In-Context Operator Networks (ICON): Towards Large Scientific Learning Models

Liu Yang, Siting Liu, Tingwei Meng, Stanley J. Osher

"In-context operator learning with data prompts for differential equation problems." PNAS, 120.39 (2023): e2310142120.
GitHub: https://github.com/LiuYangMage/in-context-operator-networks
Motivation

• Existing neural-network-based methods for solving differential equations are limited by **equation specificity**.
  • e.g., PINNs, FNO, DeepONets...
  • Need **frequent retraining** when switching to new problems.

• Inspired by Large Language Models, we propose In-Context Operator Networks (ICON).
  • Use a **single** neural network to solve a wide range of scientific machine learning tasks.
  • Get rid of retraining (even fine-tuning) the neural network.
  • Leverage commonalities shared across various tasks, so that only a few examples are needed when learning a new operator.

• We need models that can adapt to new physical systems and tasks, just as a human would.
Operator: mapping from condition to QoI, both are functions.

Examples: condition-QoI pairs associated with the same unknown operator.

Training: ICON is trained to be an "operator learner", instead of an "operator approximator". 
- Input: prompted examples and the question condition.
- The model learns the operator from the examples and apply to the question condition.
- Output: prediction of the question QoI function, evaluated at "queries".

Inference: learn and apply the new unknown operator, without weight updates.
<table>
<thead>
<tr>
<th>#</th>
<th>Problem description</th>
<th>Differential equations</th>
<th>Parameters</th>
<th>Conditions</th>
<th>Qols</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Forward problem of ODE 1</td>
<td>$\frac{d}{dt} u(t) = a_1 c(t) + a_2$</td>
<td>$a_1, a_2$</td>
<td>$u(0), c(t), t \in [0, 1]$</td>
<td>$u(t), t \in [0, 1]$</td>
</tr>
<tr>
<td>2</td>
<td>Inverse problem of ODE 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Forward problem of ODE 2</td>
<td>$\frac{d}{dt} u(t) = a_1 c(t) u(t) + a_2$</td>
<td>$a_1, a_2$</td>
<td>$u(0), c(t), t \in [0, 1]$</td>
<td>$u(t), t \in [0, 1]$</td>
</tr>
<tr>
<td>4</td>
<td>Inverse problem of ODE 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Forward problem of ODE 3</td>
<td>$\frac{d}{dt} u(t) = a_1 c(t) + a_2 c(t) + a_3$</td>
<td>$a_1, a_2, a_3$</td>
<td>$u(0), c(t), t \in [0, 1]$</td>
<td>$u(t), t \in [0, 1]$</td>
</tr>
<tr>
<td>6</td>
<td>Inverse problem of ODE 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Forward damped oscillator</td>
<td>$u(t) = A \sin(\frac{2\pi}{T} t + \eta) e^{-kt}$</td>
<td>$k$</td>
<td>$u(t), t \in [0, 0.5]$</td>
<td>$u(t), t \in [0.5, 1]$</td>
</tr>
<tr>
<td>8</td>
<td>Inverse damped oscillator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Forward Poisson equation</td>
<td>$\frac{\partial^2}{\partial x^2} u(x) = c(x)$</td>
<td>$u(0), u(1)$</td>
<td>$c(x), x \in [0, 1]$</td>
<td>$u(x), x \in [0, 1]$</td>
</tr>
<tr>
<td>10</td>
<td>Inverse Poisson equation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Forward linear reaction–diffusion</td>
<td>$-\lambda a \frac{\partial^2}{\partial x^2} u(x) + k u(x) = c$</td>
<td>$u(0), u(1), a, c$</td>
<td>$k(x), x \in [0, 1]$</td>
<td>$u(x), x \in [0, 1]$</td>
</tr>
<tr>
<td>12</td>
<td>Inverse linear reaction–diffusion</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Forward nonlinear reaction–diffusion</td>
<td>$-\lambda a \frac{\partial^2}{\partial x^2} u(x) + k u(x)^3 = c(x)$</td>
<td>$c(x), x \in [0, 1]$</td>
<td>$u(x), x \in [0, 1]$</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Inverse nonlinear reaction–diffusion</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>MFC $g$-parameter 1D → 1D</td>
<td>$\inf_{\rho, m} \int \int \frac{m^2}{2\rho} dx dt + \int g(x) \rho(1, x) dx$</td>
<td>$g(x), x \in [0, 1]$</td>
<td>$\rho(t = 0, x), x \in [0, 1]$</td>
<td>$\rho(t = 1, x), x \in [0, 1]$</td>
</tr>
<tr>
<td>16</td>
<td>MFC $g$-parameter 1D → 2D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>MFC $g$-parameter 2D → 2D</td>
<td>$\rho(t) \mu x + \nabla X \cdot m(t, x) = \mu \Delta x \rho(t, x)$</td>
<td>$\rho(t, x), x \in [0, 1]$</td>
<td>$\rho(t, x), t \in [0.5, 1], x \in [0, 1]$</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>MFC $\rho_0$-parameter 1D → 1D</td>
<td>$c = 20, \mu = 0.02$,</td>
<td>$\rho(t = 0, x)$</td>
<td>$g(x), x \in [0, 1]$</td>
<td>$\rho(t = 1, x), x \in [0, 1]$</td>
</tr>
<tr>
<td>19</td>
<td>MFC $\rho_0$-parameter 1D → 2D</td>
<td>Periodic spatial boundary condition</td>
<td>$x \in [0, 1]$</td>
<td>$\rho(t, x), t \in [0.5, 1], x \in [0, 1]$</td>
<td></td>
</tr>
</tbody>
</table>

List of the problems, including forward and inverse ODE, PDE, and mean-field control problems, solved with a single neural network, with about 30M parameters.
A Glance of ICON for ODE and PDE Problems

Colored dotted lines: condition and QoI functions in examples.
Grey dots: data of the examples used in the prompts.
Blue dots: data in the question conditions.
Red dots: prediction of the question QoI.
Solid black lines: ground truth. Note the consistency between prediction and ground truth.
Testing on In-Distribution Operators

Average relative testing errors for all 19 problems listed in the table. The error decreases with an increasing number of examples in the prompt. With only five examples, the relative error goes down to about 1%-2% for most cases.
Three examples and the question share the same terminal cost as the unknown parameter in the operator.

\[
\inf_{\rho, m} \int \int \frac{m^2}{2\rho} dx dt + \int g(x) \rho(1, x) dx
\]
\[
\text{s.t. } \partial_t \rho(t, x) + \nabla_x \cdot m(t, x) = \mu \Delta_x \rho(t, x)
\]
for \( t \in [0, 1], x \in [0, 1], \)
\( c = 20, \mu = 0.02, \)
periodic spatial boundary condition

Three examples and the question share the same terminal cost as the unknown parameter in the operator.

Plots: density field in temporal-spatial domain.

Blue dots: data for example condition (density at time from 0 to 0.5).
Red dots: data for example QoI (density at time from 0.5 to 1.0).
Black dots: data for question condition.

We make the prediction on \( \rho(t, x), (t, x) \in [0.5, 1] \times [0, 1] \)
Still the same problem (mean-field control with terminal cost as the unknown parameters).

As we increase the number of data points in each condition/QoI function, the error decreases and finally converges below 1%.

ICON is trained using 41 to 50 data points in each function, represented by the narrow red region.
Testing on Out-of-Distribution Operators

Here the coordinates are the operator parameters.

**ICON** demonstrated accurate prediction capabilities even with operator parameters extending beyond the training region.

**Black rectangle:** training region for operator parameters.

**ODE 3:**

\[ \frac{d}{dt} u(t) = a_1 u(t) + a_2 c(t) + a_3 \]

Linear reaction-diffusion PDE:

\[-\lambda a \frac{d^2}{dx^2} u(x) + k(x)u(x) = c \]

for \( x \in [0, 1], \lambda = 0.05 \)
Generalization to Equations of New Forms (New ODE)

ODE 2 (in training):
$$\frac{d}{dt} u(t) = a_1 c(t) u(t) + a_2$$

ODE 3 (in training):
$$\frac{d}{dt} u(t) = a_1 u(t) + a_2 c(t) + a_3$$

New ODE (not in training):
$$\frac{d}{dt} u(t) = a_1 u(t) c(t) + bu(t) + a_2$$

The error shows a **decreasing trend** as the training dataset becomes larger and more diversified. This is preliminary evidence of learning operators for equations of new forms that were never seen in training data.
Multi-Modal In-Context Operator Learning

The above results come from our first paper [1], where the model is a transformer-based encoder-decoder. In our second paper [2], we proposed "ICON-LM", a language-model-like architecture.
- Simplified, only one transformer.
- Fewer parameters with higher accuracy.
- Multi-modal: apart from data examples, optionally take as input the "captions" that integrate human knowledge about the operator, expressed through natural language descriptions and equations.

Key ideas of the improved ICON-LM model:

- Inspired by "next token prediction" in large language models, ICON-LM utilizes the input condition-QoI pairs in an autoregressive way.

- A special attention mask tailored for operator learning, which keeps permutation invariance for tokens in the same function.
Vague caption (without numbers): The rate of change of $u(t)$ over time is given by the equation $\frac{du(t)}{dt} = a_1 \cdot u(t) + a_2 \cdot c(t) + a_3$. condition: $u(0)$ and $c(t), t \in [0,1]$, QoI: $u(t), t \in [0,1]$

Precise caption (with numbers): The relationship between $u(t)$ and $c(t)$ is governed by the equation $\frac{du(t)}{dt} = 0.48 \cdot u(t) + 1.06 \cdot c(t) + 0.691$. condition: $u(0)$ and $c(t), t \in [0,1]$, QoI: $u(t), t \in [0,1]$
Solving Inverse Hyperbolic Conservation Laws with ICON-LM

\[ \partial_t u + \partial_x (au^3 + bu^2 + cu) = 0, \quad x \in [0, 1] \quad \text{Periodic boundary condition} \]

Unknown parameters \((a, b, c)\) that need to be inferred from prompted examples.

Forward operator: \( \mathcal{F}_\phi[u(t = 0, x)] := u(t = 0.1, x) \)
Backward operator as the reverse, not unique

The ICON-LM model is trained in a supervised manner, without auto-differentiation in the loss function.
The data are generated by solving the conservation laws numerically with the third-order WENO scheme.
- Reduce computational cost during training.
- Learn from battle-tested numerical schemes to handle discontinuities.
Randomly sample a, b, c in [-1,1]. Then for each (a, b, c):

- **Sample Initial Conditions**: Sample from periodic Gaussian process.
- **Numerical Simulation**: Use third-order WENO finite volume and fourth-order RK method to solve the conservation law. \( dx = 0.01, dt = 0.0005, t \) from 0 to 0.5.
- **Data Collection**: Consider every time step within the time interval [0,0.4], treat it as an individual initial condition. Each has an associated function that appears 0.1 time units later. They will form a condition-QoI pair for the forward/backward operator.
Example Results (Forward Operator)

![Diagrams showing example results for forward operator.](image)

(a) forward operator, $a = b = c = -0.6$

Given condition at $t = 0.0$, forward prediction at $t = 0.1$ overlaps with the ground truth.

Here the operator are inferred from 5 examples of condition-QoI pairs.
More Example Results (Forward Operator)

- $a = 0.6$, $b = -0.6$, $c = -0.6$

- $a = 0.6$, $b = 0.6$, $c = 0.6$
Example Results (Backward Operator)

(b) backward operator, \( a = b = c = -0.6 \)

(t=0.0) **backward prediction** is different from the **given label**, due to non-uniqueness of the backward solution. 
(t=0.1) If we apply the exact forward operator to the **backward prediction**, the **forward simulation** overlaps with the **input condition**.
More Example Results (Backward Operator)

a = 0.6, b = -0.6, c = -0.6

a = 0.6, b = 0.6, c = 0.6
Consistency error: apply the exact forward operator to the backward prediction at $t = 0$, then compare with the condition function at $t = 0.1$

The backward predictions are extremely accurate, evidenced by the low consistency error.

Note that all the predictions, including forward and backward predictions for different equations, are given by a single neural network!
Discussion

Why a very few examples are sufficient to learn the operator?
- Only need to learn the operator for a certain distribution of conditions.
- We leveraged the commonalities shared in training and testing operators. ICON only need to identify the equation and hidden parameters.
- For a larger family of operators, ICON requires more examples (especially for those complicated operators), as well as a larger neural network with more training cost.

What's next?
Scale up
- Scaling up large language models improves generalization, even leads to emergent abilities beyond human expectations.
- Numerical tasks is still a weak spot in the current AI ecosystem.
- We anticipate the possibility of AI for general numerical tasks with large ICON models.
Algorithm 2: The training and inference of In-Context Operator Networks (ICON).

1 // Training stage:
2 for $i = 1, 2, \ldots$, training steps do
3    for $b = 1, 2, \ldots$, batch size do
4       Randomly select a type of problem and a set of parameters from
5            dataset;
6       Randomly set the number of examples $J$, and the number of
7            key-value pairs in each condition and QoI of the examples and
8            question;
9       From $N$ pairs of conditions and QoIs, randomly select $J$ pairs as
10          examples and one pair as the question;
11       Build a prompt matrix, query vectors, and the ground truth
12          using the selected examples and question;
13    end
14 Use the batched prompts, queries and labels to calculate the MSE
15 loss and update the neural network parameters with gradients;
16 end
17 // Inference stage:
18 Given a new system with an unknown operator, collect examples and a
19     question condition, and design the queries;
20 Construct the prompt using the examples and question condition;
21 Get the prediction of the question QoI using a forward pass of the
22     neural network;