TI-DeepONet: Learnable Time Integration for Stable Long-Term Extrapolation

Dibyajyoti Nayak

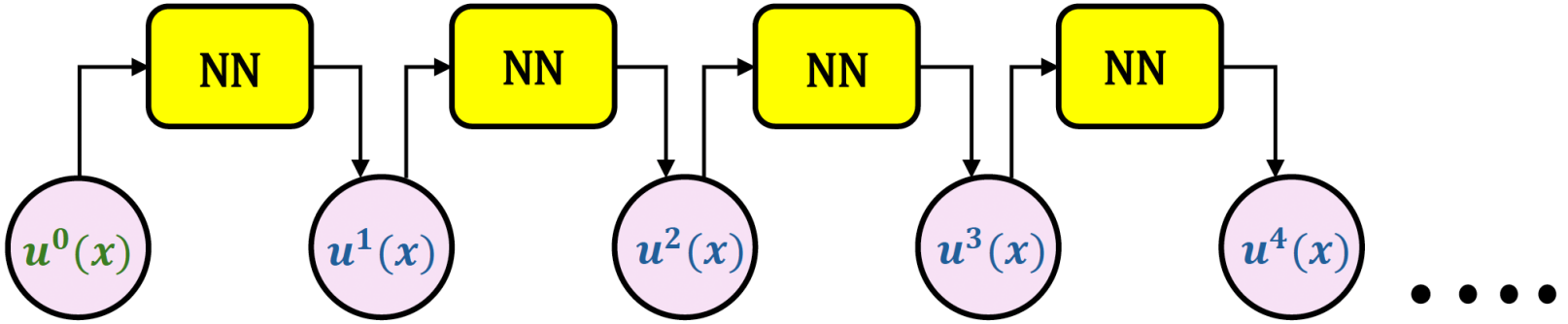
Advisor: Dr. Somdatta Goswami

Department of Civil and Systems Engineering

Johns Hopkins University

Introduction

- Operator learning frameworks inherently operate in two learning paradigms:
  - **Full rollout**: Represent entire spatiotemporal solution in terms of learnt basis functions and coefficients
  - **Autoregressive**: Recursive predictions in time

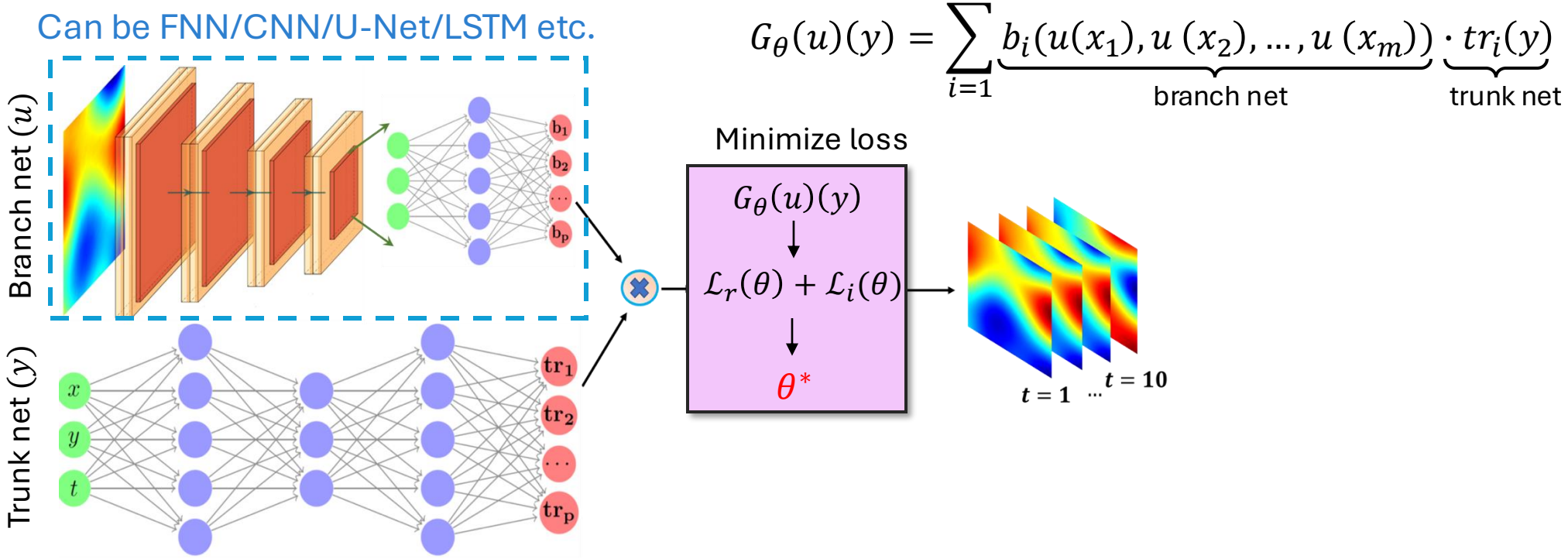


- Full rollout: Does not respect the underlying Markovian assumption, i.e., dependency between the solution states in time
- Autoregressive: Error accumulation in time
- **Idea**: To formulate a hybrid approach by combining operator learning with classical numerical schemes for accurate prediction of the future states of the system

Background – Deep Operator Network (DeepONet)

- Generalized Universal Approximation Theorem for Operator [Chen '95, Lu et al. '19]
- **Branch net**: Input $\{u(x_i)\}_{i=1}^m$, output: $\left[b_1, b_2, .., b_p\right]^T \in \mathbb{R}^p$
- **Trunk net**: Input y, output: $\left[t_1, t_2, .., t_p\right]^T \in \mathbb{R}^p$
- Input u is evaluated at the fixed locations $\{y_i\}_{i=1}^m$

$$G_\theta(u)(y) = \sum_{i=1}^{p} \underbrace{b_i(u(x_1), u(x_2), \ldots, u(x_m))}_{\text{branch net}} \cdot \underbrace{tr_i(y)}_{\text{trunk net}}$$



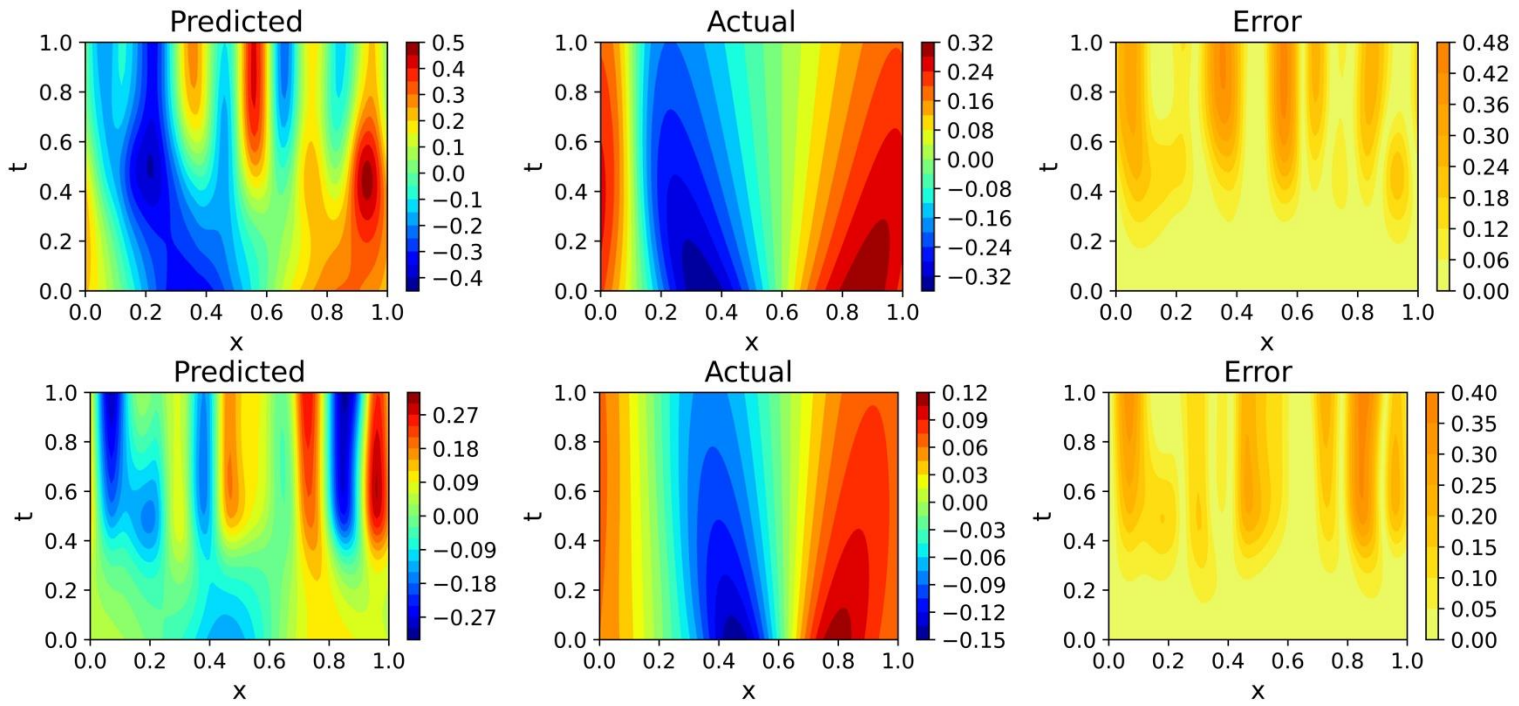
1D Burgers' Equation

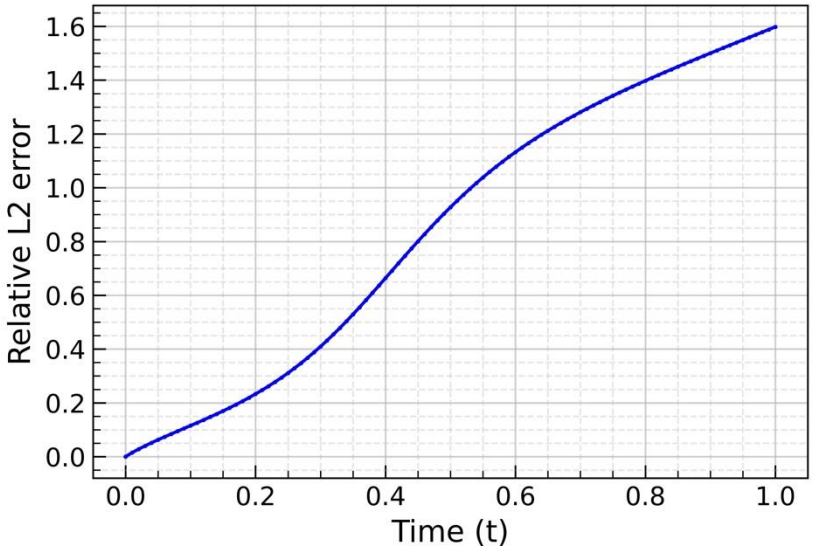
- Consider a **Burgers' PDE**:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

- **Initial condition**: $u(x, t = 0) = s(x)$
- **Boundary conditions**: $(1)\ u(x = 0, t) = u(x = 1, t)$, $(2)\ \frac{\partial u}{\partial x}(x = 0, t) = \frac{\partial u}{\partial x}(x = 1, t)$
- $\boldsymbol{N_s}$ $=$ 2500 samples of ICs, $\boldsymbol{N_t}$ $=$ 101 timesteps, $\boldsymbol{N_x}$ $=$ 101 spatial grid points
- Let $u^i(x)$ represent the solution at the i^{th} timestep
- **Branch Input**: $[u^0(x), u^1(x), u^2(x) \cdots\cdots\cdots, u^{49}(x)]$
- **Ground Truth Output**: $[u^1(x), u^2(x), u^3(x), \cdots\cdots\cdots, u^{50}(x)]$
- **Trunk Input**: $x \in [0, 1]$
- Goal is to assess autoregressive error accumulation when the DeepONet learns the mapping from $u^i(x)$ to $u^{i+1}(x)$

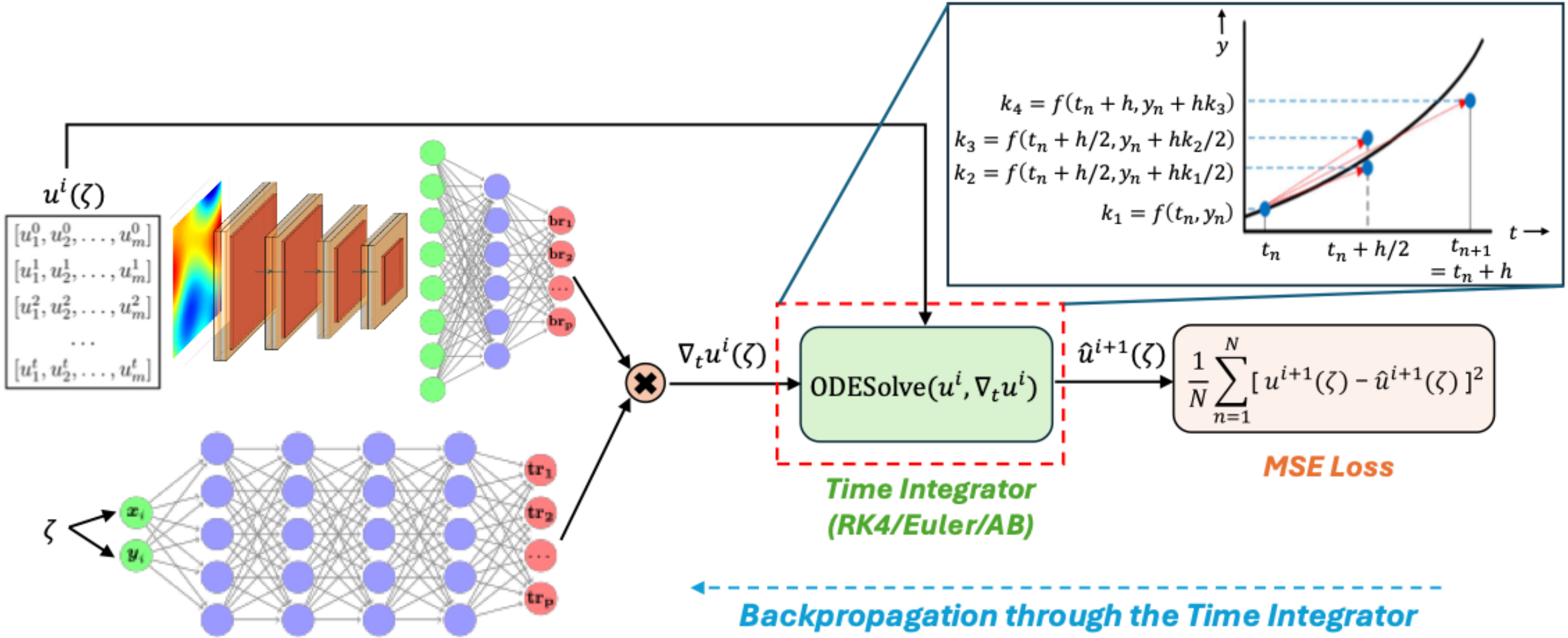
Autoregressive Errors





- Network trained with an Adam optimizer (Learning Rate = 0.001) until convergence
- Prediction done in a recursive manner starting from $u^0(x)$ $\implies$ predicted $u^{i+1}(x)$ becomes $u^i(x)$ for next timestep
- Relative L_2 error between predictions and ground truth grows quickly as we progress in time. Also, evident in the contours

Proposed Architecture: TI-DeepONet



TI(L)-DeepONet

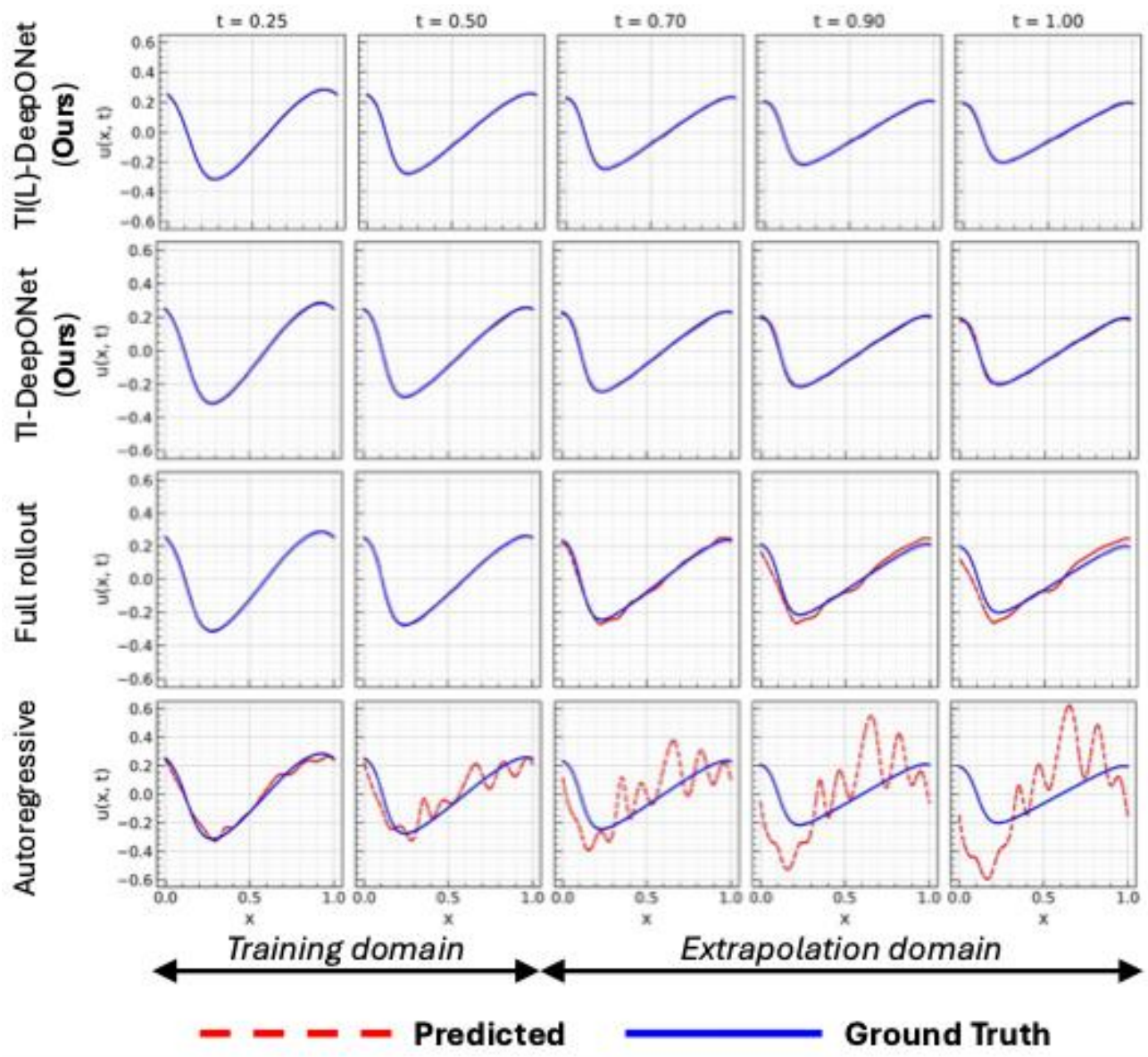
- In addition to the TI-DeepONet architecture, we also introduce an adaptive variant of this framework named **TI(L)-DeepONet**
- Instead of using fixed coefficients for the Runge-Kutta slopes, i.e., $\frac{1}{6}, \frac{2}{6}, \frac{2}{6}, \frac{1}{6}$ for k_1, k_2, k_3, k_4, we use a set of adaptive coefficients, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3, \alpha_4]$, which is a function of the current state u_i. The adaptive RK4 update then becomes:

$$u^{i+1} = u^i + \Delta t(\alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3 + \alpha_4 k_4)$$

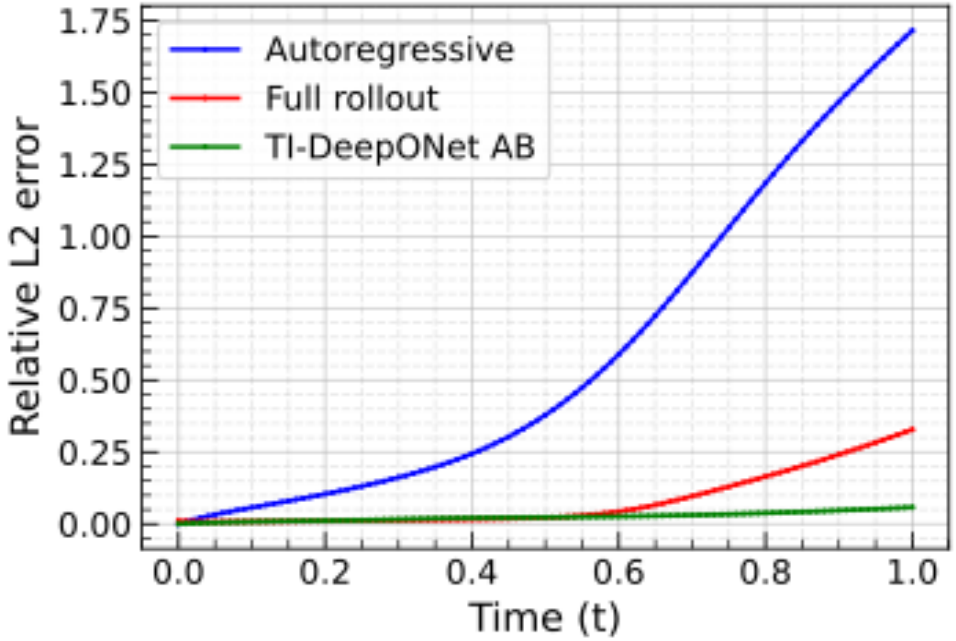
- The RK4 coefficients are learnt through another MLP: $\boldsymbol{\alpha} = NN(u^i; \theta)$
- We compare the extrapolation performance of the different frameworks for the following PDE systems:
  - 1D Burgers' Equation
  - 1D Korteweg-de Vries (KdV) Equation
  - 2D Burgers' Equation
- We use a relative L_2 error as an error metric for comparison: $L_2\ error = \frac{\|u_{pred} - u_{true}\|}{\|u_{true}\|}$

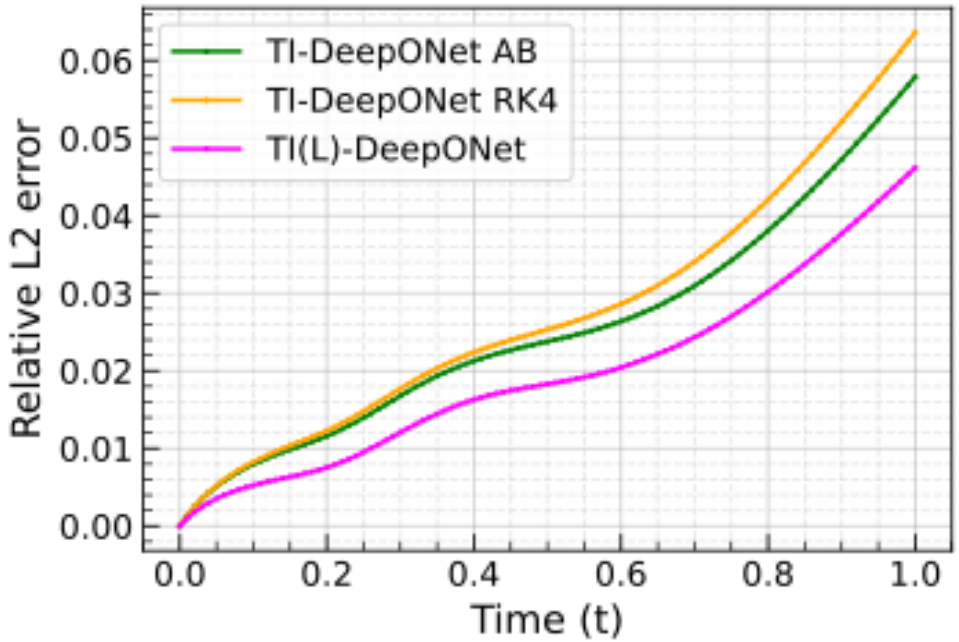
1D Burgers: Results (1/2)

- Training: $t \in [0, 0.5]$, Inference: $t \in [0, 1]$
- Prediction of the solution profiles by the four different frameworks: (1) TI(L)-DeepONet, (2) TI-DeepONet, (3) Full rollout, and (4) Autoregressive
- Autoregressive accumulates errors early on and quickly deviates from the actual profile
- Full rollout performs well in training domain but starts incurring errors upon entering the extrapolation domain
- Both TI-DeepONet and TI(L)-DeepONet maintain a stable extrapolation resulting in an accurate and reliable prediction of solution state at later timesteps
- TI(L)-DeepONet slightly performs better due to its adaptivity to the local solution



1D Burgers: Results (2/2)

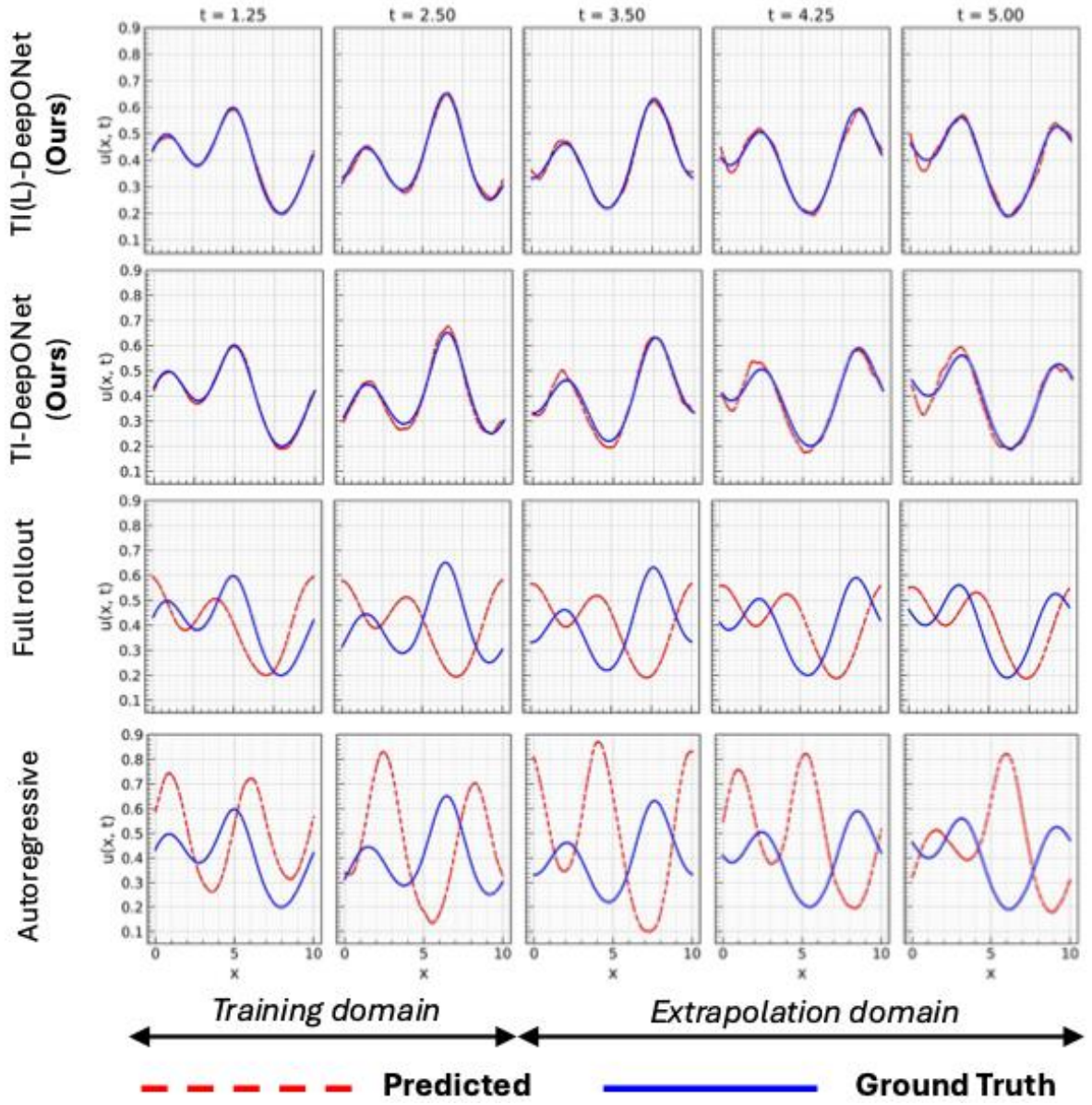




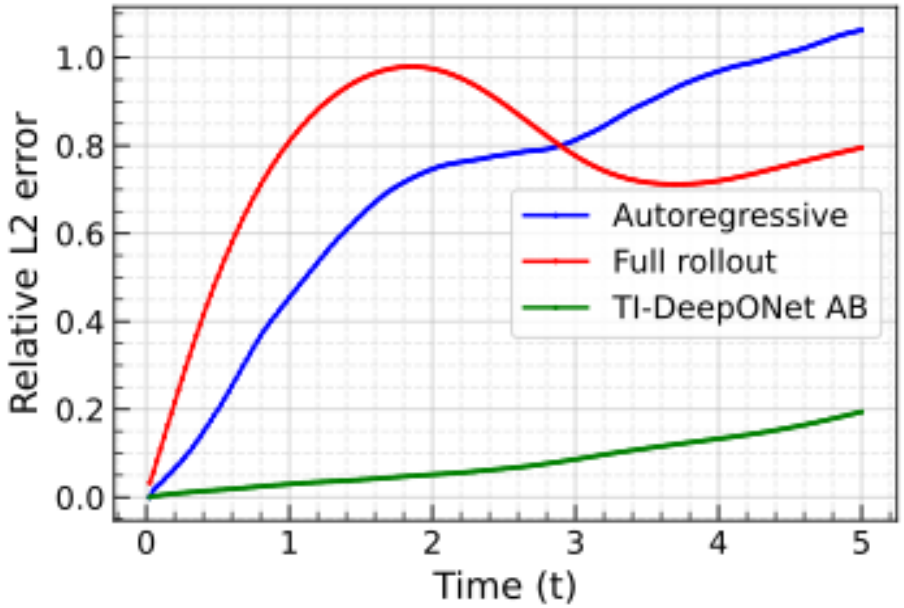
- Autoregressive: Rapid error accumulation and resembles exponential error growth beyond $t = 0.5$.
- Full rollout: Performs well until $t = 0.5$ and then starts incurring errors.
- TI-DeepONet + AB2/AM3 inference: Stable and controlled error growth especially in extrapolation domain
- With the time integrator frameworks: TI(L)-DeepONet performs best followed by TI-DeepONet AB and then TI-DeepONet RK4

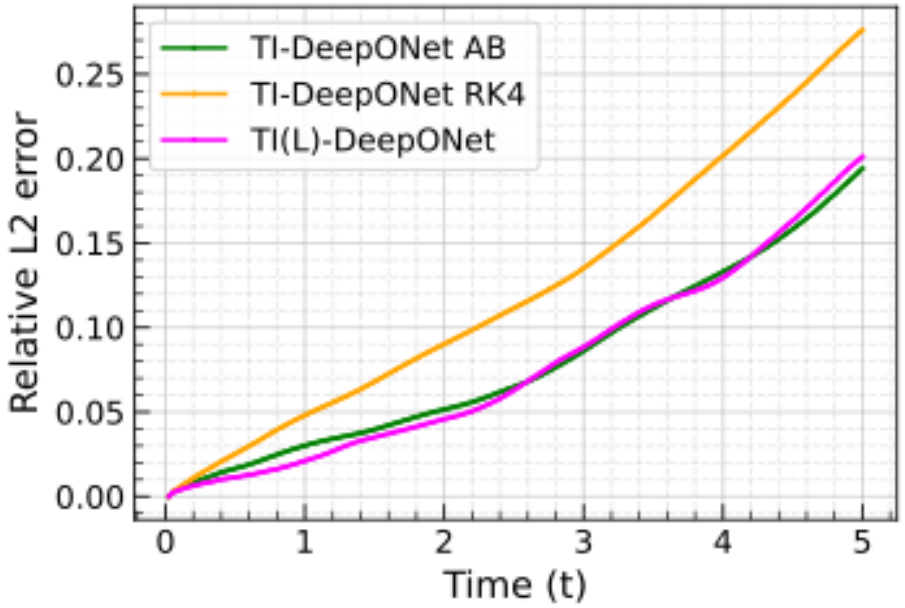
1D KdV: Results(1/2)

- Training: $t \in [0, 2.5]$, Inference: $t \in [0, 5]$
- Prediction of the solution profiles by the four different frameworks: (1) TI(L)-DeepONet, (2) TI-DeepONet, (3) Full rollout, and (4) Autoregressive
- Both autoregressive and full rollout DeepONet models fail to capture the dispersive, periodic dynamics of the solution
- Both TI-DeepONet and TI(L)-DeepONet maintain a stable extrapolation resulting in an accurate and reliable prediction of solution state at later timesteps
- TI(L)-DeepONet slightly performs better due to its adaptivity to the local solution, but the difference is minimal



1D KdV: Results (2/2)

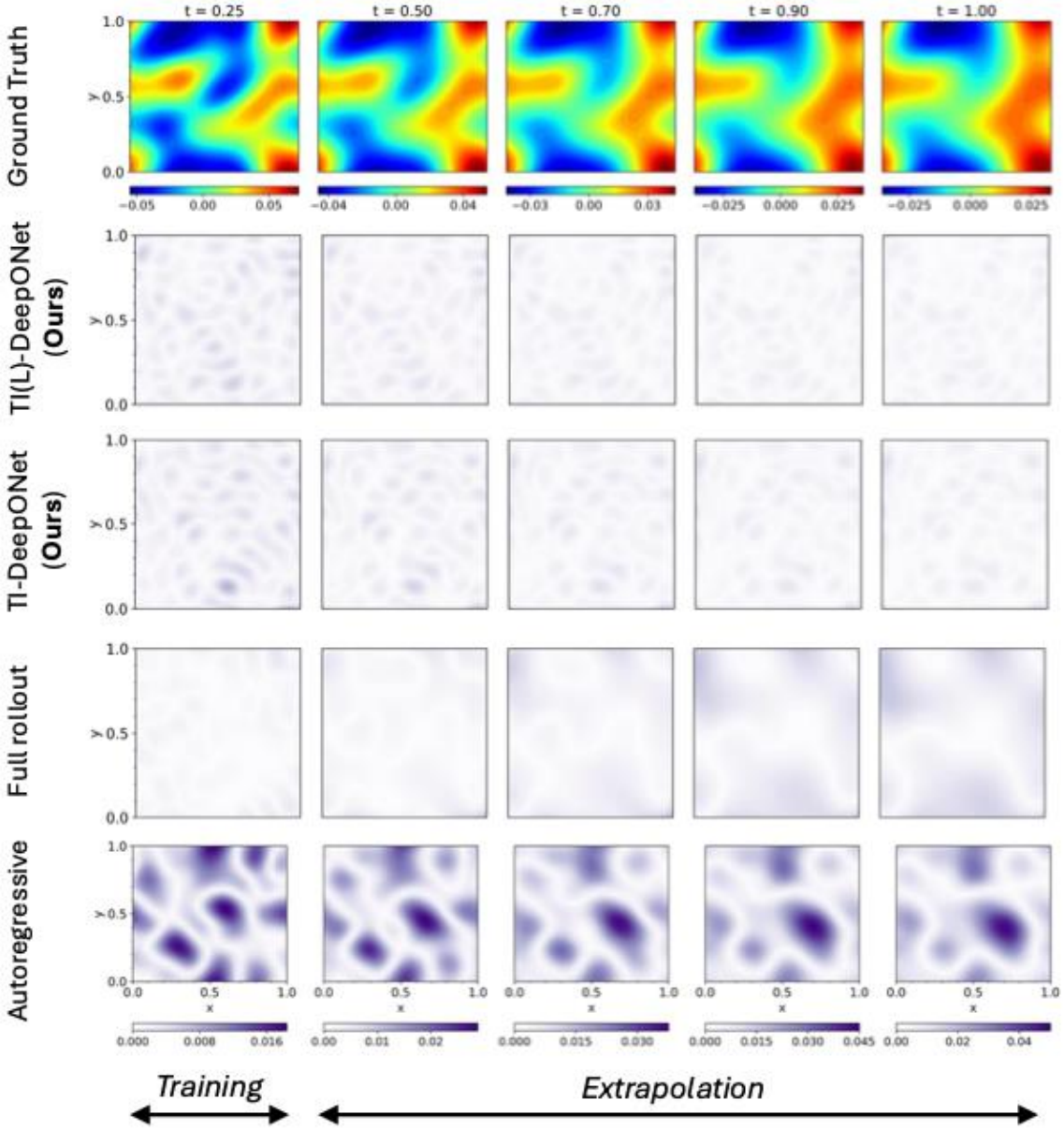




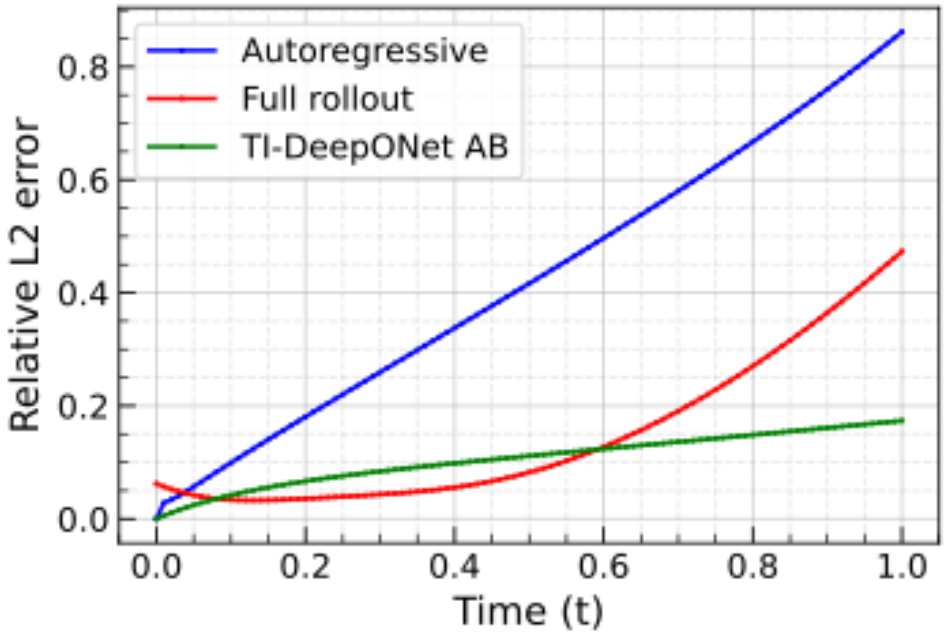
- Autoregressive: Rapid and monotonically increasing error
- Full rollout: Non-monotonic error trend. Error increases until $t = 1.75$ and then slightly falls only to increase again later
- TI-DeepONet AB: Stable and controlled error growth in the extrapolation domain
- Between the different TI-based frameworks: TI(L)-DeepONet and TI-DeepONet AB exhibit similar performance followed by TI-DeepONet with RK4 inference

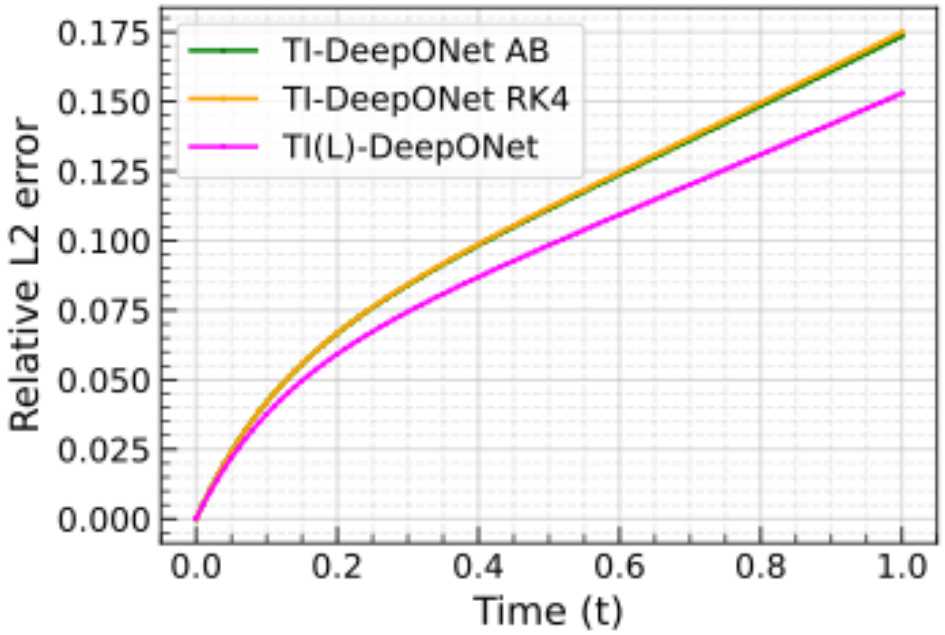
2D Burgers: Results (1/2)

- Training: $t \in [0, 0.33]$, Inference: $t \in [0, 1]$
- Prediction of the solution profiles by the four different frameworks: (1) TI(L)-DeepONet, (2) TI-DeepONet, (3) Full rollout, and (4) Autoregressive
- Autoregressive accumulates errors early on and quickly deviates from the actual profile
- Full rollout performs well up to $t = 0.5$ and then starts incurring errors in the longer time-steps
- Both TI-DeepONet and TI(L)-DeepONet incur relatively lower errors in the extrapolation regime with TI(L)-DeepONet performing slightly better due to its adaptivity to the local solution.



2D Burgers: Results (2/2)





- Autoregressive: Rapid error accumulation with a monotonically increasing trend
- Full rollout: Initially performs better but its fixed basis limits temporal generalization and errors surpass TI-DeepONet beyond $t = 0.5$.
- Overall, TI(L)-DeepONet yields the best performance with a final error at the last timestep to be 15%.

Summary

Problem	Method	Relative L_2 error			
		t+10Δt_e	t+20Δt_e	t+40Δt_e	T^*
Burgers' (1D)	TI(L)-DeepONet	**0.019±0.003**	**0.023±0.003**	**0.036±0.004**	**0.044±0.005**
	TI-DeepONet AB	0.031±0.004	0.037±0.005	0.057±0.008	0.070±0.011
	Full Rollout	0.043±0.002	0.095±0.004	0.247±0.028	0.336±0.053
	Autoregressive	0.710±0.089	1.004±0.144	1.556±0.206	1.768±0.227
KdV (1D)	TI(L)-DeepONet	**0.054±0.019**	**0.065±0.027**	**0.075±0.031**	**0.111±0.051**
	TI-DeepONet AB	0.086±0.026	0.108±0.034	0.129±0.043	0.183±0.063
	Full Rollout	0.776±0.0004	0.716±0.0005	0.719±0.0005	0.795±0.0007
	Autoregressive	0.823±0.073	0.886±0.064	0.922±0.069	0.968±0.083
Burgers' (2D)	TI(L)-DeepONet	**0.111±0.002**	**0.121±0.003**	**0.143±0.004**	**0.155±0.004**
	TI-DeepONet AB	0.121±0.002	0.133±0.002	0.157±0.003	0.169±0.003
	Full Rollout	0.131±0.007	0.194±0.014	0.357±0.035	0.453±0.049
	Autoregressive	0.503±0.017	0.590±0.024	0.783±0.052	0.894±0.075

Computational Costs

Problem	Method	Batch size	Training time (iter/sec)	N_{test}	Inference time (sec)
Burgers' (1D)	TI(L)-DeepONet (Ours)	256	163.62	2500	5.48
	TI-DeepONet AB (Ours)		195.43		5.86
	Full rollout		283.94		0.38
	Autoregressive		296.09		6.61
KdV (1D)	TI(L)-DeepONet (Ours)	256	179.82	1000	4.61
	TI-DeepONet AB (Ours)		208.59		4.47
	Full rollout		234.98		1.22
	Autoregressive		298.44		5.98
Burgers' (2D)	TI(L)-DeepONet (Ours)	128	32.83	1000	7.29
	TI-DeepONet AB (Ours)		37.42		6.98
	Full rollout		218.68		1.16
	Autoregressive		115.72		1.95

Conclusions

- Introduced TI-DeepONet and its adaptive variant TI(L)-DeepONet, which integrates classical numerical time integration with an operator learning framework (DeepONet).
- TI-DeepONet reduces relative L_2 extrapolation errors by 81% compared to autoregressive DeepONet and 70% compared to full-rollout methods, while maintaining stable predictions for temporal domains extending up to twice the training interval.
- The learnable coefficients in TI(L)-DeepONet further refine accuracy by adaptively weighting intermediate integration slopes.
- **Limitations:** Higher computational cost during training and inference due to multiple forward/backward passes through the ODE Solver


Thank You!
Questions?