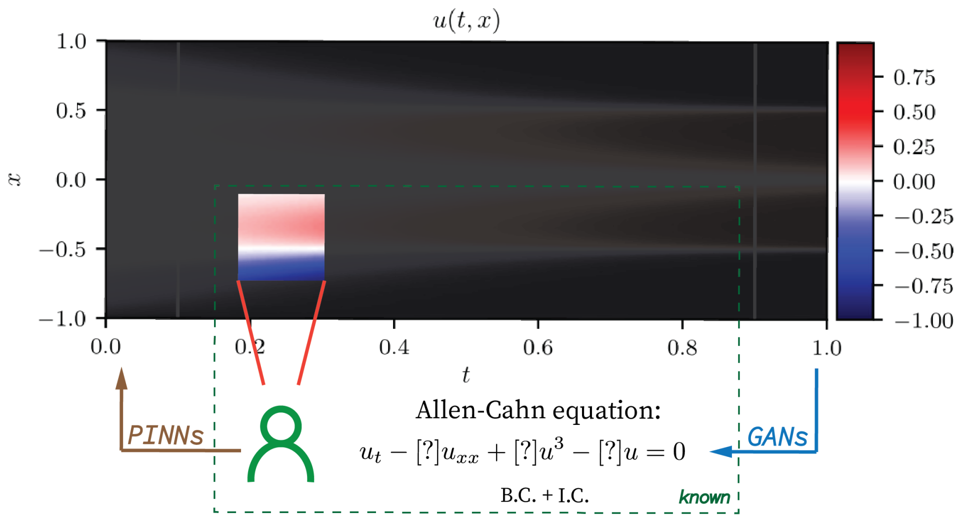


Figure 1: Schematic of sample problem

2 Motivation

In most dynamics or fluid dynamics problem, though we can affirm the governing differential equation, some parameters might remain unknown. Meanwhile, in some experiments, there are limitations that we cannot measure all the states. Therefore, it is worthwhile developing an architecture that can solve or approximate the solution only based on part of information we need. For example, with a basic pendulum problem, if the target is to draw the phase diagram without knowing the length of string and only based on some velocity and angle measurement, it is very hard to solve it. Before solving the nonlinear pendulum problem numerically, we need to find out the matching string length first as it appears as a parameter in the nonlinear differential equation.

3 Deliverable

We will deliver an algorithm that will train two neural networks simultaneously. One is a PINN which takes a guessed parameter and measurements data to solve the known governing equation of this system. The other one is a neural network which takes the guessed parameter, the measurements data, and the obtained output from the PINN on a field as input, and takes an updated parameter as an output.

4 Research plan

Firstly, we will start from a simple and well-known problem, like Allen–Cahn equation as described in Section3.2.1 in [RPK19], which has analytical solution, and has been proved to be solvable for PINN. We can easily build a dataset by solving the equation analytically for different parameter values.

Using this simple problem, we will build our network structures and test the concept. After constructing, we can test our architecture on other simple dynamics problem, such as a nonlinear pendulum. After that we can also apply it on a more complex fluid problems.

4.1 Brief description of the algorithm

We will have two neural networks, the first one is called a solver, which is a PINN. It takes $\mathbf{x}, t$ as input, and $\mathbf{u}$ as output. However, the parameters in differential equation are relied on the second part we are going to introduce.

For the other neural network which is called judger, it feeds updated parameter α back into the PINN.

4.2 Architecture

We show the general architecture of the algorithm in the following diagram 2.

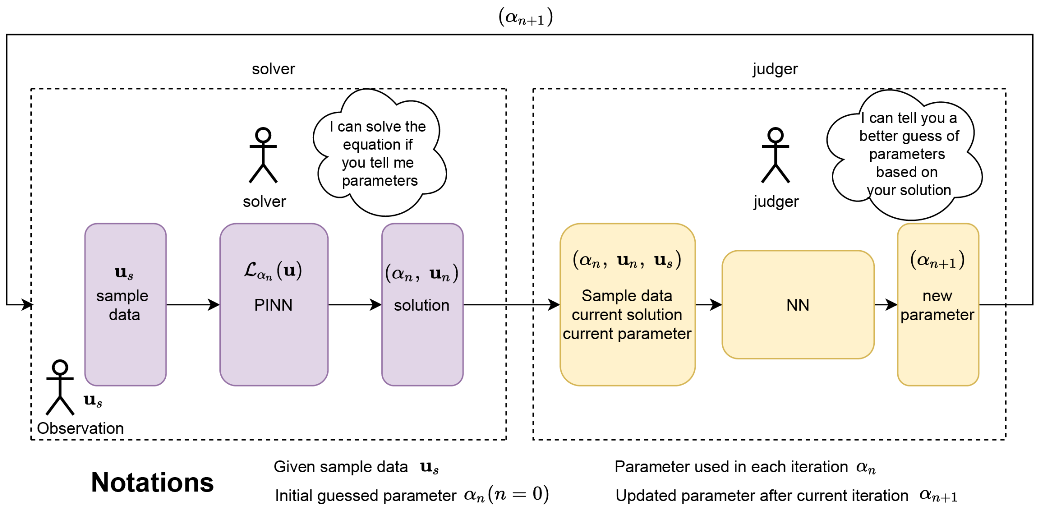

(α_{n+1})
solver
judger
I can solve the equation if you tell me parameters
solver
I can tell you a better guess of parameters based on your solution
judger
$\mathbf{u}_s$ sample data
$\mathcal{L}_{\alpha_n}(\mathbf{u})$ PINN
$(\alpha_n,\ \mathbf{u}_n)$ solution
$(\alpha_n,\ \mathbf{u}_n,\ \mathbf{u}_s)$ Sample data current solution current parameter
NN
(α_{n+1}) new parameter
$\mathbf{u}_s$
Observation
Notations
Given sample data $\mathbf{u}_s$
Initial guessed parameter $\alpha_n (n = 0)$
Parameter used in each iteration α_n
Updated parameter after current iteration α_{n+1}

Figure 2: Diagram of architecture

The loss function for the PINN will be stated in section 4.3.

For the other neural network which is called judger, it feeds updated parameter α into the PINN. As shown in Figure 3, it takes three inputs, the guessed parameter used by the PINN, the output physical field from the PINN, and the measurements. The architecture of the network is first giving several convolution layers (and pooling layers) for the output physical field, and possibly several layers of fully connected layers or convolution layers for the measurement (depends on how many measurements we have). And then we flatten the output of these two and combine them with the guessed parameter and put these to multi-layer perceptron (MLP) to get the updated parameter.

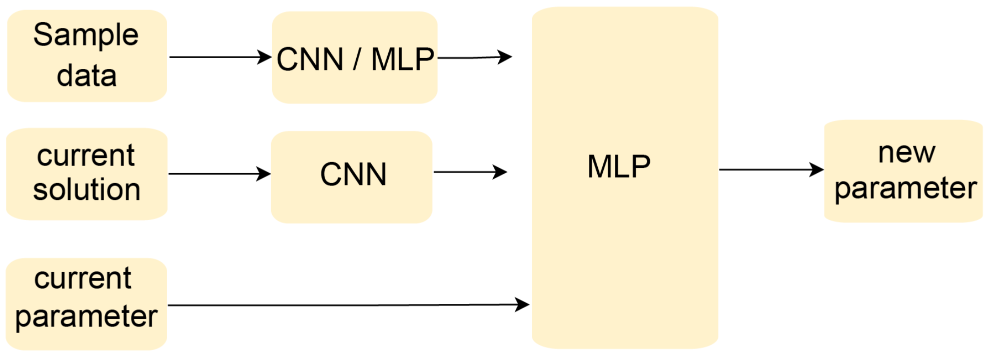


Figure 3: Diagram of judger

Before talking about the training, we would like to first talk about the dataset. The dataset a set of parameters with their corresponding solutions to this problem. We can obtain this dataset analytically or numerically.

4.3 Loss function

The loss function for the PINN composed of two parts, the first term is for satisfying the governing equation, and the second term is for matching the given measurements data. Thus, we can write the loss as

$$\Pi_1 = \frac{1}{N_1}\sum_{i=1}^{N_1}\mathcal{L}[\mathbf{u}(\mathbf{x_i},t_i);\boldsymbol{\alpha}] + \lambda_1\frac{1}{N_2}\sum_{i=1}^{N_2}||\mathbf{u}(\mathbf{x_i},t_i) - \mathbf{u_s}(\mathbf{x_i},t_i)|| + \lambda_2\text{reg}, \tag{1}$$

where N_1 is the number of points we evaluate the governing equation, α is the parameter in the equation, $\mathbf{u_s}(\mathbf{x_i},t_i)$, $i \in [1, N_2]$ is the measurements data set, and λ_1 and λ_2 are the hyper parameters we need to turn by hand.

The loss function for the second Neural Network is constructed per standard supervised learning. Thus, the loss function is mean square root (MSE) between the output of the neural network and the dataset,

$$\Pi_2 = ||\boldsymbol{\alpha} - \boldsymbol{\alpha_s}|| + \lambda_1\text{reg}, \tag{2}$$

, where $\boldsymbol{\alpha_s}$ is the actual parameter corresponding to the measurement.

4.4 Training

The training can be divided into three stages.

The first stage is called the pre-trained stage of the judger. Two pairs of corresponding parameters and solutions are randomly chosen from our dataset. For one pair, observations are sampled from the solution, with some noises added, and the corresponding parameters are used as the label. For the other pair, the parameters and solution are given as input.

In the second stage, the fine turn of the judger and training of the solver (PINN) happens simultaneously. Since the output of the solver(PINN) maybe distorted when given the observation and parameters which are not corresponding, the judger need to learn how to decode the information to give a better estimation of the parameters. At this stage, the true value of the parameter is feeded as labels for the judger, and the solver will continue to use the output of the judger to train.

In the third stage, only the solver (PINN) is trained, the judger is only used as a function to give a better estimation of parameters.

4.5 Validation

Take the Allen-Cahn as example, we will use the labelled solutions which are generated analytically as validation data set.

4.6 Testing

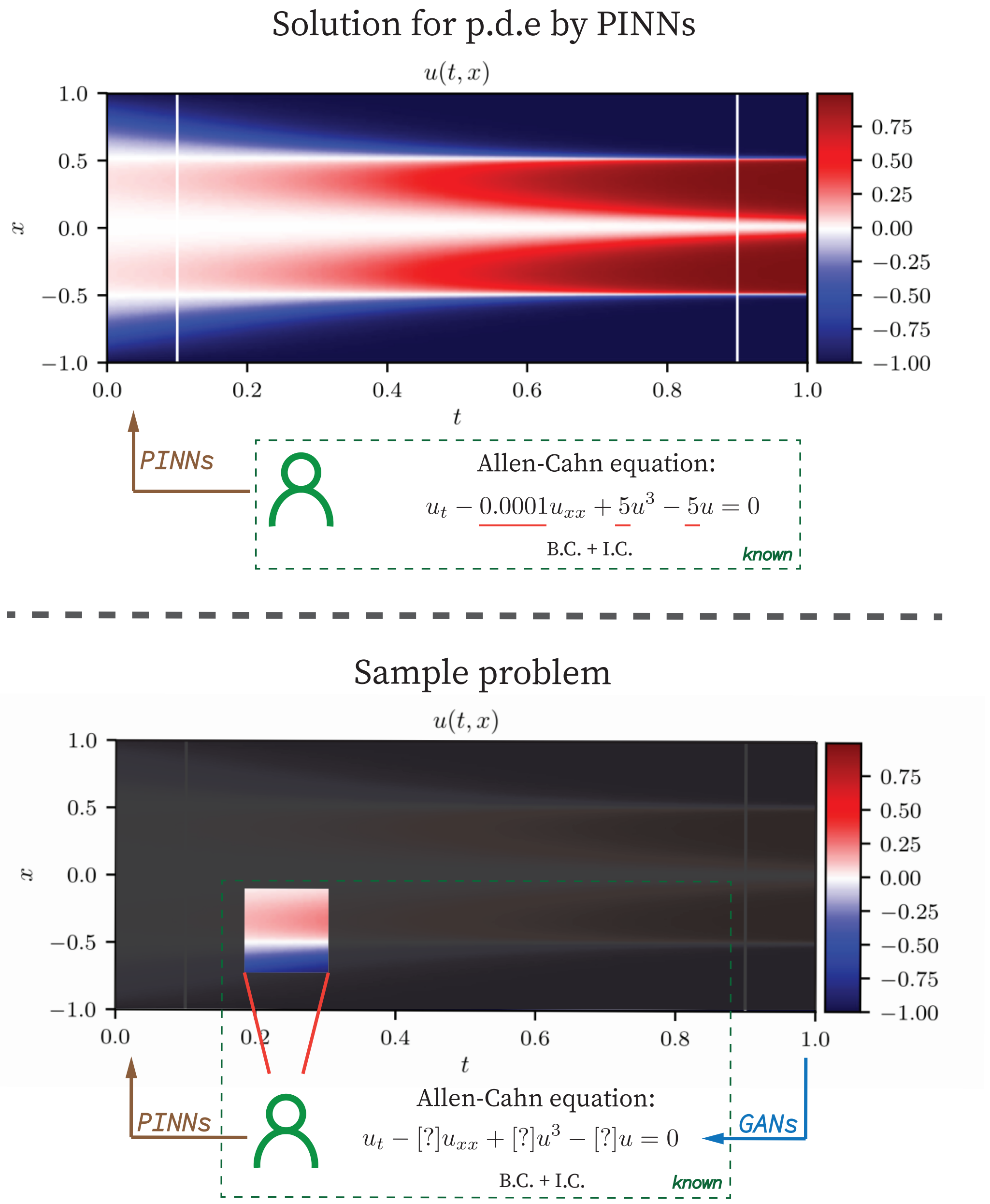
After fully trained the model for this specific problem, we will test it with another set of data that are generated analytically. For the methodology part, we will apply this architecture to other problems, e.g. nonlinear pendulum.

References

[RPK19] 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*, 378:686–707, 2019.

Appendices

A Schematic of PINN-GAN



Solution for p.d.e by PINNs
$u(t, x)$
x
t
PINNs
Allen-Cahn equation:
$u_t - \underline{0.0001} u_{xx} + \underline{5} u^3 - \underline{5} u = 0$
B.C. + I.C.
known
Sample problem
$u(t, x)$
PINNs
Allen-Cahn equation:
$u_t - [?] u_{xx} + [?] u^3 - [?] u = 0$
GANs
B.C. + I.C.
known

Figure 4: Illustration of general objective