Energy-Dissipative Evolutionary Deep Operator Neural Networks

Jiahao Zhang

Shiheng Zhang

Jie Shen

Guang Lin

MATHEMATICS

https://arxiv.org/abs/2306.06281

Objectives

- Construct an evolutionary neural network to solve Gradient flow problem in non-data driven setting.
- Perform operator learning on a kind of PDEs with different parameters in a single neural network.
- Introduce the modified energy and apply the SAV method to keep the unconditionally modified energy dissipative law.
- Introducing the adaptive time stepping and restart strategy to speed the training process.
- Developing a new gradient-based optimization algorithm for various optimization tasks.

Problem Setting

Consider a general gradient flow problem,

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}) = 0 \\ \boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{f}(\boldsymbol{x}) \end{aligned} \tag{1}$$

where $\boldsymbol{u} \in \boldsymbol{R}^l$, $\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u})$ can be written as a variational derivative of a free energy functional $E[u(\boldsymbol{x})]$ bounded from below, $\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}) = \frac{\delta E}{\delta \boldsymbol{u}}$. The first step is to approximate the initial condition operator with DeepONet.

For an operator $\mathcal{G}$, $\mathcal{G} : \boldsymbol{u}(\boldsymbol{x}) \mapsto \boldsymbol{f}(\boldsymbol{x})$, the data feed into the DeepONet is in the form $(\boldsymbol{u}, y, \mathcal{G}(\boldsymbol{u})(y))$. It is obtained by the given initial conditions.

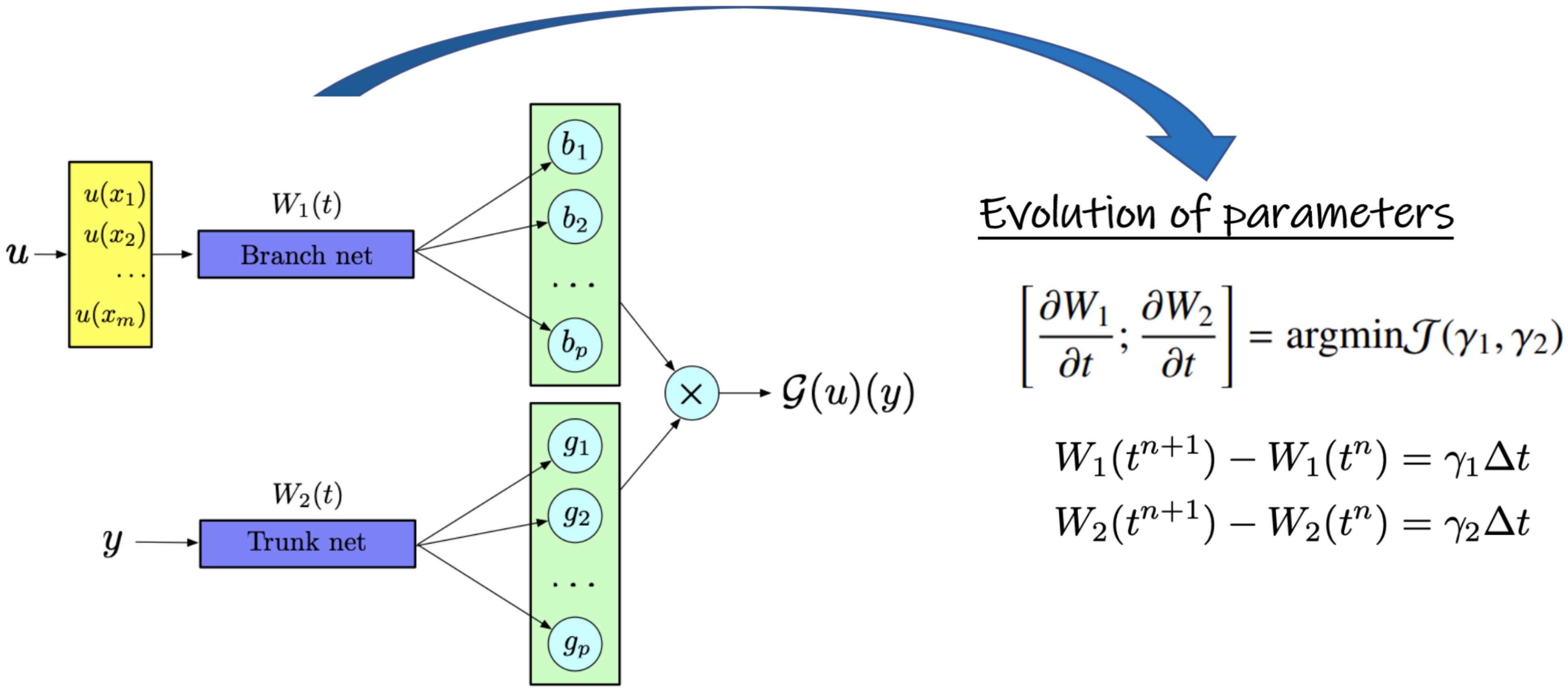
$$\mathcal{G}(\boldsymbol{u})(\boldsymbol{x}) \approx \sum_{k=1}^{p} g_k \boldsymbol{b}_k \tag{2}$$

The DeepONet Model

Theorem 0.1 (Universal Approximation Theorem for Operator) *Suppose that Ω_1 is a compact set in X, X is a Banach Space, V is a compact set in $C(\Omega_1)$, Ω_2 is a compact set in $\boldsymbol{R}^d$, σ is a continuous non-polynomial function, $\mathcal{G}$ is a nonlinear continuous operator, which maps v into $C(\Omega_2)$, then for any $\epsilon > 0$, there are positive integers M, N, m, constants $c_i^k, \zeta_k, \xi_{ij}^k \in \boldsymbol{R}$, points $\omega_k \in \boldsymbol{R}^n, x_j \in K_1, i = 1, \cdots, M$, $k = 1, \cdots, N, j = 1, \cdots, m$, such that*

$$| \mathcal{G}(u)(y) - \sum_{k=1}^{N} \sum_{i=1}^{M} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right) \cdot \sigma(\omega_k \cdot y + \zeta_k) | < \epsilon$$

holds for all $u \in V$ and $y \in \Omega_2$.



$$\left[\frac{\partial W_1}{\partial t}; \frac{\partial W_2}{\partial t}\right] = \operatorname{argmin} \mathcal{J}(\gamma_1, \gamma_2)$$

$$W_1(t^{n+1}) - W_1(t^n) = \gamma_1 \Delta t$$
$$W_2(t^{n+1}) - W_2(t^n) = \gamma_2 \Delta t$$

Evolutionary Network

Denoting the parameters in the branch network as W_1 and the parameters in the trunk network as W_2, W_1 and W_2 can be regarded a function of t since they change in every time step. According to the derivative's chain rule, we have

$$\frac{\partial \boldsymbol{u}}{\partial t} = \frac{\partial \boldsymbol{u}}{\partial W_1}\frac{\partial W_1}{\partial t} + \frac{\partial \boldsymbol{u}}{\partial W_2}\frac{\partial W_2}{\partial t} \tag{3}$$

Since $\boldsymbol{u} = \sum_{k=1}^{p} g_k \boldsymbol{b}_k = \sum_{k=1}^{p} g_k(W_1(t))\boldsymbol{b}_k(W_2(t))$, then

$$\frac{\partial \boldsymbol{u}}{\partial t} = \sum_{k=1}^{p} \frac{\partial g_k(W_1(t))}{\partial W_1}\frac{\partial W_1}{\partial t}\boldsymbol{b}_k(W_2(t)) + \sum_{k=1}^{p} g_k(W_1(t))\frac{\partial \boldsymbol{b}_k(W_2(t))}{\partial W_2}\frac{\partial W_2}{\partial t} \tag{4}$$

Our objective is to obtain $\frac{\partial W_1}{\partial t}$ and $\frac{\partial W_2}{\partial t}$, the update rule for parameters. It is equivalent to solve a minimization problem,

$$\left[\frac{\partial W_1}{\partial t}; \frac{\partial W_2}{\partial t}\right] = \text{argmin}\mathcal{J}(\gamma_1, \gamma_2) \tag{5}$$

where

$$\mathcal{J}(\gamma_1, \gamma_2) = \frac{1}{2}\left\| \sum_{k=1}^{p} \frac{\partial g_k(W_1(t))}{\partial W_1}\gamma_1 \boldsymbol{b}_k(W_2(t)) + \sum_{k=1}^{p} g_k(W_1(t))\frac{\partial \boldsymbol{b}_k(W_2(t))}{\partial W_2}\gamma_2 - \mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}) \right\|_2^2 \tag{6}$$

Evolutionary Network

By denoting

$$(\mathbf{J_1})_{ij_1} = \sum_{k=1}^{p} \frac{\partial g_k(W_1(t))}{\partial W_1^{j_1}} \boldsymbol{b}_k^i(W_2(t)) \tag{7}$$

$$(\mathbf{J_2})_{ij_2} = \sum_{k=1}^{p} g_k(W_1(t)) \frac{\partial \boldsymbol{b}_k^i(W_2(t))}{\partial W_2^{j_2}} \tag{8}$$

$$(\mathbf{N})_i = \mathcal{N}\left(\boldsymbol{u}_{\boldsymbol{x}}^i\right) \tag{9}$$

By denoting γ_i^{opt} as optimal values of γ_i, $i = 1, 2$, The minimization problem can be transformed into a linear system by the first-order optimal condition.

$$\mathbf{J_1^T}\left(\gamma_1^{opt}\mathbf{J_1} + \gamma_2^{opt}\mathbf{J_2} - \mathbf{N}\right) = 0 \tag{10}$$

$$\mathbf{J_2^T}\left(\gamma_1^{opt}\mathbf{J_1} + \gamma_2^{opt}\mathbf{J_2} - \mathbf{N}\right) = 0 \tag{11}$$

The feasible solutions of the above equations are the approximated time derivatives of W_1 and W_2.

$$\frac{dW_1}{dt} = \gamma_1^{opt} \tag{12}$$

$$\frac{dW_2}{dt} = \gamma_2^{opt} \tag{13}$$

Energy Dissipative Evolutionary Deep Operator Neural Network

Let's reconsider the given problem.

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}) &= 0 \\ \boldsymbol{u}(\boldsymbol{x},0) &= \boldsymbol{f}(\boldsymbol{x}) \end{aligned} \tag{14}$$

where $\boldsymbol{u} \in \boldsymbol{R}^l$, $\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u})$ can be written as a variational derivative of a free energy functional $E[\boldsymbol{u}(\boldsymbol{x})]$ bounded from below, $\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}) = \frac{\delta E}{\delta \boldsymbol{u}}$. Taking the inner product with $\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u})$ of the first equation, we obtain the energy dissipation property

$$\frac{dE[\boldsymbol{u}(\boldsymbol{x})]}{dt} = \left(\frac{\delta E}{\delta \boldsymbol{u}}, \frac{\partial \boldsymbol{u}}{\partial t}\right) = \left(\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}), \frac{\partial \boldsymbol{u}}{\partial t}\right) - (\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}), \mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u})) \leq 0 \tag{15}$$

However, it is usually hard for a numerical algorithm to be efficient as well as energy dissipative. The SAV approach was introduced to construct numerical schemes which is energy dissipative with a modified energy. Assuming $E[\boldsymbol{u}(\boldsymbol{x})] > 0$, it introduces a $r(t) = \sqrt{E[\boldsymbol{u}(\boldsymbol{x},t)]}$, and expands the gradient flow problem as

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} &= -\frac{r}{\sqrt{E(\boldsymbol{u})}} \mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}) \\ r_t &= \frac{1}{2\sqrt{E(\boldsymbol{u})}} \left(\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}), \frac{\partial \boldsymbol{u}}{\partial t}\right) \end{aligned} \tag{16}$$

With $r(0) = \sqrt{E[\boldsymbol{u}(\boldsymbol{x},t)]}$, the above system has a solution $r(t) \equiv \sqrt{E[\boldsymbol{u}(\boldsymbol{x},t)]}$ and $\boldsymbol{u}$ being the solution of the original problem.

First order scheme

By setting $\boldsymbol{u}^n = \sum_{k=1}^{p} g_k \boldsymbol{b}_k$, a first order scheme can be constructed as

$$\begin{aligned} \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} &= -\frac{r^{n+1}}{\sqrt{E(\boldsymbol{u}^n)}} \mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}^n) \\ \frac{r^{n+1} - r^n}{\Delta t} &= \frac{1}{2\sqrt{E(\boldsymbol{u}^n)}} \int_\Omega \mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}^n) \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} dx. \end{aligned} \tag{17}$$

Plugging the first equation into the second one, we obtain:

$$\frac{r^{n+1} - r^n}{\Delta t} = -\frac{r^{n+1}}{2E(\boldsymbol{u}^n)} \|\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}^n)\|^2, \tag{18}$$

which implies

$$r^{n+1} = \left(1 + \frac{\Delta t}{2E(\boldsymbol{u}^n)} \|\mathcal{N}_{\boldsymbol{x}}(\boldsymbol{u}^n)\|^2\right)^{-1} r^n \tag{19}$$

Theorem 0.2 (Discrete Energy Dissipation Law) *With the modified energy define above, the scheme is unconditionally energy stable, i.e.*

$$(r^{n+1})^2 - (r^n)^2 \leq 0. \tag{20}$$

Algorithm

The corresponding linear system of the first order optimal condition is

$$\mathbf{J_1^T}\left(\gamma_1^{opt}\mathbf{J_1}+\gamma_2^{opt}\mathbf{J_2}-\frac{r^{n+1}}{\sqrt{E(\boldsymbol{u^n})}}\mathbf{N}\right)=0 \tag{21}$$

$$\mathbf{J_2^T}\left(\gamma_1^{opt}\mathbf{J_1}+\gamma_2^{opt}\mathbf{J_2}-\frac{r^{n+1}}{\sqrt{E(\boldsymbol{u^n})}}\mathbf{N}\right)=0 \tag{22}$$

1. Generate input data samples in the form of $(u, y, \mathcal{G}(u)(y))$ for the DeepONet.
2. Feed $[u(x_1), u(x_2), \cdots, u(x_m)])$ into the branch network and $y \in Y$ into the trunk network. Denote the output of the DeepONet as q.
3. Update the parameters in the DeepONet by minimizing a cost function, where the cost function can be taken as the mean squared error as $\frac{1}{|Y|}\sum_{y\in Y}\|\mathcal{G}(u)(y)-q\|^2$.
4. Once the DeepONet has been trained well, solve the system of equations to obtain $\left[\frac{\partial W_1}{\partial t};\frac{\partial W_2}{\partial t}\right]$.
5.The value of $\left[\frac{\partial W_1}{\partial t};\frac{\partial W_2}{\partial t}\right]$ can be obtained in the current step. W_1^{n+1} and W_2^{n+1} for the next step can be also obtained by the Forward Euler method or Runge-Kutta method.
6. Repeat step 5 until the final time T, where $T = t_0 + s\Delta t$.
7. Output the solution at time T in the DeepONet with parameters obtained in step 6.

Adaptive Strategy

```
1. Set the tolerance for $\xi$ as $\epsilon_0$ and $\epsilon_1$, the initial time step $\Delta t$, the maximum time step $\Delta t_{max}$ and the minimum time step $\Delta t_{min}$
2. Compute $u^{n+1}$.
3. Compute $\xi^{n+1} = \frac{r^{n+1}}{\sqrt{E^n}}$.
4. **If** $|1 - \xi^{n+1}| > \epsilon_0$,
      **Then** $\Delta t = \max(\Delta t_{min}, \Delta t/2)$;
   **Else if** $|1 - \xi^{n+1}| < \epsilon_1$,
      **Then** $\Delta t = \min(\Delta t_{max}, 2\Delta t)$.
   **Go to Step 2**.
5. Update time step $\Delta t$.
```

Another popular strategy to keep r approximating the original energy E is to reset the SAV r^{n+1} to be E^{n+1} in some scenarios. The specific algorithm is as following:

```
1. Set the tolerance for $\xi$ as $\epsilon_2$.
2. Compute $u^{n+1}$.
3. Compute $\xi^{n+1} = \frac{r^{n+1}}{\sqrt{E^n}}$.
4. **If** $|1 - \xi^{n+1}| > \epsilon_2$,
      **Then** $r^{n+1} = \sqrt{E^{n+1}}$ and **Go to Step 2**.
5. Go to next iteration.
```

Example-1 Parametric Heat Equation

A general parametric heat equation in 1D can be described by

$$u_t = cu_{xx} \tag{23}$$

$$u(x,0) = sin(\pi x) \tag{24}$$

$$u(0,t) = u(2,t) = 0 \tag{25}$$

Error	$T = 0.025$	$T = 0.05$	$T = 0.075$	$T = 0.1$
$c = 1.2$	1.30×10^{-5}	1.43×10^{-5}	1.35×10^{-5}	1.20×10^{-5}
$c = 1.5$	1.35×10^{-5}	1.27×10^{-5}	9.80×10^{-6}	7.80×10^{-6}
$c = 1.8$	1.17×10^{-5}	1.03×10^{-5}	7.88×10^{-5}	1.83×10^{-5}
$c = 2.5$	2.20×10^{-4}	1.34×10^{-4}	6.02×10^{-5}	7.08×10^{-6}

Table 1: The parametric heat equation: The initial condition of the PDE is $f(x) = \sin(\pi x)$. The error is defined by $\frac{1}{N_x}\sum_{k=1}^{N_x}(u(x_k) - \hat{u}(x_k))^2$, where $N_x = 51$, u is the solution obtained by EDE-DeepONet and $\hat{u}$ is the exact solution.

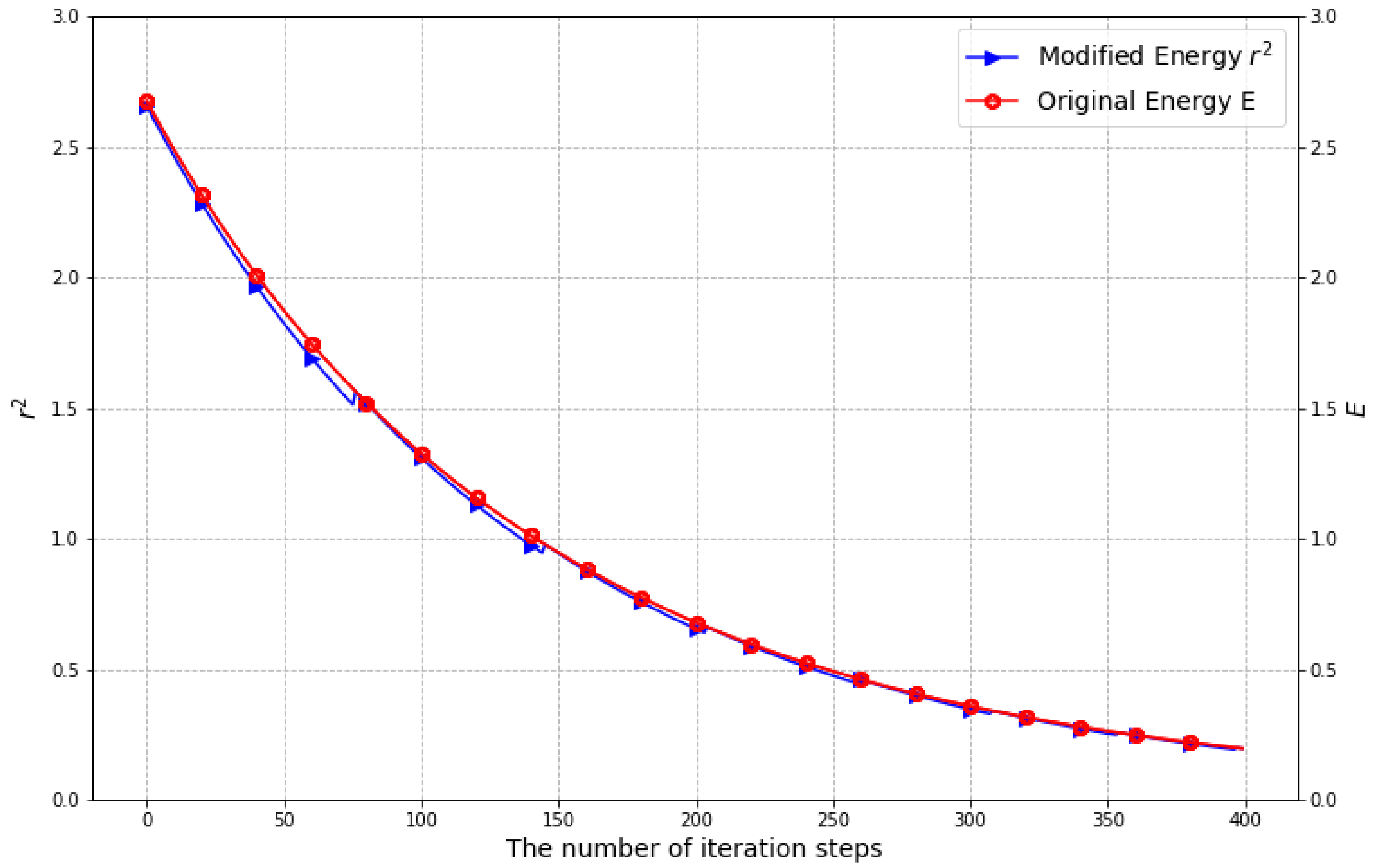
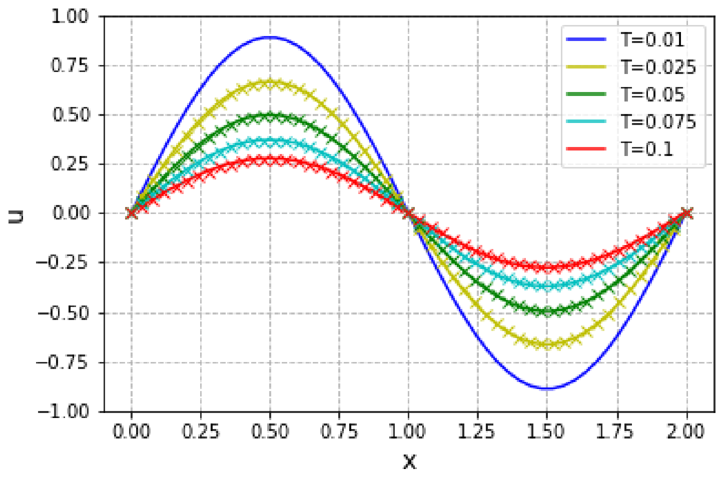
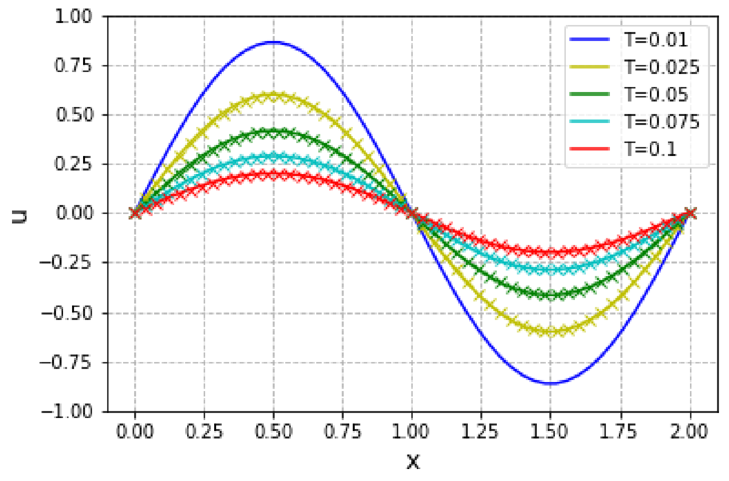


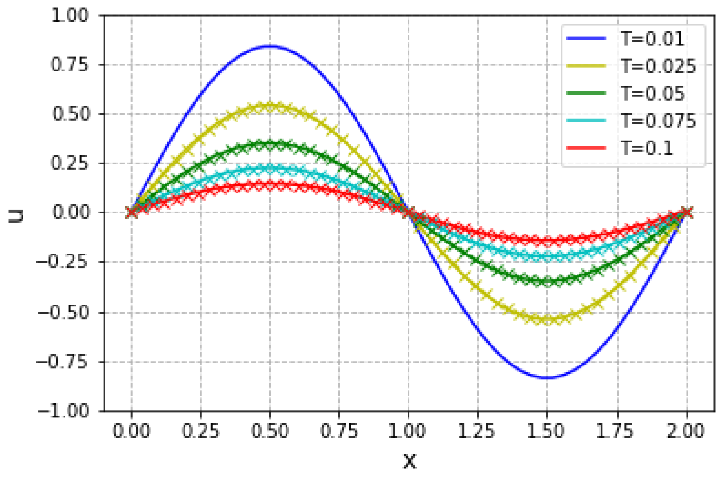
Figure 1: The parametric heat equation: The modified energy and original energy when training the network. Each iteration step represents one forward step of the PDE's numerical solution with $\Delta t = 2.5 \times 10^{-4}$. This kind of PDEs is more complicated, so it need more restarts in the training process. The original energy keeps decreasing and the modified energy also shows good approximation of the original energy.



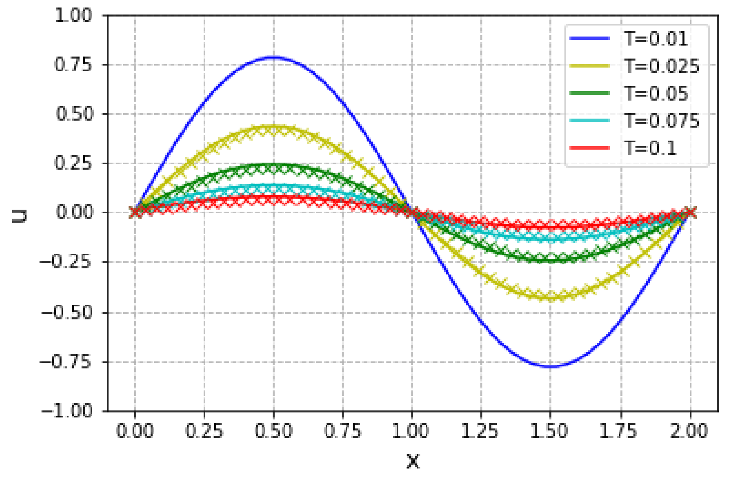
(a) $c = 1.2$



(b) $c = 1.5$



(c) $c = 1.8$



(d) $c = 2.5$

Figure 2: The parametric heat equation: The solution with 4 different parameters c. The curve represents the solution obtained by the EDE-DeepONet and xxx represents the reference solution. The training parameter c is in the range of $[1, 2)$, so we give 3 examples in this range. We also present the case out of the range in Figure 5-(d).

Example-2 Allen-Cahn Equation

1D Allen-Cahn equation with various thickness of the interface:

$$u_t = u_{xx} - \frac{1}{\epsilon^2}(u^3 - u) \tag{26}$$

$$u(-1,t) = u(1,t) = 0 \tag{27}$$

The corresponding Ginzburg–Landau free energy $E[u] = \int_0^1 \frac{1}{2}|u_x|^2 dx + \int_{x=0}^{x=1} G(u)dx$, where $G(u) = \frac{1}{4\epsilon^2}(u^2-1)^2$.

2D Allen-Cahn equation with various initial conditions:

$$u_t = \Delta u - g(u) \tag{28}$$

$$u(x,y,0) = asin(\pi x)sin(\pi y) \tag{29}$$

$$u(-1,y,t) = u(1,y,t) = u(x,-1,t) = u(x,1,t) = 0 \tag{30}$$

The corresponding Ginzburg–Landau free energy
$E[u] = \int_{-1}^1 \int_{-1}^1 \frac{1}{2}(|u_x|^2 + |u_y|^2)dxdy + \int_{-1}^1 \int_{-1}^1 G(u)dx$, where $G(u) = \frac{1}{4\epsilon^2}(u^2-1)^2$ and $g(u) = G'(u) = \frac{1}{\epsilon^2}u(u^2-1)$. Usually, we take $\epsilon = 0.1$. In the training process, we take $\Delta t = 2 \times 10^{-4}$. The number of spatial points is 51×51 and the number of training parameters a is 20.

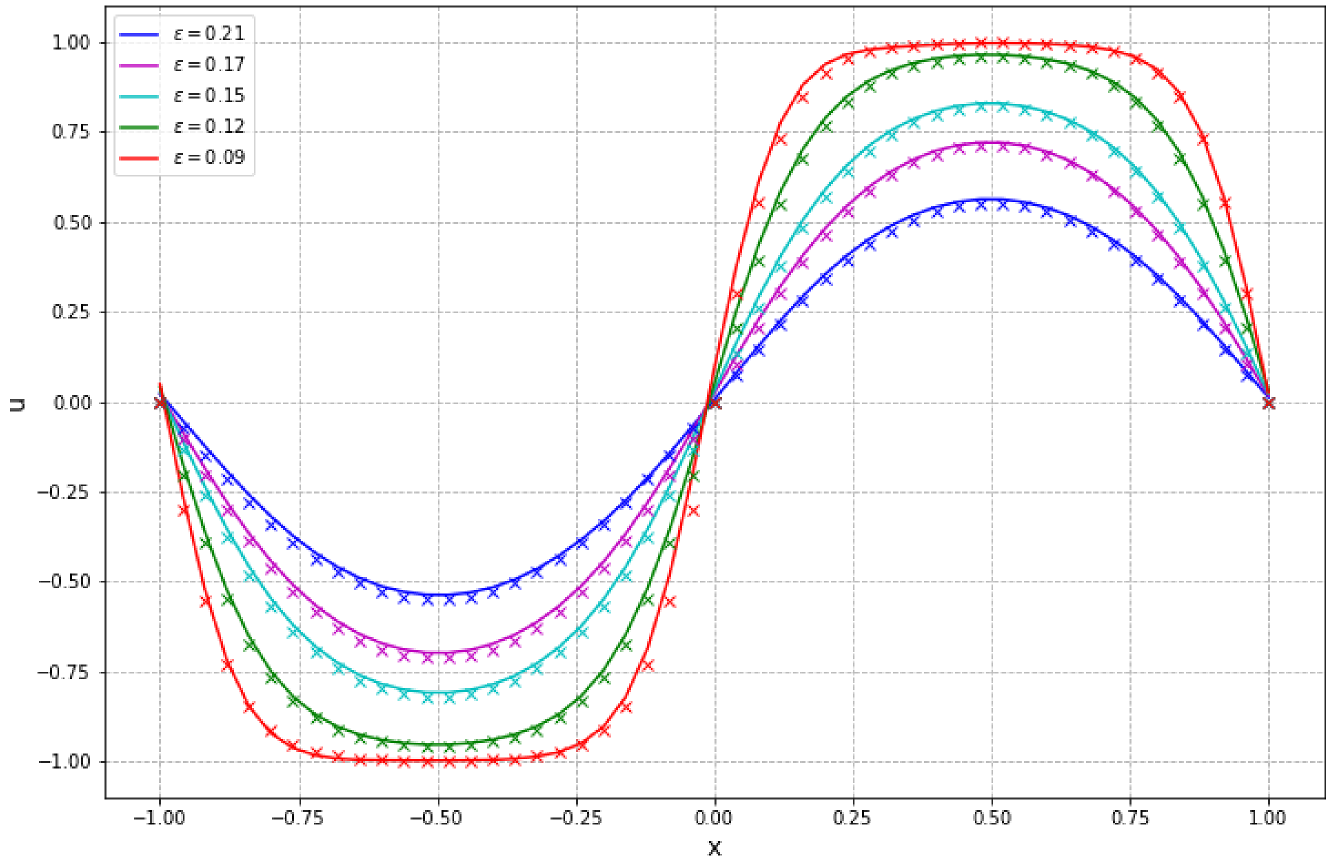
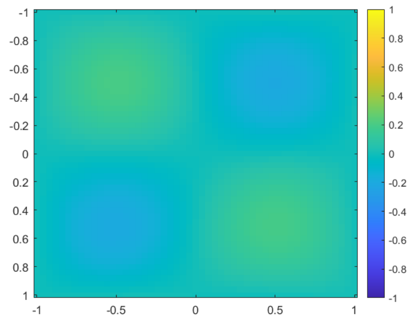
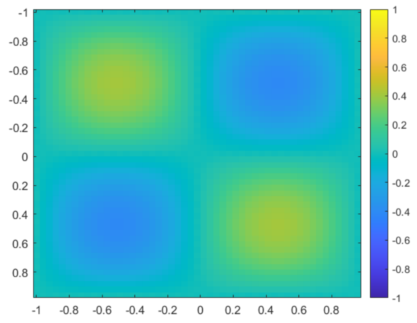


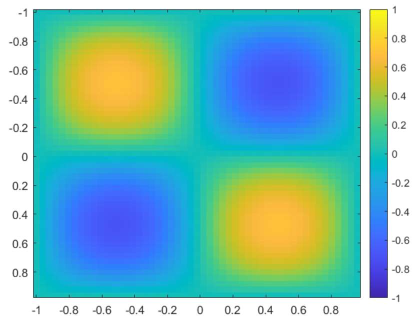
Figure 3: 1d Allen-Cahn equation: Solutions with different thickness of the interface at the same final time. The curve represents the solution obtained by EDE-DeepONet. xxx represents the reference solution.



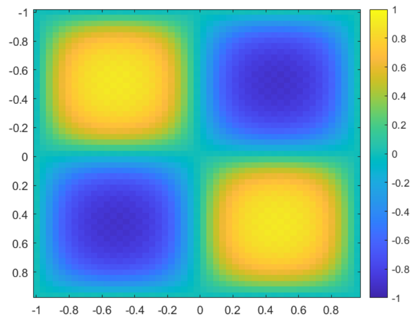
(a) $a = 0.2,\ T = 0$



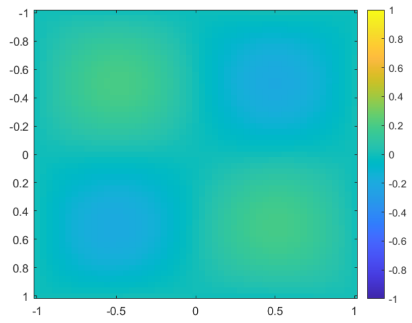
(b) $a = 0.2,\ T = 0.01$



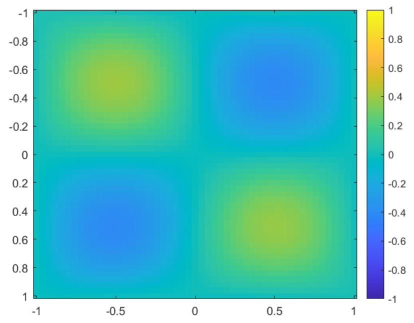
(c) $a = 0.2,\ T = 0.02$



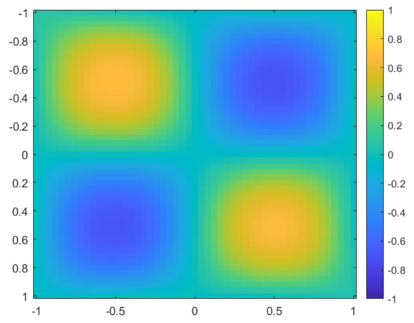
(d) $a = 0.2,\ T = 0.03$



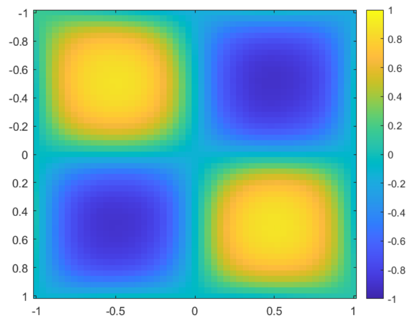
(e) $a = 0.2,\ T = 0$



(f) $a = 0.2,\ T = 0.01$



(g) $a = 0.2,\ T = 0.02$



(h) $a = 0.2,\ T = 0.03$

Figure 4: 2D Allen-Cahn equation: (a)-(d) represents the reference solution of the 2D Allen-Cahn equation with initial condition $f(x, y) = 0.3 \sin(\pi x) \sin(\pi y)$. (e)-(h) is the solution obtained by the EDE-DeepONet.

Example-3 Application to Optimization Problem

Consider the following minimization problem:

$$\min f(\boldsymbol{x}) = \sum_{k=1}^{50} x_{2k-1}^2 + \frac{1}{100}\sum_{k=1}^{50} x_{2k}^2, \tag{31}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_N)$.

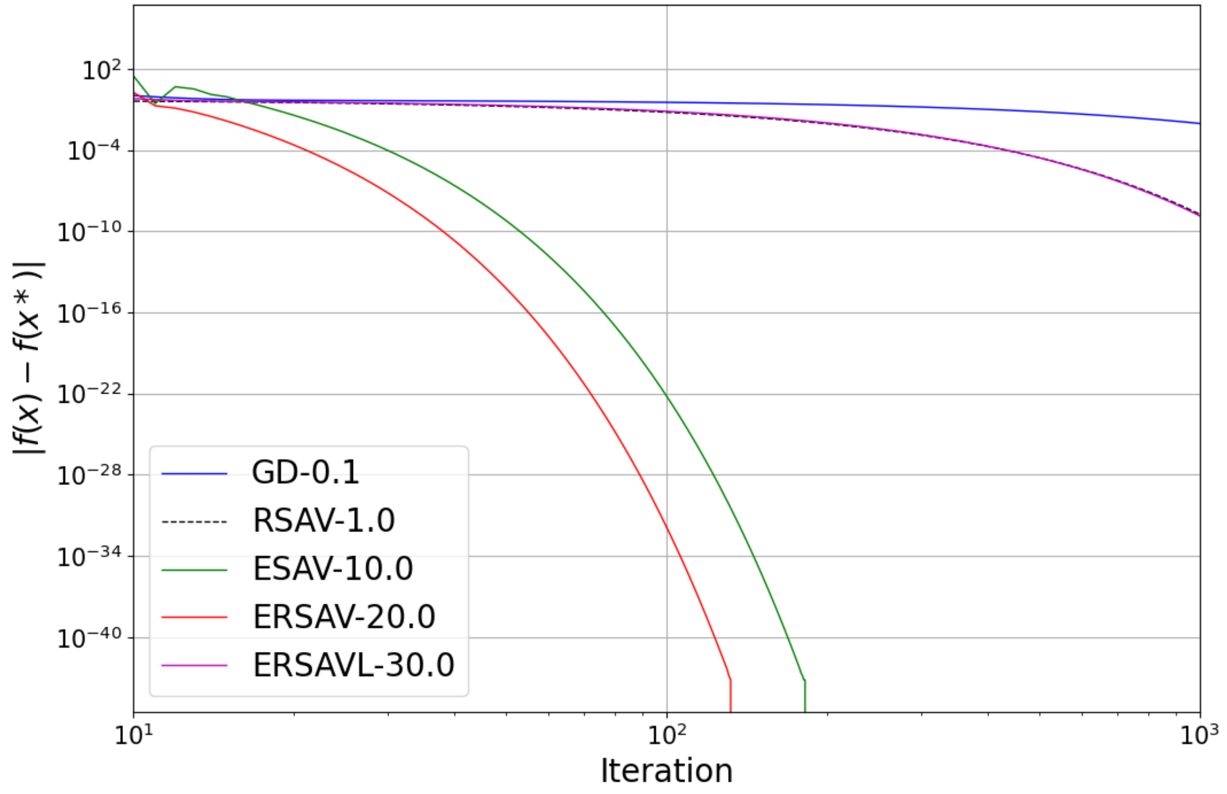


Figure 5: The loss of the convex function $f(\boldsymbol{x})$ at different step sizes.

Further Applications

- Apply EDE-DeepONet to some more complicated PDEs.
- Develop schemes on complex boundary conditions for EDE-DeepONet.
- Develop the optimization algorithm which is unconditionally energy stable.

https://arxiv.org/abs/2306.06281

Reference and Acknowledgments

Reference:

- K Wu, F Huang, J Shen, *A new class of higher-order decoupled schemes for the incompressible Navier-Stokes equations and applications to rotating dynamics, Journal of Computational Physics, 2022.*
- J Shen, *Efficient spectral-Galerkin method I. Direct solvers of second-and fourth-order equations using Legendre polynomials, SIAM Journal on Scientific Computing ,1994.*

J. Zhang, S. Zhang, J. Shen, G. Lin, Energy-Dissipative Evolutionary Deep Operator Neural Networks
https://arxiv.org/abs/2306.06281