KAN-ODEs: Kolmogorov-Arnold Network Ordinary Differential Equations for Learning Dynamical Systems and Hidden Physics

Presenter: Said Efendiyev

MODELING PHYSICAL SYSTEMS

Physical systems evolve over time and are often described by differential equations.

$$\frac{du}{dt} = g(u, t)$$

Examples:

- Fluid physics
- Population dynamics
- Chemical reactions
- Quantum systems

Goal: learn or discover the underlying dynamics from observations.

HOW DO WE MODEL DYNAMICAL SYSTEMS?

Traditional Approach

1. Derive governing equations
2. Estimate parameters from experiments
3. Solve equations numerically

Data-driven approach

Use machine learning to learn the dynamics directly from data

$$\frac{du}{dt} \approx f_\theta(u, t)$$

EXISTING ML APPROACHES

Neural ODEs

Neural network models the system dynamics

$$\frac{du}{dt} = NN(u, t)$$

Physics-Informed Neural Networks:

Use known physical equations as constraints during training

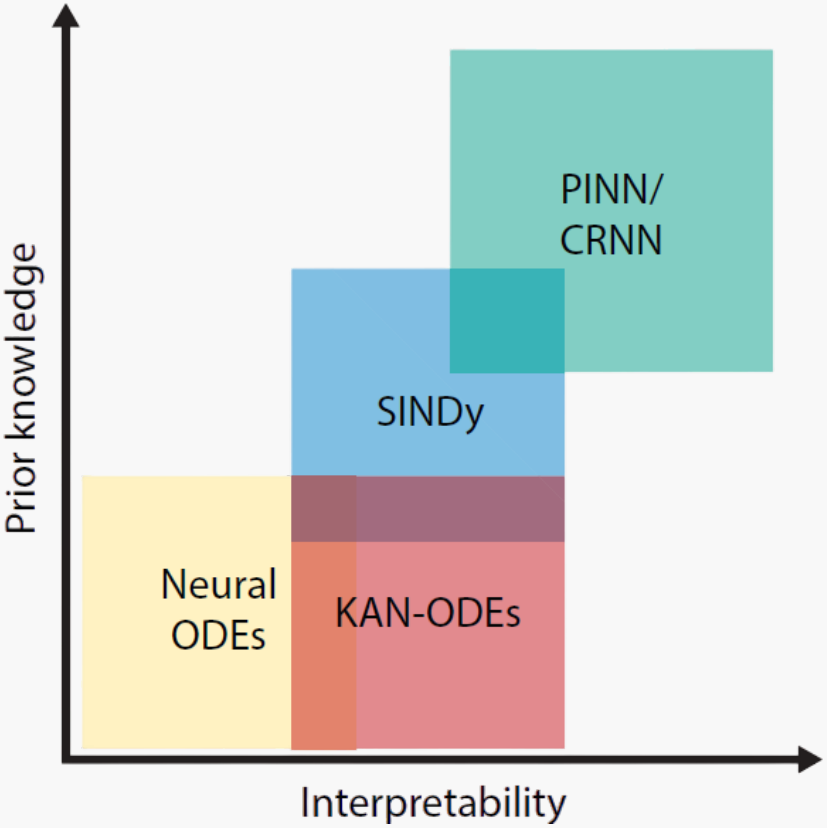
SINDy (Sparse Identification of Nonlinear Dynamics)

Discovers equations using sparse regression

THE INTERPRETABILITY VS FLEXIBILITY

Most methods lie on a trade-off curve.

Goal of this paper: Achieve interpretability without requiring prior physics knowledge



KOLMOGOROV–ARNOLD NETWORKS

Traditional NN:

$y = \sigma(Wx + b)$

KAN: Learn functions on edges instead of weights.

$$y = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}\left(x_p\right) \right)$$

Kolmogorov–Arnold representation theorem. Any multivariate function can be written as sums of univariate functions.

Benefits

- Fewer parameters
- Better scaling
- Improved interpretability

BASIS FUNCTIONS IN KAN

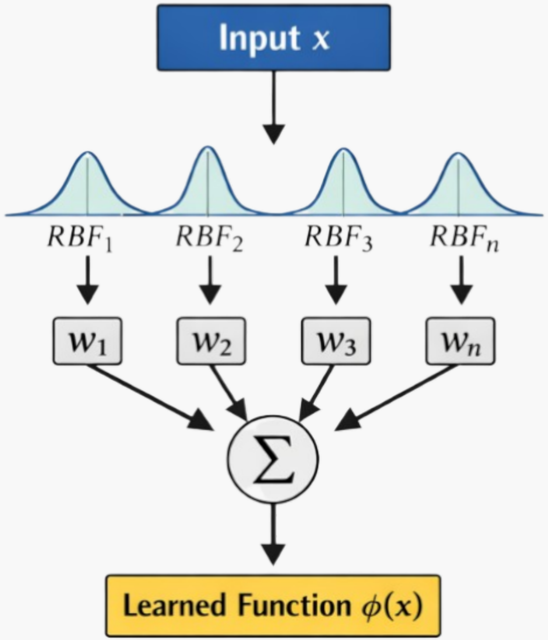
Each connection in KAN learns a function

$$\phi(x) = \sum_{i=1}^{N} w_i \psi\left(\|x - c_i\|\right) + w_i b(x)$$

Radial Basis Function (RBF)

$$\psi(r) = e^{\frac{-r^2}{2h^2}}$$

- grid of centers c_i
- combine RBFs to approximate functions
- similar to kernel methods



NEURAL ODEs

Neural ODEs learn system dynamics with neural networks

$$\frac{du}{dt} = NN(u, t)$$

Workflow

- Neural network predicts the time derivative
- ODE solver integrates the system
- Model is trained using observed trajectories

Advantages

- Continuous-time modeling
- Flexible dynamics
- Works with irregular time data

COMBINING KAN WITH NEURAL ODEs

Replace the neural network in Neural ODE with a Kolmogorov–Arnold Network

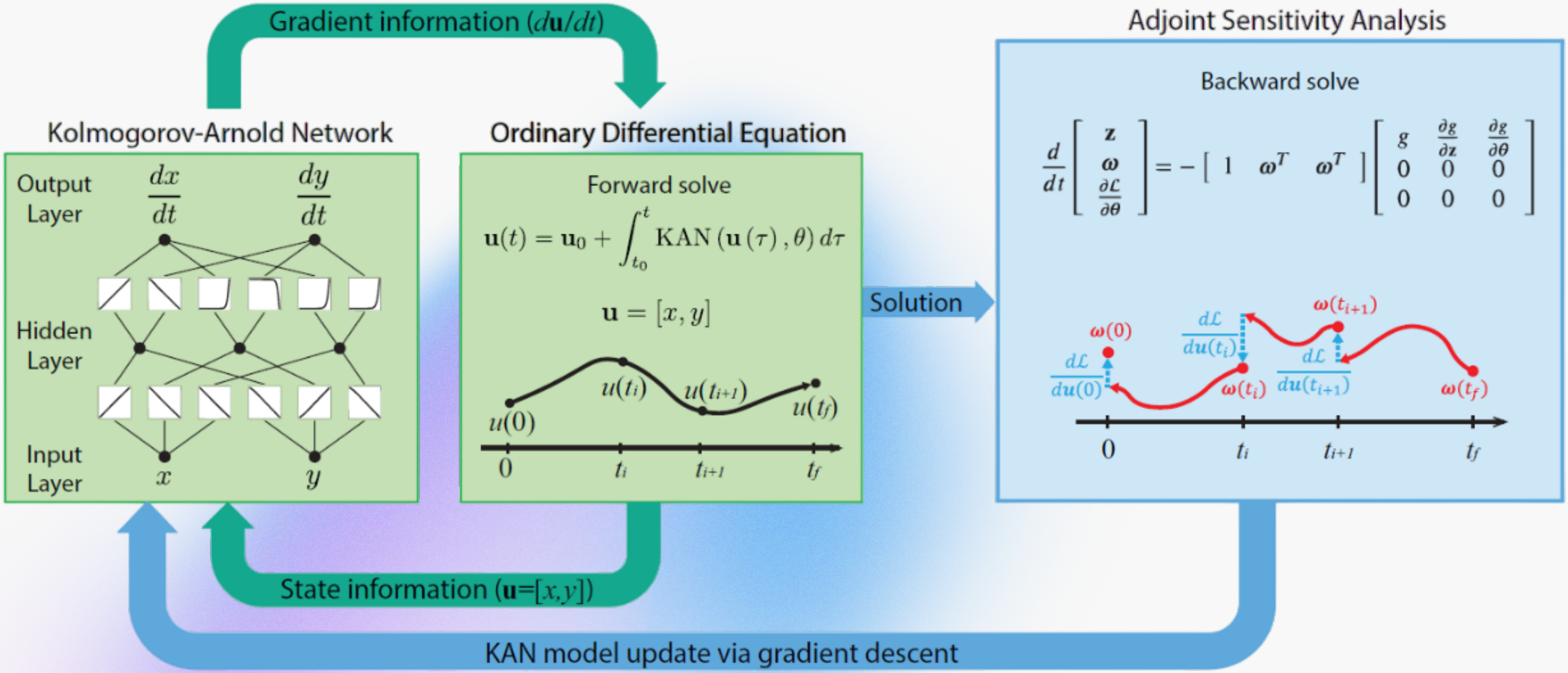
$$\frac{du}{dt} = KAN(u, t, \theta)$$

- KAN learns interpretable functional relationships
- ODE solver models continuous system evolution

Learn dynamical systems with higher accuracy, fewer parameters, and better interpretability

TRAINING KAN-ODE MODELS

KAN-ODEs: Kolmogorov-Arnold Network Ordinary Differential Equations



LOTKA–VOLTERRA SYSTEM

Lotka–Volterra equations

$$\frac{dx}{dt} = \alpha x - \beta x y$$

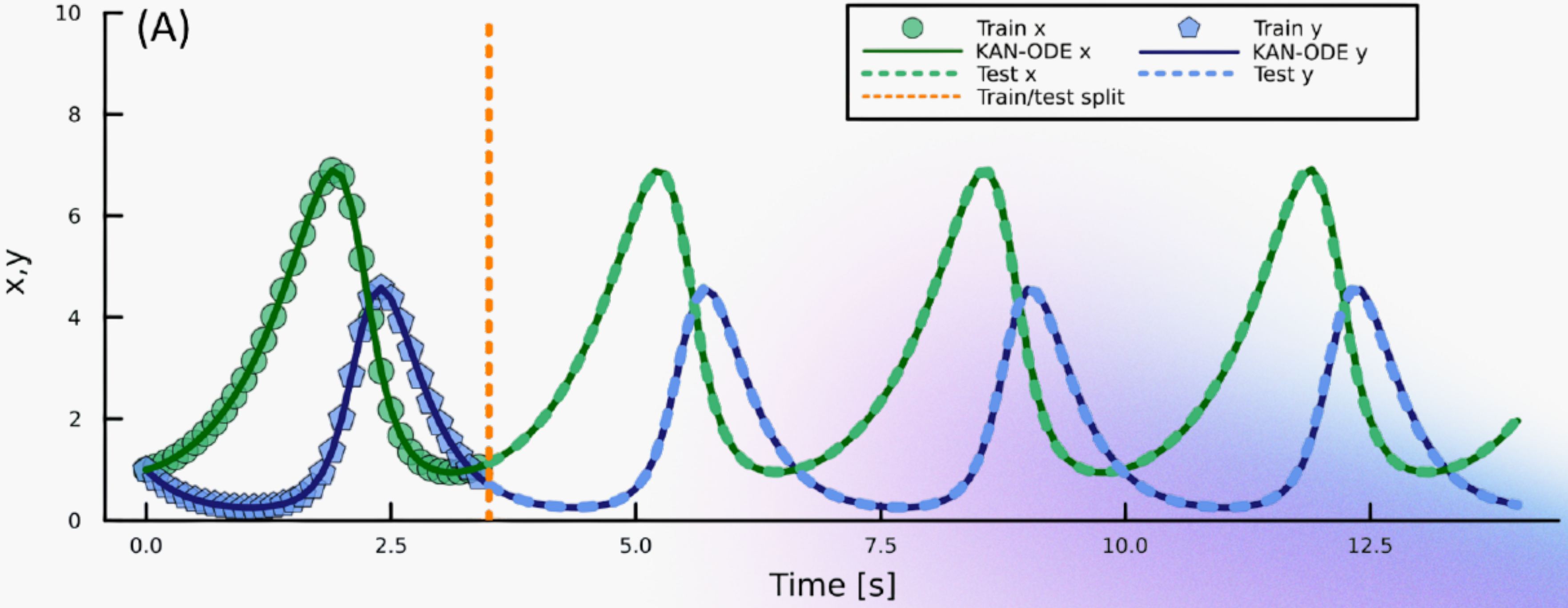
$$\frac{dy}{dt} = \gamma x y - \delta y$$

- Classic nonlinear dynamical system
- Models predator–prey interactions
- Commonly used benchmark for dynamical system learning

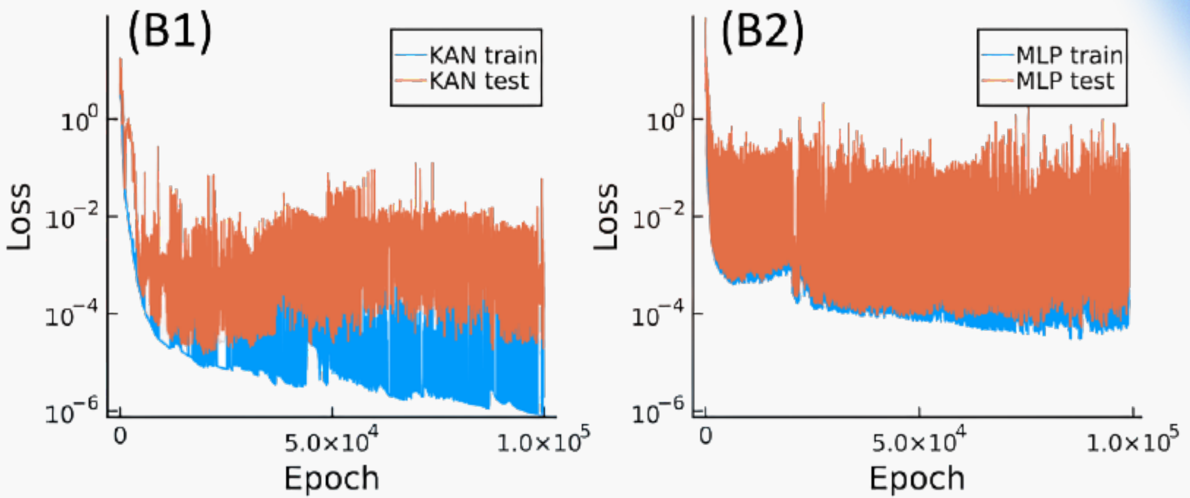
Setup

- Train model on early time data
- Test whether it predicts future dynamics

LOTKA–VOLTERRA SYSTEM



LOTKA–VOLTERRA SYSTEM



LOTKA–VOLTERRA SYSTEM

	Depth	Layer width	Grid size	Activation Function	No. Params	Train loss
Neural ODE (MLP)	2	10	N/a	tanh	52	4.7×10^{-4}
	2	**50**	**N/a**	**tanh**	**252**	$\mathbf{4.1 \times 10^{-5}}$
	2	100	N/a	tanh	502	1.6×10^{-5}
	3	3	N/a	tanh	29	2.0×10^{-4}
	3	5	N/a	tanh	57	2.6×10^{-4}
	3	8	N/a	tanh	114	4.6×10^{-5}
	3	10	N/a	tanh	162	3.7×10^{-5}
	3	20	N/a	tanh	522	3.0×10^{-5}
KAN-ODE	2	4	3	*learned*	64	1.4×10^{-4}
	2	4	4	*learned*	80	5.2×10^{-5}
	2	4	5	*learned*	96	1.2×10^{-4}
	2	6	4	*learned*	120	1.9×10^{-5}
	2	6	5	*learned*	144	1.6×10^{-5}
	2	**10**	**5**	***learned***	**240**	$\mathbf{8.3 \times 10^{-7}}$
	2	20	5	*learned*	480	6.6×10^{-7}
	2	40	5	*learned*	960	6.1×10^{-7}

LOTKA–VOLTERRA SYSTEM

Recovered equations using symbolic regression

$$\frac{dx}{dt} \approx 1.495x - 0.986xy$$

$$\frac{dy}{dt} \approx 0.970xy - 2.929y$$

Model can discover governing equations from data

LEARNING UNKNOWN TERMS IN A PDE

Fisher–KPP reaction–diffusion equation

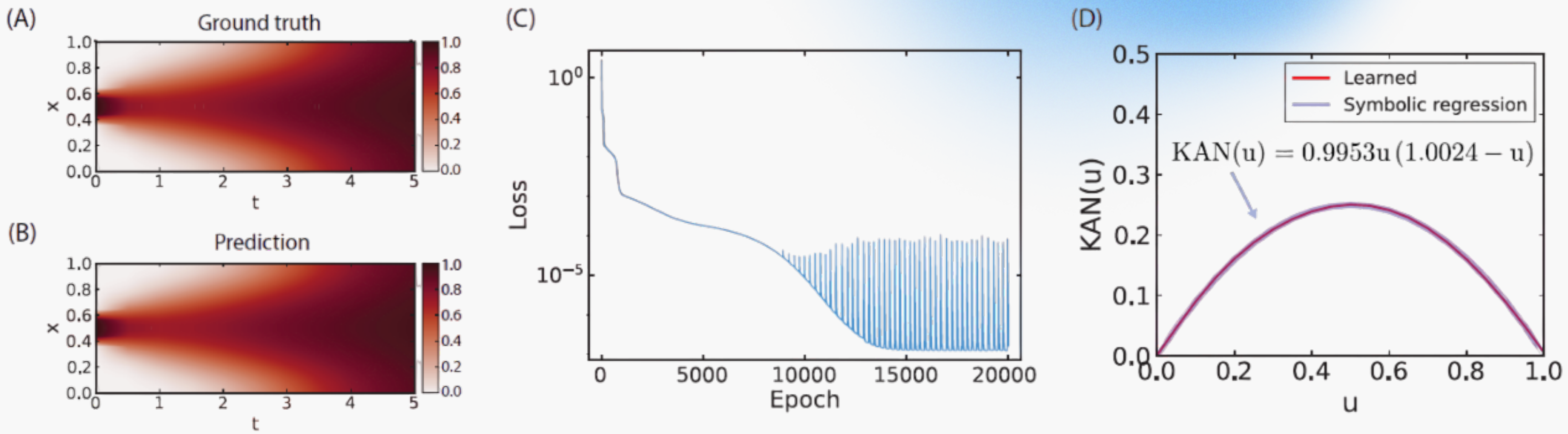
$$\frac{du}{dt} = D\frac{\partial^2 u}{\partial x^2} + ru(1-u)$$

Assumption

- Diffusion term known
- Reaction term unknown

Learn the unknown physics term from data

LEARNING UNKNOWN TERMS IN A PDE



$$KAN(u) \approx 0.995u(1.002 - u)$$

LEARNING SOLUTIONS FROM SPARSE DATA

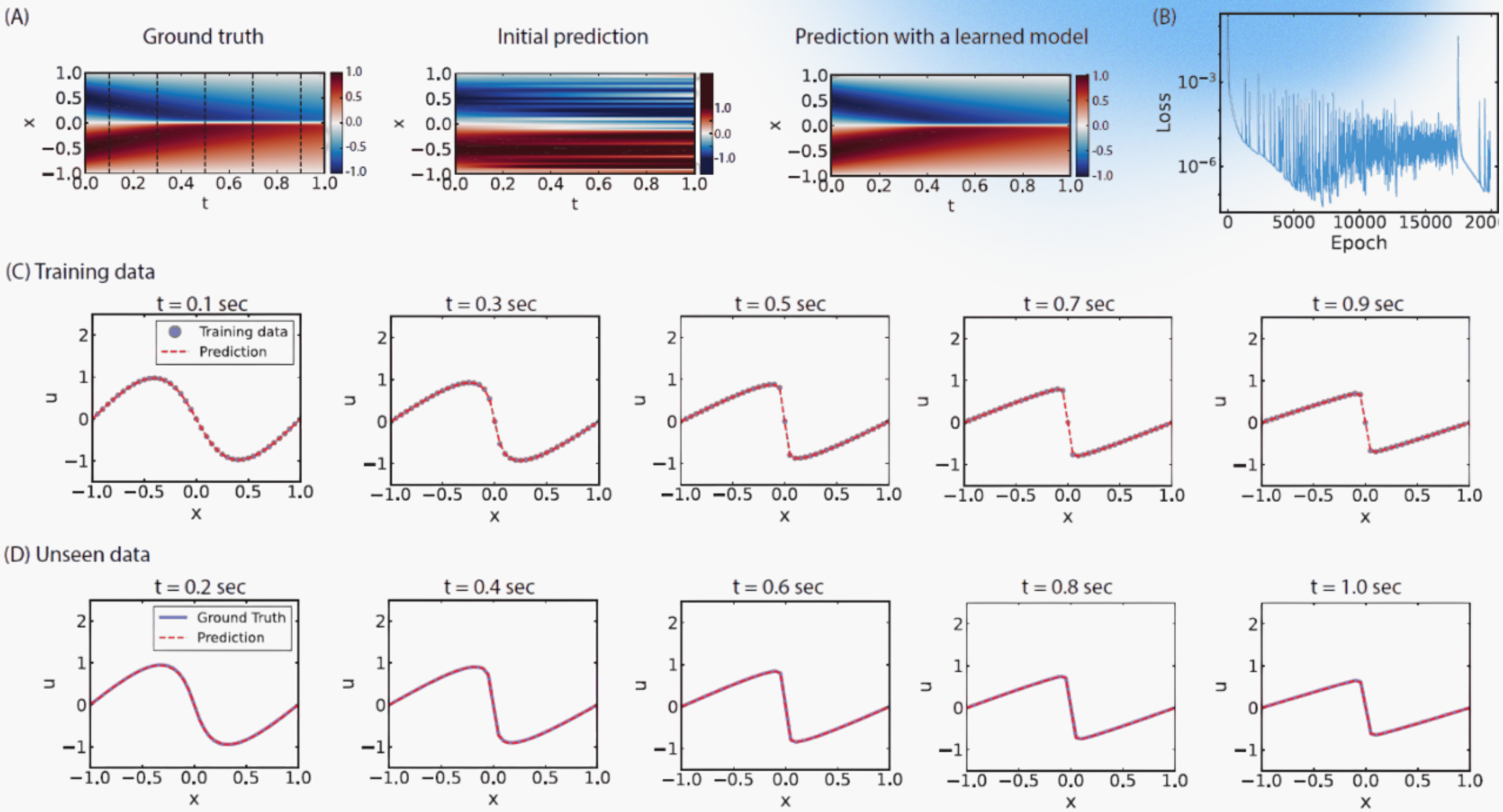
Burgers' Equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \frac{0.01}{\pi}\frac{\partial^2 u}{\partial x^2}$$

Setup

- Only 5 time snapshots for training
- Model must learn full spatiotemporal dynamics

LEARNING SOLUTIONS FROM SPARSE DATA



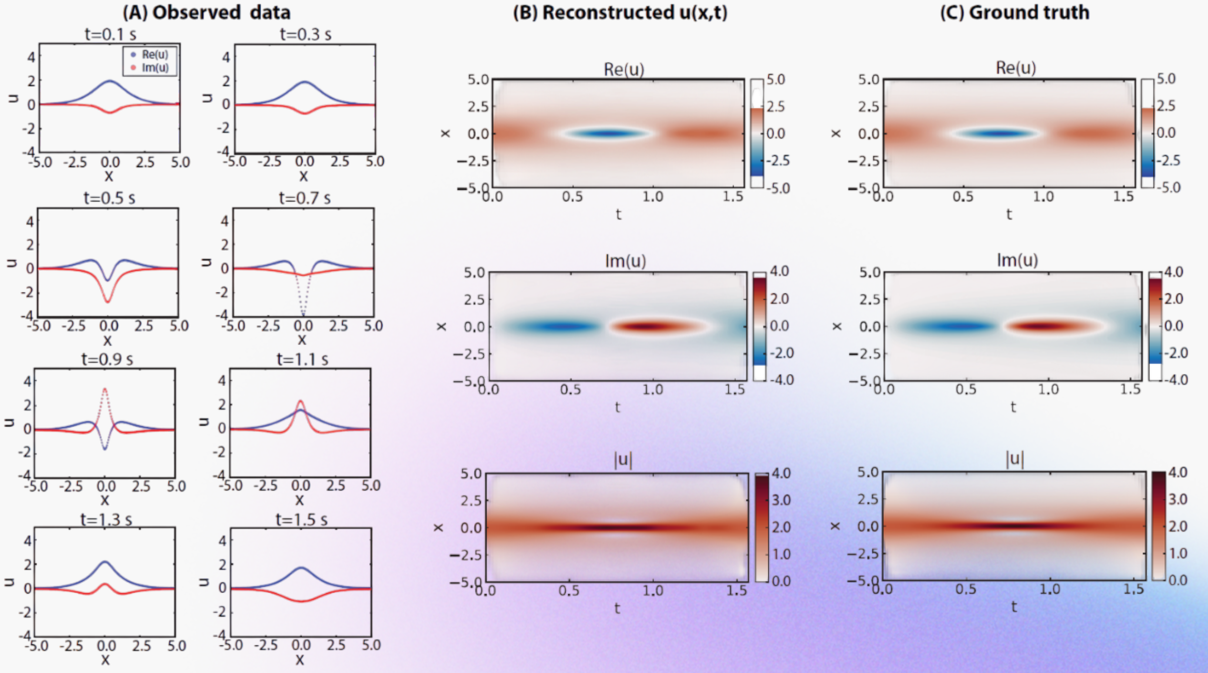
LEARNING QUANTUM WAVE DYNAMICS

Schrödinger equation

$$i\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial^2 u}{\partial x^2} + u|u|^2 = 0$$

Setup

- Only 8 time snapshots for training
- Model must learn full spatiotemporal dynamics



CONLUSION

Advantages

- Faster convergence with fewer parameters
- Better neural scaling behavior
- Interpretable learned functions

Capable of

- Learning dynamical systems
- Discovering hidden physics
- Modeling PDE dynamics from sparse data

Limitations

- Training can be computationally expensive
- Results shown mostly on synthetic benchmark systems
- Performance on very high-dimensional PDEs is unclear
- Symbolic recovery may not scale to complex systems

THANK YOU!

Do you have any questions?