Neural Parameter Regression for Explicit Representations of PDE Solution Operators

Konrad Mundinger[1,2], Max Zimmer[1] & Sebastian Pokutta[1,2]

[1]Department for AI in Society, Science, and Technology, Zuse Institute Berlin, Germany
[2]Institute of Mathematics, Technische Universität Berlin, Germany

Introduction and Preliminaries

Consider $\Omega \subset \mathbb{R}^d$ bounded with boundary $\partial\Omega$, and $T > 0$ fixed.

- Goal: Solve initial boundary value problems (**IBVPs**)

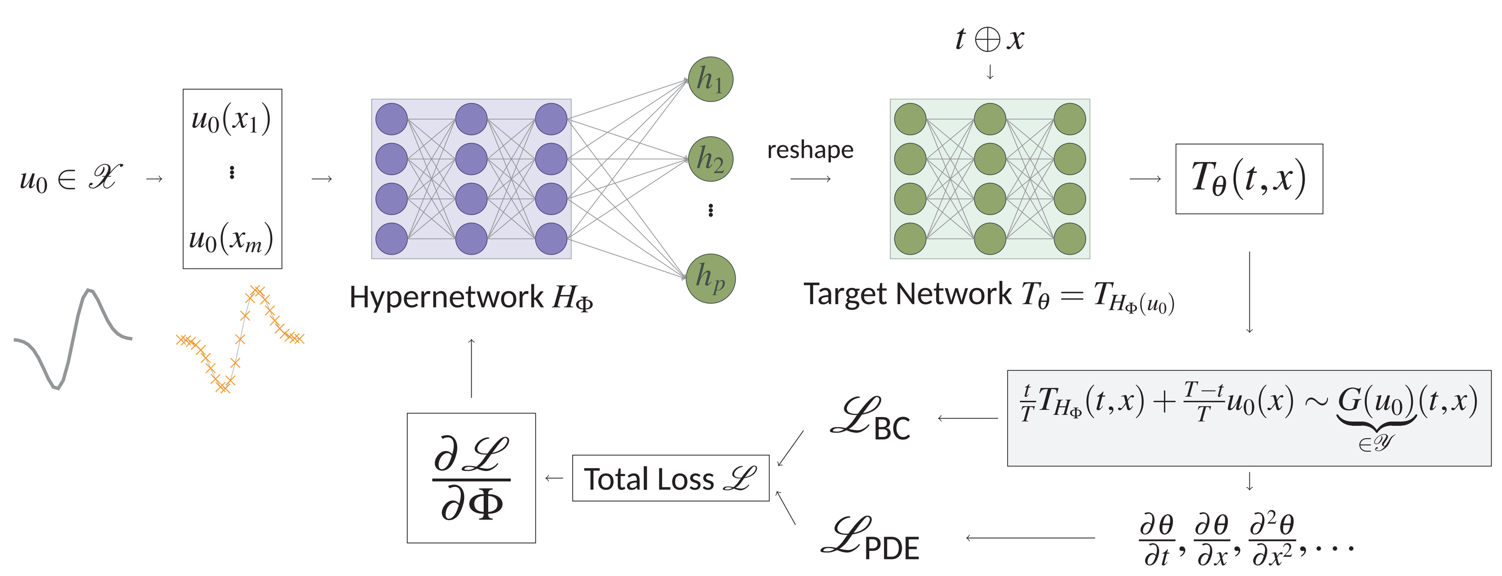
$$\partial_t u(t,x) = \mathscr{N}(u(t,x)),$$

  with initial condition $u(0,x) = u_0(x)$ and boundary condition $\mathscr{B}(u(t,x)) = 0$
- Objective: Approximate the **solution operator** G, i.e.,

$$G : \mathscr{X} \supset K \to \mathscr{Y},\; u_0 \mapsto \big((t,x) \mapsto u(t,x)\big), \qquad (1)$$

  between the infinite-dimensional Banach spaces $\mathscr{X}$ and $\mathscr{Y}$.
- Approaches:
  - **PINNs** incorporate the PDE residual and boundary conditions into a loss function to approximate solutions without training data.
  - (Physics-Informed) **DeepONets** extend PINNs to mappings between function spaces.
  - **Hypernetworks** generate weights for target networks based on input conditions.

Neural Parameter Regression



- **Explicit Parameterization:** $G(u_0)$ is parametrized by a NN for each u_0.
- **Low-rank** parametrization of the target network T_θ.
- **Data-free Training:** No need for training data.
- **Simpler Optimization:** Reparameterization of the output to enforce boundary conditions.
- **Flexibility:** Efficient adaptation to out-of-distribution examples.

Finetuning to out-of-distribution examples

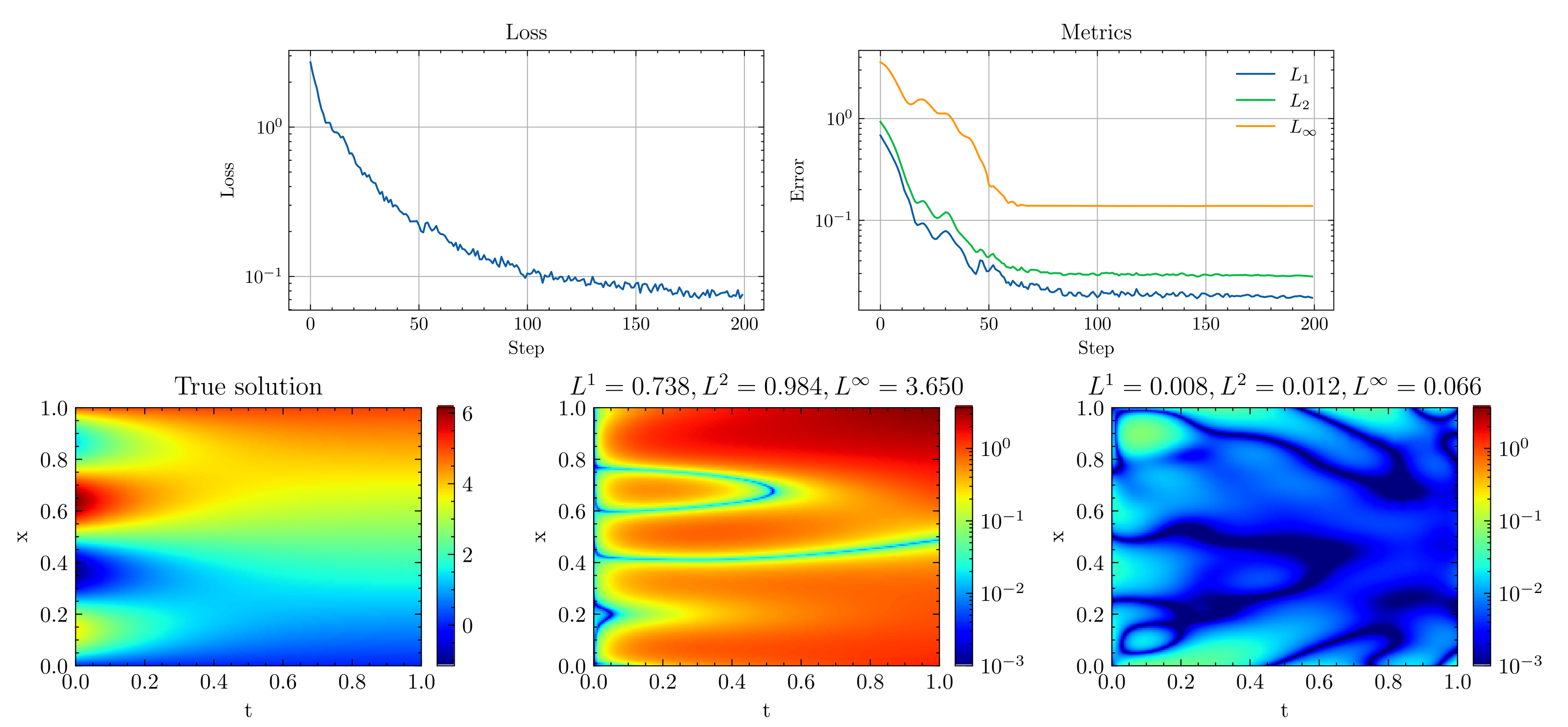


Figure 1: The heat equation with the out-of-distribution condition $u_0(x) = 5x + 3\sin(4\pi x)$. We plot the reference solution (left), the absolute difference to the reference solution before fine-tuning (middle) and after fine-tuning (right).

- **Finetuning step:** The model can quickly adapt to out-of-distribution (OOD) examples.
- For an OOD u_0, we compute the parameters of the target network from $H_\Phi(u_0)$.
- We **unfold** the target network by computing the low-rank products and make all parameters trainable.
- Using conventional PINN training, we finetune the weights T_θ in only 100-200 gradient steps (1-2 seconds on a CPU).

Linear Example: Heat Equation

- One-dimensional heat equation $\partial_t u = \kappa \partial_{xx}$ with constant Dirichlet boundary conditions on $\Omega = [0,1]$ and $T = 1$.
- Initial conditions are parametrized as Fourier polynomials.

$$f(x) = a_0 + \sum_{i=1}^{n} a_n \sin(2\pi n x) + b_n \cos(2\pi n x)$$

  with $\max_{i=1,\ldots,n} \max\{|a_i|, |b_i|\} \leq c$. In our experiments, $n = 3$ and $c = 2$.

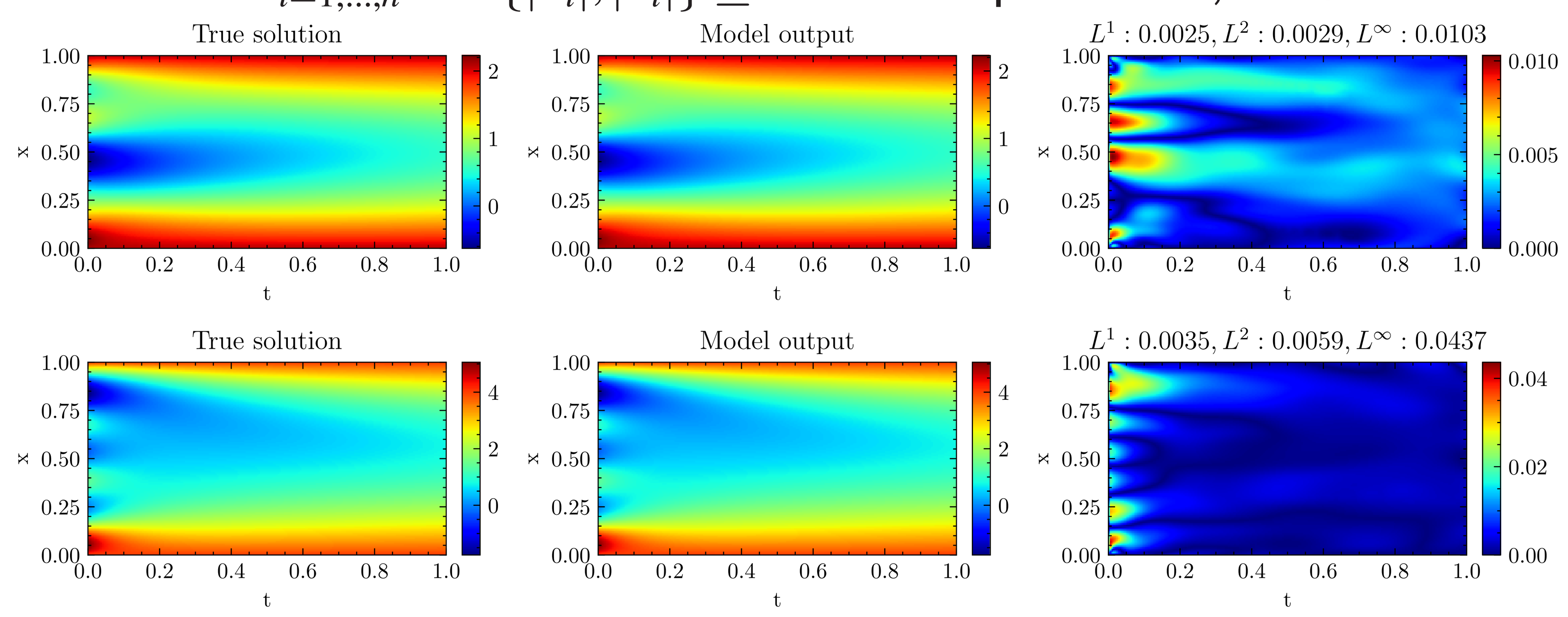


Figure 2: Results for the heat equation: The first and second row correspond to $u_0(x) = 0.5\sin(4\pi x) + \cos(2\pi x) + 0.3\cos(6\pi x) + 0.8$ and the $u_0(x) = \sum_{n=1}^{3}(\sin(n\pi x) + \cos(n\pi x)) + 1$, respectively. The columns shows the reference solutions, the model predictions and the absolute differences, respectively.

Nonlinear Example: Burgers Equation

- One-dimensional (inviscid) Burgers equation $\partial_t u = -u\partial_x u$ on $\Omega = [0,1]$ and $T = 1$ with a constant Dirichlet boundary condition at $x = 0$.
- Initial conditions are parametrized as affine functions $u_0(x) = ax + b$ with $a \in [-1,0]$ and $b \in [1,2]$.

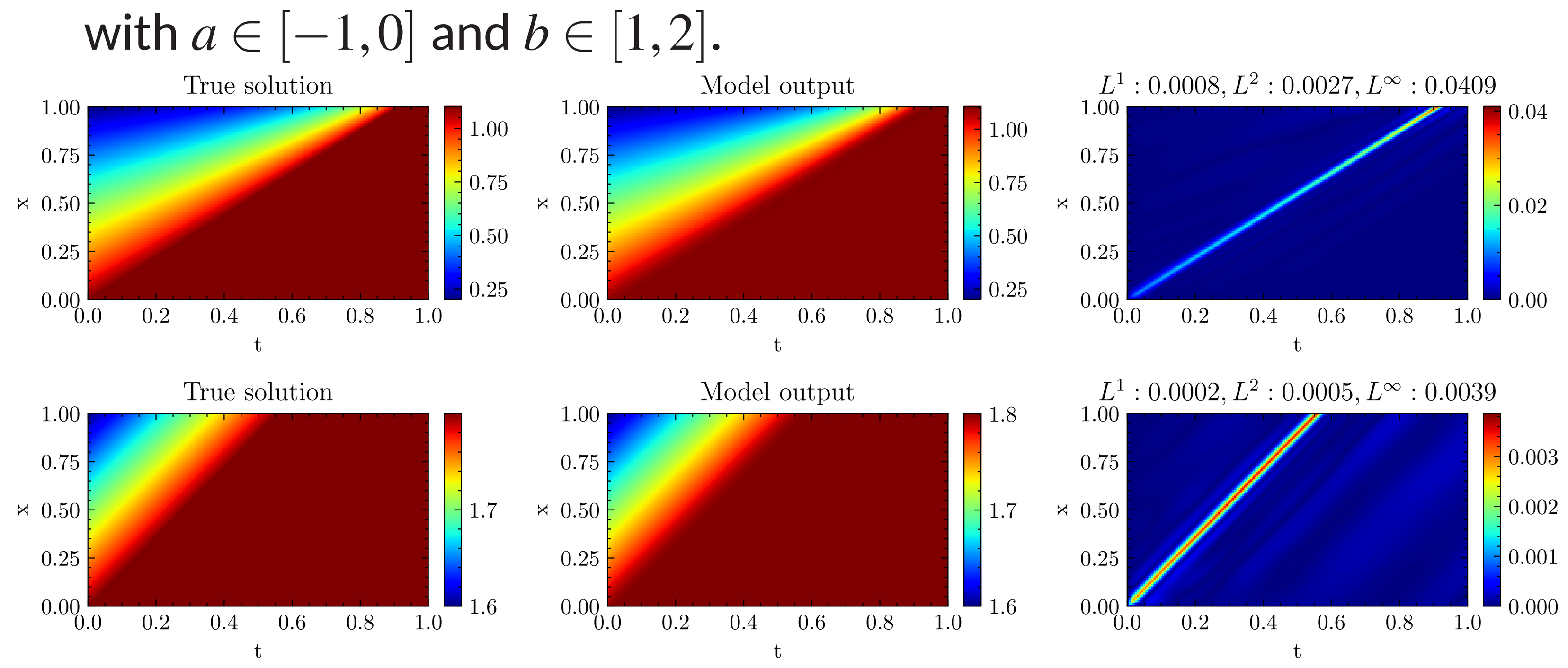


Figure 3: The first row shows the results for the initial condition $u_0(x) = -0.9x + 1.1$ and the second row for $u_0(x) = -0.2x + 1.8$

Results

Equation	Metric	Hidden dim 32			Hidden dim 64			DeepONet
		Rank 4	Rank 8	Rank 16	Rank 4	Rank 8	Rank 16	
Heat	L^1	0.0037	0.0028	0.0037	0.0036	0.0026	**0.0021**	0.0026
	L^2	0.0051	0.0037	0.0047	0.0046	0.0031	**0.0030**	0.0036
	L^∞	0.0311	0.0171	0.0165	0.0166	0.0268	**0.0159**	0.0349
	# Target	993	1761	3297	1985	3521	6593	4320
	# Hyper	79137	129057	228897	143617	243457	443137	16672
	Training Time	62 min	65 min	69 min	67 min	71 min	76 min	47 min
Burgers	L^1	0.0007	0.0006	**0.0004**	0.0005	0.0006	0.0005	0.0011
	L^2	0.0023	0.0022	**0.0014**	0.0016	0.0019	0.0017	0.0030
	L^∞	0.0282	0.0276	**0.0206**	0.0218	0.0238	0.0223	0.0328
	# Target	993	1761	3297	1985	3521	6593	14752
	# Hyper	79137	129057	228897	143617	243457	443137	57888
	Training Time	16 min	17 min	20 min	17 min	20 min	24 min	14 min

Future Work

- Rigorous analysis of the **approximation-theoretic** properties of the proposed method.
- Application to **higher-dimensional** PDEs and **more complex** boundary conditions.
- Extension to **different ansatz spaces** than multilayer perceptrons with low-rank weights.