Learned 1-D advection solver to accelerate air quality modeling

Anonymous Author(s)
Affiliation
Address
email

Abstract

Accelerating the numerical integration of partial differential equations by learned surrogate model is a promising area of inquiry in the field of air pollution modeling. Most previous efforts in this field have been made on learned chemical operators though machine-learned fluid dynamics has been a more blooming area in machine learning community. Here we show the first trial on accelerating advection operator in the domain of air quality model using a realistic wind velocity dataset. We designed a convolutional neural network-based solver giving coefficients to integrate the advection equation. We generated a training dataset using a 2nd order Van Leer type scheme with the 10-day east-west components of wind data on 39°N within North America. The trained model with coarse-graining showed good accuracy overall, but instability occurred in a few cases. Our approach achieved up to 12.5× acceleration. The learned schemes also showed fair results in generalization tests.

1 Introduction

Numerical integration of partial differential equations (PDEs) is a core element of air quality model. To run the air quality model one should solve coupled PDEs within many grid boxes and multiple time steps. Since solving many PDEs requires a huge amount of computational cost, the invention of a fast and accurate solver with has been always welcomed. Recent advancement in physics-informed machine learning [Karniadakis et al., 2021, Kashinath et al., 2021] has gained popularity to emulate existing solvers and researchers are seeking a *pareto* optimum between speed and accuracy.

So far, research efforts on learning air quality models for acceleration have mostly focused on chemistry solvers. Kelp et al. [2020] developed the encoder-operator-decoder structure neural network and achieved ×260 speedup in emulation of Carbon Bond Mechanism Z coupled to the Model for Simulating Aerosol Interactions and Chemistry. Huang and Seinfeld [2022] showed their neural integrator could solve two benchmarking problems (the H_2O_2/OH/HO_2 System and the Verwer System) with acceleration in at least one order of magnitude.

Although this type of research is an emerging area, the potential of a learned solver for transport operators in air quality modeling has not been actively investigated. However, there could be possible speedup by using a learned transport operator since researchers in computational fluid dynamics (CFD) already showed machine-learned acceleration. Kochkov et al. [2021] and Stachenfeld et al. [2021] showed convolutional neural network (CNN) based coarse-graining solver could accelerate CFD solvers. Zhuang et al. [2021]'s learned discretization to estimate coefficients in advection solver achieved 1.8× faster computing by 4× coarsening. The more examples of machine-learned fluid dynamics could be found from the review papers by Brunton et al. [2020] andKutz [2017].

Our study examines the potential of a learned advection operator for computational acceleration of air quality modeling. To build a foundation for future analysis, we first explore the potential of learned

Submitted to the DLDE Workshop in the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

solver in 1-D advection. We used a realistic wind velocity instead of a synthetic velocity to generate our baseline dataset. Since the modeling domain in the global air quality model is not rectangular, but spherical, evaluation of the model skills in different grid sizes is critical to generalization. We tested the model's generalization ability in different latitudes which have different grid sizes. Also, we tested the model's ability to integrate the wave from an initial condition shape out of training regime.

2 Numerical advection

We simulated a passive scalar advection in a horizontal line passing 39.00°N of North America (130°W - 60°W). The spatial grid size was 0.3125°. We simulated advection using east-west components of wind data from 1 to 10 January 2019 with 5-minute intervals. We obtained wind data from GEOS-FP of NASA Global Modeling and Assimilation Office [NASA, 2022]. We used a square initial condition with a 10^{-7} on the central 1/3 of domain, while other areas have 0. We implemented the L94 advection scheme [Lin et al., 1994] in the Julia computing language [Bezanson et al., 2012].

After generating this dataset by numerical integration, we down-sampled those wave datasets in lower resolutions in both space and time. To conserve mass, we averaged the scalar values in down-sampling. The sample resolutions are $\times1$, $\times2$, $\times4$, $\times8$, and $\times16$ in space and $\times1$, $\times2$, $\times4$, $\times8$, $\times16$, $\times32$, $\times64$ in time, so we have 35 different cases from one scenario.

3 Learned advection using a convolutional neural network

Figure 1 illustrates the design of the CNN-based surrogate advection solver. Our surrogate equation has information of Δt/Δx, which is critical in earth system modeling domain since grid spacing Δx can change along with latitude. The three-layer CNN receives scalar and velocity fields at nth time step as inputs and yields six coefficients to construct surrogate numerical equation. We used two GeLU activations [Hendrycks and Gimpel, 2016] to resolve sharp gradients [Kim et al., 2021], and one hypertangent since the temporal gradient can have both positive and negative signs. We used gradient scaling by adopting k_1 and k_1 to make $k_1(\Delta t/\Delta x)$ and $k_2(\Delta t/\Delta x)^2$ be in the order of 10^0.

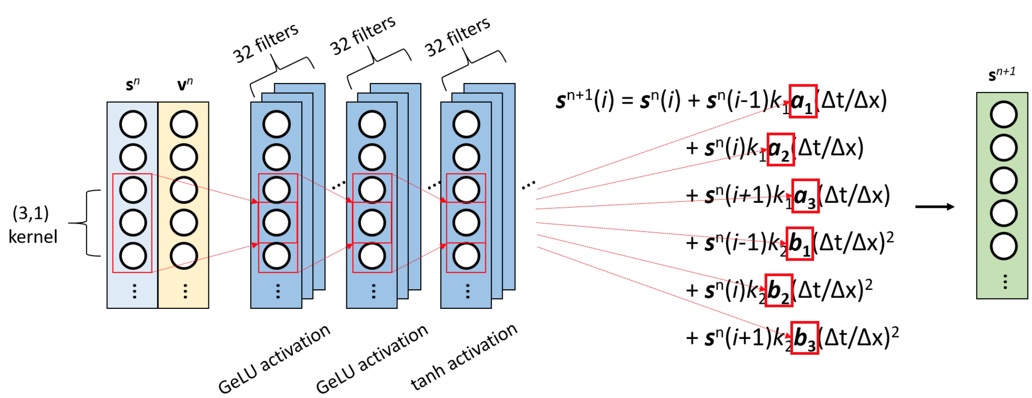

Figure 1: Illustrative diagram of convolutional neural net advection operator

We used mean absolute error (MSE) in 10-time steps as a loss function to reflect the dynamic nature of advection. To prevent error accumulation, we introduced random noise with 4×10^{-5} magnitude of initial scalar intensity. We used the ADAM optimizer [Kingma and Ba, 2014] with default parameters in Flux.jl [Innes, 2018], except for the learning rate. We used a decaying learning rate to reach the optimum. After training the model, we evaluated the model performance by feeding only the initial condition and velocity field and assessed if the model could integrate the advection process till a given period. We used a single CPU core from an HPE Apollo 6500 system with dual 6248 Cascade Lake CPUs to evaluate computational time.

Figure 2 shows the performance of the learned solver in integrating the training dataset. We normalized MSE and root mean square error (RMSE) by the initial magnitude. Here we should note that the model training was not successful in (1Δx, 8Δt), (1Δx, 16Δt), (1Δx, 32Δt), and (1Δx, 64Δt) as the outputs exploded out. Except for those cases, the learned schemes showed fair fidelity since error was usually less than 10 % (**Figure 2A** and **2B**) and r^2 was higher than 0.9 (**Figure 2C**). Maximum acceleration was achieved in (16Δx, 64Δt) and this scheme was 12.5× faster than the original solver (**Figure 2D**). As seen in **Figure 2D**, the acceleration in failure cases would be lower than the maximal acceleration case, so we may consider those resolutions to be out of our scope.

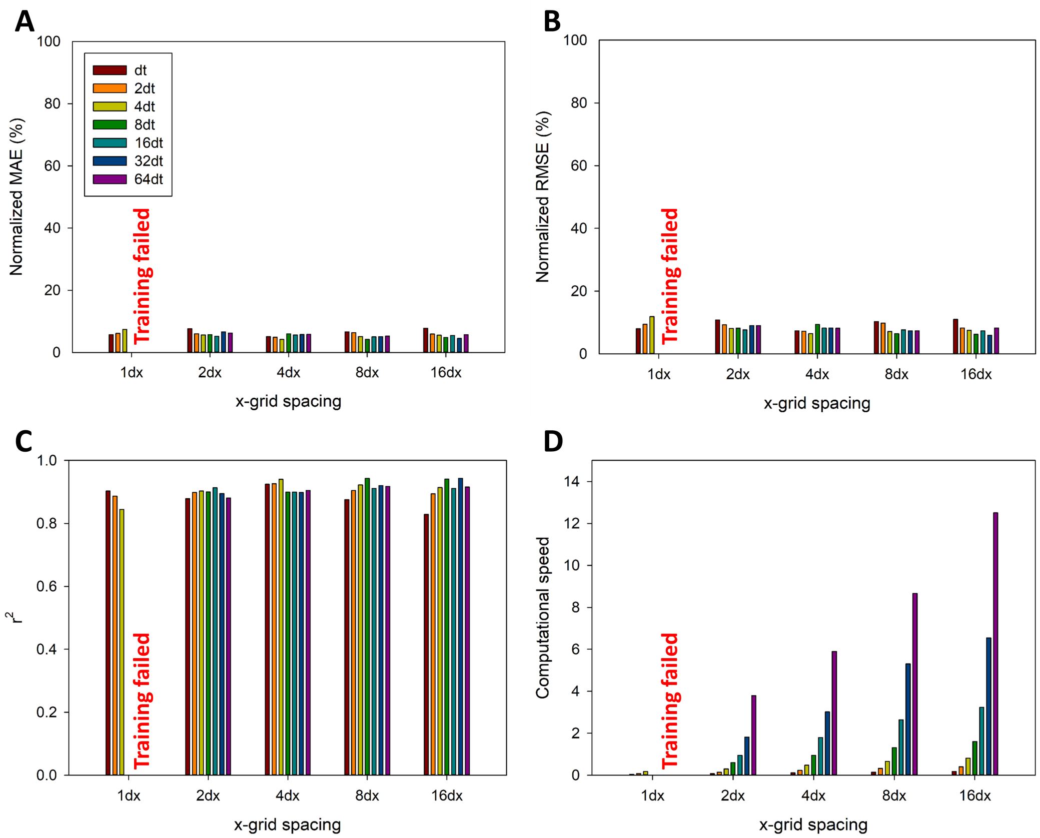

Figure 2: Performance indices of the surrogate learned solver in emulating the training dataset (A: normalized mean absolute error, B: normalized root mean square error, C: r^2, and D: speed)

Time series plots of advection in the resolution with the best accuracy (4Δx, 4Δt; **Figure 3A**) and with the maximum acceleration (16Δx, 64Δt; **Figure 3B**) show the snapshots in the first step from initial condition, 1/3 of time span, 2/3 of time span, and the final step. As seen in **Figure 3**, we could confirm that the learned model could emulate the coarsened numerical results within fair accuracy.

4 Generalization

We tested if our model could integrate the waves from outside of the training regime. We implemented two tests: 1) integrating advection in the Northern area; 2) integrating advection from Gaussian shape initial condition. In the first test, we applied our surrogate solver in a horizontal line passing 45°N. Though we used 0.3125° as grid spacing as we did on the training set, the exact spacing was shorter here because of the earth's spherical shape. In the second test, we fed a Gaussian shape initial condition to be integrated. Any conditions not mentioned are the same as the training.

The results of generalization test are summarized in **Figure 4**. **Figure 4A** and **4B** show the r^2 in the horizontal line passing 45°N and the integration Gaussian shape initial condition, respectively. As seen in **Figure 4**, we can use our learned scheme in a regime beyond the training set without substantial performance degradation. We did not test the coarsening cases in (1Δx, 8Δt), (1Δx, 16Δt), (1Δx, 32Δt), and (1Δx, 64Δt) because the surrogate models in those cases failed in training.

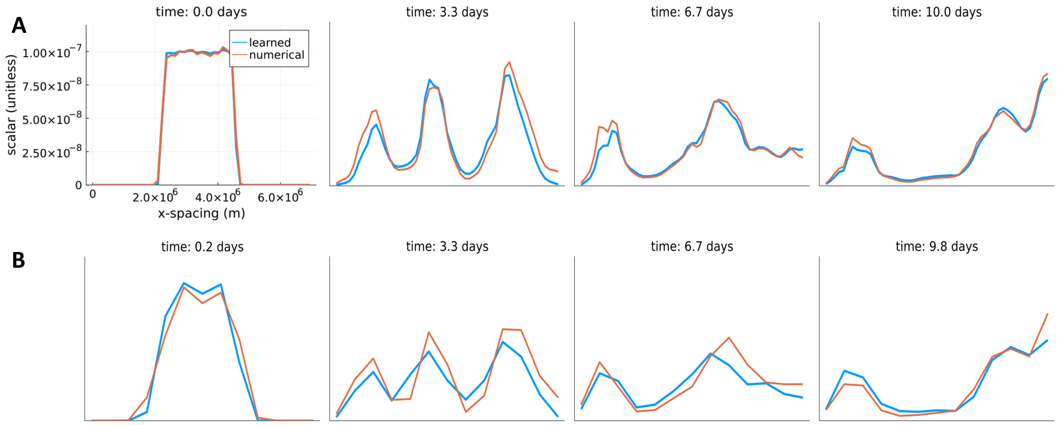

Figure 3: Time series display of both numerical (orange line) and learned (blue line) scalar advection (A: (4Δx, 4Δt) and B: (16Δx, 64Δt))

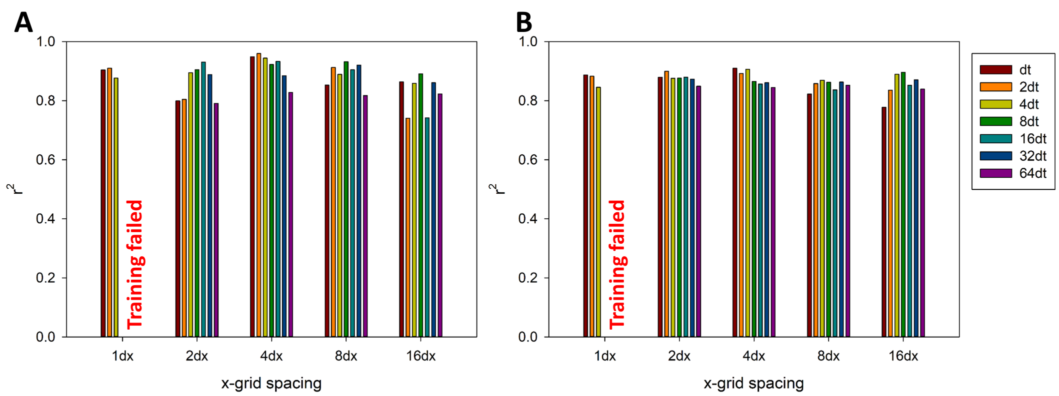

Figure 4: Generalization ability of the learned solver summarized by r^2 (A: The horizontal line passing 45°N, and B: Integration with Gaussian shape initial condition)

5 Limitation

The most noticeable limitation of this work is that the learned scheme could not work in certain resolutions. One possible explanation is that violations of the CFL condition were most extreme in those cases. Though we increased the stencil size in an attempt to resolve this issue, this approach did not fix the problem. The reason for this instability is an area for future study. Another limitation is that we did not optimize the model training in the light of hyperparameter tuning or rigorous code optimization. We would argue, however, our approach can be promising since it showed robust performance without optimization and there is still room to achieve acceleration. Finally, to use this scheme on the whole globe, we would need more generalization tests on different latitudes to see if the current scheme could work or would need more training on different regimes.

6 Conclusion

Our study revealed the learned advection scheme can be used to emulate passive scalar advection in the domain of air quality modeling using real wind-field data. The learned advection solver showed up to $\times 12.5$ with fairly accurate integration. The learned solver was robust in generalization even though we only used a single training dataset. This robustness may come from the randomness of realistic wind data, implying choice of a dataset can be crucial in physics-informed machine learning. There could be more potential acceleration by code optimization or high-performance computing. With an appropriate splitting technique, we could extend this learned 1-D advection solver to multi-dimensions and eventually accelerate air quality models.

Supplementary materials

The codes and data can be accessed at https://github.com/manozzing/Learned-1-D-advection-solver-with-grid-spacing-physics

References

J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52:477–508, 2020.

D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Y. Huang and J. H. Seinfeld. A neural network-assisted euler integrator for stiff kinetics in atmospheric chemistry. *Environmental Science & Technology*, 56(7):4676–4685, 2022.

M. Innes. Flux: Elegant machine learning with julia. *Journal of Open Source Software*, 3(25):602, 2018.

G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

K. Kashinath, M. Mustafa, A. Albert, J. Wu, C. Jiang, S. Esmaeilzadeh, K. Azizzadenesheli, R. Wang, A. Chattopadhyay, A. Singh, et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 379(2194):20200093, 2021.

M. M. Kelp, D. J. Jacob, J. N. Kutz, J. D. Marshall, and C. W. Tessum. Toward stable, general machine-learned models of the atmospheric chemical system. *Journal of Geophysical Research: Atmospheres*, 125(23):e2020JD032759, 2020.

S. Kim, W. Ji, S. Deng, Y. Ma, and C. Rackauckas. Stiff neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(9):093122, 2021.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118 (21):e2101784118, 2021.

J. N. Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

S.-J. Lin, W. C. Chao, Y. Sud, and G. Walker. A class of the van leer-type transport schemes and its application to the moisture transport in a general circulation model. *Monthly Weather Review*, 122 (7):1575–1593, 1994.

NASA. GMAO Product > GEOS Near-Real Time Data Products, 2022. URL `https://gmao.gsfc.nasa.gov/GMAO_products/NRT_products.php`.

K. Stachenfeld, D. B. Fielding, D. Kochkov, M. Cranmer, T. Pfaff, J. Godwin, C. Cui, S. Ho, P. Battaglia, and A. Sanchez-Gonzalez. Learned coarse models for efficient turbulence simulation. *arXiv preprint arXiv:2112.15275*, 2021.

J. Zhuang, D. Kochkov, Y. Bar-Sinai, M. P. Brenner, and S. Hoyer. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Physical Review Fluids*, 6(6):064605, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cited NASA GMAO webpage to cite GEOS-FP.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]