14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020)
Virtual Congress: 11-15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

LEARNING COMPUTABLE MODELS FROM DATA

Suryanarayana Maddu[1,2,3], **Dominik Sturm**[1,2,3], **Bevan L. Cheeseman**[4], **Christian L. Müller**[5], **Ivo F. Sbalzarini**[1,2,3]

[1] Institute of Artificial Intelligence, Faculty of Computer Science, Technische Universit Dresden, Dresden, Germany

[2] Max Planck Institute of Molecular Cell Biology and Genetics, Center for Systems Biology Dresden, Dresden, Germany.

[3] Cluster of Excellence Physics of Life, TU Dresden, Germany

[4] ONI, Oxford, United Kingdom

[5]Flatiron Institute, New York City, USA.

Key words: Surrogate modeling, Differential operators, Neural Networks, ENO-WENO

Abstract. Numerical methods for approximately solving partial differential equations (PDE) are at the core of scientific computing. Often, this requires high-resolution or adaptive discretization grids to capture relevant spatio-temporal features in the PDE solution, e.g., in applications like turbulence, combustion, and shock propagation. Numerical approximation also requires knowing the PDE in order to construct problem-specific discretizations. Systematically deriving such solution-adaptive discrete operators, however, is a current challenge. Here we present an artificial neural network architecture for data-driven learning of problemand resolution-specific local discretizations of nonlinear PDEs. Our proposed method achieves numerically stable discretization of the operators in an unknown nonlinear PDE by spatially and temporally adaptive parametric pooling on regular Cartesian grids, and by incorporating knowledge about discrete time integration. Knowing the actual PDE is not necessary, as solution data is sufficient to train the network to learn the discrete operators. A once-trained neural architecture model can be used to predict solutions of the PDE on larger spatial domains and for longer times than it was trained for, hence addressing the problem of PDE-constrained extrapolation from data. We present demonstrative examples on long-term forecasting of hard numerical problems including equation-free forecasting of non-linear dynamics of forced Burgers problem on coarse spatio-temporal grids.

1 INTRODUCTION

Mathematical modeling of spatio-temporal systems as Partial Differential Equations (PDEs) has been invaluable in gaining mechanistic insight from real-world dynamical processes. This has been further fueled by rapid advancements in numerical methods and parallel computing, enabling the study of increasingly complex systems in real-world geometries. Numerical methods like finite-difference (FD), finite-volume (FV), and finite-element (FE) methods are routinely used to approximate the solutions of

PDEs on computational grids or meshes [6], [5]. Such mesh/gridbased methods rely on interpolating (in some basis) the PDE solution on discrete points [5]. This often requires high-resolution or adaptive-resolution grids with problem-specific discrete operators in order to accurately and stably represent small scales and high frequencies in the PDE solution.

In addition, often in science and engineering, measurement data from a dynamical processes in space and time are available, but a first-principles mathematical model is either difficult to formulate or too simplistic to realistically capture the data. Numerical simulation methods can then not be used to predict system behavior. This problem is particularly prevalent in areas such as biology, medicine, environmental science, economy, and finance.

1.1 Background and related work

Solution-adaptive discretizations of known nonlinear PDEs have been achieved using MLPs that learn problem-specific optimal parameters for classical numerical schemes on coarser grids [4]. The resulting consistent discretization on coarser grids can then be used to accelerate numerical simulations. Recent work on solution- and resolution-specific discretization of nonlinearPDEs has shown CNN filters that are able to generalize to larger spatial solution domains than what they have been trained on [1]. However, an over-complete set of CNN filters was used, which renders learning high-level discretization features computationally expensive and data-demanding. In all previous works, the underlying PDE and the associated discrete numerical fluxes for learning accurate solution-specific discretizations had to be known beforehand.

Here, we present an approach that does not require knowing the PDE and is purely data-driven. We present the neural network architecture for data-driven, solution-adaptive discretization of unknown nonlinear PDEs. Our framework uses a “Network In Network” (NIN) [3] architecture to learn high-level discretization features from local input patches with reduced data requirements.The NIN network works by sliding a Multi-Layer Perceptron (MLP) over the input patches to perform cascaded cross-channel parametric pooling that enables learning complex solution features.

2 MATHEMATICAL FORMULATION

In general, the spatio-temporal evolution of a state variable $u \in \mathbb{R}^d$ is given by a PDE of the form

$$\alpha_1 \frac{\partial u(x,t)}{\partial t} + \alpha_2 \frac{\partial^2 u(x,t)}{\partial t^2} = \mathcal{N}\left(\Xi, u, \partial_x u, \partial_x f(u), \partial_{xx} u, \partial_{xxx} u, u^2, \ldots\right), \tag{1}$$

where $\mathcal{N}(\cdot)$ is the nonlinear function that models the dynamics of a process, Ξ is the set of PDE coefficients (e.g., diffusion constants or viscosities), and $f(u)$ are the flux terms, e.g., $f(u) = u^2, f(u) = cu$. For the time derivative, we only consider the binary and mutually exclusive case, i.e., $(\alpha_1 = 0, \alpha_2 = 1)$ or $(\alpha_1 = 1, \alpha_2 = 0)$.

We can then compactly formulate the spatially discretized version of the PDE as the system of ODEs

$$\frac{\partial u_i}{\partial t} = \mathcal{N}_d\big(\mathbf{u}_m(x_i), \Xi, \Delta x\big) + O(\Delta x^{r_1}), \quad i = 1, \ldots, N_x, \tag{2}$$

where $u_i = u(x_i)$, $\mathbf{u}_m(x_i) = \{u(x_j) : x_j \in S_m(x_i)\}$, Δx the grid resolution of the spatial discretization, and

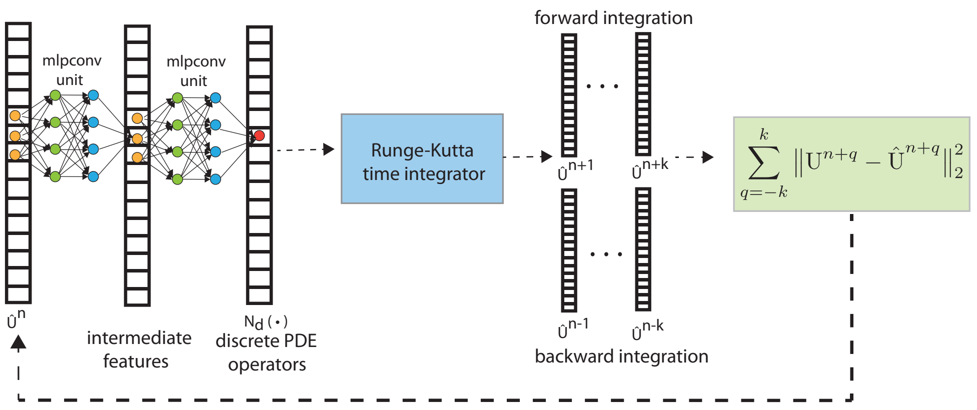

Figure 1: **Neural architecture for data-driven discretization:** The mlpconv unit performs parametric pooling by moving the MLP network across the input vector $\hat{\mathbf{U}}^n$ at time n to generate the feature maps. In our case, the features reaching the output layer are the local PDE discretization. The time integrators are then used to evolve the networks output over time for k steps forward and backward, which is then used to compute the loss.

$\mathcal{N}_d : \mathbb{R}^{2m+1} \to \mathbb{R}$ is the nonlinear discrete approximation of the continuous nonlinear right-hand side operator $\mathcal{N}(\cdot)$. The discrete approximation converges to the continuous operator with *spatial convergence rate* r_1 as $\Delta x \to 0$. Popular examples of spatial discretization methods include finite-difference, finite-volume, and finite-element methods. For simplicity of illustration, we only consider the case $\alpha_1 = 1$, $\alpha_2 = 0$; the other case is analogous.

Approximating the integral on the right-hand side of this map by quadrature, we find

$$u_i^{n+1} = \mathbf{T}_d\left(u_i^n, \mathcal{N}_d\left(\mathbf{u}_m^n(x_i), \Xi, \Delta x\right), \Delta t\right) + O(\Delta t^{r_2}), \quad n = 0, \ldots, N_t - 1, \tag{3}$$

where N_t is the total number of time steps. Here, $\mathbf{T}_d$ is the explicit discrete time integrator with time-step size Δt. Due to approximation of the integral by quadrature, the discrete time integrator converges to the continuous-time map with *temporal convergence rate* r_2 as $\Delta t \to 0$. Popular examples of explicit time-integration schemes include forward Euler, Runge-Kutta, and TVD Runge-Kutta methods. In this work, we only consider Runge-Kutta-type methods and their TVD variants as described by [8].

2.1 Optimization problem

The computation performed by a single mlpconv layer shown in Fig. 1 is given as,

$$\mathbf{y}_q^1 = \sigma\left(\mathbf{W}^1 \mathbf{x} + \mathbf{b}_1\right), \ldots\ldots \mathbf{y}_q^{n_l} = \sigma\left(\mathbf{W}^{n_l} \mathbf{y}_q^{n_l-1} + \mathbf{b}_n\right), \tag{4}$$

where $\theta = \{\mathbf{W}_q, \mathbf{b}_q\}_{q=1,2,\ldots,n_l}$ are the trainable weights and biases, respectively, and σ is the (nonlinear) activation function. The mlpconv layer can thus be seen as cascaded cross-channel parametric pooling on a normal convolutional layer [3], which allows complex and learnable interactions across channels for better abstraction of the input data.

We use the rich function approximation properties for neural networks [9] to approximate the discrete nonlinear spatial differential operator function $\mathcal{N}_d(\cdot)$ described in Eq. (3), which results in the following local non-linear parametric pooling:

$$\hat{u}_i^{n+1} = \mathbf{T}_d\left(\hat{u}_i^n, \mathcal{N}_\theta\big(\hat{\mathbf{u}}_m^n(x_i), \Xi\big), \Delta t\right), \tag{5}$$

where $\mathcal{N}_\theta : \mathbb{R}^{2m+1} \to \mathbb{R}$ are the local nonlinear mlpconv layer rules parameterized with weights θ. Based on Eq. 5, we formulate a loss function for learning the local nonlinear discretization rules $\mathcal{N}_\theta$:

$$\mathcal{L}_{MSE} = \sum_{n=1}^{N_t}\sum_{i=1}^{N_x}\sum_{k=-q}^{q} \gamma_k \left\| u_i^{n+k} - \left(\mathbf{T}_d^k\left(u_i^n, \mathcal{N}_\theta\big(\mathbf{u}_m^n, \Xi\big), \Delta t\right)\right)\right\|_2^2 \tag{6}$$

The scalar γ_k are decaying weights that account for the accumulating prediction error [7]. The positive integer q is the number of Runge-Kutta integration steps considered during optimization, which we refer to as the *training time horizon*. We penalize the noise estimates $\hat{\mathbf{N}}$ in order to prevent learning the trivial solution associated with the minimization problem given by Eq. (6). We also impose penalty on the weights of the network $\mathbf{W}$ in order to prevent over-fitting. The total loss is then given by

$$\text{Loss} = \mathcal{L}_{MSE} + \lambda_n \|\hat{\mathbf{N}}\|_F^2 + \lambda_{wd}\sum_{i=1}^{l} \|\mathbf{W}_i\|_F^2, \tag{7}$$

where l is the number of network layers, $\hat{\mathbf{N}} \in \mathbb{R}^{N_x \times N_t}$ is the matrix of point-wise noise estimates, and $\|\cdot\|_F$ is the Frobenius norm of a matrix. We find the choice $\lambda_n = 10^{-5}$ and $\lambda_{wd} = 10^{-8}$ to work well for all problems considered in this work. Training is done using an Adam optimizer [2] with learning rate $lr = 0.001$.

3 NUMERICAL EXPERIMENTS

We apply our neural architecture to learn solution-adaptive discretizations of nonlinear PDEs. For all problems discussed here, we use a single mlpconv unit with 3 hidden layers, each with 64 nodes and Exponential Linear Unit (ELU) nonlinearity. The input to the network is the solution $\mathbf{u}_m$ the stencil with $m = 3$ in all examples.

3.1 Learning non-trivial discretization

As a first simple example, we consider the problem of 1D advection equation as given by,

$$u_t + cu_x = 0 \tag{8}$$

where $c = 2$ is the constant advection velocity. We train our neural architecture to learn from data the stable computable rules for propagating sharp pulse. In Fig. 2, we show solution of sharp pulse propagation for different classical trivial and non-trivial numerics and their long-term predictions. We show that our neural architecture is able to learn local discetization rules on coarse grids and also account for the necessary up-winding numerics required for stable solution propagation. The data-driven computable model is more robust and accurate than higher-order solution adaptive WENO schemes on coarse grids.

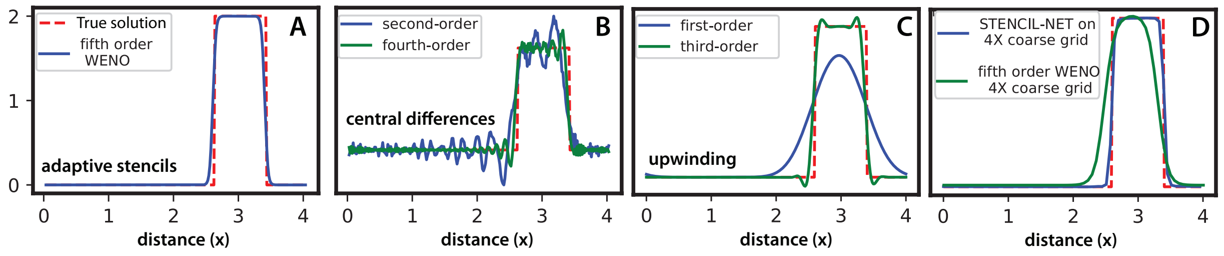

Figure 2: Numerical solutions of advection of a sharp pulse using different discretization schemes. All solutions are visualized at time $t = 3$ s.

3.2 A fast and accurate predictive surrogate

The forced Burgers' equation in 1D is given by the PDE:

$$\frac{\partial u}{\partial t} + \frac{\partial (u^2)}{\partial x} = D\frac{\partial^2 u}{\partial x^2} + f(x,t) \tag{9}$$

for the unknown function $u(x,t)$ with diffusion constant $D = 0.02$. Here, we use the forcing term

$$f(x,t) = \sum_{i=1}^{N} A_i \sin(\omega_i t + 2\pi l_i x/L + \phi_i) \tag{10}$$

with each parameter drawn independently and uniformly at random from its respective range: $A = [-0.1, 0.1]$, $\omega = [-0.4, 0.4]$, $\phi = [0, 2\pi]$, and $N = 20$. The domain size L is set to 2π (i.e., $x \in [0, 2\pi]$) with periodic boundary conditions, and $l = \{2,3,4,5\}$. We use a smooth initial condition $u(x,0) = \exp(-(x-3)^2)$.

The forced Burgers' equation with a nonlinear forcing term can produce rich and sharply varying solutions. The forcing term also introduces randomness that can help explore the solution manifold [1]. This presents an ideal challenge for testing the method's discretization capabilities. We train our architecture on fifth-order WENO solutions of the forced Burgers' equation at different spatial resolutions $\Delta x_c = (C\Delta x)$, where C is the sub-sampling factor, and Δx the resolution of the fine-grid solution (Fig. 3) on Fig. 3, we show a comparison between fifth-order accurate WENO solution on a fine grid ($\Delta x = L/N_x, N_x = 256$ with domain length $L = 2\pi$ and the corresponding neural network predictions on coarsened grids with $\Delta x_c = C\Delta x$, for sub-sampling factors $C = \{2,4,8\}$. Our method learns local discretization rules and produces stable computable solutions on coarser spatio-temporal grids for longer times with little compromise on accuracy

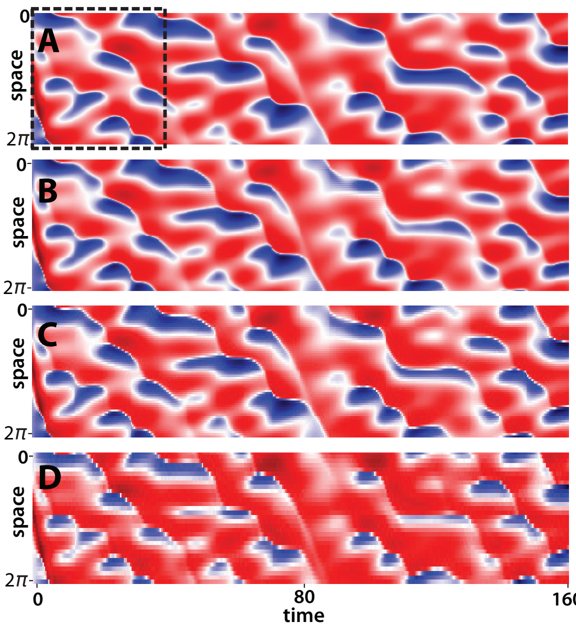

Figure 3: **Forced Burgers' prediction with neural network**: (A) Fifth-order WENO solution with $N_x = 256$. (B,C,D) Predictions of neural architecture on $(2\times, 4\times, 8\times)$ coarser grids, respectively. The dashed box in A is the domain used for training.

4 CONCLUSIONS

In this work, we proposed a neural network architecture that uses patch-wise parametric pooling and discrete time integration constraints to learn coarse-grained computable models directly from spatio-temporal data, without knowledge of the underlying symbolic PDE.

Our method fills this niche by providing a purely data-driven, equation-free way of learning predictive computable models of complex nonlinear space-time dynamics. It can be used to produce rapid coarse-grained predictions beyond the space and time horizon it was trained for. Beyond the applications shown here, we anticipate the fast and accurate data-driven predictions to be useful in areas including real-time computational steering, computer graphics, phase-space exploration, uncertainty quantification, and computer vision

Future generalizations of the idea include an extension to 2D and 3D problems, multi-resolution numerical schemes, and mesh-free particle methods. We hope that the present results provide motivation for future works and for including neural network based surrogates into existing simulations frameworks.

REFERENCES

[1] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

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

[3] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[4] S. Mishra. A machine learning framework for data driven acceleration of computations of differential equations. *arXiv preprint arXiv:1807.09519*, 2018.

[5] P. Moin. *Fundamentals of engineering numerical analysis*. Cambridge University Press, 2010.

[6] S. Patankar. *Numerical heat transfer and fluid flow*. Taylor & Francis, 2018.

[7] S. H. Rudy, J. N. Kutz, and S. L. Brunton. Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *Journal of Computational Physics*, 396:483–506, 2019.

[8] C.-W. Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In *Advanced numerical approximation of nonlinear hyperbolic equations*, pages 325–432. Springer, 1998.

[9] M. Telgarsky. Neural networks and rational functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3387–3393. JMLR. org, 2017.