Qeios

Peer Review

Review of: "Inverse Evolution Data Augmentation for Neural PDE Solvers"

Aras Bacho[1]

1. California Institute of Technology, United States

This paper proposes Inverse Evolution Data Augmentation as a method to enrich training sets for neural PDE solvers (particularly neural operators for evolution equations) by integrating the PDE backwards in time rather than solely relying on expensive forward simulations. Demonstrations on the 1D Burgers equation, 2D Allen–Cahn equation, and 2D Navier–Stokes equation show that the method can rapidly generate physically plausible solution pairs, potentially reducing the data-generation burden when training neural operators like the Fourier Neural Operator.

Although the idea is intriguing—offering speed-ups by avoiding full forward solves—it faces certain limitations:

Ill-posedness of Inverse Evolution:
In general, the time-reversal of a parabolic or dissipative PDE can be highly unstable because such PDEs exhibit a smoothing effect forward in time. Reversing that process amplifies small perturbations and noise. Thus, robust strategies for choosing initial/terminal states are critical. Indeed, Equation (3) in the paper is not strictly the time-reverse of (1) in the sense that an initial condition in (1) becomes a terminal condition in (3). The resulting PDE for backward evolution thus differs from the naive “flip the sign of t” approach and requires care in how boundary/terminal conditions are handled.

Randomized Initialization and Stability:
The authors propose a random initialization scheme for the reverse evolution and mention linear combinations of time sequences to produce more diverse states. However, it remains unclear why or how this method inherently stabilizes the backward solve. One would normally expect backward time integration to be sensitive to small errors, so additional explanation is needed. Does mixing different solution snapshots effectively “regularize” or damp potential instabilities? A more detailed argument or empirical justification would strengthen confidence in this approach.

 doi.org/10.32388/GVHKPZ

Scope: Evolution Equations with Known PDEs:

Since the method relies on integrating backward in time, it requires (i) knowledge of the PDE and (ii) that the PDE's reverse evolution is not fatally ill-posed. This narrows its applicability to scenarios where the governing equations are explicitly known and not excessively chaotic or irreversible. Purely data-driven contexts or PDEs without a stable inverse dynamic would not benefit.

Comparison with Data Augmentation Using Implicit Forward Solvers:

The paper primarily compares the proposed method to neural operators whose training data is generated via explicit forward time-stepping, where large Δt can lead to instability. It would be illuminating to see how implicit solvers (which remain stable for larger time steps but are more computationally expensive) perform as a baseline. Although the authors emphasize that inverse evolution is computationally cheaper than stable explicit forward schemes, a fair evaluation might include, for example, Crank–Nicolson or a fully implicit method, examining both accuracy and cost. If an implicit forward solver yields data of comparable accuracy—albeit at a higher cost—this would clarify the trade-off and highlight the unique advantages (or limitations) of inverse data generation.

Modest Accuracy Gains:

The numerical experiments report that adding inverse-evolution data accelerates data generation and may improve learning, but does not always yield a significant jump in accuracy. This outcome suggests that while the method can quickly provide more solution pairs, further refinements (stability mechanisms, better initialization) may be needed to boost accuracy substantially.

Overall, the Inverse Evolution Data Augmentation is an inventive and potentially very useful technique for accelerating data generation in PDE-related machine learning, provided one can address:

The instability of backward-time PDE integration, a clear rationale for how random initial states or linear combinations improve stability, and a comprehensive comparison to Neural Operators where data is generated using implicit forward solvers.

If these aspects are handled properly, the method could be a valuable tool to reduce the cost of large-scale training for neural PDE solvers.

Declarations

Potential competing interests: No potential competing interests to declare.