Unraveling Machine Learning Algorithm's Impact on Physics and Engineering through a Comprehensive Review of Three Seminal Papers on Machine Learning Applications in Physical Modeling and Simulation

Anonymous Authors[1]

Abstract

This review paper explores the groundbreaking contributions of three recent research papers that harness the power of machine learning in the fields of physics, engineering, and numerical computation. The three papers to be reviewed in this paper collectively offer innovative solutions that leverage machine learning algorithms to transcend limitations in solving complex partial differential equations to model physical phenomena. The first highlighted research addresses the capacity of deep networks to model rigid-body dynamics without explicit contact-specific modules. Through a series of experiments on diverse datasets, it establishes the efficacy of general-purpose graph network simulators in predicting contact discontinuities, showcasing the potential of deep learning models in real-world contact dynamics. The second study introduces a versatile machine learning framework, the "Graph Network-based Simulators" (GNS), designed to simulate a broad spectrum of physical domains involving fluids, rigid solids, and deformable materials. The framework, built on graph representations and learned message-passing dynamics, exhibits robust generalization capabilities across various conditions, presenting a significant leap in learned physical simulation. The third research paper addresses the computational challenges in simulating fluids, presenting an end-to-end deep-learning approach to enhance approximations within computational fluid dynamics. The findings highlight substantial improvements in accuracy, computational efficiency, and stability during long simulations, showcasing the integration of machine learning to advance scientific computing.

Introduction

The convergence of machine learning and physical modeling has ushered in a new era of possibilities, challenging established paradigms and offering innovative solutions to longstanding problems in physics and engineering. This review encapsulates recent breakthroughs that collectively redefine the landscape of learned physical simulation.

One focal point of exploration involves the capabilities of deep networks in modeling rigid-body dynamics without relying on explicit contact-specific modules. By delving into general-purpose graph network simulators, this line of research demonstrates the potential of these networks to accurately predict contact discontinuities. This discovery not only challenges prevailing beliefs but also opens doors to novel considerations regarding the applicability of deep learning models in capturing real-world contact dynamics.

Another noteworthy contribution comes in the form of the Graph Network-based Simulators (GNS) framework, a machine learning architecture designed to simulate diverse physical domains, encompassing fluids, rigid solids, and deformable materials. Founded on graph representations and learned message-passing dynamics, the GNS framework stands out for its ability to generalize across varying conditions, presenting a significant advancement in the field of learned physical simulation. This development holds promise for addressing a wide spectrum of complex forward and inverse problems.

Additionally, this review delves into the application of end-to-end deep learning to enhance approximations within computational fluid dynamics, addressing the computational challenges associated with simulating fluids. The findings showcase substantial improvements in accuracy, computational efficiency, and stability during long simulations, illustrating how scientific computing can harness machine learning to achieve enhanced simulations without compromising precision or generalization. Together, these studies underscore the transformative potential of integrating machine learning into the fabric of physical modeling, paving the way for new possibilities and insights in the realms of physics and engineering.

1. Review of paper to implement or extend: "Graph network simulators can learn discontinuous, rigid contact dynamics" by K. R. Allen, T. Lopez-Guevara,Y. Rubanova, K. Stachenfeld, A. Sanchez-Gonzalez, P. Battaglia, T. Pfaff

Storyline

High-level motivation/ problem At its core, the research paper is motivated by a high-level exploration aimed at

discerning whether deep learning-based models could offer a more fitting alternative to traditional simulation environments when it comes to precisely modeling real-world rigid-body contact dynamics. Addressing a prevalent claim that deep networks are inherently incapable of accurately representing rigid-body dynamics due to their continuous nature, the paper conducts a thorough investigation using both real and simulated datasets. The findings challenge the aforementioned claim, revealing that graph network simulators, devoid of explicit contact assumptions, exhibit the capacity to capture rigid-body trajectories with commendable accuracy. This nuanced exploration not only questions established beliefs but also opens up new avenues, suggesting that, under the right architecture and parameterization, deep learning-based models can excel in accurately modeling the complexities of real-world rigid-body dynamics, thereby presenting a transformative perspective in the field of simulation research.(2)

Prior work on this problem Dynamics characterized by discontinuities, such as rigid-body collisions or transitions involving friction, present formidable challenges due to the inherently discontinuous nature of these dynamics. Classical methods aim to represent such behavior by employing rigid contact models. However, it is evident that these methodologies encounter difficulties in accurately capturing the intricacies of real-world rigid impacts.An alternative approach involves the direct learning of a function that approximates rigid-body dynamics based on empirical data. For instance, Gaussian processes have gained popularity in controlling robotic arms, learning slip-stick transitions, and modeling planar contact. GPs are known for their data efficiency, yet they incorporate smoothness assumptions that may not generalize effectively to discontinuous dynamics.

Research gap Despite considerable efforts in crafting accurate analytical models for rigid-body dynamics, challenges persist when dealing with factors like frictional contacts and temperature-dependent attributes. These complexities impede precise system identification, resulting in suboptimal outcomes when applying analytical models to forecast real-world rigid-body dynamics. The intricate nature of these challenges underscores the limitations of traditional models, highlighting the need for more adaptive and robust approaches to ensure accurate predictions in scenarios where frictional contacts and temperature-dependent attributes play pivotal roles in system dynamics.

Contribution The study demonstrated that Graph Network Simulators exhibit the capability to accurately and precisely represent rigid-body dynamics, even when confronted with discontinuities introduced by contact. When trained, a Graph Network Simulator surpasses not only other end-to-end approaches but also alternative learning methods that incorporate explicit contact assumptions. Furthermore, when trained with a substantial volume of real-world data, the model consistently outperforms analytical physics engines adapted to the same real-world experimental settings.

Proposed Solution

The authors began by establishing a dataset containing information regarding the center position and rotation of a cube at each time step. Subsequently, these data were transformed into node positions, given their model's operation involving nodes rather than center and rotation information.

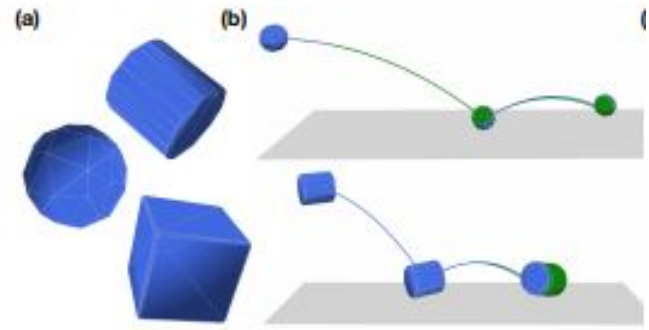
A series of metrics were computed, including the positional error (measured as the mean-squared error, MSE, as a percentage of the cube's width), rotational error (quantifying the angular disparity between ground-truth rotation and predicted values in degrees), and penetration (expressing the degree of cube penetration through the floor as a percentage of the cube's width). These metrics were then averaged over the simulation trajectory.

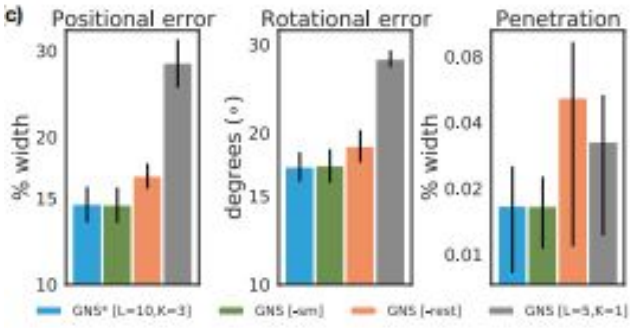
Following the dataset preparation and metric computation, the authors proceeded to train their model using a subset of trajectories from the dataset. They then presented an analysis of how the error on the test set evolved as a function of the size of the training set.

Claims-Evidence

Claim 1 The model, trained with sufficient real-world data, is on average more accurate than real-to-sim models created using analytical physics engines. Moreover, GNS performs well on other shapes, such as cylinders and spheres.

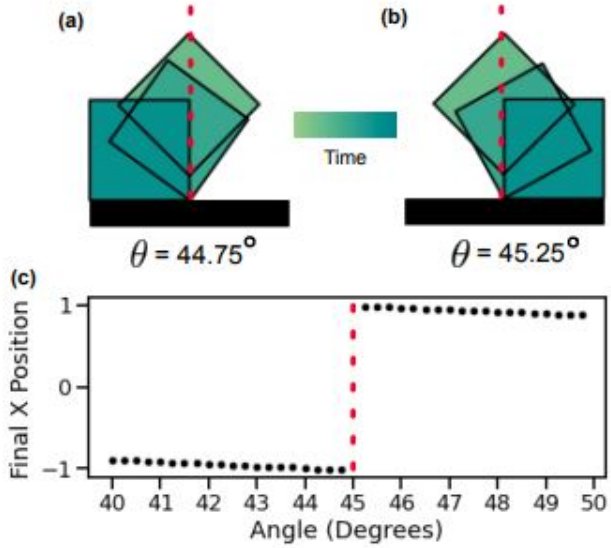
Evidence 1 As shown in the picture below the positional, rotational, and penetration error data is compared between the prediction and the ground truth and the paper showed the accuracy is with in the expected range.





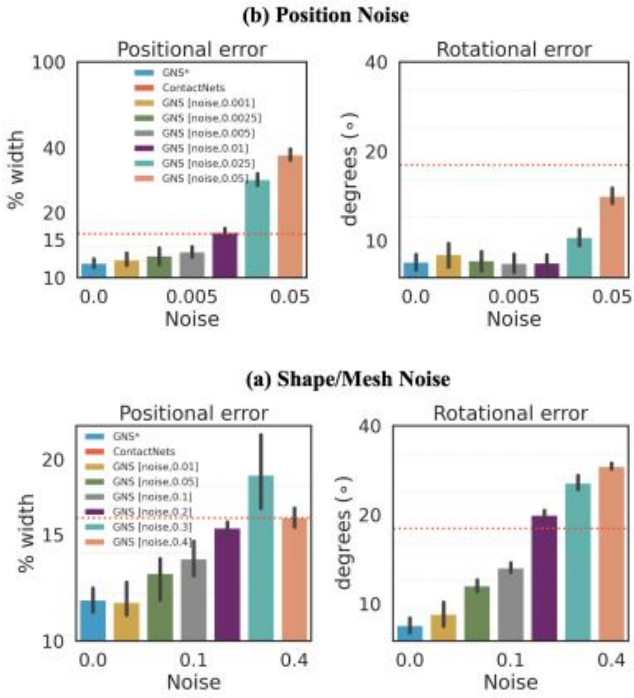
Claim 2 Graph Network Simulators seem able to model discontinuities in rigid body motion after training on very few sample trajectories.

Evidence 2 In the setup shown in the image below, a cube is thrown onto the ground such that it contacts the floor along one of its edges. This contact produces a sharp discontinuity in the velocity as the cube bounces upwards, nevertheless, the graph network simulator is able to capture the discontinuities on each bounce accurately, matching the real data closely.



Claim 3 The authors did two experiments to investigate the robustness of the GNS model to various types of noise and showed that the model is robust in both positional and shape/mesh errors.

Evidence 3 The following image shows the robustness of the model to the shape/mesh noise and position noise as modeled by jittering the position of the cube at each time-point in the trajectory.



Critique and Discussion

The paper addresses a significant and intriguing challenge in the realm of deep learning and physics simulation. The authors, embark on a systematic investigation employing experiments on both real and simulated datasets. The results presented in the paper indicate that general-purpose graph network simulators, devoid of any contact-specific assumptions, exhibit the capability to learn and predict contact discontinuities effectively. This finding challenges preconceived notions and suggests that the continuous nature of deep network parameterization may not be as limiting as previously believed. Moreover, the demonstration that graph network simulators outperform highly engineered robotics simulators in capturing real-world cube tossing trajectories, even with a limited number of trajectories provided, showcases the potential advantages of these models in specific scenarios.

The work opens up new avenues for reconsidering the role of deep learning-based models in contrast to traditional simulation environments for accurately modeling real-world contact dynamics. The paper's strength lies in its empirical approach, leveraging experiments on established datasets, which enhances the credibility of the findings. However, a comprehensive critique necessitates a closer examination of the limitations and potential biases in the experimental design. Additionally, the paper could benefit from a more in-depth discussion on the interpretability of the graph network

simulator's predictions, shedding light on the underlying mechanisms and providing insights into why it outperforms traditional simulators. Overall, the paper contributes significantly to the applicability of deep learning models in physics simulations, pushing the boundaries of understanding when and why these models may excel over traditional methods in capturing discontinuous, rigid contact dynamics.

2. Review of paper to implement or extend: "Learning to Simulate Complex Physics with Graph Networks" by Alvaro Sancez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia

This review centers on the research article titled "Learning to Simulate Complex Physics with Graph Networks," authored by Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. The study explores the conception and application of a trainable simulation methodology founded on Graph Neural Networks. This approach exhibits a capacity for acquiring knowledge and simulating diverse physical phenomena across disciplines, encompassing physics, chemistry, geology, and other domains.(3)

Storyline

High-level motivation/ problem Entitled 'Learning to Simulate Complex Physics with Graph Networks,' the paper authored by Alvaro Sancez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia introduces a machine learning framework designed to acquire the capability to simulate diverse physical phenomena. Referred to as 'Graph Network-based Simulators' (GNS) by the authors, this framework addresses the substantial computational costs and resource-intensive nature associated with traditional simulation methods. The primary impetus behind adopting a machine learning approach lies in the inherent challenges and expenses involved in creating and utilizing conventional simulations. The methodology employed in the paper centers on a particle-based simulation, where states are encapsulated as vectors represented by nodes. The dynamic aspects of these states are computed through a process of learned message-passing.

Prior work on this problem Particle-based simulations are not new in physical sciences. One among this is "smoothed particle hydrodynamics." that evaluates pressure, velocity, and positions using particle-based simulation. Graph Networks are also shown to be effective at learning and simulating some physical phenomena. There are also other techniques such as "position-based dynamics" which are more suitable for interacting, deformable materials.

Research gap All the methods mentioned earlier (in the Prior work on this problem) are effective in simulating specific physical phenomenon, but none of them are effective simulation across all physical phenomena. In addition, the "smoothed particle hydrodynamics" is not a learnable simulation.

Contribution The GNS framework addresses this gap (that the earlier models could not be applied to all the different phenomena or that they could be applied to all the different phenomena but lacks the ability to learn new phenomena with a reasonable accuracy.)

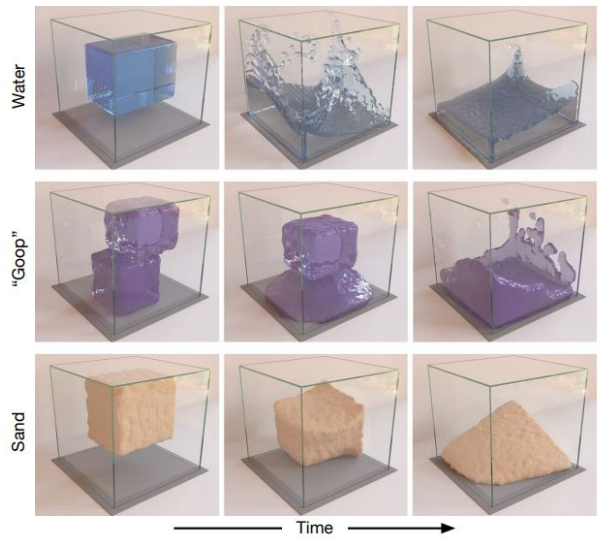
Proposed Solution

In this implementation, "the state of the world at time t is represented $X^t X$. A physical dynamics over k timesteps are applied and a trajectory of states $X^{t_0:k} = X^{t0}, \ldots, X^{t_k}$. is calculated. A mapping of $s : X \rightarrow X^{t_0:k}$ computed iteratively for each time stamp and a learnable simulator computes the dynamics information with a parameterized function approximator, $d_\theta : X \rightarrow Y$, whose parameters, θ, can be optimized for some training objective. The $Y \in \mathcal{Y}$ represents the dynamics information, whose semantics are determined by the update mechanism. The update mechanism can be seen as a function which takes $X_{\tilde{t}_k}$, and uses d_θ to predict the next state, $X_{\tilde{t}_{k+1}} = Update(X_{\tilde{t}_k}, d_\theta)$. Here we assume a simple update mechanism—an Euler integrator—and Y that represents accelerations.

Claims-Evidence

Claim 1 Previous learnable simulation methods are highly specialized for particular physical dynamics and could not be applied to a wide range of physical phenomena, but the GNS model could learn a wide array of dynamics in all physical phenomena.

Evidence 1 The paper demonstrates the learnable simulation could learn wide variety of physical domains such as fluids, rigid solids, deformable materials interacting with one another, mass-spring, robotic control systems and more.



Claim 2 Most previous simulations are expensive and difficult to implement, while the GNS framework is simpler to implement, more accurate, and less expensive (in terms of computation) across various physical settings.

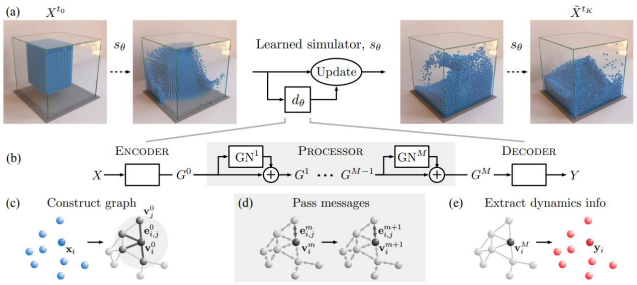
Evidence 2 The Authors provided a table of a measure of quantitative accuracy shown below.

Experimental domain	N	K	1-step ($\times 10^{-9}$)	Rollout ($\times 10^{-3}$)
WATER-3D (SPH)	13k	800	8.66	10.1
SAND-3D	20k	350	1.42	0.554
GOOP-3D	14k	300	1.32	0.618
WATER-3D-S (SPH)	5.8k	800	9.66	9.52
BOXBATH (PBD)	1k	150	54.5	4.2
WATER	1.9k	1000	2.82	17.4
SAND	2k	320	6.23	2.37
GOOP	1.9k	400	2.91	1.89
MULTIMATERIAL	2k	1000	1.81	16.9
FLUIDSHAKE	1.3k	2000	2.1	20.1
WATERDROP	1k	1000	1.52	7.01
WATERDROP-XL	7.1k	1000	1.23	14.9
WATERRAMPS	2.3k	600	4.91	11.6
SANDRAMPS	3.3k	400	2.77	2.07
RANDOMFLOOR	3.4k	600	2.77	6.72
CONTINUOUS	4.3k	400	2.06	1.06

Table 1. List of maximum number of particles N, sequence length K, and quantitative model accuracy (MSE) on the held-out test set. All domain names are also hyperlinks to the video website.

Claim 3 The GNS framework advances the state-of-the-art in learning physical dynamics and it is promising to solve other dynamic systems in both forward and inverse problems.

Evidence 3 The simulation model is based on graph networks where particles and their state is represented by nodes while their dynamics is represented by edge. This could be applied to various other phenomena including phenomena outside the physical sciences.



Critique and Discussion

The paper introduces a machine learning framework and model implementation termed "Graph Network-based Simulators" (GNS). This framework showcases a notable advance in the field of learned physical simulation by offering a versatile solution for simulating diverse physical domains, encompassing fluids, rigid solids, and deformable materials interacting with each other. The GNS framework represents the state of a physical system by employing particles expressed as nodes in a graph, and it computes dynamics through learned message-passing. The paper's strength lies in its ability to demonstrate the model's generalizability, as it successfully extends from single-timestep predictions with thousands of particles during training to different initial conditions, thousands of timesteps, and even an order of magnitude more particles at test time. Moreover, the robustness of the model to hyperparameter choices and its ability to handle noise in the training data underscore the effectiveness of the GNS framework in addressing challenges associated with complex physical simulations.

While the paper provides valuable insights into the capabilities of the GNS framework, a more in-depth discussion on the interpretability of the model's predictions could enhance the paper. Understanding how the graph-based representation captures and processes information, particularly in the context of diverse physical domains, would contribute to a deeper comprehension of the model's inner workings. Additionally, a discussion on the computational efficiency of the GNS framework in comparison to traditional simulation methods could provide valuable insights into the practical implications of adopting such machine learning-based approaches. Overall, the paper contributes significantly to the advancement of learned physical simulation, presenting a promising avenue for solving complex forward and inverse problems, yet further exploration into model interpretability and computational efficiency would enrich its impact.

3. Review of paper to implement or extend: "Machine learning–accelerated computational fluid dynamics" by Dimitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P.Brenner, and Stephan Hoyer

Machine learning-accelerated computational fluid dynamics" by Kochkov et al. is a paper that explores the utilization of deep learning techniques embedded inside traditional fluid equations to enhance the precision of fluid dynamics simulations. The study illustrates how the integration of machine learning methodologies can yield advancements in the field of scientific computation, with an emphasis on enhancing simulation accuracy and generalization in fluid mechanics.

The research paper highlights the application of end-to-end deep learning methodologies to refine approximations within the domain of computational fluid dynamics, specifically for the modeling of two-dimensional turbulent flows.

Storyline

Despite the straightforward relationship between the equations of motion in classical mechanics and applications ranging from weather, climate, and fluid mechanics, to particle and plasma physics, it is nearly impossible to solve some equations in these areas using direct numerical methods due to the complexity of the non-linear partial differential equations which are central in physics and engineering. In this paper, the authors introduce a machine-learning algorithm to replace the components of traditional solvers with learned alternatives.

The paper's approach is different from other machine learning algorithms because the machine learning algorithms are embedded inside the traditional fluid dynamics equations which makes it less costly than just direct machine learning algorithms.

The comparison in computation time is shown in the table below.

High-level motivation/ problem It is known that machine learning algorithms simulate fluid mechanics very quickly but less accurately while traditional fluid mechanics yields accurate results but with significant lag in computation time. The high-level motivation for this research is combining traditional fluid equations and machine learning algorithms to combine the efficiency of machine learning algorithms with the accuracy of traditional fluid mechanics computation.

Modeling fluids using machine learning algorithms embedded in classical fluid dynamics equations uses the strengths of both worlds to simulate fluid mechanics with less computation complexity and more accuracy.

Prior work on this problem

Research gap Classically incompressible fluids are modeled by the Navier Stockes equations shown here.

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)=-\nabla p+\mu\nabla^2\mathbf{u}+\mathbf{f} \tag{1}$$

$$\nabla\cdot\mathbf{u}=0 \tag{2}$$

where ρ is the fluid density, u is the velocity vector, p is the pressure, μ is the dynamic viscosity, and f is the body force per unit volume.

Direct Numerical Simulation (DNS) which is a computational fluid dynamics (CFD) simulation technique solves the Navier-Sokes equations. In other words, DNS resolves the whole range of spatial and temporal scales of the turbulence, from the smallest dissipative scales to the integral scale but it is computationally expensive and requires a large amount of memory storage.

Another modeling technique models the large-scale structures of turbulence and filters out the small-scale structures. Large Eddy Simulation (LES) is computationally less expensive than DNS and requires less memory storage. However, LES still requires a turbulence model to account for the effects of the unresolved scales of turbulence.

Contribution The contribution of the paper is creating a learned model that solves the Napier-Stokes equations more efficiently than the traditional DNS. The research's principal aim is to accelerate DNS without compromising accuracy or generalization. To that end, the authors considered ML modeling approaches that enhance the standard computational fluid mechanics solver when run on coarse grids.

The ML models can improve the accuracy of the numerical solver via effective super-resolution of missing details. Unlike traditional numerical methods, the learned solvers here are optimized to fit the observed manifold of solutions to the equations they solve, rather than arbitrary polynomials.

Proposed Solution

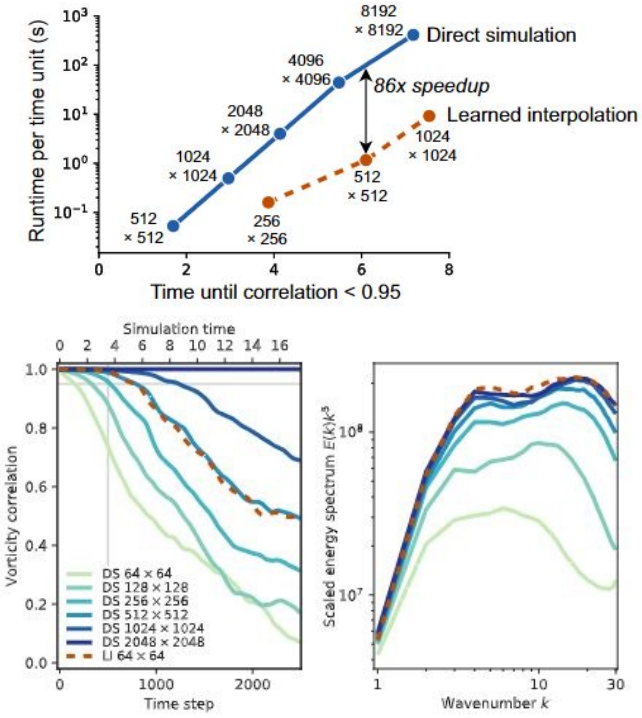
The algorithm operates in the following manner. At each time step, neural networks create a latent vector at every grid point using information from the present velocity field. These vectors are subsequently harnessed by the solver's sub-components to consider the local solution structure.

Employing a convolutional methodology, the neural networks enforce translation invariance, allowing them to operate within a local spatial context. Standard numerical methods' elements are then integrated to introduce inductive biases aligning with the principles underlying the Navier–Stokes equations.

Claims-Evidence

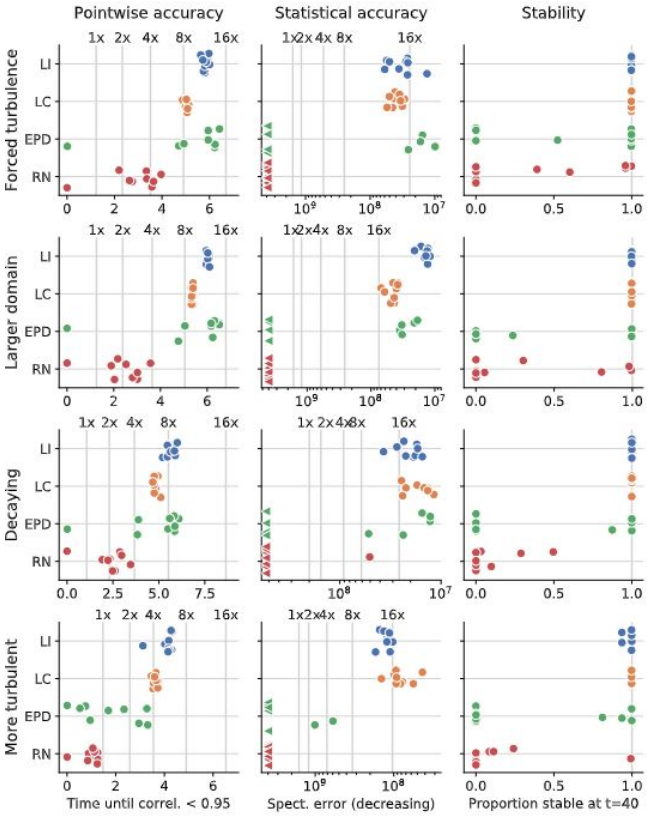
Claim 1 Direct Numerical Simulation (DNS) is distinguished by its exceptional accuracy and generalization, albeit it falls short in terms of operational efficiency. Machine learning methods tailored for fluid dynamics must outperform conventional benchmarks, like DNS, in computational speed while upholding the same high levels of accuracy.

Evidence 1 The figure depicted below illustrates the outcomes of training and evaluating our model on Kolmogorov flows at a Reynolds number of Re = 1,000. All datasets were created through the utilization of high-resolution DNS, subsequently followed by a coarsening step.



Claim 2 When compared to other ML models, learned interpolation performs the best.

Evidence 2 The figure presented below provides a comparative analysis of results obtained across all configurations under consideration.



Claim 3 The model is useful across a wide range of fluid mechanical phenomena for it could learn local operators.

Evidence 3 The model is applied to larger domain sizes and small-scale Komologrov flow it showed the exact same performance as on the training domain.(1)

Critique and Discussion

In this study, the researchers present a data-driven numerical approach that achieves commensurate accuracy with conventional finite difference/finite volume methods, albeit with significantly reduced resolution. This method is capable of learning precise local operators for convective fluxes and residual terms, thereby attaining a level of accuracy comparable to that of an advanced numerical solver operating at an 8 to 10 times finer resolution, while expediting computations by a factor of 40 to 80.

Leveraging machine learning (ML) techniques, the proposed method facilitates enhanced interpolation at a coarser scale, operating within the established framework of traditional numerical discretizations. Notably, this methodology inherently encompasses the scaling and symmetry attributes inherent in the original governing Navier–Stokes equations.

The method demonstrates superior generalizability compared to purely black-box machine-learned approaches, extending its applicability not only to diverse forcing functions but also to varying parameter domains.

Implementation

Implementation motivation

My motivation behind implementing the proposed model lies in addressing the inherent challenges associated with numerical simulations of fluid dynamics, a pivotal aspect in modeling diverse physical phenomena ranging from weather and climate to aerodynamics and plasma physics. While the Navier-Stokes equations serve as a comprehensive descriptor for fluids, the computational demands of solving these equations at scale pose a formidable obstacle, often necessitating trade-offs between accuracy and tractability. The decision to implement the end-to-end deep learning approach stems from the overarching goal of enhancing approximations within computational fluid dynamics, particularly in modeling two-dimensional turbulent flows. The promise of achieving results on par with baseline solvers but at significantly reduced computational costs, coupled with the demonstrated stability during long simulations and the ability to generalize beyond the training conditions, highlights the transformative potential of this method.

Implementation setup and plan

The authors furnish a comprehensive implementation setup and plan, offering a detailed procedure, a coherent code base, and sample data. The setup is presented with a straightforward approach, enhancing its accessibility for replication by researchers and practitioners. Python libraries relevant to the methodology are included, with an accompanying installation guide for users to execute the provided demo notebook effortlessly. Following the instructions, I successfully replicated the authors' experiment, facilitating a systematic exploration of the simulation's behavior through parameter manipulation.

Details

JAX-CFD demo

This initial demonstration shows how to use JAX-CFD to simulate decaying turbulence in 2D.

```
import jax
import jax.numpy as jnp
import jax_cfd.base as cfd
import numpy as np
import seaborn
import xarray
```

```
size = 256
density = 2.
viscosity = 1e-3
seed = 0
inner_steps = 25
outer_steps = 200

max_velocity = 2.0
cfl_safety_factor = 0.5

# Define the physical dimensions of the simulation.
grid = cfd.grids.Grid((size, size), domain=((0, 2 * jnp.pi), (0, 2 * jnp.pi)))

# Construct a random initial velocity. The `filtered_velocity_field` function
# ensures that the initial velocity is divergence free and it filters out
# high frequency fluctuations.
v0 = cfd.initial_conditions.filtered_velocity_field(
    jax.random.PRNGKey(seed), grid, max_velocity)

# Choose a time step.
dt = cfd.equations.stable_time_step(
    max_velocity, cfl_safety_factor, viscosity, grid)

# Define a step function and use it to compute a trajectory.
step_fn = cfd.funcutils.repeated(
    cfd.equations.semi_implicit_navier_stokes(
        density=density, viscosity=viscosity, dt=dt, grid=grid),
    steps=inner_steps)
rollout_fn = jax.jit(cfd.funcutils.trajectory(step_fn, outer_steps))
%time _, trajectory = jax.device_get(rollout_fn(v0))
```

```
CPU times: total: 18.5 s
Wall time: 11.1 s
```

```
# JAX-CFD uses GridVariable objects for input/output. These objects contain:
#  - array data
#  - an "offset" that documents the position on the unit-cell where the data
#    values are located
#  - grid properties
#  - boundary conditions on the variable
with np.printoptions(edgeitems=1):
  for i, u in enumerate(trajectory):
    print(f'Component {i}: {u}')
```

```
      ...,

      [[-0.06033939, ..., -0.08106222],
       ...,
       [-0.05940307, ..., -0.08062349]]], dtype=float32), offset=(1.0, 0.5), grid=Grid(shape=(256, 256), step=(0.024543692606170
6), domain=((0.0, 6.28318530717586), (0.0, 6.28318530717586))), bc=BoundaryConditions(boundaries=('periodic', 'periodic')))
Component 1: GridVariable(array=GridArray(data=array([[[[0.14287749, ..., 0.15110306],
        ...,
        [0.17213465, ..., 0.18034112]],

       ...,

       [[0.28383735, ..., 0.28290105],
        ...,
        [0.2726322 , ..., 0.27045414]]], dtype=float32), offset=(0.5, 1.0), grid=Grid(shape=(256, 256), step=(0.024543692606170
domain=((0.0, 6.28318530717586), (0.0, 6.28318530717586))), bc=BoundaryConditions(boundaries=('periodic', 'periodic')))
```

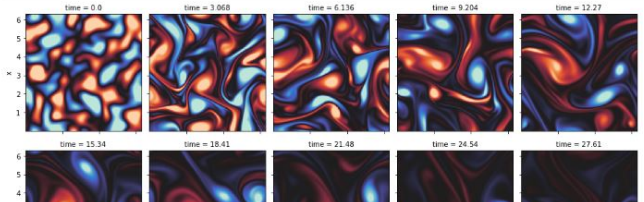
```
# Load into xarray for visualization and analysis
ds = xarray.Dataset(
    {
        'u': (('time', 'x', 'y'), trajectory[0].data),
        'v': (('time', 'x', 'y'), trajectory[1].data),
    },
    coords={
        'x': grid.axes()[0],
        'y': grid.axes()[1],
        'time': dt * inner_steps * np.arange(outer_steps)
    }
)
```

```
ds
```

```
<xarray.Dataset>
Dimensions:  (time: 200, x: 256, y: 256)
Coordinates:
  * x        (x) float32 0.01227 0.03682 0.06136 0.0859 ... 6.222 6.246 6.271
  * y        (y) float32 0.01227 0.03682 0.06136 0.0859 ... 6.222 6.246 6.271
  * time     (time) float64 0.0 0.1534 0.3068 0.4602 ... 30.07 30.22 30.37 30.53
Data variables:
    u        (time, x, y) float32 -0.8935 -0.9323 -0.9674 ... -0.1009 -0.08062
    v        (time, x, y) float32 0.1429 0.1357 0.1293 ... 0.2681 0.2691 0.2705
```

```
def vorticity(ds):
  return (ds.v.differentiate('x') - ds.u.differentiate('y')).rename('vorticity')

(ds.pipe(vorticity).thin(time=20)
 .plot.imshow(col='time', cmap=seaborn.cm.icefire, robust=True, col_wrap=5));
```



Preliminary results and interpretation

I have repeated the training, changing the density, viscosity, seed, and number of inner steps and outer steps and I have compared the outputs with direct numerical simulations of Napier-Stock equations and found the simulation works as expected.

References

[1] Gideon Dresdner, Dmitrii Kochkov, Peter Norgaard, Leonardo Zepeda-Núñez, Jamie A. Smith, Michael P. Brenner, and Stephan Hoyer. Learning to correct spectral methods for simulating turbulent flows. 2022.

[2] Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021.

[3] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, 2020.