



**HAL**  
open science

# Time and State Dependent Neural Delay Differential Equations

Thibault Monsel, Onofrio Semeraro, Lionel Mathelin, Guillaume Charpiat

► **To cite this version:**

Thibault Monsel, Onofrio Semeraro, Lionel Mathelin, Guillaume Charpiat. Time and State Dependent Neural Delay Differential Equations. ML-DE@ECAI 2024 : Machine Learning Meets Differential Equations: From Theory to Applications, Sep 2024, Santiago de compostela, Galicia, Spain. ⟨hal-04794800⟩

**HAL Id: hal-04794800**

**<https://hal.science/hal-04794800v1>**

Submitted on 21 Nov 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

## Motivation

- ➔ Discontinuities and delayed terms are encountered in many domains (physics, engineering, medicine, economics, etc...).
- ➔ These systems cannot be modelled and simulated with ODEs.
- ➔ Latent variables are introduced to solve this issue which lack physical interpretability.
- ➔ Delay Differential Equations (DDEs) naturally appear as good candidates for such systems.

We revisit the recently proposed Neural DDE by introducing **Neural State-Dependent DDE (SDDDE)**, a general and flexible framework that can model multiple and state- and time-dependent delays.

## Modelling choice

### Neural ODE

$$\frac{dy(t)}{dt} = \mathbf{f}_\theta(t, \mathbf{y}(t)), \quad \mathbf{y}(0) = \mathbf{y}_0.$$

### ANODE

$$\frac{d}{dt} \begin{bmatrix} \mathbf{y}(t) \\ \mathbf{a}(t) \end{bmatrix} = \mathbf{f}_\theta \left( t, \begin{bmatrix} \mathbf{y}(t) \\ \mathbf{a}(t) \end{bmatrix} \right), \quad \begin{bmatrix} \mathbf{y}(0) \\ \mathbf{a}(0) \end{bmatrix} = \begin{bmatrix} \mathbf{y}_0 \\ \mathbf{0} \end{bmatrix}.$$

### Neural DDE

$$\frac{dy(t)}{dt} = \mathbf{f}_\theta(t, \mathbf{y}(t), \mathbf{y}(t - \tau)), \quad \tau \in \mathbb{R}^+$$

$$\mathbf{y}(t < 0) = \phi(t),$$

### Neural Laplace

$$\mathbf{p} = \mathbf{h}_\gamma((\mathbf{y}(t_1), t_1), \dots, (\mathbf{y}(t_T), t_T))$$

$$\mathbf{F}(\mathbf{s}) = v(\mathbf{g}_\beta(\mathbf{p}, u(\mathbf{s})))$$

### Neural SDDDE

$$\frac{dy(t)}{dt} = \mathbf{f}_\theta(t, \mathbf{y}(t), \mathbf{y}(t - \tau_1(t, \mathbf{y}(t))), \dots, \mathbf{y}(t - \tau_k(t, \mathbf{y}(t))))$$

$$\mathbf{y}(t < 0) = \phi(t)$$

Our method can handle :

- ➔ multiple delays
- ➔ constant, state and time dependent delays

It cannot handle delays defined through integrals as typically found in integro-differential equations.

Delay types	$\tau$ 's Definition	Neural DDE	NPCDDEs	Neural Laplace	Neural SDDDE
References	-	[1]	[2]	[3]	This work
Constant	$\tau = a$	✓	×	✓	✓
Piece-wise constant	$\tau = \lfloor \frac{t-a}{a} \rfloor a$	×	✓	✓	✓
Time-dependent	$\tau = g(t)$	×	×	✓	✓
State-dependent	$\tau = g(t, y(t))$	×	×	×	✓
Continuous	$\tau = \int_0^t g(s, y(s)) ds$	×	×	×	×

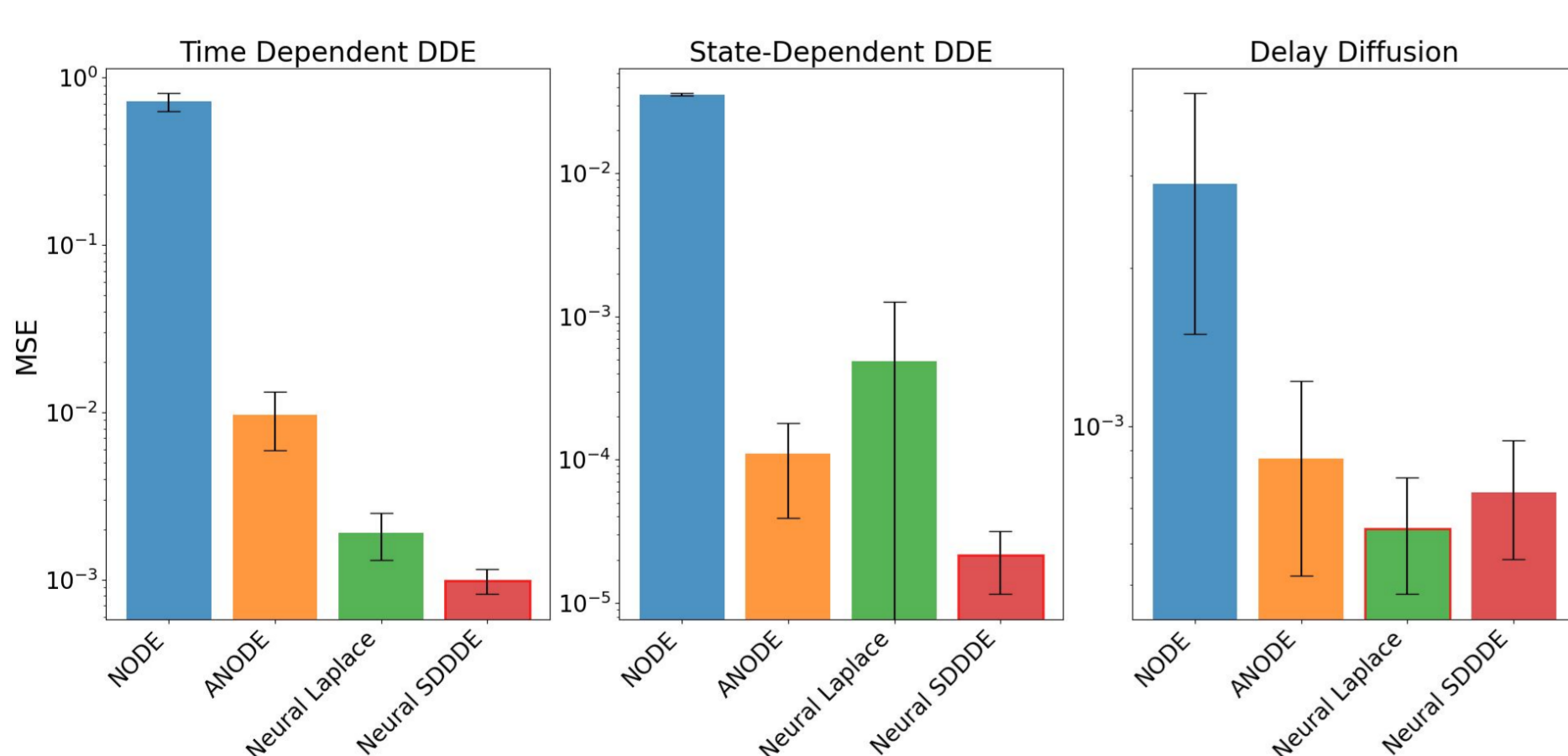
## Experiments & Results

### Time Dependent DDE

$$\tau(t) = 2 + \sin(t)$$

$$\frac{dy(t)}{dt} = y(t)[1 - y(t - \tau(t))],$$

#### Test dataset

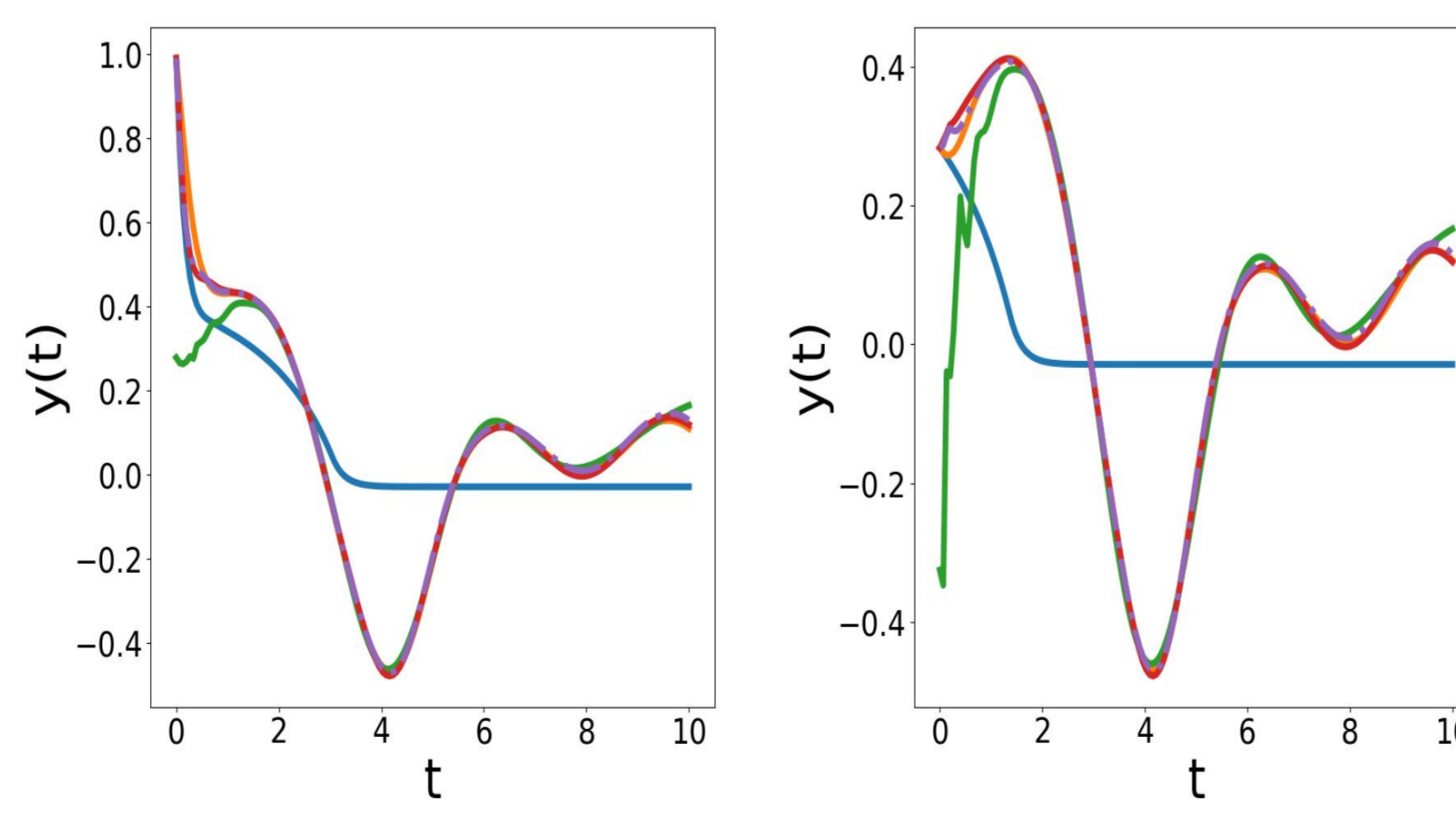


### State Dependent DDE

$$\tau(y) = \frac{1}{2} \cos(y(t))$$

$$\frac{dy(t)}{dt} = -\alpha(t)y(t) + \beta(t) \frac{y^2(t - \tau(y))}{1 + y^2(t - \tau(y))} + \gamma(t)$$

#### Changing history function dataset



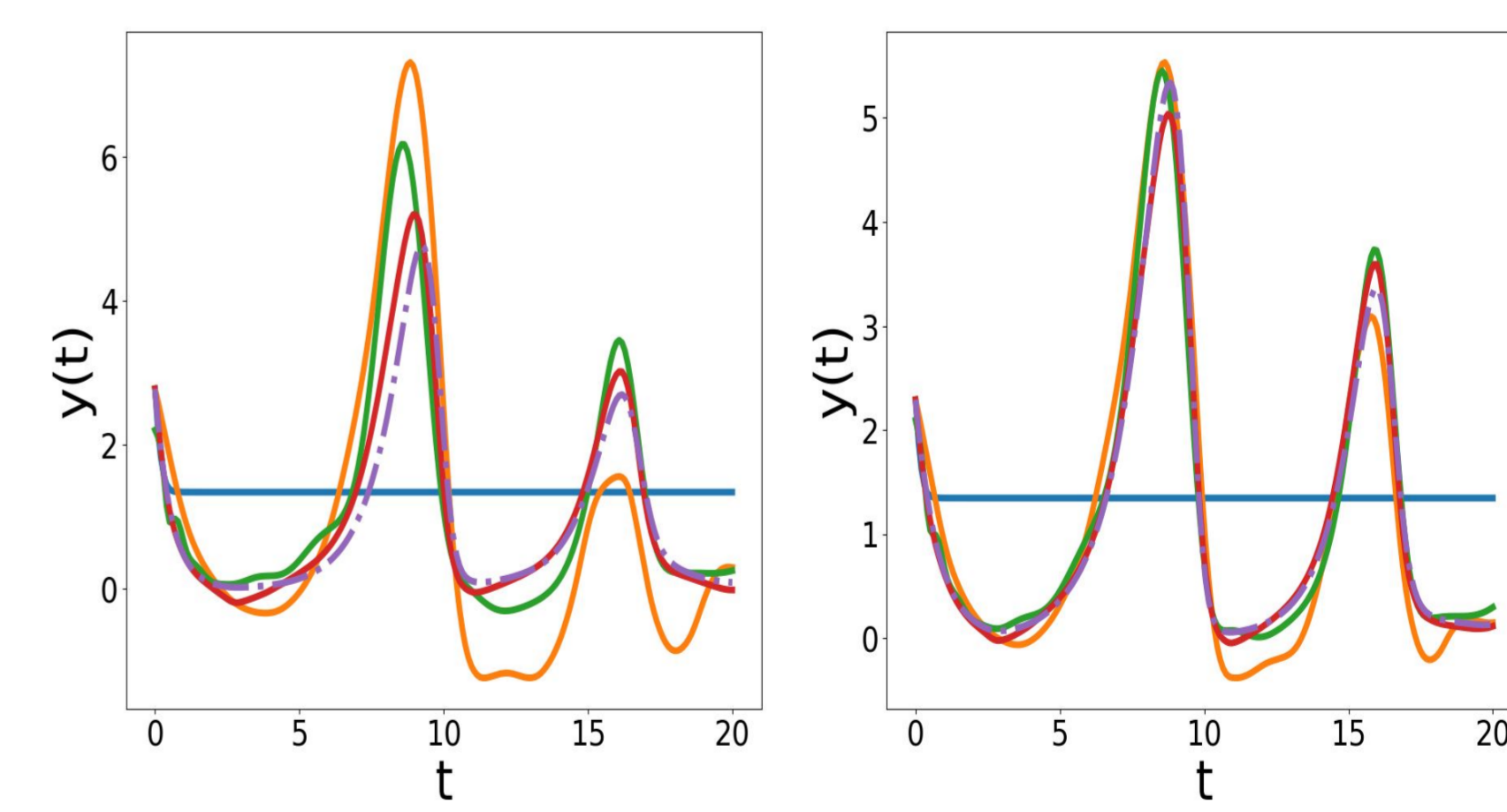
State dependent sample with new history function

### Delay Diffusion

$$\tau = 2$$

$$\frac{\partial u}{\partial t}(x, t) = D \frac{\partial^2 u}{\partial x^2}(x, t) + ru(x, t)(1 - u(x, t - \tau)),$$

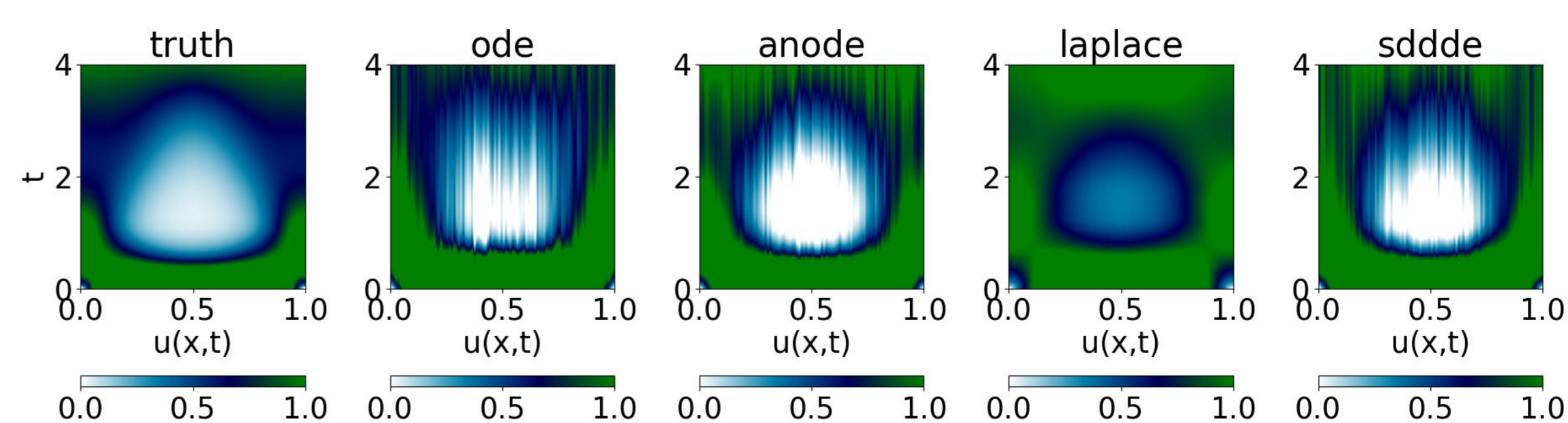
#### Out of distribution (OOD) dataset



Time dependent OOD sample

## Conclusion

- ➔ More experiments and discussions are in the paper !
- ➔ Neural SDDDE almost consistently outperforms all other models.
- ➔ With the OOD and changing history function dataset, we show SDDDE's robustness.



Delay Diffusion OOD sample