

Enforcing Analytic Constraints in Neural-Networks Emulating Physical Systems

Tom Beucler,^{1, 2,*} Michael Pritchard,¹ Stephan Rasp,³ Jordan Ott, Pierre Baldi,⁴ and Pierre Gentine²

¹*Department of Earth System Science, University of California, Irvine, CA, USA*

²*Department of Earth and Environmental Engineering, Columbia University, New York, NY, USA*

³*Technical University of Munich, Munich, Germany*

⁴*Department of Computer Science, University of California, Irvine, CA, USA*

(Dated: January 29, 2021)

Neural networks can emulate nonlinear physical systems with high accuracy, yet they may produce physically-inconsistent results when violating fundamental constraints. Here, we introduce a systematic way of enforcing nonlinear analytic constraints in neural networks via constraints in the architecture or the loss function. Applied to convective processes for climate modeling, architectural constraints enforce conservation laws to within machine precision without degrading performance. Enforcing constraints also reduces errors in the subsets of the outputs most impacted by the constraints.

Main Repository: <https://github.com/raspstephan/CBRAIN-CAM>

Figures and Tables: https://github.com/tbeucler/CBRAIN-CAM/blob/master/notebooks/tbeucler_devlog/042_Figures_PRL_Submission.ipynb

I. INTRODUCTION

Many fields of science and engineering (e.g., fluid dynamics, hydrology, solid mechanics, chemistry kinetics) have exact, often *analytic*, closed-form constraints, i.e. constraints that can be explicitly written using analytic functions of the system's variables. Examples include translational or rotational invariance, conservation laws, or equations of state. While physically-consistent models should enforce constraints to within machine precision, data-driven algorithms often fail to satisfy well-known constraints that are not explicitly enforced. In particular, neural networks (NNs, [1]), powerful regression tools for nonlinear systems, may severely violate constraints on individual samples while optimizing overall performance.

Despite the need for physically-informed NNs for complex physical systems [2–5], enforcing *hard* constraints [6] has been limited to physical systems governed by specific equations, such as advection equations [7–9], Reynolds-averaged Navier-Stokes equations [10, 11], boundary conditions of idealized flows [12], or quasi-geostrophic equations [13]. To address this gap, we introduce a systematic method to enforce analytic constraints arising in more general physical systems to within machine precision, namely the Architecture-Constrained NN or ACnet. We then compare ACnets to unconstrained (UCnets) and loss-constrained NNs (LCnets, in which soft constraints are added through a penalization term in the loss function [e.g., 14–16]) in the particular case of climate modeling, where the system is high-dimensional and the constraints (such as mass and energy conservation) are few but crucial [17].

II. THEORY

A. Formulating the Constraints

Consider a NN mapping an input vector $\mathbf{x} \in \mathbb{R}^m$ to an output vector $\mathbf{y} \in \mathbb{R}^p$. Enforcing constraints is easiest for linearly-constrained NNs, i.e. NNs for which the constraints (\mathcal{C}) can be written as a linear system of rank n :

$$(\mathcal{C}) \stackrel{\text{def}}{=} \left\{ \mathbf{C} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \mathbf{0} \right\}. \quad (1)$$

We call $\mathbf{C} \in \mathbb{R}^{n \times \mathbb{R}^{m+p}}$ the constraints matrix, and use bold font for vectors and tensors to distinguish them from scalars. For the regression problem to have non-unique solutions, the number of independent constraints n has to be strictly less than $m + p$.

In Figure 1, we consider a generic regression problem subject to analytic constraints (\mathcal{C}) that may be nonlinear, and propose how to formulate a linearly-constrained NN. First, define the regression's inputs \mathbf{x}_0 and outputs \mathbf{y}_0 , which respectively become the *temporary* NN's features and targets. Then (**Formulation 1**), write the constraints (\mathcal{C}) as an identically zero function \mathbf{c} of the inputs, the outputs, and additional parameters \mathbf{z} the constraints may involve. We recommend non-dimensionalizing all variables to facilitate the design, interpretation, and performance of the loss function. While the function \mathbf{c} may be nonlinear, it can always be written as the sum of: (1) terms \mathbf{x} that *only* depend on inputs and (2) terms \mathbf{y} that depend on inputs, outputs and additional parameters. Thus the constraints can be written as:

$$\mathbf{c}(\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}) = \mathbf{C} \begin{bmatrix} \mathbf{x}(\mathbf{x}_0) \\ \mathbf{y}(\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}) \end{bmatrix}, \quad (2)$$

where \mathbf{C} is a matrix. Finally (**Formulation 2**), choose \mathbf{x} and \mathbf{y} as the NN's new inputs and outputs. If \mathbf{x} and \mathbf{y} are not bijective functions of $(\mathbf{x}_0, \mathbf{y}_0)$, add variables to the

* tom.beucler@gmail.com

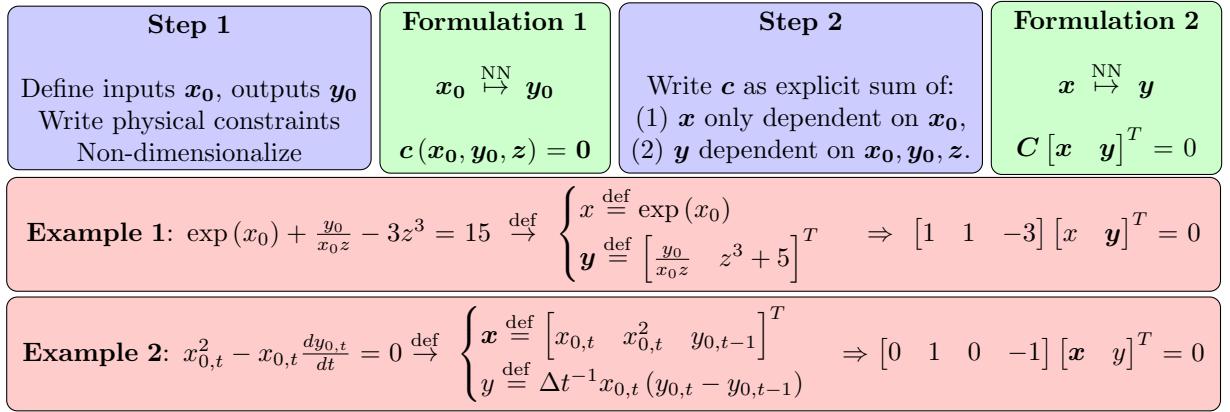


FIG. 1. Framework to treat constrained regression problems using linearly-constrained NNs, with two examples: (1) A regression problem with one nonlinear constraint, and (2) a time-prediction problem with one differential nonlinear constraint that we discretize using a forward Euler method of timestep Δt . Note that the choice of \mathbf{x}, \mathbf{y} , and \mathbf{C} is not unique.

NN’s inputs and outputs to recover \mathbf{x}_0 and \mathbf{y}_0 after optimization (e.g., we add $x_{0,t}$ and $y_{0,t-1}$ to \mathbf{x} in **Example 2**). We are now in a position to build a computationally-efficient NN that satisfies the linear constraints (\mathcal{C}).

B. Enforcing the Constraints

Consider a NN trained on preexisting measurements of \mathbf{x} and \mathbf{y} . For simplicity’s sake, we measure the quality of its output \mathbf{y}_{NN} using a standard mean-squared error (MSE) misfit:

$$\text{MSE}(\mathbf{y}_{\text{Truth}}, \mathbf{y}_{\text{NN}}) \stackrel{\text{def}}{=} \|\mathbf{y}_{\text{Err}}\|_2 \stackrel{\text{def}}{=} \frac{1}{p} \sum_{k=1}^p y_{\text{err},k}^2, \quad (3)$$

where we have introduced the error vector, defined as the difference between the NN’s output and the “truth”:

$$\mathbf{y}_{\text{Err}} \stackrel{\text{def}}{=} \mathbf{y}_{\text{NN}} - \mathbf{y}_{\text{Truth}}. \quad (4)$$

In the reference case of an “unconstrained network” (UCnet), we optimize a multi-layer perceptron [e.g., 18, 19] using MSE as its loss function \mathcal{L} . To enforce the constraints (\mathcal{C}) within NNs, we consider two options:

(1) Constraining the loss function (LCnet, soft constraints): We first test a *soft* penalization of the NN for violating physical constraints using a penalty \mathcal{P} , defined as the mean-squared residual from the constraints:

$$\begin{aligned} \mathcal{P}(\mathbf{x}, \mathbf{y}_{\text{NN}}) &\stackrel{\text{def}}{=} \left\| \mathbf{C} \begin{bmatrix} \mathbf{x} \\ \mathbf{y}_{\text{NN}} \end{bmatrix} \right\|_2, \\ &= \frac{1}{n} \sum_{i=1}^n \left(\sum_{j=1}^m C_{ij} x_j + \sum_{k=1}^p C_{i(k+m)} y_{\text{NN},k} \right)^2, \end{aligned} \quad (5)$$

and given a weight $\alpha \in [0, 1]$ in the loss function \mathcal{L} :

$$\mathcal{L}(\alpha) = \alpha \mathcal{P}(\mathbf{x}, \mathbf{y}_{\text{NN}}) + (1 - \alpha) \text{MSE}(\mathbf{y}_{\text{Truth}}, \mathbf{y}_{\text{NN}}). \quad (6)$$

(2) Constraining the architecture (ACnet, hard constraints): Alternatively, we treat the constraints as *hard* and augment a standard, optimizable NN with n fixed conservation layers that sequentially enforce the constraints (\mathcal{C}) to within machine precision (Figure 2), while keeping the MSE as the loss function:

$$(\text{ACnet}) \Rightarrow \left\{ \min \text{MSE} \text{ s.t. } \mathbf{C} [\mathbf{x} \quad \mathbf{y}_{\text{NN}}]^T = \mathbf{0} \right\} \quad (7)$$

The optimizable NN calculates a “direct” output whose size is $p - n$. We then calculate the remaining output’s components of size n as exact “residuals” from the constraints. Concatenating the “direct” and “residual” vectors results in the full output \mathbf{y}_{NN} that satisfies the constraints to within machine precision. Since our loss uses the full output \mathbf{y}_{NN} , the gradients of the loss function are passed through the constraints layers during optimization, meaning that the final NN’s weights and biases depend on the constraints (\mathcal{C}). ACnet improves upon the common approach of calculating “residual” outputs *after* training because ACnet exposes the NN to “residual” output data *during* training (SM C.3). A possible implementation of the constraints layer uses custom (Tensorflow in our case) layers with fixed parameters that solve the system of equations (\mathcal{C}), in row-echelon form, from the bottom to the top row (SM B.1). Note that we are free to choose which outputs to calculate as “residuals”, which introduces n new hyperparameters (SM B.2).

C. Linking Constraints to Performance

Intuitively, we might expect the NNs’ performance to improve once we enforce constraints arising in physical systems with few degrees of freedom, but this may not hold true with many degrees of freedom. We formalize the link between constraints and performance by: (1) decomposing the NN’s prediction into the “truth” and

error vectors following equation 4; and (2) assuming that constraints exactly hold for the “truth” (no errors in measurement). This yields:

$$\mathbf{C} \begin{bmatrix} \mathbf{x} \\ \mathbf{y}_{\text{NN}} \end{bmatrix} \stackrel{\text{def}}{=} \overbrace{\mathbf{C} \begin{bmatrix} \mathbf{x} \\ \mathbf{y}_{\text{Truth}} \end{bmatrix}}^{\mathbf{0}} + \mathbf{C} \begin{bmatrix} \mathbf{0} \\ \mathbf{y}_{\text{Err}} \end{bmatrix}. \quad (8)$$

Equation 8 relates how much the constraints are violated to the error vector. More explicitly, if we measure performance using the MSE, we may square each component of Equation 8. The resulting equation links how much physical constraints are violated to the squared error for each constraint of index $i \in [1, n]$:

$$\underbrace{\left(\mathbf{C} \begin{bmatrix} \mathbf{x} \\ \mathbf{y}_{\text{NN}} \end{bmatrix} \right)_i^2}_{\text{Physical constraints}} = \underbrace{\sum_{k=1}^p C_{i(k+m)}^2 y_{\text{Err},k}^2}_{\text{Squared-error}>0} + \underbrace{\sum_{k=1}^p \sum_{l \neq k} C_{i(k+m)} C_{i(l+m)} y_{\text{Err},k} y_{\text{Err},l}}_{\text{Cross-term}} \quad (9)$$

In ACnets, we strictly enforce physical constraints, setting the left-hand side of Equation 9 to 0, within numerical errors. As the squared error is positive-definite, the cross-term is always negative in ACnets as both terms sum up to 0. It is difficult to predict the cross-term before optimization, hence Equation 9 does not provide a-priori predictions of performance, even for ACnets. Instead, it links how much the NN violates constraints to how well it predicts outputs that appear in the constraints equations: the more negative the cross-term, the larger the squared error for a given violation of physical constraints.

III. APPLICATION

A. Convective Parameterization for Climate Modeling

The representation of subgrid-scale processes in coarse-scale, numerical models of the atmosphere, referred to as

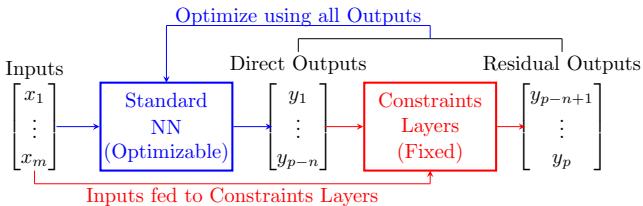


FIG. 2. ACnet: Direct outputs are calculated using a standard NN, while the remaining outputs are calculated as residuals from the fixed constraints layers.

subgrid *parameterization*, is a large source of error and uncertainty in numerical weather and climate prediction [e.g., 20, 21]. Machine-learning algorithms trained on fine-scale, process-resolving models can improve subgrid parameterizations by faithfully emulating the effect of fine-scale processes on coarse-scale dynamics [e.g., 22–25, see Section 2 of Rasp [26] for a detailed review]. The problem is that none of these parameterizations exactly follow conservation laws (e.g., conservation of mass, energy). This is critical for long-term climate projections, as the spurious energy production may both exceed the projected radiative forcing from greenhouse gases and result in large thermodynamic drifts or biases over a long time-period. Motivated by this shortcoming, we build a NN parameterization of convection and clouds that we *constrain* to conserve 4 quantities: column-integrated energy, mass, longwave radiation, and shortwave radiation.

B. Model and Data

We use the Super-Parameterized Community Atmosphere Model 3.0 [27] to simulate the climate for two years in aquaplanet configuration [28], where the surface temperatures are fixed with a realistic equator-to-pole gradient [29]. Following [24]’s sensitivity tests, we use 42M samples from the simulation’s first year to train the NN (training set) and 42M samples from the simulation’s second year to validate the NN (validation set). Since we use the validation set to adjust the NN’s hyperparameters and avoid overfitting, we additionally introduce a test set using 42M different samples from the simulation’s second year to provide an unbiased estimator of the NNs’ performances. Note that each sample represents a single atmospheric column at a given time, longitude, and latitude.

C. Formulating the Conservation Laws in a Neural Network

The parameterization’s goal is to predict the rate at which sub-grid convection vertically redistributes heat and water based on the current large-scale thermodynamic state. We group all variables describing the local climate in an input vector \mathbf{x} of size 304 (5 vertical profiles with 30 levels each, prescribed large-scale conditions \mathbf{LS} for all profiles of size 150, and 4 scalars):

$$\mathbf{x} = [(\mathbf{q}_v, \mathbf{q}_l, \mathbf{q}_i, \mathbf{T}, \mathbf{v}, \mathbf{LS}, p_s, S_0) \text{ SHF LHF}]^T, \quad (10)$$

where all variables are defined in SM A. We then concatenate the time-tendencies from convection and the additional variables involved in the conservation laws to form an output vector \mathbf{y} of size 216 (7 vertical profiles with 30 levels, followed by 6 scalars):

$$\mathbf{y} = [\dot{\mathbf{q}}_v \ \dot{\mathbf{q}}_l \ \dot{\mathbf{q}}_i \ \dot{\mathbf{T}} \ \dot{\mathbf{T}}_{KE} \ \mathbf{lw} \ \mathbf{sw} \ \mathbf{LW}_t \ \mathbf{LW}_s \ \mathbf{SW}_t \ \mathbf{SW}_s \ P \ P_i]^T, \quad (11)$$

We normalize all variables to the same units before non-dimensionalizing them using the constant 1W m^{-2}

(SM A.5). Finally, we derive the dimensionless conservation laws (SM A.1-A.4) and write them as a sparse matrix of size $4 \times (304 + 218)$:

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & \ell_s & -\ell_s \delta p & -\ell_f \delta p & 0 & -\delta p & \delta p & 0 & 0 & -1 & 1 & 1 & -1 & -\ell_f & \ell_f \\ 0 & 0 & 1 & -\delta p & -\delta p & -\delta p & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta p & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta p & 0 & 0 & -1 & 1 & 0 & 0 \end{bmatrix}, \quad (12)$$

that acts on \mathbf{x} and \mathbf{y} to yield Equation 1.

Each row of the constraints matrix \mathbf{C} describes a different conservation law: The first row is column-integrated enthalpy conservation (here equivalent to energy conservation), the second row is column-integrated water conservation (here equivalent to mass conservation), the third row is column-integrated longwave radiation conservation and the last row is column-integrated shortwave radiation conservation.

D. Implementation

We implement the three NN types and a multi-linear regression baseline using the Tensorflow library [30] version 1.13 with Keras [31] version 2.2.4: (1) *LCnets* for which we vary the weight α given to conservation laws from 0 to 1 (Equation 6), (2) our reference *ACnet*, and (3) *UCnet*, i.e. an unconstrained LCnet of weight $\alpha = 0$. In our reference ACnet, we write the constraints layers in Tensorflow to solve the system of equations (\mathcal{C}) from bottom to top, and calculate surface tendencies as residuals of the conservation equations (SM B.1); switching the “residual” outputs to different vertical levels does not significantly change the validation loss nor the constraints penalty (SM B.3). After testing multiple architectures and activation functions (SM C.2), we chose 5 hidden layers of 512 nodes with leaky rectified linear-unit activations as our standard multi-layer perceptron architecture, resulting in $\sim 1.3\text{M}$ trainable parameters. We optimized the NN’s weights and biases with the RMSprop optimizer [32] for LCnets (because it was more stable than the Adam optimizer [33]), used Sherpa for hyperparameter optimizations [34], and saved the NN’s state of minimal validation loss over 20 epochs.

E. Results

In Figure 3a, we compare mean performance (measured by MSE) and by how much physical constraints

are violated (measured by \mathcal{P}) for the three NN types. As expected, we note a monotonic trade-off between performance and constraints as we increase α from 0 to 1 in the loss function. This trade-off is well-measured by MSE and \mathcal{P} across the training, validation, and test sets (SM Table V). Interestingly, the physical constraints are easier to satisfy than reducing MSE in our case, likely because it is difficult to deterministically predict precipitation, which is strongly non-Gaussian, inherently stochastic, and whose error contributes to a large portion of MSE. Despite this, UCnet may violate physical constraints more than our multi-linear regression baseline.

Our first key result is that *ACnet performs nearly as well as our lowest-MSE UCnet on average* (to within 3%) while satisfying constraints to $\sim (10^{-9}\%)$ (SM C.1). This result holds across the training, validation and test sets (SM Table IV). In our case, ACnets perform slightly less well than UCnet because they are harder to optimize and the “residual” outputs exhibit systematically larger errors (SM B.2). This systematic, unphysical bias can be remedied by multiplying the weights of these “residual” outputs in the loss function (SM B.3) by a factor $\beta > 1$ (SM Equation 12 and SM Figure 2). β can be objectively chosen alongside the “residual” outputs via formal hyperparameter optimization (SM C.2).

In Figure 3b, we compare how much the NNs violate column energy conservation (RESID) to the prediction of a variable that appears in that constraint: the total thermodynamic tendency in the enthalpy conservation equation (THERMO):

$$\overbrace{\left(\mathbf{C} \begin{bmatrix} \mathbf{x} \\ \mathbf{y}_{NN} \end{bmatrix} \right)_1}^{\text{RESID}} = \overbrace{\delta p \cdot (\dot{\mathbf{T}}_{KE} - \dot{\mathbf{T}} - \ell_s \dot{\mathbf{q}}_v - \ell_f \dot{\mathbf{q}}_l)}^{\text{THERMO}} + \dots, \quad (13)$$

where the ellipsis includes the surface fluxes, radiation, and precipitation terms. ACnet predicts THERMO more accurately than all NNs (full blue line) by an amount closely related to how much each NN violates enthalpy conservation (dashed lines), followed by LCnet (full green line). This yields our second key result: *Enforcing con-*

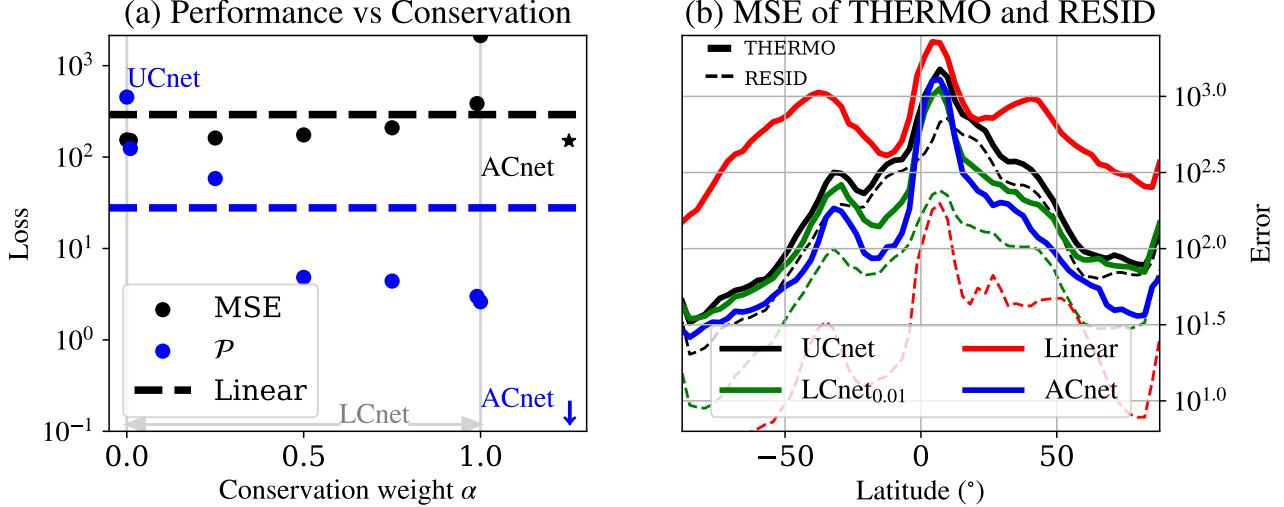


FIG. 3. (a) MSE and \mathcal{P} averaged over all samples of the test dataset for UCnet, LCnets of varying α , and ACnet. The dashed lines indicate MSE and \mathcal{P} for our multi-linear regression baseline. (b) Mean-squared error in the thermodynamic term (THERMO) and the enthalpy residual (RESID) versus latitude for our lowest-MSE NN in each category.

straints, whether in the architecture or the loss function, can systematically reduce the error of variables that appear in the constraints. This result holds true across the training, validation, and test sets (SM Figure 4). However, possibly since our case has many degrees of freedom, it does not hold true for individual components of THERMO as their cross-term in Equation 9 is more negative for ACnet, nor does it hold for variables that are hard to predict deterministically (e.g., precipitation). Additionally, obeying conservation laws does not guarantee the ability to generalize well far outside of the training set, e.g. in the Tropics of a warmer climate (see Figure 3 of [35]). These results nuance the finding that physically constraining NNs systematically improves their generalization ability, which has been documented for machine learning emulation of low-dimensional idealized flows [5, 12], and motivate physically-constraining machine-learning algorithms capable of stochastic predictions [36] that are consistent across climates [35].

Finally, although the mapping presented in Section III has linear constraints, ACnets can also be applied to non-linearly constrained mappings by using the framework presented in Figure 1. We give a concrete example in SM D, where we introduce the concept of ‘‘conversion layers’’

that transform non-linearly constrained mappings into linearly-constrained mappings within NNs and without overly degrading performance (SM Table IX). Additionally, ACnets can be extended to incorporate inequality constraints on their ‘‘direct’’ outputs (by using positive-definite activation functions, discussed in SM E), making ACnets applicable to a broad range of constrained optimization problems.

ACKNOWLEDGMENTS

TB is supported by NSF grants OAC-1835769, OAC-1835863, and AGS-1734164. PG acknowledges support from USMILE ERC synergy grant. The work of JO and PB is in part supported by grants NSF 1839429 and NSF NRT 1633631 to PB. We thank Eric Christiansen, Imme Ebert-Uphoff, Bart Van Merriënboer, Tristan Abbott, Ankitesh Gupta, and Derek Chang for advice. We also thank the meteorology department of LMU Munich and the Extreme Science and Engineering Discovery Environment supported by NSF grant number ACI-1548562 (charge numbers TG-ATM190002 and TG-ATM170029) for computational resources.

- [1] P. Baldi, *Deep Learning in Science: Theory, Algorithms, and Applications* (Cambridge University Press, Cambridge, UK, 2021) in press.
- [2] M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, and Prabhat, Deep learning and process understanding for data-driven Earth system science, *Nature* **566**, 195 (2019).
- [3] K. J. Bergen, P. A. Johnson, M. V. De Hoop, and G. C.

- Beroza, Machine learning for data-driven discovery in solid Earth geoscience (2019).
- [4] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar, Theory-guided data science: A new paradigm for scientific discovery from data, *IEEE Transactions on Knowledge and Data Engineering* **29**, 2318 (2017).
- [5] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Ku-

- mar, Integrating Physics-Based Modeling with Machine Learning: A Survey, (2020), arXiv:2003.04919.
- [6] P. Márquez-Neila, M. Salzmann, and P. Fua, Imposing Hard Constraints on Deep Networks: Promises and Limitations, (2017), arXiv:1706.02025.
- [7] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations, (2017), arXiv:1711.10561.
- [8] 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**, 15344 (2019).
- [9] E. de Bezenac, A. Pajot, and P. Gallinari, Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge, (2017), arXiv:1711.07970.
- [10] J. Ling, A. Kurzawski, and J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, Journal of Fluid Mechanics **807**, 155 (2016).
- [11] J. L. Wu, H. Xiao, and E. Paterson, Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework, Physical Review Fluids **7**, 074602 (2018).
- [12] L. Sun, H. Gao, S. Pan, and J. X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Computer Methods in Applied Mechanics and Engineering **361**, 112732 (2020).
- [13] T. Bolton and L. Zanna, Applications of Deep Learning to Ocean Data Inference and Subgrid Parameterization, Journal of Advances in Modeling Earth Systems **11**, 376 (2019).
- [14] A. Karpatne, W. Watkins, J. Read, and V. Kumar, Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling, (2017), arXiv:1710.11431.
- [15] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar, Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles, in *SIAM International Conference on Data Mining, SDM 2019* (2019) pp. 558–566, arXiv:1810.13075v2.
- [16] M. Raissi, A. Yazdani, and G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science **367**, 1026 (2020).
- [17] T. Beucler, S. Rasp, M. Pritchard, and P. Gentine, Achieving Conservation of Energy in Neural Network Emulators for Climate Modeling, (2019), arXiv:1906.06622.
- [18] A. K. Jain, J. Mao, and K. M. Mohiuddin, Artificial neural networks: A tutorial (1996).
- [19] M. W. Gardner and S. R. Dorling, Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences, Atmospheric Environment **32**, 2627 (1998).
- [20] T. Palmer, G. Shutts, R. Hagedorn, F. Doblas-Reyes, T. Jung, and M. Leutbecher, Representing Model Uncertainty in Weather and Climate Prediction, Annual Review of Earth and Planetary Sciences **33**, 163 (2005).
- [21] T. Schneider, J. Teixeira, C. S. Bretherton, F. Bréit, K. G. Pressel, C. Schär, and A. P. Siebesma, Climate goals and computing the future of clouds, Nature Climate Change **7**, 3 (2017).
- [22] V. M. Krasnopol'sky, M. S. Fox-Rabinovitz, and A. A. Belochitski, 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, Advances in Artificial Neural Systems **2013**, 1 (2013).
- [23] P. Gentine, M. Pritchard, S. Rasp, G. Reinaudi, and G. Yacalis, Could Machine Learning Break the Convection Parameterization Deadlock?, Geophysical Research Letters **45**, 5742 (2018).
- [24] S. Rasp, M. S. Pritchard, and P. Gentine, Deep learning to represent sub-grid processes in climate models, Proceedings of the National Academy of Sciences of the United States of America **115**, 9684 (2018), arXiv:1806.04731.
- [25] N. D. Brenowitz and C. S. Bretherton, Prognostic Validation of a Neural Network Unified Physics Parameterization, Geophysical Research Letters **45**, 6289 (2018).
- [26] S. Rasp, Coupled online learning as a way to tackle instabilities and biases in neural network parameterizations 10.5194/gmd-2019-319 (2019), arXiv:1907.01351.
- [27] M. Khairoutdinov, D. Randall, and C. DeMott, Simulations of the Atmospheric General Circulation Using a Cloud-Resolving Model as a Superparameterization of Physical Processes, Journal of the Atmospheric Sciences **62**, 2136 (2005).
- [28] M. S. Pritchard, C. S. Bretherton, and C. A. Demott, Restricting 32–128 km horizontal scales hardly affects the MJO in the Superparameterized Community Atmosphere Model v.3.0 but the number of cloud-resolving grid columns constrains vertical mixing, Journal of Advances in Modeling Earth Systems **6**, 723 (2014).
- [29] J. A. Andersen and Z. Kuang, Moist static energy budget of MJO-like disturbances in the atmosphere of a zonally symmetric aquaplanet, Journal of Climate **25**, 2782 (2012).
- [30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, (2016), arXiv:1603.04467.
- [31] F. Chollet, Keras (2015).
- [32] T. Tieleman, G. E. Hinton, N. Srivastava, and K. Swersky, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Networks for Machine Learning **4**, 26 (2012).
- [33] D. P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, (2014), arXiv:1412.6980.
- [34] L. Hertel, J. Collado, P. Sadowski, J. Ott, and P. Baldi, Sherpa: Robust hyperparameter optimization for machine learning, SoftwareX (2020), in press.
- [35] T. Beucler, M. Pritchard, P. Gentine, and S. Rasp, Towards Physically-consistent, Data-driven Models of Convection, (2020), arXiv:2002.08525.
- [36] J.-L. Wu, K. Kashinath, A. Albert, D. Chirila, Prabhat, and H. Xiao, Enforcing Statistical Constraints in Generative Adversarial Networks for Modeling Chaotic Dynamical Systems, (2019), arXiv:1905.06841.

Supplemental Material

Enforcing Analytic Constraints in Neural-Networks Emulating Physical Systems

Tom Beucler,^{1, 2,*} Michael Pritchard,¹ Stephan Rasp,³ Jordan Ott, Pierre Baldi,⁴ and Pierre Gentine²

¹*Department of Earth System Science, University of California, Irvine, CA, USA*

²*Department of Earth and Environmental Engineering, Columbia University, New York, NY, USA*

³*Technical University of Munich, Munich, Germany*

⁴*Department of Computer Science, University of California, Irvine, CA, USA*

(Dated: January 29, 2021)

The supplemental material (SM) has 5 Sections that can be read independently: (A) introduces the variables and constraints specific to our application, (B) details the implementation of our architecture-constrained network, (C) compares different network types and architectures, (D) presents an example where *nonlinear* constraints are enforced in neural networks emulating physical systems, and (E) discusses how to enforce *inequality* constraints.

A. Derivation of Dimensionless Conservation Equations

The Super-Parameterized Community Atmosphere Model 3.0 embeds a convection-permitting model, namely the System for Atmospheric Modeling [1], in each grid cell of the Community Atmosphere Model 3.0 [2]. In the absence of convective momentum transfer, our convection-permitting model conserves two quantities: liquid/ice water static energy and total water. We recast these conservation equations as an energy (A.1) and mass (A.2) conservation, before adding the conservation of longwave (A.3) and shortwave (A.4) radiation as our network predicts both radiative heating profiles and boundary fluxes at the top and bottom of the atmosphere, whose difference must match the mass-weighted vertical integral of the heating rate profiles. Finally, we non-dimensionalize all conservation equations in SM A.5. We define all variables in Table I.

A.1. Conservation of Energy

We define the enthalpy H of an atmospheric column as the mass-weighted vertical integral of the sum of its sensible heat and latent heat, in J m^{-2} , where ice is used as the reference phase of zero energy:

$$H \stackrel{\text{def}}{=} \int_0^{p_s} \frac{dp}{g} \underbrace{(c_p T + L_s q_v + L_f q_l)}_h, \quad (1)$$

where p is atmospheric pressure in Pa, p_s is surface pressure in Pa, $g \approx 9.81 \text{m s}^{-2}$ is the gravity constant,

Variable	Name
δp	Normalized differential pressure reference profile
ℓ_f	Normalized latent heat of fusion of water
ℓ_s	Normalized latent heat of sublimation of water
LHF	Latent heat flux
LS	Large-scale forcings in water, temperature, velocity
Iw	Longwave heating rate profile
LW _s	Net surface longwave flux
LW _t	Net top-of-atmosphere longwave flux
P	Total precipitation rate
P_i	Solid precipitation rate
p_s	Surface pressure
S_0	Solar insolation
SHF	Sensible heat flux
sw	Shortwave heating rate profile
SW _s	Net surface shortwave flux
SW _t	Net top-of-atmosphere shortwave flux
T	Absolute temperature profile
\dot{T}	Convective heating profile
\dot{T}_{KE}	Heating from turbulent kinetic energy dissipation
q_i	Ice concentration profile
\dot{q}_i	Convective ice tendency profile
q_l	Liquid water concentration profile
\dot{q}_l	Convective liquid water tendency profile
q_v	Specific humidity profile
\dot{q}_v	Convective water vapor tendency profile
v	North-South velocity profile
z	Vertical level profile

TABLE I. Definition of physical variables: Variables that depend on height are (boldfaced) vectors, referred to as “profiles”.

$c_p \approx 1.00 \cdot 10^3 \text{J kg}^{-1} \text{K}^{-1}$ is the specific heat of water at constant pressure in standard atmospheric conditions, T is the absolute temperature in K, $L_s \approx 2.83 \cdot 10^6 \text{J kg}^{-1}$ is the latent heat of sublimation of water in standard conditions, q_v is the specific humidity or water vapor mass concentration in kg/kg, $L_f \approx 3.34 \cdot 10^5 \text{J kg}^{-1}$ is the latent heat of fusion of water in standard conditions, q_l is the liquid water mass concentration in kg/kg and h is the specific enthalpy in J kg^{-1} . We isolate the atmospheric column’s time-tendency that is due to water phase changes only (Δ_φ) for each variable of Equation 1:

$$\Delta_\varphi h = -L_f (P - P_i), \quad (2a)$$

* tom.beucler@gmail.com

$$c_p \Delta_\varphi T = -\text{SHF} + \int_0^{p_s} \frac{dp}{g} c_p (\dot{T} + \dot{T}_{KE}) + \text{SW}_s - \text{SW}_t + \text{LW}_t - \text{LW}_s, \quad (2b)$$

$$L_v \Delta_\varphi q_v = -\text{LHF} + \int_0^{p_s} \frac{dp}{g} L_v \dot{q}_v, \quad (2c)$$

$$\Delta_\varphi q_l = \int_0^{p_s} \frac{dp}{g} \dot{q}_l, \quad (2d)$$

where P is the total surface precipitation rate in $\text{kg m}^{-2} \text{s}^{-1}$, P_i is the surface solid precipitation rate in $\text{kg m}^{-2} \text{s}^{-1}$, SHF is the surface sensible heat flux in W m^{-2} , \dot{T} is the time-tendency of temperature in K s^{-1} , \dot{T}_{KE} is the time-tendency of temperature due to frictional dissipation of kinetic energy in K s^{-1} , SW_s is the net surface downwards shortwave radiative flux in W m^{-2} , SW_t is the net top-of-atmosphere downwards shortwave radiative flux in W m^{-2} , LW_t is the net top-of-atmosphere upwards longwave radiative flux in W m^{-2} , LW_s is the net surface upwards longwave radiative flux in W m^{-2} , $L_v \approx 2.50 \cdot 10^6 \text{ J kg}^{-1}$ is the latent heat of vaporization of water in standard conditions, LHF is the surface latent heat flux in W m^{-2} , \dot{q}_v is the time tendency of specific humidity in $\text{kg kg}^{-1} \text{s}^{-1}$ and \dot{q}_l is the time tendency of liquid water concentration in $\text{kg kg}^{-1} \text{s}^{-1}$. We then equate the time-tendencies due to water phase changes in Equation 1 to form the equation describing the conservation of column enthalpy:

$$\Delta_\varphi h - c_p \Delta_\varphi T - L_s \Delta_\varphi q_v - L_f \Delta_\varphi q_l = 0. \quad (3)$$

A.2. Conservation of Mass

The conservation of water states that the change in total column water concentration must balance the sources and sinks of water at the surface, namely the surface evaporation and precipitation rates:

$$\int_0^{p_s} \frac{dp}{g} (\dot{q}_v + \dot{q}_l + \dot{q}_i) = \frac{\text{LHF}}{L_v} - P. \quad (4)$$

A.3. Conservation of Longwave Radiation

The conservation of longwave radiation states that the difference between the top-of-atmosphere and surface longwave radiative fluxes must balance the net longwave (or longwave) radiative cooling of the atmospheric column to space:

$$\text{LW}_t - \text{LW}_s = - \int_0^{p_s} \frac{dp}{g} c_p \text{lw}, \quad (5)$$

where lw is the vertically-resolved temperature tendency due to longwave heating in K s^{-1} .

A.4. Conservation of Shortwave Radiation

The conservation of shortwave radiation states that the difference between the top-of-atmosphere insolatation and the incoming shortwave radiation at the surface must balance the net shortwave radiative heating of the atmospheric column:

$$\text{SW}_t - \text{SW}_s = \int_0^{p_s} \frac{dp}{g} c_p \text{sw}. \quad (6)$$

A.5. Non-dimensionalization of Conservation Equations

We non-dimensionalize the conservation equations by converting all tendencies to units W m^{-2} before dividing them by 1W m^{-2} . For numerical modeling purposes, each vertical profile is discretized to 30 vertical levels z of varying pressure thickness δP_z . This means that a continuous conservation equations becomes a linear constraint on discrete variables; for instance Equation 6 becomes:

$$\text{SW}_t - \text{SW}_s = \sum_{z=1}^{30} \frac{\delta P_z}{g} c_p \text{sw}_z. \quad (7)$$

To make Equation 7 non-dimensional, we introduce a fixed, normalized differential pressure coordinate δp to make the atmospheric pressure p non-dimensional: $\forall z, \delta \tilde{p}_z \stackrel{\text{def}}{=} \delta P_z / \delta p_z$. This motivates the following non-dimensionalizations:

$$\widetilde{\text{SW}}_t \stackrel{\text{def}}{=} \frac{\text{SW}_t}{1\text{W m}^{-2}}, \quad \widetilde{\text{SW}}_s \stackrel{\text{def}}{=} \frac{\text{SW}_s}{1\text{W m}^{-2}}, \quad \widetilde{\text{sw}}_z \stackrel{\text{def}}{=} \frac{c_p \delta p_z \text{sw}_z}{g}, \quad (8)$$

which leads to the simpler form of the shortwave conservation equation presented in the main text:

$$\widetilde{\text{SW}}_t - \widetilde{\text{SW}}_s = \sum_{z=1}^{30} \widetilde{\text{sw}}_z \delta \tilde{p}_z = \widetilde{\text{sw}} \cdot \delta \tilde{p}. \quad (9)$$

For simplicity, the tildes are dropped in the main text and the following appendices.

B. Implementation of the Architecture-constrained Network

In this section, we present our standard implementation of ACnets (B.1), which is application-specific, investigate the sensitivity of ACnets to the choice of “residual”

outputs (B.2), and introduce a method to decrease the “residual” outputs’ biases by preferentially weighting the loss function (B.3).

Note that there are multiple ways of implementing the ACnet defined in our manuscript. The first step involves choosing $(p - n)$ “direct” NN outputs calculated from the NN’s weights and biases, and n “residual” outputs calculated using the fixed constraints matrix \mathbf{C} . If a number q of outputs appear in the n constraints, this results in $\frac{n!}{q!(n-q)!}$ options to choose from. The second step involves calculating the n residual outputs every time a batch is passed to the network. We can implement this by either:

- adding custom layers to the NN that use the physical constraints (\mathcal{C}) to calculate the “residual” outputs, which are then concatenated with the “direct” outputs to pass an output vector of length p to the loss function, or
- passing the “direct” outputs of length $(p - n)$ to the loss function and modifying the loss function’s code to calculate the “residual” outputs within the loss function before e.g. calculating the mean-squared error.

We adopt the former approach below.

B.1. Standard Implementation

In our standard implementation, we first output a vector of size $(218 - 4) = 214$, before calculating the “residual” component of the output vector by solving the system of equations $\mathbf{C} [\mathbf{x} \ \mathbf{y}]^T = \mathbf{0}$ from bottom to top and within the network. All layers are implemented in Tensorflow using the functional application programming interface. Here, we choose to implement one layer per physical constraint, so that each layer solves one equation described by one row of the constraints matrix \mathbf{C} . We could equivalently have grouped all constraints in a single constraints layer, which would then have solved the entire system of equations described by the full constraints matrix \mathbf{C} .

The first conservation layer (CL_1) calculates the net shortwave surface flux as a residual of the conservation of shortwave radiation (last row of \mathbf{C}):

$$\sum_{i=1}^{30} \text{sw}_z \delta p_z - \text{SW}_t + \text{SW}_s = 0. \quad (10)$$

$$(\text{CL}_1) \quad \underbrace{\text{SW}_s}_{\text{Residual}_4} = \text{SW}_t - \sum_{z=1}^{30} \text{sw}_z \delta p_z. \quad (11a)$$

The second conservation layer (CL_2) calculates the net longwave surface flux as a residual of the conservation of longwave radiation (third row of \mathbf{C}):

$$(\text{CL}_2) \quad \underbrace{\text{LW}_s}_{\text{Residual}_3} = \text{LW}_t - \sum_{z=1}^{30} \text{lw}_z \delta p_z. \quad (11b)$$

The third conservation layer (CL_3) calculates the lowest-level specific humidity tendency as a residual of the conservation of mass (second row of \mathbf{C}):

$$(\text{CL}_3) \quad \delta p_{30} \underbrace{\dot{q}_{v,30}}_{\text{Residual}_2} = \text{LHF} - \text{P} - \sum_{z=1}^{29} \delta p_z \dot{q}_{v,z} - \sum_{z=1}^{30} \delta p_z (\dot{q}_{l,z} + \dot{q}_{i,z}). \quad (11c)$$

The fourth conservation layer (CL_4) calculates the lowest-level temperature tendency as a residual of the conservation of energy (first row of \mathbf{C}):

$$(\text{CL}_4) \quad \delta p_{30} \underbrace{\dot{T}_{30}}_{\text{Residual}_1} = \text{SHF} + \ell_s \text{LHF} + \ell_f (P - P_i) - \text{LW}_t + \text{LW}_s + \text{SW}_t - \text{SW}_s - \sum_{z=1}^{29} \delta p_z \dot{T}_z + \sum_{z=1}^{30} \delta p_z (\dot{T}_{\text{KE},z} - \ell_s \dot{q}_{v,z} - \ell_f \dot{q}_{l,z}). \quad (11d)$$

B.2. Sensitivity to Residual Index

The indices of the output’s components calculated as “residuals” – i.e. which vertical level and specific variable is chosen to residually enforce the constraints on the column – are new hyperparameters of ACnets: While we chose the lowest-level convective heating and moistening tendencies as the “residuals” of our reference ACnet, we are free to choose other vertical levels (e.g., top-of-atmosphere) or other variables (e.g., convective liquid or ice tendencies) as “residuals”. To probe the sensitivity of ACnets to this unfamiliar hyperparameter choice, we train 5 ACnets with different vertical “residual”-levels over 20 epochs, save their states of minimal validation loss to avoid overfitting, and report their performance and conservation properties in Table II. Each ACnet is referred to as $q_{z1} T_{z2}$, where $z1$ is the vertical level index (increasing downward from 0 at top-of-atmosphere to 29 at the surface) of the “residual” convective moistening for mass conservation, and $z2$ the index of the “residual” convective heating for energy conservation.

Reassuringly, all ACnets conserve column mass, energy and radiation to $\sim (10^{-9} \text{W}^2 \text{m}^{-4})$ over the training, validation and test sets (Table II). However, a problem is

apparent in the underlying vertical structure: Despite similar overall MSEs, ACnets' squared-error is larger at the “residual” vertical level. SM Figure 1 illustrates this by focusing on errors in the convective moistening and heating profiles averaged over the entire validation and test sets. Note that averaging the squared error over the entire validation set is equivalent to averaging the squared error in latitude, longitude, and over all days of our reference simulation used in the validation set.

To first order, all ACnets exhibit similar vertical structures in their squared error: The squared error scales like the variance at each vertical level, except near the surface ($z > 22$) where a large portion of the signal cannot be deterministically predicted (see Section 4.2 of [3] and the discussion on the performance for different horizontal grid spacings in [4]). The scaling also does not hold in the upper atmosphere ($z < 10$) where the true convective moistening \dot{q}_v variance is so small that the squared error in \dot{q}_v is negligible and does not contribute to the overall MSE. Interestingly, each ACnet performs worse at its “residual” levels on both sets. For instance, $q_{14}T_{14}$ (orange line) always produces the largest error at its residual level (middle horizontal black line) relative to other ACnets. Similarly, $q_{29}T_{29}$ (green line) and q_0T_{29} (red line) produce the largest convective heating error at the lowest model level (bottom horizontal black line). Although the sample-to-sample variability is large (e.g., MSE standard deviation in Table II), latitude-pressure and longitude-pressure plots of convective heating and moistening errors show a systematic error increase at the “residual level” (not shown), confirming this is a robust error associated with the “residual level” hyperparameter. Additional errors in radiative fluxes and precipitation result in the total MSE, reported on the first, third, and fifth rows of Table II. Since this unphysical vertical structure is a disadvantage of ACnets we propose a simple solution below.

B.3 Decreasing the Residual Index Bias by Weighting the Loss Function

A simple way to address the systematic bias induced at an ACnet’s “residual” level is to increase the weight of these “residual” outputs in the loss function \mathcal{L} :

$$\mathcal{L}(\beta) = \underbrace{\frac{1}{p-n} \sum_{k=1}^{p-n} y_{\text{Err},k}^2}_{\text{Direct Outputs MSE}} + \beta \times \underbrace{\frac{1}{n} \sum_{k=p-n+1}^p y_{\text{Err},k}^2}_{\text{Residual Outputs MSE}}, \quad (12)$$

where we have modified the MSE loss defined in Equation 3 by introducing a loss multiplier $\beta > 1$.

We implement the weighted loss as a custom Tensorflow loss and train ACnets of residual index $z = 14$ (i.e., $q_{14}T_{14}$) to test the sensitivity of the spurious error feature near 300 hPa (Fig 1, orange line) with five different values of the loss multiplier β : 1, 2, 5, 10, and 20. We re-

port the MSE and conservation penalty in Table III, and show the squared-error profiles averaged over the test set in SM Figures 2a and 2b. Reassuringly, all ACnets still conserve mass, energy and radiation to within machine precision. As β increases, the spurious squared error anomaly at the “residual” index (horizontal black line at $z = 14$) becomes indistinguishable from the squared error at adjacent levels, successfully removing the anomaly. To quantitatively validate this observation and better visualize the “residual” bias, we introduce the absolute value of the logarithmic vertical gradient in the squared error **Log.bias** as a proxy for the “residual” bias at each vertical level z :

$$\begin{aligned} \text{Log.bias}_z &\stackrel{\text{proxy}}{\approx} \frac{|\partial_z y_{\text{Err},z}^2|}{y_{\text{Err},z}^2}, \\ &\stackrel{\text{numerics}}{\approx} \frac{|y_{\text{Err},z+1}^2 - y_{\text{Err},z}^2| + |y_{\text{Err},z}^2 - y_{\text{Err},z-1}^2|}{y_{\text{Err},z+1}^2 + y_{\text{Err},z-1}^2}, \end{aligned} \quad (13)$$

and depict it in SM Figures 2c and 2d. These figures underline the trade-off between the total MSE and the squared error at the residual level: As β increases, the “direct” outputs are given less and less weight in the loss function \mathcal{L} , and while the bias at the “residual” levels decreases, the overall MSE increases. This is confirmed by (1) reading the MSE line of Table III from left to right, (2) the general increase of the squared-error profile with β in SM Figures 2a and 2b; and (3) the large Log.bias introduced in the upper atmosphere ($z < 10$) in SM Figures 2c and 2d.

In conclusion, introducing a moderate loss multiplier (e.g., $\beta = 2$) can systematically address the “residual” bias of ACnets at the cost of total MSE. β can be seen as a new hyperparameter of ACnets, which can be tuned in conjunction with the “residual” indices to guarantee high performance and minimal “residual” biases as detailed in SM C.2.b.

C. Comparison of Neural Network Types and Architectures

In this section, we run sensitivity tests to probe the effect of our NN’s characteristics on their performance and constraints penalty. Note that we distinguish NN types (i.e., LCnet, UCnet, ACnet; see SM C.1) from NN hyperparameters (i.e., number of layers, etc.; see SM C.2). In SM C.3, we compare ACnets that enforce constraints during training to NNs enforcing constraints after training.

C.1 Comparison between LCnets, UCnet and ACnet

The performance and constraints penalties of the different NN types depicted in Figure 3b are compared in Table IV:

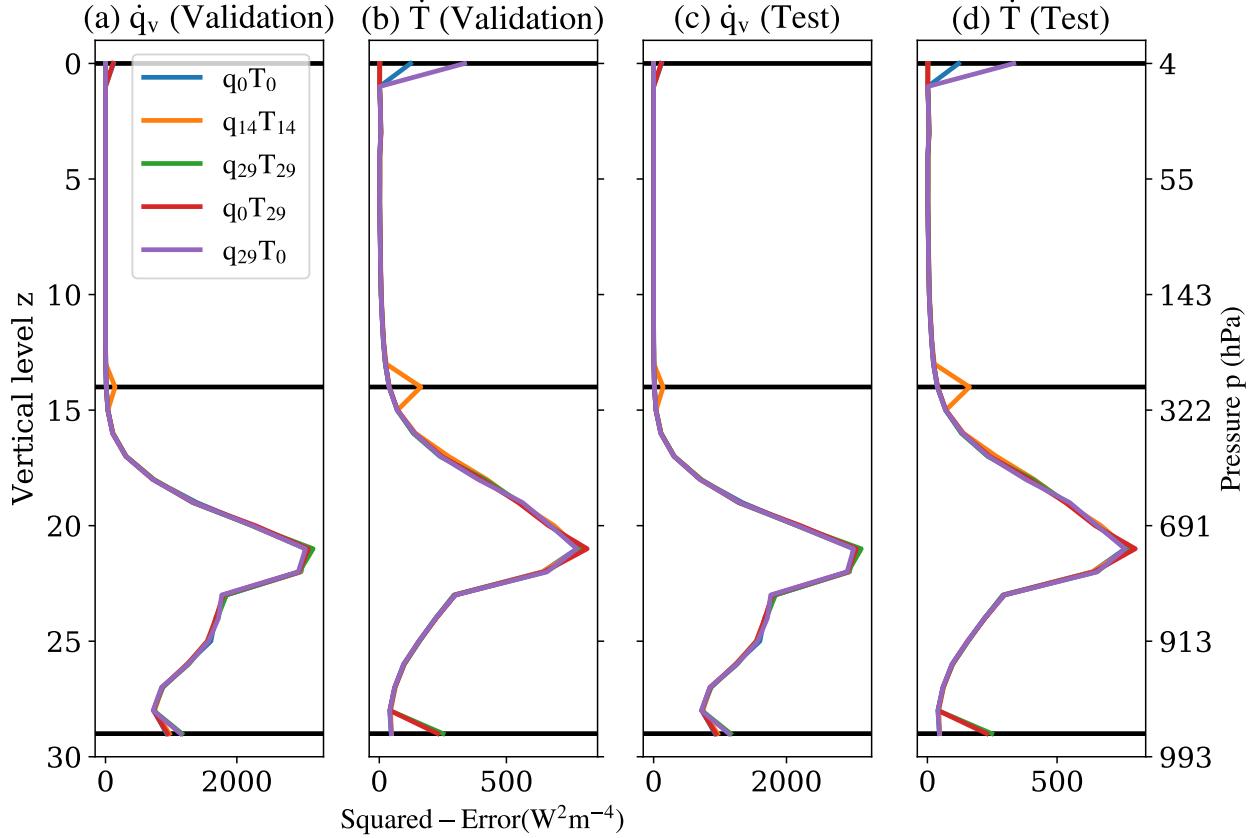


FIG. 1. For various residual levels: Squared error in convective moistening \dot{q}_v and heating \dot{T} versus pressure, for the validation and test sets. We indicate “residual” levels where squared errors differ the most using black horizontal lines: The vertical level of index 0 is at the top of the atmosphere (0hPa), the vertical level of index 14 in the upper troposphere (274hPa), and the vertical level of index 29 at the surface (1000hPa).

Dataset	Metric	q_0T_0	$q_{14}T_{14}$	$q_{29}T_{29}$	q_0T_{29}	$q_{29}T_0$
Training set	MSE	$1.6 \cdot 10^{+02} \pm 9.6 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 9.7 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 9.7 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 9.6 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 9.5 \cdot 10^{+02}$
	\mathcal{P}	$7.8 \cdot 10^{-10} \pm 1.2 \cdot 10^{-09}$	$7.6 \cdot 10^{-10} \pm 1.2 \cdot 10^{-09}$	$8.1 \cdot 10^{-10} \pm 1.3 \cdot 10^{-09}$	$8.0 \cdot 10^{-10} \pm 1.4 \cdot 10^{-09}$	$7.6 \cdot 10^{-10} \pm 1.2 \cdot 10^{-09}$
Validation set	MSE	$1.6 \cdot 10^{+02} \pm 9.8 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 9.9 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 9.9 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 9.8 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 9.8 \cdot 10^{+02}$
	\mathcal{P}	$7.8 \cdot 10^{-10} \pm 1.3 \cdot 10^{-09}$	$7.7 \cdot 10^{-10} \pm 1.3 \cdot 10^{-09}$	$8.2 \cdot 10^{-10} \pm 1.4 \cdot 10^{-09}$	$8.0 \cdot 10^{-10} \pm 1.4 \cdot 10^{-09}$	$7.6 \cdot 10^{-10} \pm 1.3 \cdot 10^{-09}$
Test set	MSE	$1.6 \cdot 10^{+02} \pm 9.7 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 9.8 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 9.8 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 9.7 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 9.7 \cdot 10^{+02}$
	\mathcal{P}	$7.9 \cdot 10^{-10} \pm 1.3 \cdot 10^{-09}$	$7.7 \cdot 10^{-10} \pm 1.3 \cdot 10^{-09}$	$8.3 \cdot 10^{-10} \pm 1.4 \cdot 10^{-09}$	$8.1 \cdot 10^{-10} \pm 1.5 \cdot 10^{-09}$	$7.7 \cdot 10^{-10} \pm 1.3 \cdot 10^{-09}$

TABLE II. ACnets of varying residual levels for mass (m) and enthalpy (e) conservation, presented in SM Figure 1 (Mean MSE/Penalty \pm Standard deviation)

Validation	Metric	$\beta = 1$	$\beta = 2$	$\beta = 5$	$\beta = 10$	$\beta = 20$
Validation set	MSE	$1.6 \cdot 10^{+02} \pm 6.1 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 6.2 \cdot 10^{+02}$	$1.7 \cdot 10^{+02} \pm 6.4 \cdot 10^{+02}$	$1.9 \cdot 10^{+02} \pm 7.0 \cdot 10^{+02}$	$2.4 \cdot 10^{+02} \pm 8.8 \cdot 10^{+02}$
	\mathcal{P}	$7.9 \cdot 10^{-10} \pm 8.1 \cdot 10^{-09}$	$8.2 \cdot 10^{-10} \pm 8.6 \cdot 10^{-09}$	$7.7 \cdot 10^{-10} \pm 7.7 \cdot 10^{-09}$	$7.0 \cdot 10^{-10} \pm 7.4 \cdot 10^{-09}$	$5.2 \cdot 10^{-10} \pm 5.4 \cdot 10^{-09}$
Test set	MSE	$1.6 \cdot 10^{+02} \pm 6.0 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 6.0 \cdot 10^{+02}$	$1.7 \cdot 10^{+02} \pm 6.2 \cdot 10^{+02}$	$1.8 \cdot 10^{+02} \pm 6.9 \cdot 10^{+02}$	$2.4 \cdot 10^{+02} \pm 8.6 \cdot 10^{+02}$
	\mathcal{P}	$7.9 \cdot 10^{-10} \pm 8.2 \cdot 10^{-09}$	$8.2 \cdot 10^{-10} \pm 8.7 \cdot 10^{-09}$	$7.8 \cdot 10^{-10} \pm 7.8 \cdot 10^{-09}$	$7.1 \cdot 10^{-10} \pm 7.6 \cdot 10^{-09}$	$5.3 \cdot 10^{-10} \pm 5.5 \cdot 10^{-09}$

TABLE III. ACnets with varying loss multiplier β given to the residual levels ($q_{14}T_{14}$) (Mean MSE/Penalty \pm Standard deviation)

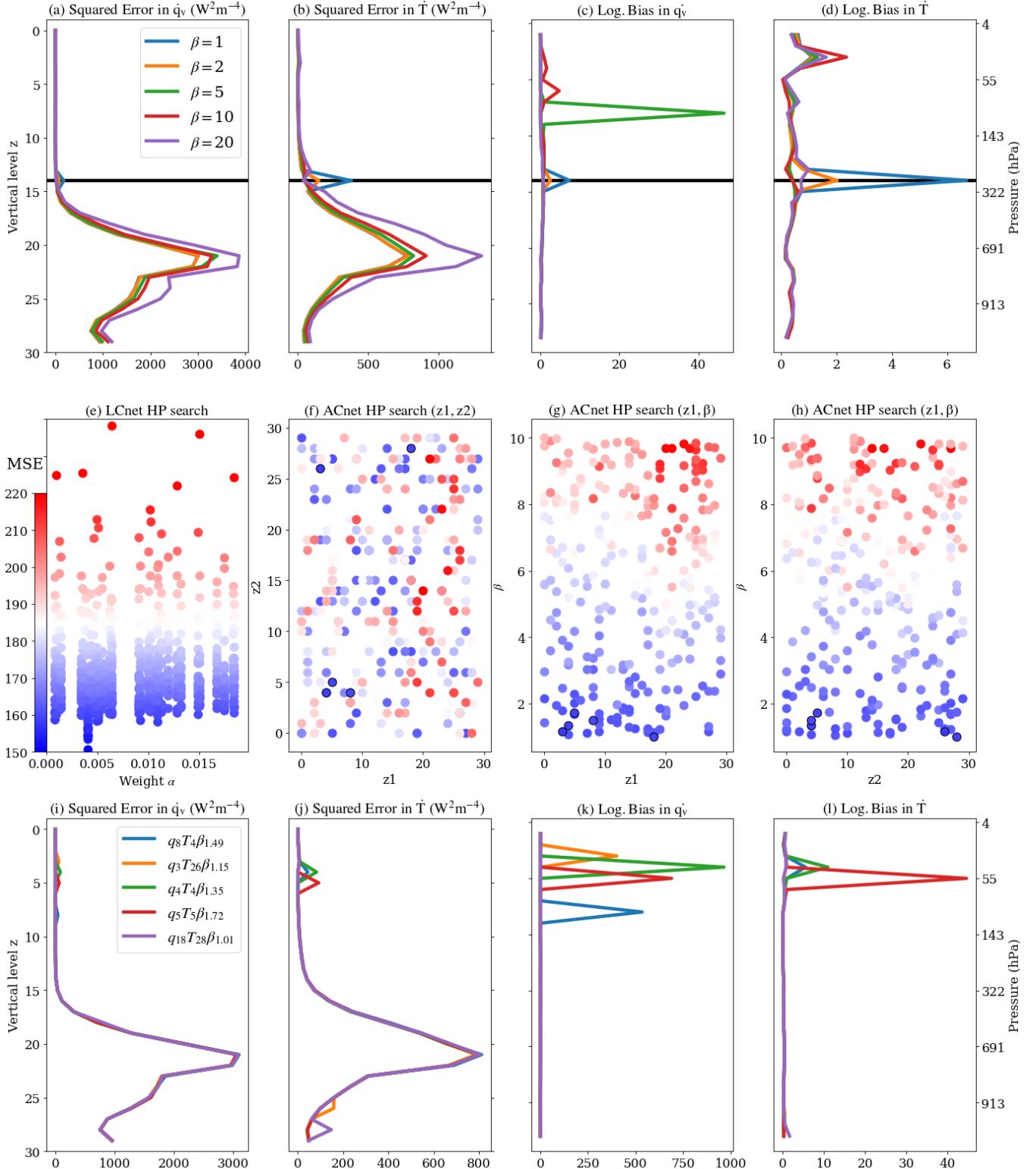


FIG. 2. For various loss multipliers β : Squared error (a-b) and Log.Bias (c-d) in convective moistening \dot{q}_v and heating \dot{T} versus pressure, averaged over the test set. (e) LCnet HP search for low physical constraints weight α , where dots are colored according to their MSE evaluated over the validation set. (f-h) ACnet HP search over (β, z_1, z_2) , where the five NNs of minimal validation MSE are circled in black. For these five NNs of minimal validation MSE: Squared error (i-j) and Log.Bias (k-l) in convective moistening \dot{q}_v and heating \dot{T} versus pressure, averaged over the test set.

1. “Linear” is the multi-linear regression baseline, derived by replacing all of UCnet’s leaky rectified linear-unit activations with the identity function.
2. “UCnet” is our best-performing NN, i.e. our NN of lowest MSE ($149 \text{ W}^2\text{m}^{-4}$). Despite its high performance, it violates conservation laws more than our multi-linear regression, motivating ACnet and LCnet.
3. “LCnet ($\alpha = 0.01$)” is our LCnet with strictly-positive conservation weight α of lowest MSE ($151 \text{ W}^2\text{m}^{-4}$). The 1% conservation weight is enough to divide the mean penalty of “UCnet” by a factor 2.4 over the baseline validation dataset. Despite this improvement, samples at $+1$ standard deviation have conservation penalties of $\sim 50 \text{ W}^2\text{m}^{-4}$, further motivating ACnet.
4. “ACnet” is our reference ACnet described in SM B.1. Its MSE is $152 \text{ W}^2\text{m}^{-4}$, which is only $3\text{W}^2\text{m}^{-4}$ more than our lowest-MSE UCnet.

We present the performance and constraints penalties of LCnets of varying conservation weight in Table V to better characterize the trade-off between performance and physical consistency depicted in Figure 3a. Note the large values of the conservation penalty’s standard deviation for LCnets, illustrating the limits of enforcing constraints in the loss function. We further show the coefficient of determination R^2 for convective moistening and heating for the four different NN types in SM Figure 3. NNs with nonlinear activation functions consistently outperform the multiple-linear regression baseline, while ACnets show a slight “residual” bias at the vertical level $z = 29$ (horizontal black line in SM Figure 3).

Finally, as we produced Figure 3 using data from the test set, we reproduce Figure 3 using data from the training and validation sets below (SM Figure 4). The two top rows of SM Figure 4 are nearly indistinguishable from Figure 3, confirming the robustness of our conclusions across the training, validation, and test sets. To explain why the two top rows are nearly indistinguishable in our case, we note that because all three sets are made of samples randomly drawn from a statistically-steady aquaplanet simulation, we expect statistical metrics to converge in the limit of large sample number. As this convergence is in contrast with standard training-validation splits in deep learning benchmarks (e.g., CIFAR-10 [5], ImageNet [6]), we illustrate it in the bottom row of SM Figure 4 by plotting the squared mean of the true output vector (left) and the mean-squared error of UCnet (right) as a function of sample number from the training (red), validation (blue), and test (black) sets.

C.2 Formal Hyperparameter Optimization

To optimize the performance of our NN emulators and address some of their recurring biases, we conduct four

distinct hyperparameter (HP) searches. We formally implement each HP search using SHERPA [7], a Python library for HP tuning. We choose to conduct *random* searches to explore a wide range of HP settings and avoid making assumption about the structure of the HP search problem [8]. Each search involves training more than 200 trial models for at least 20 epochs to find the NN yielding the lowest MSE over the validation set. We follow the structure of the main text and first conduct one UCnet HP search (C.2.a), followed by two LCnet (C.2.b) and one ACnet (C.2.c) searches.

a. UCnet HP Optimization

We follow common practices to tune UCnets and conduct our searches over the number of layers, the number of nodes per layer, the properties of the activation function, the presence of batch normalization, and the dropout coefficient (see Table VI for the range of available options during the search). The parameters of the best-performing UCnet are listed in Table VII, while the results notebook can be found at: https://github.com/jordanott/CBRAIN-CAM/blob/master/notebooks/tbeucler_devlog/hp_opt_conservation/NormalMSE.ipynb. As tuning LCnets and ACnets involves additional HPs (α , β , and the residual indices), we use Table VII to guide our choice of baseline HPs, which we round to: 5 layers of 512 nodes, Leaky ReLU coefficient of 0.3, no dropout and no batch normalization. Finally, we use this first HP search to objectively choose an appropriate number of epochs. SM Figure 5 shows training and validation MSE curves for more than 200 models. Both metrics plateau after 20 epochs, indicating this is an appropriate time to terminate training.

Using the baseline HPs identified via the UCnet HP search, we are now interested in the values of α for LCnets and $(\beta, z1, z2)$ for ACnets that yield the best performance. In the following paragraphs, we quantify performance using MSE averaged over the validation set for consistency, but note that HP searches are flexible enough to target any performance metric that can be calculated from the NN’s input and output vectors.

b. LCnet HP Optimization

Our first LCnet HP search (not shown) indicates that as suggested in Figure 3, SM Figure 4, and Table V, MSE typically increases with α for LCnets. This means that low values of α systematically lead to the lowest MSEs if LCnets are trained for enough epochs: While our LCnets of lowest MSEs use $\alpha \approx 0.01$, their MSE is still larger than our best UCnet’s MSE. This suggests a dominant trade-off between performance and physical constraints. For completeness, we conduct a second HP search restricted to $\alpha \in]0, 0.02[$ that we depict in Figure 2e and identify $\alpha = 4.1 \cdot 10^{-4}$ as the weight leading to the minimal validation MSE of $151 \text{ W}^2\text{m}^{-4}$.

c. ACnet HP Optimization

Finally, we conduct a HP search over $(\beta, z1, z2) \in]1, 10] \times [0, 29]^2$ to find the set of loss multiplier and residual indices that maximize performance while still

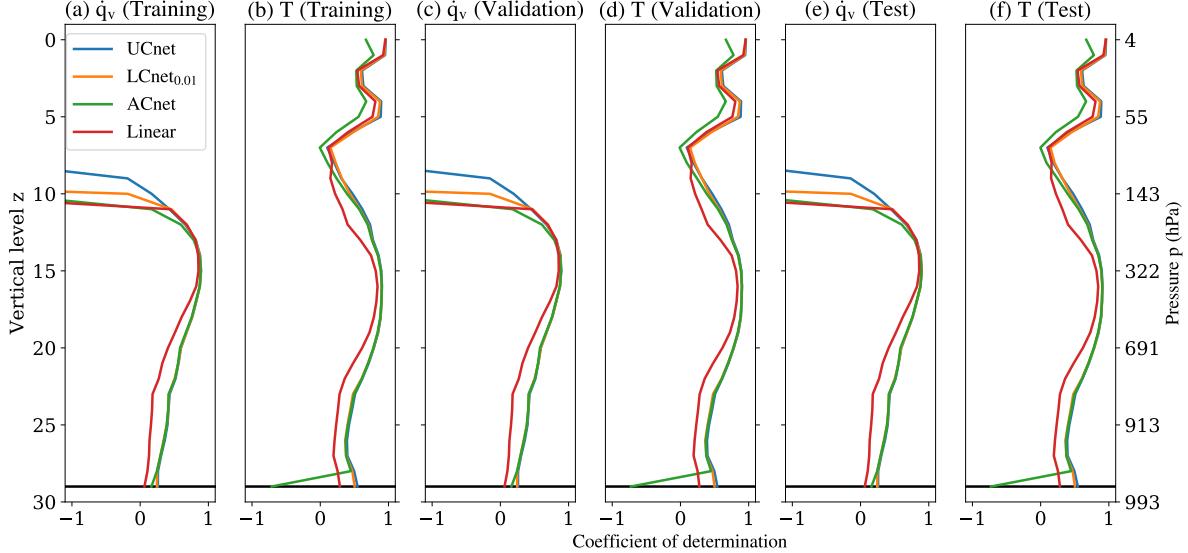


FIG. 3. For various NN types and architectures: Coefficient of determination R^2 for convective moistening \dot{q}_v and heating \dot{T} versus pressure, for the training, validation and test sets. Note that squared errors in \dot{q}_v from levels 0 to 10 never exceed $10^{-3}\%$ of the total squared error in \dot{q}_v , making them irrelevant for our regression purposes.

Validation	Metric	Linear	UC _{net}	LC _{net} ($\alpha = 0.01$)	AC _{net}
Training set	MSE	$2.9 \cdot 10^{+02} \pm 1.6 \cdot 10^{+03}$	$1.5 \cdot 10^{+02} \pm 9.2 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 9.2 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 9.4 \cdot 10^{+02}$
	\mathcal{P}	$2.7 \cdot 10^{+01} \pm 2.2 \cdot 10^{+01}$	$8.9 \cdot 10^{+01} \pm 8.0 \cdot 10^{+01}$	$3.7 \cdot 10^{+01} \pm 2.8 \cdot 10^{+01}$	$8.3 \cdot 10^{-10} \pm 1.4 \cdot 10^{-09}$
Validation set	MSE	$3.0 \cdot 10^{+02} \pm 1.7 \cdot 10^{+03}$	$1.5 \cdot 10^{+02} \pm 9.4 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 9.5 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 9.6 \cdot 10^{+02}$
	\mathcal{P}	$2.8 \cdot 10^{+01} \pm 2.3 \cdot 10^{+01}$	$9.1 \cdot 10^{+01} \pm 8.2 \cdot 10^{+01}$	$3.8 \cdot 10^{+01} \pm 2.8 \cdot 10^{+01}$	$8.4 \cdot 10^{-10} \pm 1.5 \cdot 10^{-09}$
Test set	MSE	$2.9 \cdot 10^{+02} \pm 1.7 \cdot 10^{+03}$	$1.5 \cdot 10^{+02} \pm 9.4 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 9.4 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 9.6 \cdot 10^{+02}$
	\mathcal{P}	$2.8 \cdot 10^{+01} \pm 2.4 \cdot 10^{+01}$	$9.0 \cdot 10^{+01} \pm 8.2 \cdot 10^{+01}$	$3.8 \cdot 10^{+01} \pm 2.8 \cdot 10^{+01}$	$8.5 \cdot 10^{-10} \pm 1.5 \cdot 10^{-09}$

TABLE IV. NNs presented in Figure 3b (Mean MSE/Penalty \pm Standard deviation)

addressing biases at the “residual” vertical level. While good performance can be obtained for all values of z_2 , it is preferable to use $z_1 \lesssim 20$ (Figures 2e). Consistently with Figures 2a-d, there is a general trade-off between overall MSE and decreasing the “residual” bias, which suggests using $\beta \lesssim 2$ to maintain competitive performance (Figures 2f-g). For completeness, we identify the 5 NNs of lowest validation MSE (circled in black) and evaluate their squared error and Log.Bias over the test set (Figures 2i-l).

All NNs exhibit biases at their “residual” levels, which we list in Figure 2i’s legend. We can conveniently visualize these “residual” biases using the Log.Bias defined in equation 13, except near the surface in which case we may directly use the squared error. Finally, we note that the highest-performing NNs use larger β for lower z_2 , confirming that $(\beta, z_1, z_2) \in [1, 10] \times [0, 29]^2$ must be optimized simultaneously as part of the HP search as their effects on MSE are not independent.

C.3. Advantages over enforcing constraints after training

ACnets enforce constraints to within numerical precision *during* training because they use constraints layers within their architecture. A more common method to ensure NNs satisfy constraints to within numerical precision is to enforce constraints *after* training, referred to as “post-processing” [9] (abbreviated to pp hereafter): An unconstrained NN only predicts $(p - n)$ outputs and the n other outputs are calculated as residuals *after* the NN is trained (SM Figure 6). To clarify how ACnets compare to pp, we train 5 pp UCnets that have the same architecture as our 5 ACnets of lowest validation MSE from SM C.2.c, except for the constraints layers that are moved outside of the NN. By construction, pp UCnets enforce constraints to within numerical precision.

To evaluate the performance of pp UCnets, we first compare the MSE of their “direct” outputs with that of ACnets in SM Table VIII over the validation (Line 1) and test (Line 5) datasets. Both “direct” MSEs are systematically to within 3% of each other, confirming that UCnets have been trained well enough to perform

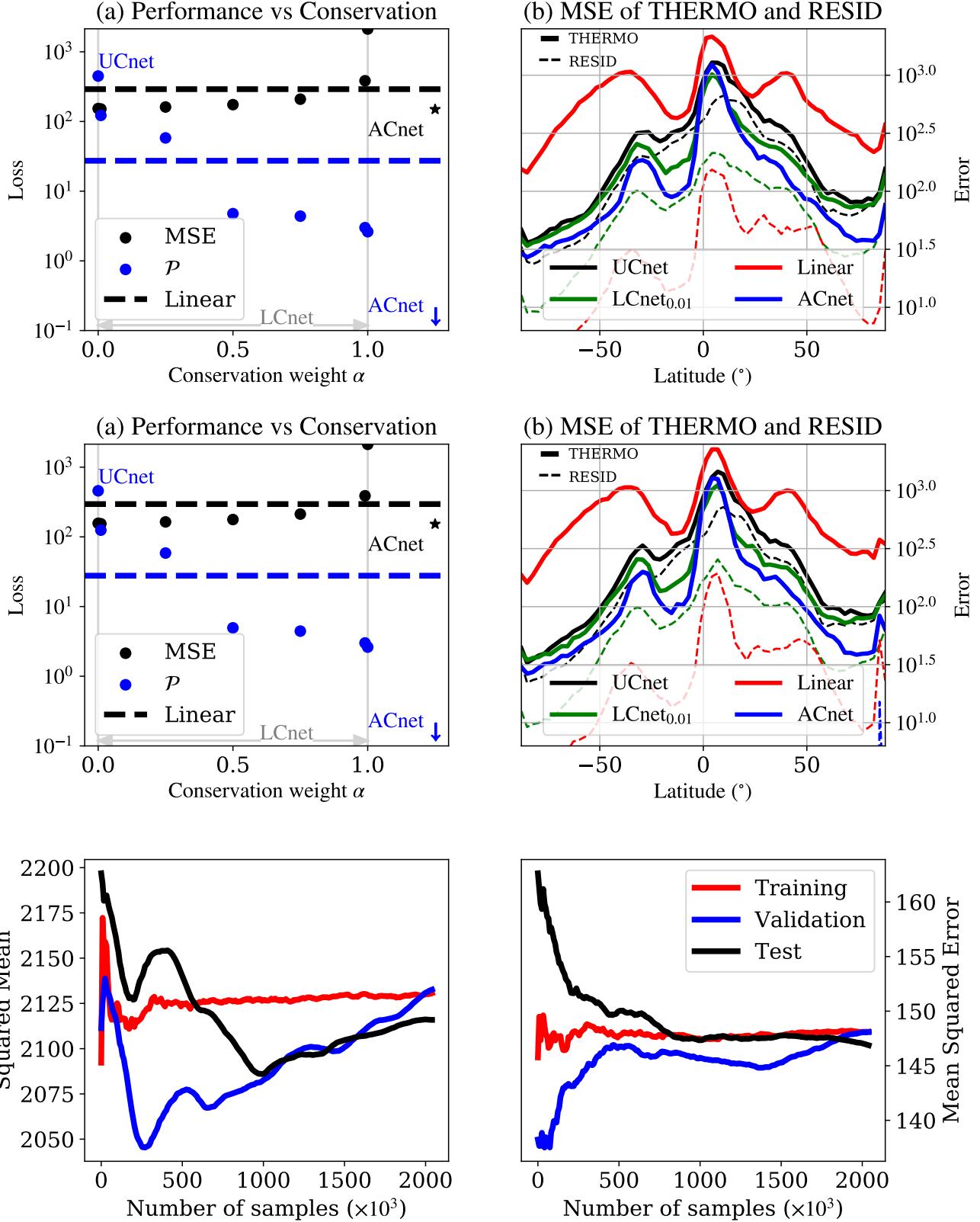


FIG. 4. Figure 3 reproduced for the training (top row) and validation (middle row) sets. (Bottom row) Convergence of the squared mean output and UCnet's mean-squared error calculated over the training, validation, and test sets for large sample size.

Validation	Metric	$\alpha = 0$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 0.99$
Training set	MSE	$1.5 \cdot 10^{+02} \pm 9.6 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 9.9 \cdot 10^{+02}$	$1.7 \cdot 10^{+02} \pm 1.0 \cdot 10^{+03}$	$2.1 \cdot 10^{+02} \pm 1.2 \cdot 10^{+03}$	$3.8 \cdot 10^{+02} \pm 1.8 \cdot 10^{+03}$
	\mathcal{P}	$4.5 \cdot 10^{+02} \pm 4.5 \cdot 10^{+02}$	$5.8 \cdot 10^{+01} \pm 6.2 \cdot 10^{+01}$	$4.8 \cdot 10^{+00} \pm 4.7 \cdot 10^{+00}$	$4.4 \cdot 10^{+00} \pm 2.6 \cdot 10^{+00}$	$3.0 \cdot 10^{+00} \pm 1.9 \cdot 10^{+00}$
Validation set	MSE	$1.6 \cdot 10^{+02} \pm 9.8 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 1.0 \cdot 10^{+03}$	$1.8 \cdot 10^{+02} \pm 1.1 \cdot 10^{+03}$	$2.1 \cdot 10^{+02} \pm 1.2 \cdot 10^{+03}$	$3.9 \cdot 10^{+02} \pm 1.8 \cdot 10^{+03}$
	\mathcal{P}	$4.6 \cdot 10^{+02} \pm 4.7 \cdot 10^{+02}$	$5.9 \cdot 10^{+01} \pm 6.4 \cdot 10^{+01}$	$4.9 \cdot 10^{+00} \pm 4.9 \cdot 10^{+00}$	$4.4 \cdot 10^{+00} \pm 2.7 \cdot 10^{+00}$	$3.0 \cdot 10^{+00} \pm 1.9 \cdot 10^{+00}$
Test set	MSE	$1.5 \cdot 10^{+02} \pm 9.7 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 1.0 \cdot 10^{+03}$	$1.7 \cdot 10^{+02} \pm 1.0 \cdot 10^{+03}$	$2.1 \cdot 10^{+02} \pm 1.2 \cdot 10^{+03}$	$3.8 \cdot 10^{+02} \pm 1.8 \cdot 10^{+03}$
	\mathcal{P}	$4.5 \cdot 10^{+02} \pm 4.7 \cdot 10^{+02}$	$5.8 \cdot 10^{+01} \pm 6.5 \cdot 10^{+01}$	$4.8 \cdot 10^{+00} \pm 4.8 \cdot 10^{+00}$	$4.4 \cdot 10^{+00} \pm 2.6 \cdot 10^{+00}$	$3.0 \cdot 10^{+00} \pm 1.9 \cdot 10^{+00}$

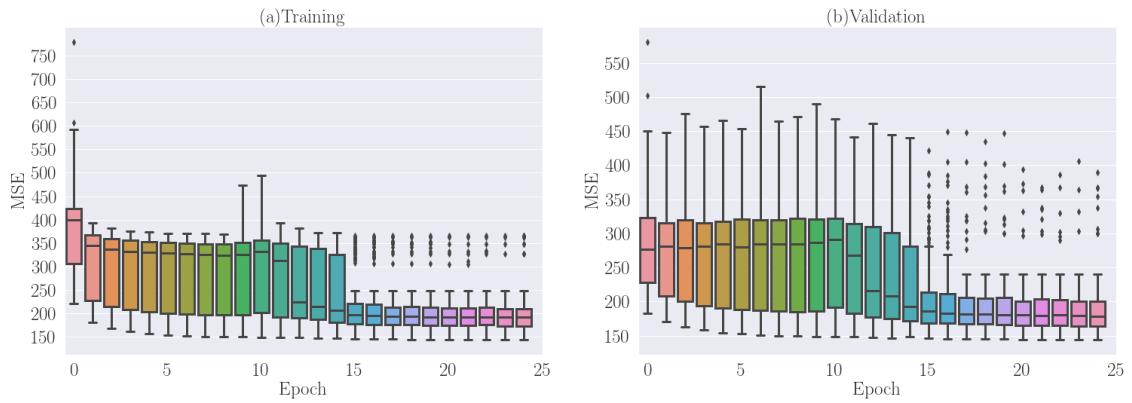
TABLE V. LCnets of varying weight α , presented in Figure 3a (Mean MSE/Penalty \pm Standard deviation)

FIG. 5. Training and validation MSE versus number of epochs from more than 200 models.

TABLE VI. Hyperparameter Space

Name	Options	Parameter Type
Batch Normalization	[yes, no]	Choice
Dropout	[0, 0.25]	Continuous
Leaky ReLU coefficient	[0 - 0.4]	Continuous
Learning Rate	[0.0001 - 0.01]	Continuous (log)
Nodes per Layer	[300 - 700]	Discrete
Number of layers	[3 - 8]	Discrete

TABLE VII. Best HP configuration for UCnets ($\alpha = 0$)

Batch Normalization	No
Dropout	0.00975
Leaky ReLU coefficient	0.25373
Learning Rate	0.000977
Number of layers	5
Nodes per Layer	[625, 517, 543, 538, 692]

as well as ACnets. However, as “residual” outputs are calculated after training and hence not optimized using data, their MSEs are significantly larger for pp UCnets than for ACnets, except for the $q_4 T_4 \beta_{1.35}$ ACnet whose MSE for \dot{T}_4 exceeds that of the corresponding pp UCnet (green spike in SM Figure 2j and 2l). Nevertheless, the $q_8 T_4 \beta_{1.49}$ ACnet example proves that increasing β can decrease the MSE for \dot{T}_4 so that it performs better than the corresponding pp UCnet (blue spike in SM Figure 2j and 2l). This affirms the superiority of ACnets over pp UCnets in predicting “residual” outputs as ACnets allow to decrease residual biases by weighting the loss function (SM B.3), which is unfeasible for pp UCnets whose loss function only includes “direct” outputs. That being said, the overall MSE of ACnets is always smaller but to within 3% of the overall MSE of pp UCnets. This means that the main advantage of ACnet over pp UCnets is exposing “residual” outputs to data during training, thereby improving corresponding predictions.

D. Extending to Nonlinear Constraints: An Example

In Section III of the main text, we focused on the mapping $\mathbf{x} \mapsto \mathbf{y}$, where \mathbf{x} is given by Equation 10 and \mathbf{y} is given by Equation 11. For this mapping, the constraints – conservation of column enthalpy, mass and radiation

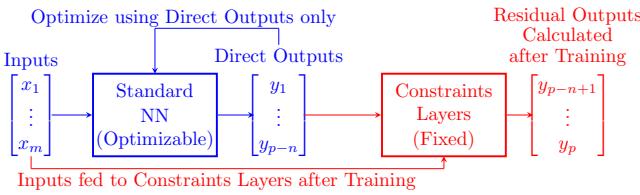


FIG. 6. “Post-processing” UCnet: A standard NN is trained on “direct” outputs only (blue) and the “residual” outputs are calculated after training, outside of the NN (red).

– were conveniently *linear* and given by $\mathbf{C} [\mathbf{x} \ \mathbf{y}]^T = \mathbf{0}$, where the constraints matrix \mathbf{C} is defined by Equation 12.

It is natural to wonder whether the approach demonstrated in this context is extendable to *nonlinear* constraints as occur in many other physical systems such as enforcing kinetic energy conservation in a mapping from momentum to momentum time-tendencies.

Here, we show that indeed a similar strategy can be used to successfully enforce *nonlinear* physical constraints in a neural-network emulating a different mapping $\mathbf{x}_0 \mapsto \mathbf{y}_0$. For simplicity, we will work on the same problem of thermodynamic convective parameterization for climate modeling. A nonlinearity is introduced in the formulation of the humidity variable. In SM D.1, we closely follow the steps of Figure 1 to enforce the resulting nonlinear constraints in a version of our architecture-constrained neural network that is adapted with “conversion layers”. In SM D.2, we show that the resulting network enforces nonlinear constraints to within excellent precision while maintaining high performance. Detailed documentation of the thermodynamics equations used for this network are given in SM D.3. The code for this network can be found at https://github.com/tbeucler/CBRAIN-CAM/blob/master/notebooks/tbeucler_devlog/041_ACnet_Non_Linear.ipynb.

D.1 Formulation

Following Figure 1, the first step is to define the inputs and outputs of the mapping. Again, we map the local climate to how water vapor, liquid water, ice, and temperature are redistributed in the vertical, but this time we describe water vapor in the input vector \mathbf{x}_0 and output vector \mathbf{y}_0 using *relative humidity* (definition below) instead of specific humidity. In addition to being a helpful illustration of how to handle nonlinearities, the physical context here is that there are reasons to think that reformulating moisture in this way will have advantages in improving the generalization of convection NNs trained in one climate to make predictions in another (e.g. warmer) one – an active research frontier. While specific humidity increases sharply with temperature, relative humidity normalizes specific humidity so as to maintain values between 0 and 1 across climates, helpfully avoiding out-of-sample issues.

Mathematically, relative humidity is defined as the ratio of the partial pressure of water vapor $e(\mathbf{p}, \mathbf{q}_v)$ to its saturation value $e_{\text{sat}}(T)$, and can be expressed analytically:

$$\text{RH} \stackrel{\text{def}}{=} \frac{e(\mathbf{p}, \mathbf{q}_v)}{e_{\text{sat}}(T)} \stackrel{\text{def}}{=} \frac{R_v}{R_d} \frac{\mathbf{p}\mathbf{q}_v}{e_{\text{sat}}(T)}, \quad (14)$$

where $R_v \approx 461 \text{ J kg}^{-1} \text{ K}^{-1}$ is the specific gas constant for water vapor, $R_d \approx 287 \text{ J kg}^{-1} \text{ K}^{-1}$ is the specific gas

Dataset	MSE	$q_8 T_4 \beta_{1.49}$	$q_3 T_{26} \beta_{1.15}$	$q_4 T_4 \beta_{1.35}$	$q_5 T_5 \beta_{1.72}$	$q_{18} T_{28} \beta_{1.01}$
Validation set	Direct	$1.6 \cdot 10^{+02}, 1.6 \cdot 10^{+02}$				
	z_1	$4.4 \cdot 10^{+01}, 2.4 \cdot 10^{+02}$	$5.9 \cdot 10^{+01}, 1.7 \cdot 10^{+02}$	$8.5 \cdot 10^{+01}, 1.6 \cdot 10^{+02}$	$6.6 \cdot 10^{+01}, 1.4 \cdot 10^{+02}$	$8.2 \cdot 10^{+02}, 9.4 \cdot 10^{+02}$
	z_2	$4.6 \cdot 10^{+01}, 5.8 \cdot 10^{+01}$	$1.6 \cdot 10^{+02}, 2.5 \cdot 10^{+02}$	$8.4 \cdot 10^{+01}, 5.0 \cdot 10^{+01}$	$9.3 \cdot 10^{+01}, 1.1 \cdot 10^{+02}$	$1.5 \cdot 10^{+02}, 2.2 \cdot 10^{+02}$
	Total	$1.6 \cdot 10^{+02}, 1.6 \cdot 10^{+02}$				
Test set	Direct	$1.6 \cdot 10^{+02}, 1.6 \cdot 10^{+02}$	$1.5 \cdot 10^{+02}, 1.6 \cdot 10^{+02}$			
	z_1	$4.4 \cdot 10^{+01}, 2.4 \cdot 10^{+02}$	$5.8 \cdot 10^{+01}, 1.7 \cdot 10^{+02}$	$8.5 \cdot 10^{+01}, 1.6 \cdot 10^{+02}$	$6.5 \cdot 10^{+01}, 1.4 \cdot 10^{+02}$	$8.0 \cdot 10^{+02}, 9.1 \cdot 10^{+02}$
	z_2	$4.5 \cdot 10^{+01}, 5.7 \cdot 10^{+01}$	$1.6 \cdot 10^{+02}, 2.5 \cdot 10^{+02}$	$8.4 \cdot 10^{+01}, 5.0 \cdot 10^{+01}$	$9.3 \cdot 10^{+01}, 1.1 \cdot 10^{+02}$	$1.5 \cdot 10^{+02}, 2.2 \cdot 10^{+02}$
	Total	$1.6 \cdot 10^{+02}, 1.6 \cdot 10^{+02}$				

TABLE VIII. Comparing ACnets to “post-processed” UCnets: MSE of “direct”, “residual”, and overall outputs. In each cell, the MSE for ACnets (left) is separated by a comma from the MSE for “post-processed” UCnets (right).

constant for dry air, \mathbf{p} (in units Pa) is the total atmospheric pressure, \mathbf{q}_v (in units kg/kg) is specific humidity, and $e_{\text{sat}}(\mathbf{T})$ (in units Pa) is the saturation pressure of water vapor, whose analytic expression in our case is given in SM D.3. As $e_{\text{sat}}(\mathbf{T})$ increases approximately exponentially with absolute temperature, relative humidity is a strongly nonlinear function of specific humidity and temperature. Therefore, the two first constraints in the present example, namely conservation of mass and energy, are strongly nonlinear with respect to

the new relative humidity input and output. In summary, our input vector \mathbf{x}_0 is equal to \mathbf{x} (Equation 10) with \mathbf{q}_v replaced by \mathbf{RH} , our output vector \mathbf{y}_0 is equal to \mathbf{y} (Equation 11) with $\dot{\mathbf{q}}_v$ replaced by $\dot{\mathbf{RH}}$, and our constraints $\mathbf{c}(\mathbf{x}_0, \mathbf{y}_0) = \mathbf{0}$ are strongly nonlinear:

$$\mathbf{x}_0 = [(\mathbf{RH}, \mathbf{q}_l, \mathbf{q}_i, \mathbf{T}, \mathbf{v}, \mathbf{LS}, p_s, S_0) \quad \text{SHF} \quad \text{LHF}]^T, \quad (15)$$

$$\mathbf{y}_0 = [\dot{\mathbf{RH}} \quad \dot{\mathbf{q}}_l \quad \dot{\mathbf{q}}_i \quad \dot{\mathbf{T}} \quad \dot{\mathbf{T}}_{KE} \quad \text{lw} \quad \text{sw} \quad \text{LW}_t \quad \text{LW}_s \quad \text{SW}_t \quad \text{SW}_s \quad P \quad P_i]^T. \quad (16)$$

The second step is to write \mathbf{c} as an explicit sum of (1) \mathbf{x} only dependent on \mathbf{x}_0 ; and (2) \mathbf{y} dependent on \mathbf{x}_0 and \mathbf{y}_0 . In our case, we can choose the mapping $\mathbf{x} \mapsto \mathbf{y}$ described in the main paper, where conservation of mass and energy can be written as an explicit sum of specific humidity tendencies. This natural choice yields Formulation 2 in Figure 1, i.e. a linearly-constrained mapping related to the nonlinearly constrained mapping of interest via a bijective function. The last step is to build the network that maps \mathbf{x}_0 to \mathbf{y}_0 while enforcing the constraints $\mathbf{c}(\mathbf{x}_0, \mathbf{y}_0) = \mathbf{C} [\mathbf{x} \quad \mathbf{y}]^T = \mathbf{0}$, which are nonlinear with respect to $(\mathbf{x}_0, \mathbf{y}_0)$ but linear with respect to (\mathbf{x}, \mathbf{y}) . For that purpose, we augment the ACnet architecture described in Figure 2 with two “conversion” layers encoding the moist thermodynamics described in SM D.3:

1. A conversion layer ($\mathbf{RH} \mapsto \mathbf{q}_v$) calculating \mathbf{q}_v based on $(\mathbf{RH}, \mathbf{T})$ to convert \mathbf{x}_0 to \mathbf{x} before ACnet.
2. A conversion layer ($\dot{\mathbf{q}}_v \mapsto \dot{\mathbf{RH}}$) calculating $\dot{\mathbf{RH}}$ based on $(\dot{\mathbf{q}}_v, \dot{\mathbf{T}})$ to convert \mathbf{y} to \mathbf{y}_0 after ACnet.

The resulting network, which we refer to as the nonlinear ACnet (ACnet_{NL} for short), is depicted in SM Figure 7c. It is worth remarking that the idea of “conversion layers” should easily be adaptable to other physical systems

with nonlinear constraints, such as “converting” velocity components into a kinetic energy upstream of an ACnet that enforces its linear conservation.

D.2 Results

In this section, we compare the performance and constraints penalty of three types of NNs, all mapping $\mathbf{x}_0 \mapsto \mathbf{y}_0$:

1. An unconstrained network (UCnet),
2. An unconstrained network using the two conversion layers ($\mathbf{RH} \mapsto \mathbf{q}_v$) and ($\dot{\mathbf{q}}_v \mapsto \dot{\mathbf{RH}}$), referred to as the nonlinear UCnet (UCnet_{NL} for short), to assess the effect of using conversion layers on optimization independently of their actual purpose to manage nonlinear constraints,
3. A nonlinearly constrained network that exploits the two conversion layers to adapt the idea of ACnet to a nonlinear setting (ACnet_{NL}).

UCnet_{NL} and ACnet_{NL} are implemented using custom Tensorflow layers for the conversion layers. For each network type, we train three networks (total of 9 NNs) for 20 epochs using the RMSprop optimizer and save the state of minimal validation loss to avoid overfitting. We report

the performance and constraints penalty of UCnet_{NL} and ACnet_{NL} for the validation and test sets in Table IX. We complement Table IX with the vertical profile of the squared error in (\dot{q}_v, \dot{T}) for all three networks in SM Figure 8.

The main result of this section is that we *successfully enforced nonlinear constraints in NNs to excellent approximation*, as can be seen by the constraints penalty of ACnet_{NL} (Table VIII, bottom-right cell), which is 8 orders of magnitude smaller than that of UCnet_{NL} for both validation and test sets. Interestingly, the constraints penalty is more than one order of magnitude lower for UCnet_{NL} than for UCnet, suggesting that unconstrained NNs are able without direction to approximate linear constraints better than strongly nonlinear constraints.

Table VIII also reveals how a reliance on conversion layers to handle nonlinear constraints does impact optimization with some training trade-offs in ways that are worth future research. As expected, UCnets are easiest to optimize. They exhibit mean-squared errors that are lower by a factor ~ 1.5 compared to NNs using “conversion” layers (Table VIII). This suggests that NNs with nonlinear conversion layers are somewhat harder to optimize. However, the absolute difference in MSE between UCnet_{NL} and ACnet_{NL} is smaller than $10 \text{ W}^2 \text{m}^{-4}$, analogous to the small difference in MSE between UCnets and ACnets reported in Section III of the main text – a price worth paying for the benefit of machine precision adherence to nonlinear constraints.

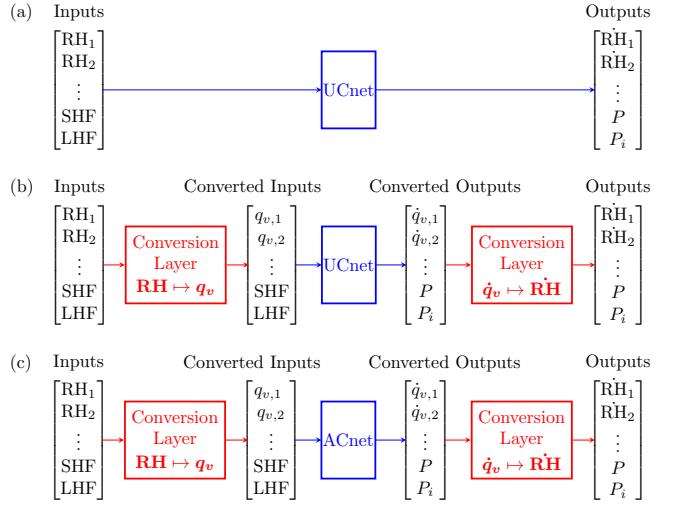


FIG. 7. (a) UCnet: Directly maps relative humidity inputs to relative humidity tendency outputs.
(b) UCnet_{NL}: Inputs are converted from relative humidity to specific humidity before UCnet, after which outputs are converted back from specific humidity tendencies to relative humidity tendencies.
(c) ACnet_{NL}: Same as (b) with ACnet instead of UCnet.

D.3 Saturation pressure of water vapor

In this section, we present the thermodynamics equations used to calculate the saturation pressure of water vapor $e_{\text{sat}}(\mathbf{T})$, which allows to convert between specific and relative humidities. The saturation pressure of water vapor can be found by integrating the Clausius-Clapeyron equation with respect to temperature. Under the microphysical assumptions of our fine-scale, cloud-resolving model [10], it can be expressed analytically as:

$$e_{\text{sat}}(\mathbf{T}) = \begin{cases} e_{\text{liq}}(\mathbf{T}) & \mathbf{T} > T_0 = 273.16\text{K} \\ e_{\text{ice}}(\mathbf{T}) & \mathbf{T} < T_{00} = 253.16\text{K} . \\ \omega e_{\text{liq}}(\mathbf{T}) + (1 - \omega) e_{\text{ice}}(\mathbf{T}) & \mathbf{T} \in [T_{00}, T_0] \end{cases} \quad (17)$$

In Equation 17, as temperature increases, the saturation pressure of water vapor goes from the saturation

vapor pressure with respect to liquid e_{liq} , given by the following polynomial approximation:

$$e_{\text{liq}}(\mathbf{T}) = 100\text{Pa} \times \sum_{i=0}^8 a_{\text{liq},i} [\max(-193.15\text{K}, T - T_0)]^i , \quad (18)$$

where \mathbf{a}_{liq} is a vector of length 9 containing nonzero polynomial coefficients, to the saturation vapor pressure with

respect to ice e_{ice} , given by a different polynomial approximation with temperature switches:

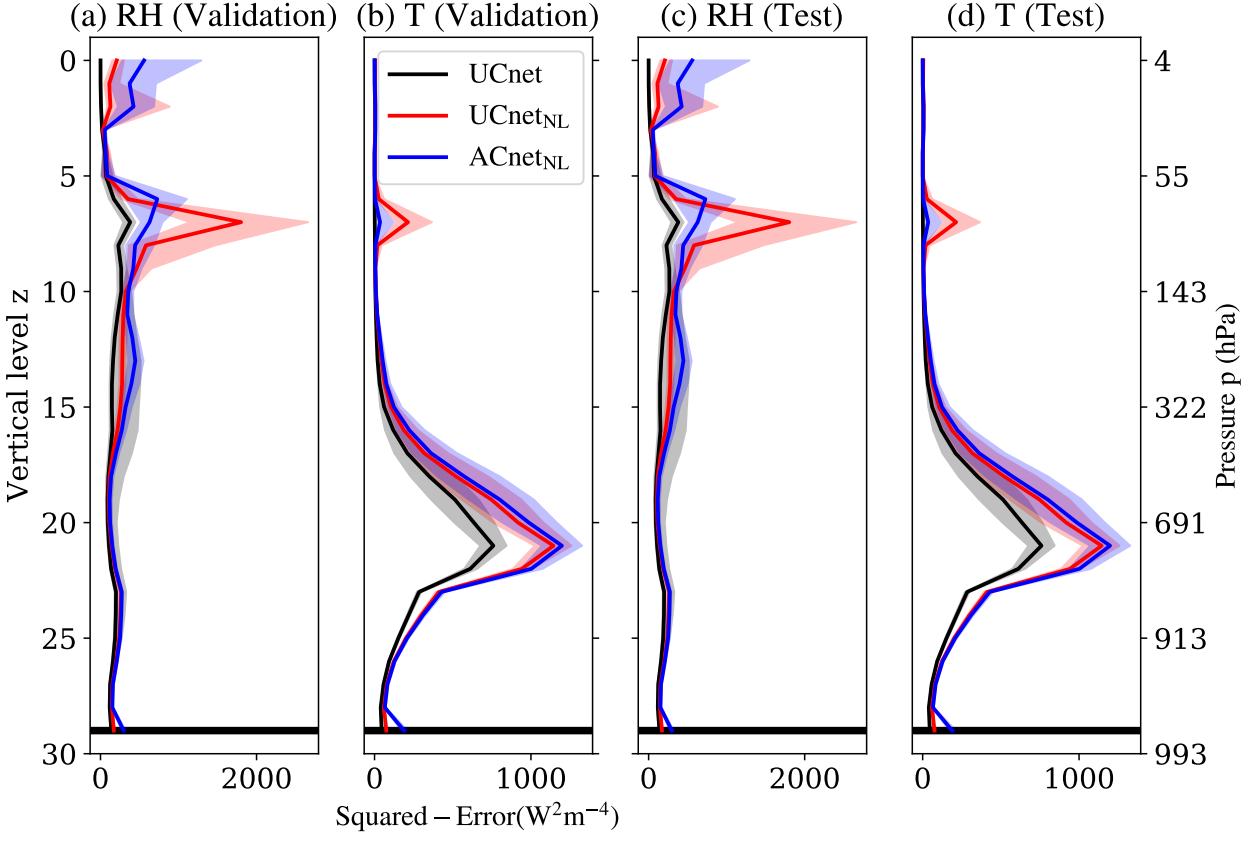


FIG. 8. For UCnet, UCnet_{NL}, and ACnet_{NL}: Squared error in convective moistening \dot{q}_v and heating \dot{T} versus pressure, for the validation and test sets. Each color represents a different NN type. We depict the median squared-error using full lines and shade the region between the first and third quartiles. Statistics use the full dataset and three NNs for each type (nine total).

Validation	Metric	UCnet	UCnet _{NL}	ACnet _{NL}
Validation set	MSE	$9.3 \cdot 10^{+01} \pm 4.4 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 5.9 \cdot 10^{+02}$	$1.6 \cdot 10^{+02} \pm 5.7 \cdot 10^{+02}$
	\mathcal{P}	$7.2 \cdot 10^{+04} \pm 7.3 \cdot 10^{+05}$	$3.9 \cdot 10^{+03} \pm 4.0 \cdot 10^{+04}$	$2.1 \cdot 10^{-04} \pm 7.3 \cdot 10^{-04}$
Test set	MSE	$9.3 \cdot 10^{+01} \pm 4.5 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 6.0 \cdot 10^{+02}$	$1.5 \cdot 10^{+02} \pm 5.9 \cdot 10^{+02}$
	\mathcal{P}	$6.9 \cdot 10^{+04} \pm 7.3 \cdot 10^{+05}$	$3.8 \cdot 10^{+03} \pm 4.2 \cdot 10^{+04}$	$2.1 \cdot 10^{-04} \pm 7.1 \cdot 10^{-04}$

TABLE IX. NNs presented in SM Figure 7 (Ensemble mean of Mean MSE/Penalty \pm Standard deviation). MSE is calculated using relative humidity tendencies, explaining its smaller values.

$$e_{\text{ice}}(\mathbf{T}) = \begin{cases} e_{\text{liq}}(\mathbf{T}) & \mathbf{T} > T_0 \\ 100\text{Pa} \times \{c_{\text{ice},1} + \mathcal{C}(\mathbf{T}) [c_{\text{ice},4} + c_{\text{ice},5}\mathcal{C}(\mathbf{T})]\} & \mathbf{T} < T_{00} \\ 100\text{Pa} \times \sum_{i=0}^8 a_{\text{ice},i} (\mathbf{T} - T_0)^i & \mathbf{T} \in [T_{00}, T_0] \end{cases}, \quad (19)$$

where $\mathcal{C}(\mathbf{T})$ is a ramp function of temperature given by:

$$\mathcal{C}(\mathbf{T}) \stackrel{\text{def}}{=} \max(c_{\text{ice},2}, \mathbf{T} - T_0), \quad (20)$$

and $(\mathbf{a}_{\text{ice}}, \mathbf{c}_{\text{ice}})$ are vectors of length 9 and 5 containing nonzero elements, respectively. Between temperatures of T_{00} and T_0 , the saturation pressure of water vapor is a

weighted mean of e_{liq} and e_{ice} , where the weight ω is a linear function of the absolute temperature:

$$\omega \stackrel{\text{def}}{=} \frac{\mathbf{T} - T_{00}}{T_0 - T_{00}}. \quad (21)$$

The reader interested in the numerical details of this calculation is referred to our implementation of rel-

ative humidity at https://github.com/tbeucler/CBRAIN-CAM/blob/master/notebooks/tbeucler_devlog/041_ACnet_Non_Linear.ipynb.

E. Enforcing Inequality Constraints in Architecture-Constrained Networks

In this section, we briefly discuss how to enforce inequality constraints in ACnets using a concrete example: the positivity of liquid water concentration in this letter’s NN parameterization of convection. This inequality requires the liquid water concentration $q_{l,z}(t)$ at a given vertical level z and at the current timestep t to be positive. In practice, the liquid water concentration $q_{l,z}(t)$ is obtained from the liquid water concentration $q_{l,z}(t-1)$

and the liquid water tendency $\dot{q}_{l,z}(t-1)$ at the previous timestep $t-1$ through time-stepping, which means that we can write the positivity constraint as:

$$q_{l,z}(t) = \overbrace{q_{l,z}(t-1)}^{\text{Input}} + \Delta t \times \overbrace{\dot{q}_{l,z}(t-1)}^{\text{Output}} \geq 0, \quad (22)$$

where we note that $q_{l,z}(t-1)$ is an NN input while $\dot{q}_{l,z}(t-1)$ is an NN output.

To enforce this inequality constraint in ACnet, we choose $\dot{q}_{l,z}(t-1)$ to be a “direct” NN output and add an inequality constraints layer (ICL) before the constraints layer (CL). Although any positive-definite activation function can be used, a possible implementation of (ICL) uses the rectified linear unit (ReLU) activation function:

$$\underbrace{\dot{q}_{l,z}(t-1)}_{\text{After (ICL)}} = \text{ReLU} \left[\underbrace{\dot{q}_{l,z}(t-1)}_{\text{Before (ICL)}} + \Delta t^{-1} q_{l,z}(t-1) \right] - \Delta t^{-1} q_{l,z}(t-1), \quad (23)$$

so that the liquid water tendency after (ICL) yields a positive liquid water concentration $q_{l,z}(t)$ at the current timestep. Since the inequality constraints layer (ICL) comes before the constraints layer, which do not change $q_{l,z}(t)$, we can still enforce the equality constraints (\mathcal{C})

to within machine precision. Finally, note that the previous example can be generalized to nonlinear analytic constraints involving less than $(p-n)$ inputs, allowing ACnets to enforce a broad range of inequality constraints.

-
- [1] M. F. Khairoutdinov and D. a. Randall, Cloud Resolving Modeling of the ARM Summer 1997 IOP: Model Formulation, Results, Uncertainties, and Sensitivities, *Journal of the Atmospheric Sciences* **60**, 607 (2003).
 - [2] W. D. Collins, P. J. Rasch, B. A. Boville, J. J. Hack, J. R. McCaa, D. L. Williamson, B. P. Briegleb, C. M. Bitz, S. J. Lin, and M. Zhang, The formulation and atmospheric simulation of the Community Atmosphere Model version 3 (CAM3), *Journal of Climate* **19**, 2144 (2006).
 - [3] P. Gentine, M. Pritchard, S. Rasp, G. Reinaudi, and G. Yacalis, Could Machine Learning Break the Convection Parameterization Deadlock?, *Geophysical Research Letters* **45**, 5742 (2018).
 - [4] J. Yuval and P. A. O’Gorman, Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions, *Nature Communications* **11**, 1 (2020).
 - [5] A. Krizhevsky and G. Hinton, Learning multiple layers of features from tiny images.(2009), Cs.Toronto.Edu (2009).
 - [6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, *International Journal of Computer Vision* **115**, 211 (2015), arXiv:1409.0575.
 - [7] L. Hertel, P. Sadowski, J. Collado, and P. Baldi, Sherpa : Hyperparameter Optimization for Machine Learning Models, *Conference on Neural Information Processing Systems (NIPS)* (2018).
 - [8] J. Bergstra and Y. Bengio, Random search for hyperparameter optimization, *Journal of Machine Learning Research* **13**, 281 (2012).
 - [9] T. Bolton and L. Zanna, Applications of Deep Learning to Ocean Data Inference and Subgrid Parameterization, *Journal of Advances in Modeling Earth Systems* **11**, 376 (2019).
 - [10] M. F. Khairoutdinov and D. A. Randall, Cloud Resolving Modeling of the ARM Summer 1997 IOP: Model Formulation, Results, Uncertainties, and Sensitivities, *Journal of the Atmospheric Sciences* **60**, 607 (2003).