Learning data-driven discretizations for partial differential equations

Yohai Bar-Sinai[a,1,2], Stephan Hoyer[b,1,2], Jason Hickey[b], and Michael P. Brenner[a,b]

[a]School of Engineering and Applied Sciences, Harvard University, Cambridge MA 02138; and [b]Google Research, Mountain View, CA 94043

Edited by John B. Bell, Lawrence Berkeley National Laboratory, Berkeley, CA, and approved June 21, 2019 (received for review August 14, 2018)

The numerical solution of partial differential equations (PDEs) is challenging because of the need to resolve spatiotemporal features over wide length- and timescales. Often, it is computationally intractable to resolve the finest features in the solution. The only recourse is to use approximate coarse-grained representations, which aim to accurately represent long-wavelength dynamics while properly accounting for unresolved small-scale physics. Deriving such coarse-grained equations is notoriously difficult and often ad hoc. Here we introduce data-driven discretization, a method for learning optimized approximations to PDEs based on actual solutions to the known underlying equations. Our approach uses neural networks to estimate spatial derivatives, which are optimized end to end to best satisfy the equations on a low-resolution grid. The resulting numerical methods are remarkably accurate, allowing us to integrate in time a collection of nonlinear equations in 1 spatial dimension at resolutions 4× to 8× coarser than is possible with standard finite-difference methods.

coarse graining | machine learning | computational physics

Solutions of nonlinear partial differential equations can have enormous complexity, with nontrivial structure over a large range of length- and timescales. Developing effective theories that integrate out short lengthscales and fast timescales is a long-standing goal. As examples, geometric optics is an effective theory of Maxwell equations at scales much longer than the wavelength of light (1); density functional theory models the full many-body quantum wavefunction with a lower-dimensional object—the electron density field (2); and the effective viscosity of a turbulent fluid parameterizes how small-scale features affect large-scale behavior (3). These models derive their coarse-grained dynamics by more or less systematic integration of the underlying governing equations (by using, respectively, WKB theory, local density approximation, and a closure relation for the Reynolds stress). The gains from coarse graining are, of course, enormous. Conceptually, it allows a deep understanding of emergent phenomena that would otherwise be masked by irrelevant details. Practically, it allows computation of vastly larger systems.

Averaging out unresolved degrees of freedom invariably replaces them by effective parameters that mimic typical behavior. In other words, we identify the salient features of the dynamics at short and fast scales and replace these with terms that have a similar average effect on the long and slow scales. Deriving reliable effective equations is often challenging (4). Here we approach this challenge from the perspective of statistical inference. The coarse-grained representation of the function contains only partial information about it, since short scales are not modeled. Deriving coarse-grained dynamics requires first inferring the small-scale structure using the partial information (reconstruction) and then incorporating its effect on the coarse-grained field. We propose to perform reconstruction using machine-learning algorithms, which have become extraordinarily efficient at identifying and reconstructing recurrent patterns in data. Having reconstructed the fine features, modeling their effect can be done using our physical knowledge about the system. We call our method data-driven discretization. It is qualitatively different from coarse-graining techniques that are currently in use: Instead of analyzing equations of motion to derive effective behavior, we directly learn from high-resolution solutions to these equations.

Related Work

Several related approaches for computationally extracting effective dynamics have been previously introduced. Classic works used neural networks for discretizing dynamical systems (5, 6). Similarly, equation-free modeling approximates coarse-scale derivatives by remapping coarse initial conditions to fine scales which are integrated exactly (7). The method has a similar spirit to our approach, but it does not learn from fine-scale dynamics and use the memorized statistics in subsequent times to reduce the computational load. Recent works have applied machine learning to partial differential equations (PDEs), either focusing on speed (8–10) or recovering unknown dynamics (11, 12). Models focused on speed often replace the slowest component of a physical model with machine learning, e.g., the solution of Poisson's equation in incompressible fluid simulations (9), subgrid cloud models in climate simulations (10), or building reduced-order models that approximate dynamics in a lower-dimensional space (8, 13, 14). These approaches are promising, but learn higher-level components than our proposed method. An important development is the ability to satisfy some physical constraints exactly by plugging learned models into a fixed equation of motion. For example, valid fluid dynamics can be

Significance

In many physical systems, the governing equations are known with high confidence, but direct numerical solution is prohibitively expensive. Often this situation is alleviated by writing effective equations to approximate dynamics below the grid scale. This process is often impossible to perform analytically and is often ad hoc. Here we propose data-driven discretization, a method that uses machine learning to systematically derive discretizations for continuous physical systems. On a series of model problems, data-driven discretization gives accurate solutions with a dramatic drop in required resolution.

Author contributions: Y.B.-S., S.H., J.H., and M.P.B. designed research; Y.B.-S. and S.H. performed research; Y.B.-S. and S.H. analyzed data; and Y.B.-S., S.H., J.H., and M.P.B. wrote the paper.

The authors declare no conflict of interest.

This article is a PNAS Direct Submission.

Data deposition: Source code is available on GitHub (https://github.com/google/data-driven-discretization-1d).

[1]Y.B.-S. and S.H. contributed equally to this work.

[2]To whom correspondence may be addressed. Email: ybarsinai@gmail.com or shoyer@google.com.

This article contains supporting information online at www.pnas.org/lookup/suppl/doi:10.1073/pnas.1814058116/-/DCSupplemental.

Published online July 16, 2019.

guaranteed by learning either velocity fields directly (12) or a vector potential for velocity in the case of incompressible dynamics (8). Closely related to this work, neural networks can be used to calculate closure conditions for coarse-grained turbulent flow models (15, 16). However, these models rely on existing coarse-grained schemes specific to turbulent flows and do not discretize the equations directly. Finally, ref. 17 suggested discretizations whose solutions can be analytically guaranteed to converge to the center manifold of the governing equation, but not in a data-driven manner.

Data-Driven Subgrid-Scale Modeling

Consider a generic PDE, describing the evolution of a continuous field $v(x,t)$,

$$\frac{\partial v}{\partial t} = F\left(t, x, v, \frac{\partial v}{\partial x_i}, \frac{\partial v}{\partial x_i \partial x_j}, \cdots\right). \quad [1]$$

Most PDEs in the exact sciences can be cast in this form, including equations that describe hydrodynamics, electrodynamics, chemical kinetics, and elasticity. A common algorithm to numerically solve such equations is the method of lines (18): Given a spatial discretization $x_1, \ldots, x_N$, the field $v(x,t)$ is represented by its values at node points $v_i(t) = v(x_i, t)$ (finite differences) or by its averages over a grid cell, $v_i(t) = \Delta x^{-1} \int_{x_i-\Delta x/2}^{x_i+\Delta x/2} v(x',t)\,dx'$ (finite volumes), where $\Delta x = x_i - x_{i-1}$ is the spatial resolution (19). The time evolution of v_i can be computed directly from Eq. **1** by approximating the spatial derivatives at these points. There are various methods for this approximation—polynomial expansion, spectral differentiation, etc.—all yielding formulas resembling

$$\frac{\partial^n v}{\partial x^n} \approx \sum_i \alpha_i^{(n)} v_i, \quad [2]$$

where the $\alpha_i^{(n)}$ are precomputed coefficients. For example, the 1-dimensional (1D) finite-difference approximation for $\frac{\partial v}{\partial x}$ to first-order accuracy is $\partial_x v(x_i) = \frac{v_{i+1}-v_i}{\Delta x} + \mathcal{O}(\Delta x)$.

Standard schemes use one set of precomputed coefficients for all points in space, while more sophisticated methods alternate between different sets of coefficients according to local rules (20, 21). This discretization transforms Eq. **1** into a set of coupled ordinary differential equations of the form

$$\frac{\partial v_i}{\partial t} = F\left(t, x, v_1, \ldots, v_N\right) \quad [3]$$

that can be numerically integrated using standard techniques. The accuracy of the solution to Eq. **3** depends on Δx, converging to a solution of Eq. **2** as $\Delta x \to 0$. Qualitatively, accuracy requires that Δx is smaller than the spatial scale of the smallest feature of the field $v(x,t)$.

However, the scale of the smallest features is often orders of magnitude smaller than the system size. High-performance computing has been driven by the ever increasing need to accurately resolve smaller-scale features in PDEs. Even with petascale computational resources, the largest direct numerical simulation of a turbulent fluid flow ever performed has Reynolds number of order 1,000, using about 5×10^{11} grid points (22–24). Simulations at higher Reynolds number require replacing the physical equations with effective equations that model the unresolved physics. These equations are then discretized and solved numerically, e.g., using the method of lines. This overall procedure essentially modifies Eq. **2**, by changing the α_i to account for the unresolved degrees of freedom, replacing the discrete equations in Eq. **3** with a different set of discrete equations.

The main idea of this work is that unresolved physics can instead be learned directly from data. Instead of deriving an approximate coarse-grained continuum model and discretizing it, we suggest directly learning low-resolution discrete models that encapsulate unresolved physics. Rigorous mathematical work shows that the dimension of a solution manifold for a nonlinear PDE is finite (25, 26) and that approximate parameterizations can be constructed (27–29). If we knew the solution manifold, we could generate equation-specific approximations for the spatial derivatives in Eq. **2**, approximations that have the potential to hold even when the system is underresolved. In contrast to standard numerical methods, the coefficients $\alpha_i^{(n)}$ are equation dependent. Different regions in space (e.g., inside and outside a shock) will use different coefficients. To discover these formulas, we use machine learning: We first generate a training set of high-resolution data and then learn the discrete approximations to the derivatives in Eq. **2** from this dataset. This produces a tradeoff in computational cost, which can be alleviated by carrying out high-resolution simulations on small systems to develop local approximations to the solution manifold and using them to solve equations in much larger systems at significantly reduced spatial resolution.

Burgers' Equation. For concreteness, we demonstrate this approach with a specific example in 1 spatial dimension. Burgers' equation is a simple nonlinear equation which models fluid dynamics in 1D and features shock formation. In its conservative form, it is written as

$$\frac{\partial v}{\partial t} + \frac{\partial}{\partial x} J\left(v, \frac{\partial v}{\partial x}\right) = f(x,t), \quad J \equiv \frac{v^2}{2} - \eta \frac{\partial v}{\partial x}, \quad [4]$$

where $\eta > 0$ is the viscosity and $f(x,t)$ is an external forcing term. J is the flux. Generically, solutions of Eq. **4** spontaneously develop sharp shocks, with specific relationships between the shock height, width, and velocity (19) that define the local structure of the solution manifold.

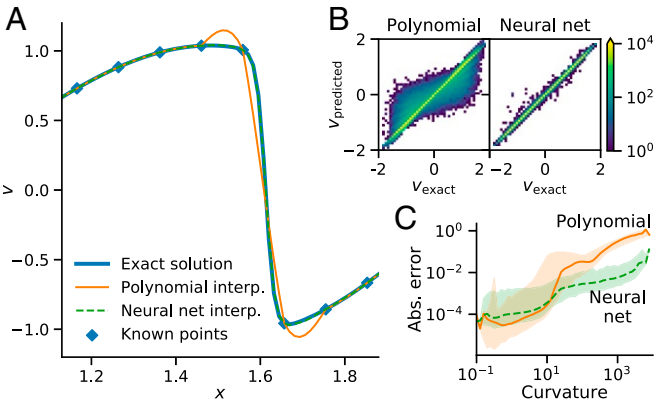

Fig. 1. Polynomial vs. neural net-based interpolation. (*A*) Interpolation between known points (blue diamonds) on a segment of a typical solution of Burgers' equation. Polynomial interpolation exhibits spurious "overshoots" in the vicinity of shock fronts. These errors compound when integrated in time, such that a naive finite-difference method at this resolution quickly diverges. In contrast, the neural network interpolation is so close to the exact solution that it cannot be visually distinguished. (*B*) Histogram of exact vs. interpolated function values over our full validation dataset. The neural network vastly reduces the number of poor predictions. (*C*) Absolute error vs. local curvature. The thick line shows the median and the shaded region shows the central 90% of the distribution over the validation set. The neural network makes much smaller errors in regions of high curvature, which correspond to shocks.

With this in mind, consider a typical segment of a solution to Burgers' equation (Fig. 1*A*). We want to compute the time derivative of the field given a low-resolution set of points (blue diamonds in Fig. 1). Standard finite-difference formulas predict this time derivative by approximating v as a piecewise-polynomial function passing through the given points (orange curves in Fig. 1). But solutions to Burger's equations are not polynomials: They are shocks with characteristic properties. By using this information, we can derive a more accurate, albeit equation-specific, formula for the spatial derivatives. For the method to work it should be possible to reconstruct the fine-scale solution from low-resolution data. To this end, we ran many simulations of Eq. **4** and used the resulting data to train a neural network. Fig. 1 compares the predictions of our neural net (details below and in *SI Appendix*) to fourth-order polynomial interpolation. This learned model is clearly far superior to the polynomial approximation, demonstrating that the spatial resolution required for parameterizing the solution manifold can be greatly reduced with equation-specific approximations rather than finite differences.

Models for Time Integration

The natural question to ask next is whether such parameterizations can be used for time integration. For this to work well, integration in time must be numerically stable, and our models need a strong generalization capacity: Even a single error could throw off the solution for later times.

To achieve this, we use multilayer neural networks to parameterize the solution manifold, because of their flexibility, including the ability to impose physical constraints and interpretability through choice of model architecture. The high-level aspects of the network's design, which we believe are of general interest, are described below. Additional technical details are described in *SI Appendix* and source code is available online at https://github.com/google/data-driven-discretization-1d.

Pseudolinear Representation. Our network represents spatial derivatives with a generalized finite-difference formula similar to Eq. **2**: The output of the network is a list of coefficients $\alpha_1, \ldots, \alpha_N$ such that the nth derivative is expressed as a pseudolinear filter, Eq. **2**, where the coefficients $\alpha_i^{(n)}(v_1, v_2, \ldots)$ depend on space and time through their dependence on the field values in the neighboring cells. Finding the optimal coefficients is the crux of our method.

The pseudolinear representation is a direct generalization of the finite-difference scheme of Eq. **2**. Moreover, exactly as in the case of Eq. **2**, a Taylor expansion allows us to guarantee formal polynomial accuracy. That is, we can impose that approximation errors decay as $\mathcal{O}(\Delta x^m)$ for some $m \leq N - n$, by layering a fixed affine transformation (*SI Appendix*). We found the best results when imposing linear accuracy, $m = 1$ with a 6-point stencil ($N = 6$), which we used for all results shown here. Finally, we note that this pseudolinear form is also a generalization of the popular essentially nonoscillatory (ENO) and weighted ENO (WENO) methods (20, 21), which choose a local linear filter (or a combination of filters) from a precomputed list according to an estimate of the solution's local curvature. WENO is an efficient, human-understandable, way of adaptively choosing filters, inspired by nonlinear approximation theory. We improve on WENO by replacing heuristics with directly optimized quantities.

Physical Constraints. Since Burgers' equation is an instance of the continuity equation, as with traditional methods, a major increase in stability is obtained when using a finite-volume scheme, ensuring the coarse-grained solution satisfies the conservation law implied by the continuity equation. That is, coarse-grained equations are derived for the cell averages of the field v, rather than its nodal values (19). During training we provide the cell average to the network as the "true" value of the discretized field.

Integrating Eq. **4**, it is seen that the change rate of the cell averages is completely determined by the fluxes at cell boundaries. This is an exact relation, in which the only challenge is estimating the flux given the cell averages. Thus, prediction is carried out in 3 steps: First, the network reconstructs the spatial derivatives on the boundary between grid cells (staggered grid). Then, the approximated derivatives are used to calculate the flux J using the exact formula Eq. **4**. Finally, the temporal derivative of the cell averages is obtained by calculating the total change at each cell by subtracting J at the cell's left and right boundaries. The calculation of the time derivative from the flux can also be done using traditional techniques that promote stability, such as monotone numerical fluxes (19). For some experiments, we use Godunov flux, inspired by finite-volume ENO schemes (20, 21), but it did not improve predictions for our neural networks models.

Dividing the inference procedure into these steps is favorable in a few aspects: First, it allows us to constrain the model at the various stages using traditional techniques; the conservative constraint, numerical flux, and formal polynomial accuracy constraints are what we use here, but other constraints are also conceivable. Second, this scheme limits the machine-learning part to reconstructing the unknown solution at cell boundaries, which is the main conceptual challenge, while the rest of the scheme follows either the exact dynamics or traditional approximations for them. Third, it makes the trained model more interpretable since the intermediate outputs (e.g., J or α_i) have clear physical meaning. Finally, these physical constraints contribute to more accurate and stable models, as detailed in the ablation study in *SI Appendix*.

Choice of Loss. The loss of a neural net is the objective function minimized during training. Rather than optimizing the prediction accuracy of the spatial derivatives, we optimize the accuracy of the resulting time derivative*. This allows us to incorporate physical constraints in the training procedure and directly optimize the final predictions rather than intermediate stages. Our loss is the mean-squared error between the predicted time derivative and labeled data produced by coarse graining the fully resolved simulations.

Note that a low value of our training loss is a necessary but not sufficient condition for accurate and stable numerical integration over time. Many models with low training loss exhibited poor stability when numerically integrated (e.g., without the conservative constraint), particularly for equations with low dissipation. From a machine-learning perspective, this is unsurprising: Imitation learning approaches, such as our models, often exhibit such issues because the distribution of inputs produced by the model's own predictions can differ from the training data (30). Incorporating the time-integrated solution into the loss improved predictions in some cases (as in ref. 9), but did not guarantee stability, and could cause the training procedure itself to diverge due to decreased stability in calculating the loss. Stability for learned numerical methods remains an important area of exploration for future work.

Learned Coefficients. We consider 2 different parameterizations for learned coefficients. In our first parameterization, we learn optimized time- and space-independent coefficients. These fixed

*For one specific case, namely the constant-coefficient model of Burgers' equation with Godunov flux limiting, trained models showed poor performance (e.g., not monotonically increasing with resample factor) unless the loss explicitly included the time-integrated solution, as done in ref. 9. Results shown in Figs. 3 and 4 use this loss for the constant-coefficient models with Burgers' equation. See details in *SI Appendix*.

coefficients minimize the loss when averaged over the whole training set for a particular equation, without allowing the scheme to adapt the coefficients according to local features of the solution. Below, we refer to these as "optimized constant coefficients." In our second parameterization, we allow the coefficients to be an arbitrary function of the neighboring field values $\{v_i\}$, implemented as a fully convolutional neural network (31). We use the exact same architecture (3 layers, each with 32 filters, kernel size of 5, and ReLU nonlinearity) for coarse graining all equations discussed in this work.

Example coefficients predicted by our trained models are shown in Fig. 2 and *SI Appendix*, Fig. S3. Both the optimized constant and data-dependent coefficients differ from baseline polynomial schemes, particularly in the vicinity of the shock. The neural network solutions are particularly interesting: They do not appear to be using 1-sided stencils near the shock, in contrast to traditional numerical methods such as WENO (21) which avoid placing large weights across discontinuities.

The output coefficients can also be interpreted physically. For example, coefficients for both $\partial v/\partial x$ (Fig. 2 *B*, *Inset*) and v (*SI Appendix*, Fig. S3*C*) are either right or left biased, opposite the sign of v. This is in line with our physical intuition: Burgers' equation describes fluid flow, and the sign of v corresponds to the direction of flow. Coefficients that are biased in the opposite direction of v essentially look "upwind," a standard strategy in traditional numerical methods for solving hyperbolic PDEs (19), which helps constrain the scheme from violating temporal causality. Alternatively, upwinding could be built into the model structure by construction, as we do in models which use Godunov flux.

Results

Burgers' Equation. To assess the accuracy of the time integration from our coarse-grained model, we computed "exact" solutions to Eq. **4** for different realizations of $f(x,t)$ at high enough resolution to ensure mesh convergence. These realizations of f were drawn from the same distribution as those used for training, but were not in the training set. Then, for the same realization of the forcing we solved the equation at a lower, possibly underresolved resolution using 4 different methods for calculating the flux: 1) A standard finite-volume scheme with either first-order or third-order accuracy; 2) a fifth-order upwind-biased WENO scheme with Godunov flux (21); 3) spatial derivatives estimated by constant optimized coefficients, with and without Godunov flux; and 4) spatial derivatives estimated by the space- and time-dependent coefficients, computed with a neural net.

Results are shown in Fig. 3. Fig. 3*A* compares the integration results for a particular realization of the forcing for different values of the resample factor, that is, the ratio between the number of grid points in the low-resolution calculation and that of the fully converged solution[†]. Our learned models, with both constant and solution-dependent coefficients, can propagate the solution in time and dramatically outperform the baseline method at low resolution. Importantly, the ringing effect around the shocks, which leads to numerical instabilities, is practically eliminated.

Since our model is trained on fully resolved simulations, a crucial requirement for our method to be of practical use is that training can be done on small systems, but still produce models that perform well on larger ones. We expect this to be the case, since our models, being based on convolutional neural networks, use only local features and by construction are translation invariant. Fig. 3*B* illustrates the performance of our model trained on the domain $[0, 2\pi]$ for predictions on a 10-times larger spatial domain of size $[0, 20\pi]$. The learned model generalizes well. For example, it shows good performance when function values are all positive in a region of size greater than 2π, which due to the conservation law cannot occur in the training dataset.

To make this assessment quantitative, we averaged over many realizations of the forcing and calculated the mean absolute error integrated over time and space. Results on the 10-times larger inference domain are shown in Fig. 3*C*: The solution from the full neural network has equivalent accuracy to increasing the resolution for the baseline by a factor of about $8\times$. Interestingly, even the simpler constant-coefficient method significantly outperforms the baseline scheme. The constant-coefficient model with Godunov flux is particularly compelling. This model is faster than WENO, because there is no need to calculate coefficients on the fly, with comparable accuracy and better numerical stability at coarse resolution, as shown in Figs. 3*A* and 4.

These calculations demonstrate that neural networks can carry out coarse graining. Even if the mesh spacing is much larger than the shock width, the model is still able to accurately propagate dynamics over time, showing that it has learned an internal representation of the shock structure.

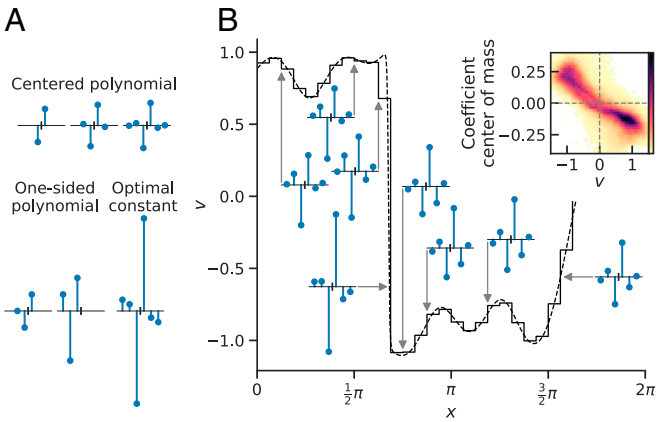

Fig. 2. Learned finite-volume coefficients for Burgers' equation. Shown are fixed and spatiotemporally varying finite-volume coefficients $\alpha_1^{(1)}, \ldots, \alpha_6^{(1)}$ (Eq. **2**) for $\partial v/\partial x$. (*A*) Various centered and 1-sided polynomial finite-volume coefficients, along with optimized constant coefficients trained on this dataset ($16\times$ resample factor in Fig. 3). The vertical scale, which is the same for all coefficient plots, is not shown for clarity. (*B*) An example temporal snapshot of a solution to Burgers' equation (Eq. **4**), along with data-dependent coefficients produced by our neural network model at each of the indicated positions on cell boundaries. The continuous solution is plotted as a dashed line, and the discrete cell-averaged representation is plotted as a piecewise constant solid line. The optimized constant coefficients are most similar to the neural network's coefficients at the shock position. Away from the shock, the solution resembles centered polynomial coefficients. (*B*, *Inset*) Relative probability density for neural network coefficient "center of mass" vs. field value v across our full test dataset. Center of mass is calculated by averaging the positions of each element in the stencil, weighted by the absolute value of the coefficient.

[†]Physically, the natural measure of the spatial resolution is with respect to the internal lengthscale of the equation which in the case of Burgers' equation is the typical shock width. However, since this analysis is meant to be applicable also to situations where the internal lengthscale is a priori unknown, we compare here to the lengthscale at which mesh convergence is obtained.

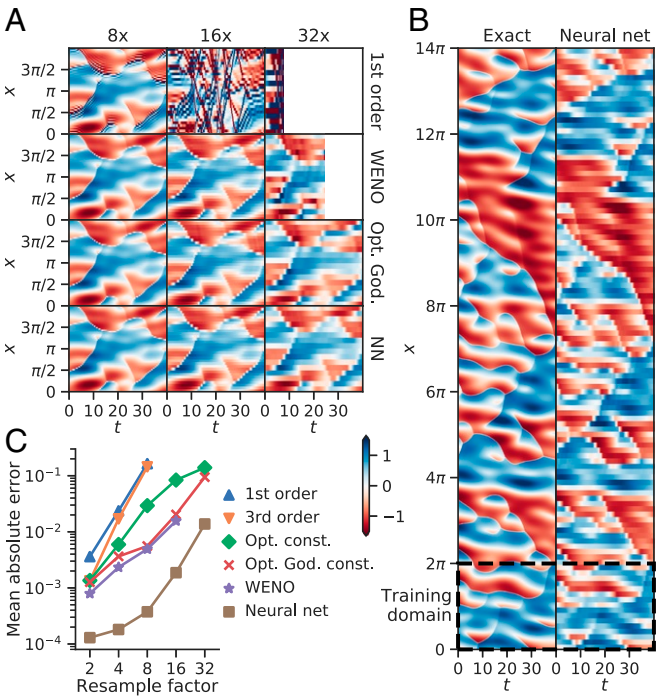

Fig. 3. Time integration results for Burgers' equation. (*A*) A particular realization of a solution at varying resolution solved by the baseline first-order finite-volume method, WENO, optimized constant coefficients with Godunov flux (Opt. God.), and the neural network (NN), with the white region indicating times when the solution diverged. Both learned methods manifestly outperform the baseline method and even outperform WENO at coarse resolutions. (*B*) Inference predictions for the 32× neural network model, on a 10 times larger spatial domain (only partially shown). The box surrounded by the dashed line shows the spatial domain used for training. (*C*) Mean absolute error between integrated solutions and the ground truth, averaged over space, times less than 15, and 10 forcing realizations on the 10-times larger inference domain. These metrics almost exactly match results on the smaller training domain $[0, 2\pi]$ (*SI Appendix*, Fig. S8). As ground truth, we use WENO simulations on a 1× grid. Markers are omitted if some simulations diverged or if the average error is worse than fixing $v = 0$.

train equation-specific estimators of the spatial derivative based on a coarse grid. These equations are essentially nondissipative, so we do not include a forcing term. The solution manifold is explored by changing the initial conditions, which are taken to be a superposition of long-wavelength sinusoidal functions with random amplitudes and phases (see *SI Appendix* for details).

To assess the accuracy of the integrated solution, for each initial condition we define "valid simulation time" as the first time that the low-resolution integrated solution deviates from the cell-averaged high-resolution solution by more than a given threshold. We found this metric more informative to compare across very different equations than absolute error.

Fig. 4 shows the median valid simulation time as a function of the resample factor. For all equations and resolutions, our neural network models have comparable or better performance than all other methods. The neural network is particularly advantageous at low resolutions, demonstrating its improved ability to solve coarse-grained dynamics. The optimized constant coefficients perform better at coarse resolution than baseline methods, but not always at high resolutions. Finally, at large enough resample factors the neural network approximations also fail to reproduce the dynamics, as expected. These results also hold on a 10-times larger spatial domain, as shown in *SI Appendix*, along with figures illustrating specific realizations and mean absolute error (*SI Appendix*, Figs. S8 and S9).

Discussion and Conclusion

It has long been remarked that even simple nonlinear PDEs can generate solutions of great complexity. But even very complex, possibly chaotic, solutions are not just arbitrary functions: They are highly constrained by the equations they solve. In mathematical terms, despite the fact that the solution set of a PDE is nominally infinite dimensional, the inertial manifold of solutions is much smaller and can be understood in terms of interactions between local features of the solutions to nonlinear PDEs. The dynamical rules for interactions between these features have been well studied over the past 50 years. Examples include, among many others, interactions of shocks in complex media, interactions of solitons (32), and the turbulent energy cascade (34).

Machine learning offers a different approach for modeling these phenomena, by using training data to parameterize the inertial manifold itself; said differently, it learns both the features and their interactions from experience of the solutions. Here we propose a simple algorithm for achieving this, motivated by coarse graining in physical systems. It is often the case that coarse graining a PDE amounts to modifying the weights in a discretized numerical scheme. Instead, we use known solutions to learn these weights directly, generating data-driven discretizations. This effectively parameterizes the solution manifold of the PDE, allowing the equation to be solved at high accuracy with an unprecedented low resolution.

Faced with this success, it is tempting to try and leverage the understanding the neural network has developed to gain new insights about the equation or its coarse-grained representation. Indeed, in Fig. 2 we could clearly interpret the directionality of the weights as an upwind bias, the pseudolinear representation providing a clear interpretation of the prediction in a physically sensible way. However, extracting more abstract insight from the network, such as the scaling relation between the shock height and width, is a difficult challenge. This is a general problem in the field of machine learning, which is under intensive current research (35, 36).

Our results are promising, but 2 challenges remain before our approach can be deployed at large scales. The first challenge is speed. We showed that optimized constant coefficients can already improve accuracy, but our best models rely on the flexibility of neural networks. Unfortunately, our neural nets use many more convolution operations than the single convolution required to implement finite differences, e.g., $32^2 = 1{,}024$

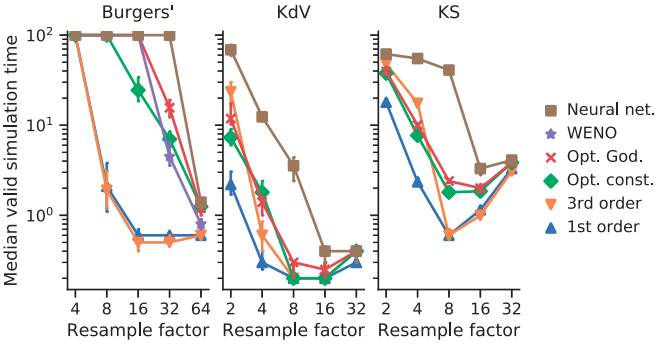

Fig. 4. Model performance across all of our test equations. Each plot shows the median time for which an integrated solution remains "valid" for each equation, defined by the absolute error on at least 80% of grid points being less than the 20th percentile of the absolute error from predicting all 0s. These thresholds were chosen so that valid corresponds to a relatively generous definition of an approximately correct solution. Error bars show the 95% confidence interval for the median across 100 simulations for each equation, determined by bootstrap resampling. Simulations for each equation were run out to a maximum of time of 100.

convolutions with a 5-point stencil between our second and third layers. We suspect that other machine-learning approaches could be dramatically faster. For example, recent work on a related problem—inferring subpixel resolution from natural images—has shown that banks of pretrained linear filters can nearly match the accuracy of neural nets with orders of magnitude better performance (37, 38). The basic idea is to divide input images into local patches, classify patches into classes based on fixed properties (e.g., curvature and orientation), and learn a single optimal linear filter for each class. Such computational architectures would also facilitate extracting physical insights from trained filters.

A second challenge is scaling to higher-dimensional problems and more complex grids. Here we showcased the approach for regular grids in 1D, but most problems in the real world are higher dimensional, and irregular and adaptive grids are common. We do expect larger potential gains in 2D and 3D, as the computational gain in terms of the number of grid points would scale like the square or the cube of the resample factor. Irregular grids may be more challenging, but deep learning methods that respect appropriate invariants have been developed both for arbitrary graphs (39) and for collections of points in 3D space (40). Similar to what we found here, we expect that hand-tuned heuristics for both gridding and grid coefficients could be improved upon by systematic machine learning. More broadly, data-driven discretization suggests the potential of data-driven numerical methods, combining the optimized approximations of machine learning with the generalization of physical laws.

ACKNOWLEDGMENTS. We thank Peyman Milanfar, Pascal Getreur, Ignacio Garcia Dorado, and Dmitrii Kochkov for collaboration and important conversations; Peter Norgaard and Geoff Davis for feedback on drafts of the manuscript; and Chi-Wang Shu for guidance on the implementation of WENO. Y.B.-S. acknowledges support from the James S. McDonnel postdoctoral fellowship for the study of complex systems. M.P.B. acknowledges support from NSF Grants DMS-1715477, ONR N00014-17-1-3029, as well as the Simons Foundation.

1. J. D. Jackson, *Classical Electrodynamics* (John Wiley & Sons, 1999).
2. D. Sholl, J. A. Steckel, *Density Functional Theory: A Practical Introduction* (Wiley & Sons, 2011).
3. C. J. Chen, *Fundamentals of Turbulence Modelling* (CRC Press, 1997).
4. M. Van Dyke, Perturbation methods in fluid mechanics (NASA STI/Recon Technical Report A 75, 1975).
5. R. Gonzalez-Garcia, R. Rico-Martinez, I. Kevrekidis, Identification of distributed parameter systems: A neural net based approach. *Comput. Chem. Eng.* **22**, S965–S968 (1998).
6. R. Rico-Martinez, I. Kevrekidis, K. Krischer, "Nonlinear system identification using neural networks: Dynamics and instabilities" in *Neural Networks for Chemical Engineers*, A. B. Bulsari, Ed. (Elsevier, Amsterdam, The Netherlands, 1995), pp. 409–442.
7. I. G. Kevrekidis, G. Samaey, Equation-free multiscale computation: Algorithms and applications. *Annu. Rev. Phys. Chem.* **60**, 321–344 (2009).
8. B. Kim *et al.*, Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum* **38**, 59–70 (2019).
9. J. Tompson, K. Schlachter, P. Sprechmann, K. Perlin, "Accelerating Eulerian fluid simulation with convolutional networks" in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup, Y. W. Teh, eds. (PMLR, 2017), vol. 70, pp. 3424–3433.
10. S. Rasp, M. S. Pritchard, P. Gentine, Deep learning to represent subgrid processes in climate models. *Proc. Natl. Acad. Sci. U.S.A.* **115**, 9684–9689 (2018).
11. S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci. U.S.A.* **113**, 3932–3937 (2016).
12. E. de Bezenac, A. Pajot, P. Gallinari, "Deep learning for physical processes: Incorporating prior scientific knowledge" in *International Conference on Learning Representations* (2018). https://iclr.cc/Conferences/2018/Schedule?showEvent=40. Accessed 11 July 2019.
13. B. Lusch, J. N. Kutz, S. L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics. *Nat. Commun.* **9**, 4950 (2018).
14. J. Morton, F. D. Witherden, A. Jameson, M. J. Kochenderfer, "Deep dynamical modeling and control of unsteady fluid flows" in *Advances in Neural Information Processing Systems*, S. Bengio *et al.*, Eds. (Curran Associates, Inc., 2018), vol. 31, pp. 9258–9268.
15. J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J. Fluid Mech.* **807**, 155–166 (2016).
16. A. D. Beck, D. G. Flad, C. D. Munz, Deep neural networks for data-driven turbulence models. arXiv:1806.04482 (15 June 2018).
17. A. Roberts, Holistic discretization ensures fidelity to Burgers' equation. *Appl. Numer. Math.* **37**, 371–396 (2001).
18. W. E. Schiesser, *The Numerical Method of Lines: Integration of Partial Differential Equations* (Academic Press, San Diego, 1991).
19. R. J. LeVeque, *Numerical Methods for Conservation Laws* (Birkhauser Verlag, 1992).
20. A. Harten, B. Engquist, S. Osher, S. R. Chakravarthy, Uniformly high order accurate essentially non-oscillatory schemes, III. *J. Comput. Phys.* **71**, 231–303 (1987).
21. C. W. Shu, "Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws" in *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, A. Quarteroni, Ed. (Springer, 1998), pp. 325–432.
22. M. Lee, R. D. Moser, Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$. *J. Fluid Mech.* **774**, 395–415 (2015).
23. M. Clay, D. Buaria, T. Gotoh, P. Yeung, A dual communicator and dual grid-resolution algorithm for petascale simulations of turbulent mixing at high Schmidt number. *Comput. Phys. Commun.* **219**, 313–328 (2017).
24. K. P. Iyer, K. R. Sreenivasan, P. K. Yeung, Reynolds number scaling of velocity increments in isotropic turbulence. *Phys. Rev. E* **95**, 021101 (2017).
25. P. Constantin, C. Foias, B. Nicolaenko, R. Temam, *Integral Manifolds and Inertial Manifolds for Dissipative Partial Differential Equations* (Springer Science & Business Media, 2012), vol. 70.
26. C. Foias, G. R. Sell, R. Temam, Inertial manifolds for nonlinear evolutionary equations. *J. Differ. Equations* **73**, 309–353 (1988).
27. M. Jolly, I. Kevrekidis, E. Titi, Approximate inertial manifolds for the Kuramoto-Sivashinsky equation: Analysis and computations. *Physica D Nonlinear Phenom.* **44**, 38–60 (1990).
28. E. S. Titi, On approximate inertial manifolds to the Navier-Stokes equations. *J. Math. Anal. Appl.* **149**, 540–557 (1990).
29. M. Marion, Approximate inertial manifolds for reaction-diffusion equations in high space dimension. *J. Dyn. Differ. Equations* **1**, 245–267 (1989).
30. S. Ross, D. Bagnell, "Efficient reductions for imitation learning" in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, Y. W. Teh, M. Titterington, eds. (PMLR, Chia Laguna Resort, Sardinia, Italy, 2010), vol. 9, pp. 661–668.
31. I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, *Deep Learning* (MIT Press, Cambridge, MA, 2016).
32. N. J. Zabusky, M. D. Kruskal, Interaction of "solitons" in a collisionless plasma and the recurrence of initial states. *Phys. Rev. Lett.* **15**, 240–243 (1965).
33. D. Zwillinger, *Handbook of Differential Equations* (Gulf Professional Publishing, 1998).
34. U. Frisch, *Turbulence: The Legacy of A. N. Kolmogorov* (Cambridge University Press, 1996).
35. M. Sundararajan, A. Taly, Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, D. Precup, Y. W. Teh, Eds. (PMLR, 2017), vol. 70, pp. 3319–3328.
36. A. Shrikumar, P. Greenside, A. Kundaje, "Learning important features through propagating activation differences" in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, D. Precup, Y. W. Teh, Eds. (PMLR, 2017), vol. 70, pp. 3145–3153.
37. Y. Romano, J. Isidoro, P. Milanfar, RAISR: Rapid and accurate image super resolution. *IEEE Trans. Comput. Imaging* **3**, 110–125 (2017).
38. P. Getreuer *et al.*, "BLADE: Filter learning for general purpose computational photography" in *2018 IEEE International Conference on Computational Photography (ICCP)* (IEEE, 2018).
39. J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, "Neural message passing for quantum chemistry" in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, D. Precup, Y. W. Teh, Eds. (PMLR, 2017), vol. 70, pp. 1263–1272.
40. C. R. Qi, L. Yi, H. Su, L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space" in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds. (Curran Associates, Inc., 2017), vol. 30.