12th International Symposium on Turbulence and Shear Flow Phenomena (TSFP12)
Osaka, Japan (Online), July 19-22, 2022

SUBGRID-SCALE SURROGATE MODELING OF IDEALIZED ATMOSPHERIC FLOWS: A DEEP LEARNED APPROACH USING HIGH-RESOLUTION SIMULATION DATA

Muralikrishnan Gopalakrishnan Meena
National Center for Computational Sciences
Oak Ridge National Laboratory
Oak Ridge, TN, 37831-6008, USA
gopalakrishm@ornl.gov

Matthew R. Norman
National Center for Computational Sciences
Oak Ridge National Laboratory
Oak Ridge, TN, 37831-6008, USA
normanmr@ornl.gov

David M. Hall
NVIDIA
Lafayette, CO, USA
dhall@nvidia.com

ABSTRACT

We introduce a deep learned subgrid-scale surrogate model for dry, stratified idealized atmospheric flows from high-resolution simulation data. Deep neural networks (NNs) are used to model the full state differences between a coarse resolution simulation and a high-resolution simulation, run simultaneously with the coarse resolution simulation forced by the high-resolution simulation, hence capturing both dissipative and anti-dissipative effects. The NN model is able to accurately capture the state differences in *a priori* tests outside the training regime. In *a posteriori* tests intended for production use, the NN coupled coarse simulation is accurate compared to the high-resolution simulation over a finite period in time. With the accumulation of the errors, the NN-coupled simulation becomes computationally unstable after a while. These surrogate models further pave the way for formulating stable, complex, physics-based NN models which are driven by traditional subgrid-scale turbulence closure models.

Introduction

The challenge for numerically analyzing atmospheric turbulence is tied to the broad range of multi-scale physics constituting such flows. More specifically, the added complexity of density stratification involving cloud microphysics and turbulence leads to greater challenges (Wyngaard, 1992). Capturing the complex subgrid-scale processes at the scale of clouds is important for constructing accurate global climate models (Schneider *et al.*, 2017). There have been various efforts in the atmospheric science community to capture the effects of the subgrid-scale processes using some form of parametrization. Such parametrizations, like large-eddy simulations, rely on filtering the high-wavenumber (subgrid-scale) structures and resolving only a coarser domain with embedded subgrid-scale modeling (SGS modeling) (Pressel *et al.*, 2017). The SGS models parametrize the viscous dissipation from the filtered subgrid-scale structures. Cloud resolving models (CRM) are another example for such a modeling framework used in simulating global climate models (Randall, 2013; Hannah *et al.*, 2020). CRMs are comprised of a high-resolution cloud model which parametrizes deep convection for the fluid state at that location in the host model.

Such models are limited by the appropriate choice of the parametrization. Moreover, simulating such models with the global climate model at large climate time scales is still a computational challenge, even with the current generation of accelerator-based high-performance computing systems (Norman *et al.*, 2021). Recently, the use of artificial intelligence has been at the forefront to form surrogate models parametrizing the subgrid-scale processes (Duraisamy *et al.*, 2019; Brunton *et al.*, 2020; Kochkov *et al.*, 2021), taking advantage of the availability of high-resolution simulation data. These models generally rely on components in the eddy-viscosity term of the filtered Navier-Stokes equation, which concentrate on capturing the dissipative effects (Gamahara & Hattori, 2017; Rasp *et al.*, 2018; Maulik *et al.*, 2019; Yuval & O' Gorman, 2020). Moreover, deep learning techniques have been used to capture and parametrize the subgrid-scale processes in global climate models (Krasnopolsky *et al.*, 2013; Gentine *et al.*, 2018; Brenowitz & Bretherton, 2019).

As stratified turbulent flows encountered in cloud physics also are comprised of significant anti-dissipative effects, it is beneficial to capture both the dissipative and anti-dissipative effects through the SGS closures. We approach this problem by directly modeling the full state difference between a coarse simulation and a simultaneously running very fine resolution simulation, enabling us to capture both the dissipative and anti-dissipative effects from the subgrid-scale processes. An overview of the modeling framework is shown in Fig. 1. We use the two-dimensional (2D) inviscid, non-hydrostatic, compressible Euler equations with density stratification for solving canonical LES and mesoscale atmospheric flows and to generate subgrid-scale data as the difference between high- and low-resolution flow field data.

Instead of handling the problem on the global spatial domain all at once, we choose to handle it locally using stencils of data (not unlike the nature of convolutions), to reduce the deep learning model size, improve generalization, and increase the number of samples available for training. To avoid chaotic divergence between the two simulations, the states of coarse sim-

ulation are updated by the states of fine simulation after each time step. The dissipative and anti-dissipative components are embedded into the coarse simulation as it is integrated with the high-resolution simulation as a parallel driver. The current modeling approach for 2D flows is useful as a pilot project to develop the surrogate modeling techniques, particularly since 2D CRMs have been found to be accurate and computationally cheaper parametrizations for global climate models. Incorporating cloud microphysics to the present supervised approach could serve as a novel alternative to the SGS models of CRMs, capturing both dissipative and anti-dissipative effects of the subgrid-scale processes.

General procedure

We use NNs to augment the flow states of the coarse resolution simulation, q_{coarse}, with states of a simultaneously running fine resolution simulation, $\mathbf{q}_{\text{fine}}$. The majority of subgrid-scale effects missing from the coarse resolution simulation are captured by the full state difference

$$\Delta\mathbf{q} = \overline{\mathbf{q}}_{\text{fine}} - \mathbf{q}_{\text{coarse}}, \tag{1}$$

where $\overline{\mathbf{q}}_{\text{fine}}$ denotes $\mathbf{q}_{\text{fine}}$ interpolated to the coarse resolution grid stencil by summing over the stencil around a given cell in the fine grid to reduce the dimension to that of the coarse grid. A sample portrayal of the procedure is shown in Fig. 1. Given a coarse resolution data, the objective is to obtain a NN learned state difference, $\Delta\mathbf{q}_{\text{NN}}$, and correct the coarse states to get

$$\mathbf{q}_{\text{NN}} = \mathbf{q}_{\text{coarse}} + \Delta\mathbf{q}_{\text{NN}}. \tag{2}$$

The procedure resembles a mapping from fine grid to coarse grid. Modeling the full state difference enables us to capture both the dissipative and anti-dissipative effects of the subgrid-scale processes. In the following section, we will introduce the flow problem we use to demonstrate the modeling technique. We will discuss the details of the NN architecture and training procedures in Sections and .

Fluid flow problem

We consider a 2D thermal collision of two hot and cold thermals as a sample problem to demonstrate the modeling framework (Norman, 2021). The colliding thermals create strong discontinuities, strong winds, and significant turbulent regimes as the flow evolves in time. The flow evolution is described by the 2D, dry, compressible, non-hydrostatic Euler equations, given by

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho u\\ \rho w\\ \rho\theta\end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uw\\ \rho u\theta\end{bmatrix} + \frac{\partial}{\partial z}\begin{bmatrix}\rho w\\ \rho uw\\ \rho w^2+p-p_H\\ \rho w\theta\end{bmatrix} = \begin{bmatrix}0\\ 0\\ -(\rho-\rho_H)g\\ 0\end{bmatrix} \tag{3}$$

$$\rho_H = -\frac{1}{g}\frac{\partial p_H}{\partial z} \tag{4}$$

where ρ is density, u and w are wind velocities in x- and z-directions (horizontal and vertical directions, respectively), θ is potential temperature, $p = C_0(\rho\theta)^\gamma$ is pressure (C_0 is a constant, $\gamma = c_p/c_v$, c_p is specific heat of dry air at constant pressure, and c_v is specific heat of dry air at constant volume), p_H and ρ_H are the hydrostatic pressure and density, and g is acceleration due to gravity. Equation 3 can be represented in vector form as $\partial_t\mathbf{q} + \partial_x\mathbf{f} + \partial_z\mathbf{g} = \mathbf{s}$ with the state vector $\mathbf{q} = [\rho \quad \rho u \quad \rho w \quad \rho\theta]^T$. We only model the perturbed scalars, ρ' and $(\rho\theta)'$), removing the dominant underlying hydrostatic balance described by Eq. (4). Although inviscid, the Euler equations form the basis of atmospheric flow simulations, and inherent numerical dissipation maintains stability. Moreover, the 2D setting is an apt choice for idealized simulation as the majority of idealized test cases in literature are 2D.

The flow is perturbed from a neutrally stratified dry atmosphere. The domain is of size $(x,z) \in [0,20]\times[0,10]$ km. Slip, solid wall boundary conditions are prescribed to the top and bottom walls. The left and right walls are prescribed with periodic boundary conditions. The time steps are adjusted so that the Courant–Friedrichs–Lewy number is 0.8. We choose the grid resolution, $n_x \times n_z$, of the coarse simulation to be 200×100 (grid spacing of 100 m in all directions). The fine resolution domain has a resolution of 1000×500 (grid spacing of 20 m in all directions), resulting in a grid mapping ratio of $5\times$ between the coarse and fine domains. We simulate the flows till 1000 secs. After the initial laminar regime dominated with anti-dissipative effects, the flow evolves to a highly turbulent regime from about 500 secs onward. The temporal regimes consisting of both anti-dissipative and dissipative effects makes the colliding thermal test case a useful model problem to test the capability of the current framework to capture these effects. Further details of the test case, solver, and numerical schemes can be found in Norman (2021).

Neural network setup

We deploy a supervised deep learning technique to model the relation between the inputs, a local stencil of coarse resolution states ($\mathbf{q}_{\text{coarse}}$), and the outputs, state difference in the center cell of the stencil with the fine resolution states ($\Delta\mathbf{q}$). We model this mapping between $\mathbf{q}_{\text{coarse}}$ and $\Delta\mathbf{q}$ using deep NNs of the form $\mathcal{N}(\mathbf{q}_{\text{coarse}},\mathbf{w})$ with $\mathbf{w}$ representing the parameters ("weights") of the NN. The learning process involves optimizing the parameters $\mathbf{w}$ of the NN by minimizing the loss computed by the function $L[\Delta\mathbf{q},\mathcal{N}(\mathbf{q}_{\text{coarse}},\mathbf{w})]$. In the current analysis, we use a squared L_2 norm as the loss function, $L[\Delta\mathbf{q},\mathcal{N}(\mathbf{q}_{\text{coarse}},\mathbf{w})] = \|\Delta\mathbf{q}-\mathcal{N}(\mathbf{q}_{\text{coarse}},\mathbf{w})\|_2^2$.

A stencil-based approach is used to build the architecture of $\mathcal{N}(\mathbf{q}_{\text{coarse}},\mathbf{w})$. We use the stencil around each grid cell in the coarse simulation to model the $\Delta\mathbf{q}$ at the respective locations as shown in Fig. 2. The framework of $\mathcal{N}(\mathbf{q}_{\text{coarse}},\mathbf{w})$ is composed of 36 inputs, the 4 flow states of the 3×3 grid stencil around a given coarse cell. The outputs (4 values) are the $\Delta\mathbf{q}$ at the given grid cell (the center of the stencil of inputs). The hidden layer(s) of the NN are comprised of neurons of various configurations, enabling the nonlinear activation function to capture the relation between the inputs and outputs (Goodfellow *et al.*, 2016). We choose 3 different types of NN architectures for the hidden layer(s) with a *Leaky ReLU* activation function having a slope of 0.1 (Glorot *et al.*, 2011): (a) a single layer of 45 neurons (single-layer model), (b) 10 layers with 45 neurons/layer in a ResNet-based configuration (ResNet model) (He *et al.*, 2016), and (c) 10 layers with 45 neurons/layer in a DenseNet-based configuration (DenseNet model) (Huang *et al.*, 2017). We tune various hyperparameters of the NN architecture to arrive at these architectures. These hyperparameters for the current approach involve the number of neurons per layer, number of layers, activation function, values for regularization at the input, hidden, and output layers, optimization routine, and batch size of training samples.

The single-layer model is used as a shallow, simple NN model to test the capability of NNs to capture the grid mapping. Drop-out regularization is applied to the input layer to stabilize the single-layer model. Adding more layers to this

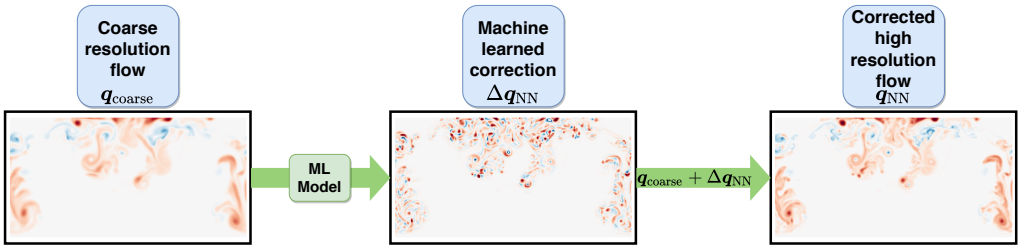

Figure 1. An overview of the NN-based approach to capture the subgrid-scale effects missing in a coarse resolution simulation.

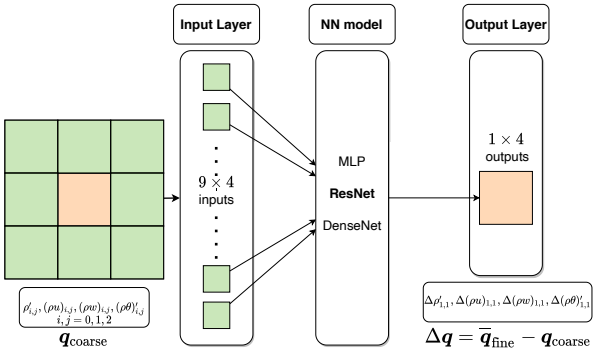

Figure 2. Illustration of the network model for modeling the full state difference using stencil data.

model reduced the stability of the model during testing without considerable increase in accuracy and ability to capture the nonlinear physics. This lead us to the last two models. The last two models are deep NNs, used to increase the complexity of the network and accurately capture the highly non-linear physics. These deep NNs take advantage of the so-called skip-connections to enable sparse, complex, non-linear relations between the input and output (He *et al.*, 2016).

We have found that the simple, small framework of using a stencil-based input and output to the network is advantageous over convolutional neural networks for higher accuracy and better stability of the model during testing in unknown data regions. The architecture intrinsically is similar to convolution operations. Moreover, the stencil-based approach mimics a higher-order finite difference scheme. The current formulation does not assume a dissipation- or advection-based mapping for the relationship between $\mathbf{q}_{\text{coarse}}$ and $\Delta\mathbf{q}$. Our approach is to let the NN capture the complex mathematical relations between the states of the center cell and neighboring cells with the state difference at the center cell in the coarse domain.

It is important to note that the actual domain of constraint for the fine-scale model is *wider* than the 3×3 coarse cells over the temporal domain of a coarse-grid time step. Therefore, full constraint of the state difference is likely not possible. Still, we wished to maintain a simpler NN model with minimal inputs, and results will show that the 3×3 stencil is effective in providing a constraining set of inputs.

Training and validation of model

We have learned that sampling the data for training is a crucial step for building an accurate and stable NN model, which we elaborate here. The training dataset is comprised of 1M samples collected between the temporal regime of $[0, 900]$ secs (both laminar and turbulent regimes). The small size of the stencil-based framework allows us to use significantly fewer samples for training. We have found that the accuracy of the model is significantly deteriorated when trained with data randomly sampled in space and time, particularly in regions with high gradients of the states. Thus, we use a total variation (TV) weighted random sampling technique. The TV of a sample at the cell (i, j) in the 2D domain is given by $V(\mathbf{q}_{i,j}) = \sum_{k=-1}^{0} \{|\mathbf{q}_{i+(2k+1),j} - \mathbf{q}_{i,j}| + |\mathbf{q}_{i,j+(2k+1)} - \mathbf{q}_{i,j}|\}$. TV identifies regions with high variations in states. As the majority of the domain is comprised of quiescent regions with small changes in the states during the initial transient regime, these regions are not captured by the TV weighting. This limitation makes the models biased against the background mean. To alleviate this bias issue, we use a $50:50$ split to curate samples with and without high TV weighting. Selecting the threshold to identify the $50:50$ split poses further challenges as the flow evolves through both laminar and turbulent regimes, leading to a broad range of values for the states over time. Thus, the threshold for identifying the $50:50$ split is manually adjusted for various flow regimes.

The input and output data for the NN is scaled according to min-max normalization. For a vector of input and output variables of the network, $\mathbf{Q} = [\mathbf{q} \quad \Delta\mathbf{q}]^T$, the normalization is given by $\tilde{\mathbf{Q}} = (\mathbf{Q} - \min_{\text{train}} \mathbf{Q})/(\max_{\text{train}} \mathbf{Q} - \min_{\text{train}} \mathbf{Q})$, where $\tilde{\mathbf{Q}}$ is the rescaled vector and the operators $\min_{\text{train}}$ and $\max_{\text{train}}$ refers to the minimum and maximum operators over the training sample space of the variables, respectively.

We train the model in Python environment using PyTorch and then deploy the model in the C++ solver with GPU acceleration for production usage. The "NAdam" optimizer is used for optimizing the weights of the NNs (Dozat, 2016). For training the 1M data points (with a $70:30$ split between training and validation loops, and random shuffling of training samples), we use mini-batches of 1024 samples. Even with the use of mini-batches, the model incurs high variance in accuracy over the training epochs. We have learned that using a higher learning rate at initial training epochs till saturation of the model accuracy and then lowering the value helps to reduce this high variance in accuracy. Furthermore, for the deep NNs, we also rely on an averaging technique to ensemble the model weights across various training epochs to alleviate the high variance in the training accuracy. After a regime of stable accuracy is reached over epochs, we collect ensembles of model weights from models with error below a particular threshold. The weights of the final model are an ensemble of these collected weights. The procedure helps average the gradient basin in the optimization manifold, aiding to form a stable solution. This procedure has similarities with the stochastic weighted averaging (SWA) method introduced in Izmailov *et al.* (2018).

Results

For an effective model predicting the subgrid-scale effects, the model should be accurate as well as computationally stable when tested in temporal regimes outside the training regime. As with subgrid-scale turbulence models, we perform both *a priori* and *a posteriori* testing (Piomelli *et al.*, 1988) of

the NN model in turbulent regimes outside the training regime. The results reveal that the ResNet-based model gives the most accurate and computationally stable NN-coupled solver. Herein, we only show the results of the ResNet-based model, while we also discuss insights gained from the other models.

A priori testing For the *a priori* testing, the results from the NN model is directly compared with the expected solution, evaluated independently without coupling the NN model with the numerical solver. We observe the behavior of the model by using data from a full snapshot in a turbulent regime outside the training regime, as shown in Fig. 3 for θ'. The ResNet model predicts the state difference accurately with an overall Euclidean error norm of $L_2 = \|\Delta\mathbf{q} - \Delta\mathbf{q}_{\text{NN}}\|_2/\|\Delta\mathbf{q}\|_2 = 0.195$. We also plot the L_1 error, $\varepsilon = |\Delta\mathbf{q} - \Delta\mathbf{q}_{\text{NN}}|/\max(|\Delta\mathbf{q}|)$, to study the results locally in space, as shown in Fig. 3 (c). The main structures are accurately captured whereas the smaller-scale features have lower accuracy. Nonetheless, the maximum error is 10%, which is isolated to a small region in the lower-left side of the domain. The DenseNet model performed close to that by the ResNet model for both the random samples (normalized errors have a mean and standard deviation of $\mathscr{O}(10^{-3})$ and $\mathscr{O}(10^{-2})$, respectively) and full flow field ($L_2 = 0.243$) test datasets. Surprisingly, the single-layer model did give accurate results (normalized errors have a mean and standard deviation of $\mathscr{O}(10^{-2})$ and $\mathscr{O}(10^{-2})$, respectively, and $L_2 = 0.629$) but did not perform as well as the ResNet model.

A posteriori testing The *a posteriori* testing is done by implementing the NN model with the flow simulation. A sample depiction of the NN-coupled solver is shown in Fig. 4. The fine simulation is used as correction until the testing time, t_0, after which the NN model is invoked. Therefore, at time t_0, the model state is essentially perfect with respect to the fine-grid model. We use the PyTorch C++ frontend API[1] to couple the NN model with the C++ solver, performing the so-called in-the-loop ML integration with the scientific solver. A GPU kernel is used to update the $\mathbf{q}_{\text{coarse}}$ with the machine learned $\Delta\mathbf{q}_{\text{NN}}$ after every time step, following Eq. 2. Even without special batching consideration of online inferencing and refactoring the code for GPU performance, we attain speed-ups up to $8\times$ when using the NN-coupled solver compared to a fine resolution simulation. Since the current discussion is concentrated more towards the scientific aspects of the model, we reserve the full computational aspects of the framework for future investigation.

We invoke the NN model at time $t_0 = 900$ secs, which is outside the training regime and well within the turbulent regime. Recall that the training data was sampled from $t \in [0, 900]$ seconds. The results we show here are those after the simulation has run for $t_0 + 5$ secs, which is approximately 25 coarse-grid time steps after t_0. The results for θ' of the coarse simulation with correction using fine simulation, without correction (after t_0), and with correction using the NN model are shown in Figs. 5 (a-c), respectively. Without correction after t_0 (Fig. 5 (b)), diffusive effects start to dominate the coarse simulation and the subgrid-scale effects are not captured. The NN model accurately predicts the flow evolution with the sharp gradients, maintaining a difference of $L_2 = 0.118$ and maximum L_1 difference of 20%, as shown in Figs. 5 (c-d), respectively. Note that, the chaotic divergence of the modeled flow and its difference from the ideal states are convolved inseparably in the results of the L_1 and L_2 difference norms. The main coherent structures are accurately captured and the larger L_1 errors are from the small-scale fluctuations, isolated to a few regions in the flow field. Note that these regions of high local errors are located at high-gradient regions. Note that we also test the models in the initial laminar regime where anti-dissipative effects dominate, which are not discussed here. We have found the models effectively capture high-gradient regions encountered in the laminar flow regimes as compared to the coarse simulations.

With spatial accuracy comes the next challenge of temporal stability of the NN-coupled solver. The high spatial accuracy attained by the NN does not remain numerically stable indefinitely. As the flow evolves, the simulation becomes unstable. The normalized error of the rms value of wind speeds and total kinetic energy (KE) over time are shown in Fig. 6. The rms value (spatial) of wind speeds and the total kinetic energy are defined by $\mathbf{u}_{\text{rms}} = [1/(n_x n_z)\sum_{i,j}^{n_x,n_z}\mathbf{u}_{i,j}^2]^{0.5}$ and Total KE $= 0.5[\sum_{i,j}^{n_x,n_z}(\rho u)_{i,j}^2 + (\rho w)_{i,j}^2]$, respectively, where $\mathbf{u} = [\rho u \quad \rho w]^T$. Overestimates of variability, as seen previously in Fig. 5 (d), accumulate over time until the solver becomes unstable at $t_0 + 17$ secs, which is after ~ 85 coarse-grid time steps. The error in $(\rho u)_{\text{rms}}$ by the coarse simulation with correction using the NN model is always lesser than that of the coarse simulation without correction while in the stable regime (see Fig. 6 (a) solid and dashed blue lines, respectively). The error in $(\rho w)_{\text{rms}}$ by NN model is comparable to that of the coarse simulation without correction. The total KE shows that the NN-based correction indeed outperforms the coarse simulation without correction during most of the stable temporal regime.

The results suggests that the NN-based correction is highly accurate over a finite time interval, whereas further research needs to be done to make the solver stable from the accumulation of the errors at high-gradient regions. Note the "spike" at the far right of Fig. 6. After this time, the model becomes unstable. We note that the DenseNet model is able to perform similarly to that by the ResNet model, becoming unstable after $t_0 + 12$ secs but with higher levels of error before it becomes unstable. Whereas, the single-layer model is highly unstable, as it becomes unstable after only $t_0 + 3$ secs. This inability of the single-layer model emphasizes the need for a complex formulation of the NN model with multiple layers and skip-connections. We provide insights from our ongoing work on improving the stability of the NN-coupled solver in the Conclusion section.

Concluding remarks and future work

We use deep learning to generate surrogate models from high-resolution data for capturing subgrid-scale effects in dry, stratified turbulence in idealized atmospheric flows. Starting from the inviscid Euler equations with density stratification, we generate subgrid-scale data as a difference between high- and low-resolution flow field data. The coarse resolution simulation is kept in sync with the fine resolution simulation via coarsening interpolation after each time step to avoid chaotic non-linear divergence. The results show that a deep ResNet type model using the stencil information around a given grid cell is able to accurately perform *a posteriori* tests and is stable over a significant time period. Such accuracy is consistent over both laminar as well as turbulent regimes, which are dominated by anti-dissipative and disspative effects, respectively. The present supervised approach serves as a novel alternative to the subgrid-scale models in large eddy simulations and CRMs, capturing both dissipative and anti-dissipative effects from the subgrid-scale phenomena.

[1] https://pytorch.org/cppdocs/. A sample implementation of the API is provided in https://github.com/muralikrishnangm/pytorch-cpp-example.

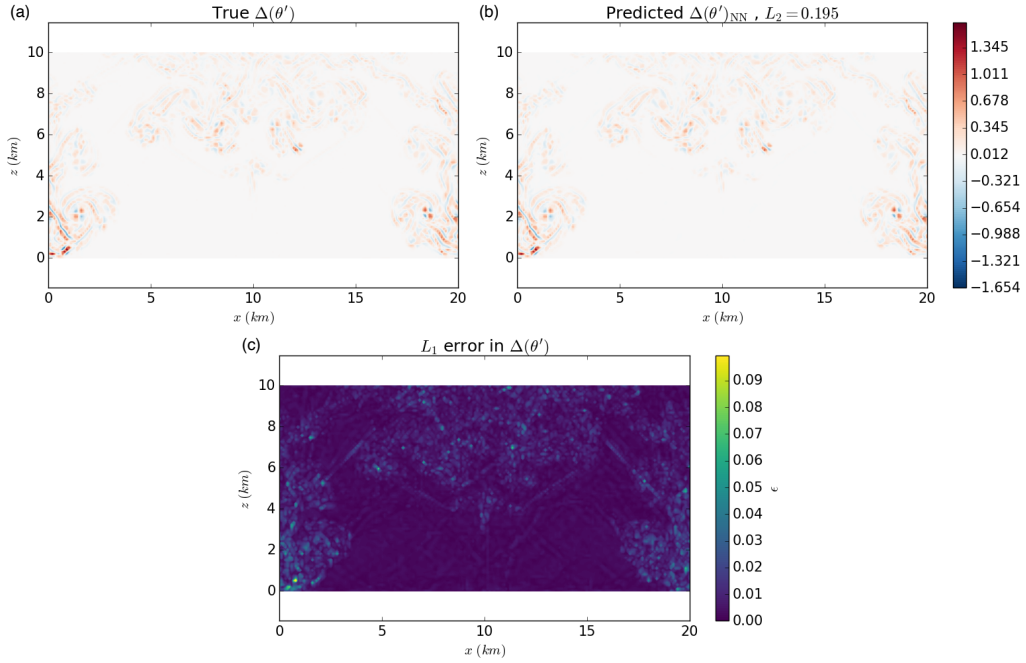

Figure 3. Full flow field prediction by the neural network model. Potential temperature corrections are shown along with the L_1 error norm, $\varepsilon = |\Delta\mathbf{q} - \Delta\mathbf{q}_{\mathrm{NN}}|/\max(|\Delta\mathbf{q}|)$ and L_2 norm of the full flow field, $L_2 = \|\Delta\mathbf{q} - \Delta\mathbf{q}_{\mathrm{NN}}\|_2/\|\Delta\mathbf{q}\|_2$.

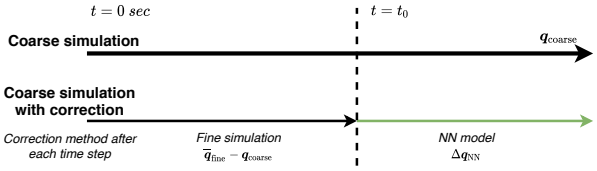

Figure 4. Sample depiction of *a posteriori* testing using the neural network model.

We are currently investigating alternate formulations to enhance the stability of the model. Complex diffusion-based approaches inspired from traditional SGS models are an alternative approach in the physics-informed realm. Such models along with enforcing conservation of various conserved quantities can help stabilize the NN coupled solver (Ling *et al.*, 2016). Moreover, recurrent neural networks and generative adversarial networks can be used for correcting the emulation. One aspect we have not explored in this manuscript is the explainability and interpretability of the NN models. Recently, various techniques have been used to understand the physical implications of machine learning, particularly in the meteorological domains (McGovern *et al.*, 2019). Finally, performant and scalable integration of the model to the solver in hybrid architectures (Partee *et al.*, 2021) is of utmost urgency for practical production use of such models in climate models.

Acknowledgments This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

REFERENCES

Brenowitz, N. D. & Bretherton, C. S. 2019 Spatially extended tests of a neural network parametrization trained by coarse-graining. *J. Adv. Modeling Earth Sys.* **11** (8), 2728–2744.

Brunton, S. L. *et al.* 2020 Machine learning for fluid mechanics. *Ann. Rev. Fluid Mechanics* **52**, 477–508.

Dozat, T. 2016 Incorporating Nesterov momentum into Adam.

Duraisamy, K. *et al.* 2019 Turbulence modeling in the age of data. *Ann. Rev. Fluid Mechanics* **51**, 357–377.

Gamahara, M. & Hattori, Y. 2017 Searching for turbulence models by artificial neural network. *Phy. Rev. Fluids* **2** (5), 054604.

Gentine, P. *et al.* 2018 Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters* **45** (11), 5742–5751.

Glorot, X. *et al.* 2011 Deep sparse rectifier neural networks.

Goodfellow, I. *et al.* 2016 *Deep learning*. MIT press.

Hannah, W. M. *et al.* 2020 Initial results from the super-parameterized E3SM. *J. Adv. Modeling Earth Sys.* **12** (1).

He, K. *et al.* 2016 Deep residual learning for image recognition.

Huang, G. *et al.* 2017 Densely connected convolutional networks.

Izmailov, P. *et al.* 2018 Averaging weights leads to wider optima and better generalization.

Kochkov, D. *et al.* 2021 Machine learning–accelerated computational fluid dynamics. *Proc. National Academy of Sciences* **118** (21).

Krasnopolsky, V. M. *et al.* 2013 Using ensemble of neural networks to learn stochastic convection parameterizations for climate and numerical weather prediction models from data simulated by a cloud resolving model. *Adv. Artificial Neural Systems* **2013**.

Ling, J. *et al.* 2016 Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J. Fluid Mechanics* **807**, 155–166.

Maulik, R. *et al.* 2019 Subgrid modelling for two-dimensional turbulence using neural networks. *J. Fluid Mechanics* **858**,

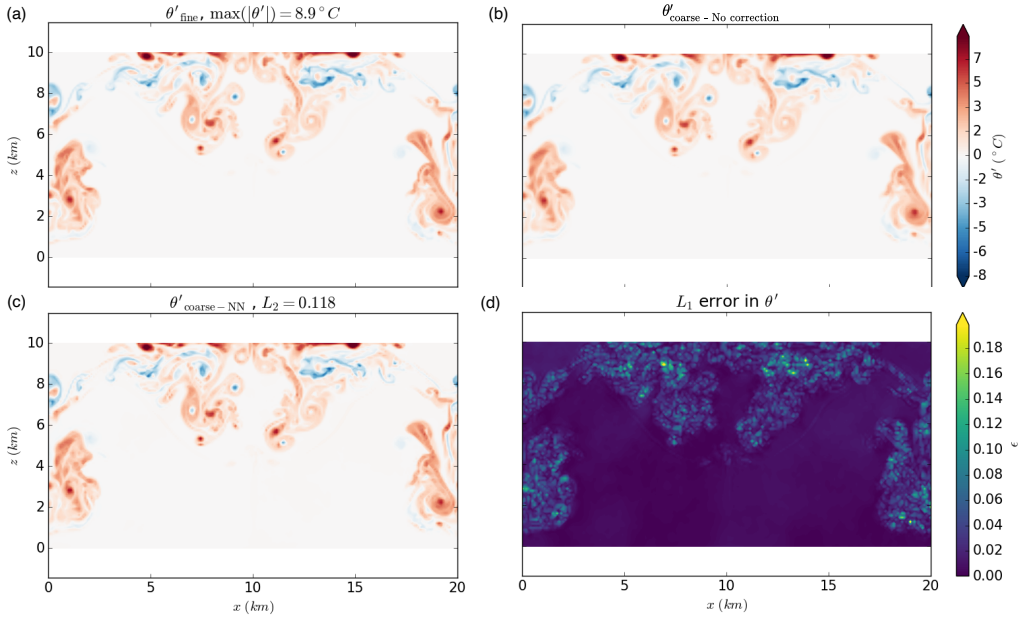

Figure 5. Full flow state prediction using neural network based emulation in turbulent regime outside of training regime. Snapshots of potential temperature perturbation (θ') after 5 secs from the initial state of testing regime are shown for: (a) coarse simulation with correction using fine simulation (θ'_{fine}), (b) coarse simulation without correction ($\theta'_{\text{coarse-no correction}}$), and (c) coarse simulation with correction using NN model ($\theta'_{\text{coarse-NN}}$), along with the $L_2 = \|\mathbf{q}_{\text{fine}} - \mathbf{q}_{\text{coarse-NN}}\|_2 / \|\mathbf{q}_{\text{fine}}\|_2$ difference norm of the flow field. (d) L_1 difference of $\theta'_{\text{coarse-NN}}$ with respect to θ'_{fine}, $\varepsilon = |\mathbf{q}_{\text{fine}} - \mathbf{q}_{\text{coarse-NN}}| / \max(|\mathbf{q}_{\text{fine}}|)$.

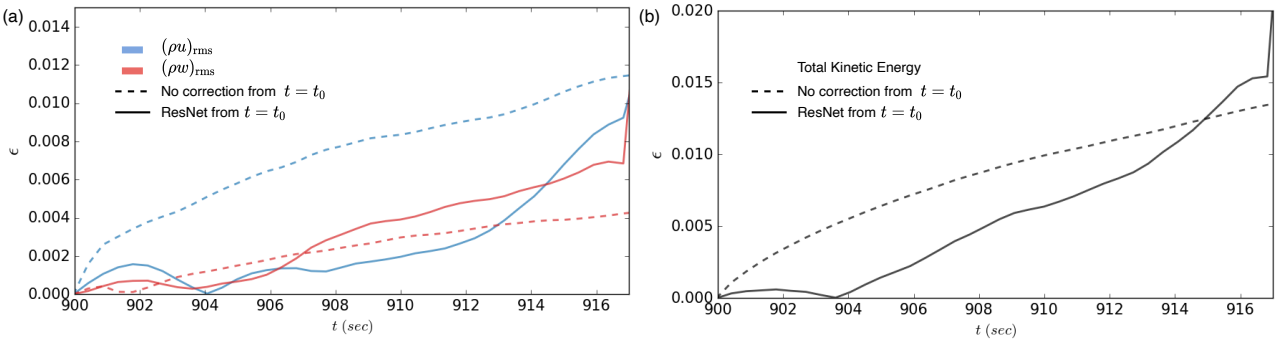

Figure 6. Comparison of L_1 errors in the (a) rms values of flow velocity and (b) total kinetic energy over time. The dashed lines are the variables from the coarse simulation without correction and the solid lines are those of the coarse simulation with correction using NN model. The L_1 error norm for the time series data is defined by $\varepsilon = |\mathbf{f}_{\text{coarse-no correction/coarse-NN}} - \mathbf{f}_{\text{fine}}| / |\mathbf{f}_{\text{fine}}|$, where $\mathbf{f} = [\mathbf{u}_{\text{rms}} \quad \text{Total KE}]^T$.

122–144.

McGovern, A. *et al.* 2019 Making the black box more transparent: Understanding the physical implications of machine learning. *Bulletin of the American Meteorological Society* **100** (11), 2175–2199.

Norman, M. R. 2021 A high-order WENO-limited finite-volume algorithm for atmospheric flow using the ADER-differential transform time discretization. *Quarterly J. Royal Meteorological Society* **147** (736), 1661–1690.

Norman, M. R. *et al.* 2021 Unprecedented cloud resolution in a GPU-enabled full-physics atmospheric climate simulation on OLCFs summit supercomputer. *Int. J. HPC Applications* **36(1)**, 93–105.

Partee, S. *et al.* 2021 Using machine learning at scale in HPC simulations with SmartSim: An application to ocean climate modeling.

Piomelli, U. *et al.* 1988 Model consistency in large eddy simulation of turbulent channel flows. *Physics of Fluids* **31** (7), 1884–1891.

Pressel, K. G. *et al.* 2017 Numerics and subgrid-scale modeling in large eddy simulations of stratocumulus clouds. *J. Adv. Modeling Earth Sys.* **9** (2), 1342–1365.

Randall, D. 2013 Beyond deadlock. *Geophysical Research Letters* **40** (22), 5970–5976.

Rasp, S. *et al.* 2018 Deep learning to represent subgrid processes in climate models. *Proc. National Academy of Sciences* **115** (39), 9684–9689.

Schneider, Tapio *et al.* 2017 Climate goals and computing the future of clouds. *Nature Climate Change* **7** (1), 3–5.

Wyngaard, J. C 1992 Atmospheric turbulence. *Ann. Rev. Fluid Mechanics* **24** (1), 205–234.

Yuval, J. & O' Gorman, P. A. 2020 Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nature Comm.* **11** (1), 1–10.