LESSONS LEARNED

Two-Step Hyperparameter Optimization Method: Accelerating Hyperparameter Search by Using a Fraction of a Training Dataset

SUNGDUK YU, a PO-LUN MA, b BALWINDER SINGH, b SAM SILVA, c AND MIKE Pritchard a,d

a Department of Earth System Sciences, University of California, Irvine, Irvine, California
b Pacific Northwest National Laboratory, Richland, Washington
c Department of Earth Sciences, University of Southern California, Los Angeles, Los Angeles, California
d NVIDIA, Santa Clara, California

(Manuscript received 10 February 2023, in final form 17 August 2023, accepted 23 October 2023)

ABSTRACT: Hyperparameter optimization (HPO) is an important step in machine learning (ML) model development, but common practices are archaic—primarily relying on manual or grid searches. This is partly because adopting advanced HPO algorithms introduces added complexity to the workflow, leading to longer computation times. This poses a notable challenge to ML applications, as suboptimal hyperparameter selections curtail the potential of ML model performance, ultimately obstructing the full exploitation of ML techniques. In this article, we present a two-step HPO method as a strategic solution to curbing computational demands and wait times, gleaned from practical experiences in applied ML parameterization work. The initial phase involves a preliminary evaluation of hyperparameters on a small subset of the training dataset, followed by a reevaluation of the top-performing candidate models postretraining with the entire training dataset. This two-step HPO method is universally applicable across HPO search algorithms, and we argue it has attractive efficiency gains. As a case study, we present our recent application of the two-step HPO method to the development of neural network emulators for aerosol activation. Although our primary use case is a data-rich limit with many millions of samples, we also find that using up to 0.0025% of the data—a few thousand samples—in the initial step is sufficient to find optimal hyperparameter configurations from much more extensive sampling, achieving up to 135× speedup. The benefits of this method materialize through an assessment of hyperparameters and model performance, revealing the minimal model complexity required to achieve the best performance. The assortment of top-performing models harvested from the HPO process allows us to choose a higher-performing model with a low inference cost for efficient use in global climate models (GCMs).

KEYWORDS: Climate models; Optimization; Artificial intelligence; Deep learning; Machine learning; Neural networks

1. Introduction

The application of artificial intelligence/machine learning (AI/ML) techniques in Earth system science is becoming increasingly popular in recent years. However, our field’s adaptation of modern hyperparameter optimization (HPO) or tuning techniques—that is, advanced search algorithms and extensive search space—has been slow. For example, a manual or a grid search is still commonly used in published studies despite the availability of ample HPO software packages and GPU resources. The question naturally arises: Do our ML applications exploit the full potential of ML models? This issue becomes more pressing as ML architectures are getting more sophisticated, that is, more hyperparameters need to be optimized, “the curse of dimensionality.”

We suspect that the perceived large investment of time and computing power is a major barrier preventing the widespread adoption of modern HPO practices. There are certainly modern ways to reduce computation time: for example, distributed search (i.e., evaluating multiple hyperparameter configurations simultaneously), adaptive search algorithms [i.e., selecting hyperparameter configurations based on the result of preceding hyperparameter evaluations, e.g., Bayesian methods (Snoek et al. 2012)], and adaptive resource allocations [i.e., allocating more resources to promising hyperparameter configurations using a tournament selection, e.g., hyperband (Li et al. 2016)]. Nonetheless, an HPO task can still take a significant amount of time, as both training dataset sizes and ML model complexity have been increasing.

Beyond the conventional strategies such as parallel computing and adaptive algorithms, we further explore accelerating HPO tasks based on a core-set (also known as instance selection) approach, which focuses on reducing the size of the dataset by selecting samples representative of the original dataset. The core-set approach has gained much attention in the past decade in the computer science domain to shorten model training and data processing time for big data (Feldman 2020; Mirzasoleiman et al. 2020; Killamsetty et al. 2021a,b, 2022). Specifically for HPO, Wendt et al. (2020) presented an interesting two-phase HPO approach: using a small subset for a wide search with low granularity and then using a full dataset...
for a final search within the narrowed domain identified from the wide search. They showed a 10% subset combined with a random search algorithm can reliably return the same results as a single-phase random search but 7 times faster. Furthermore, Visalpara et al. (2021) showed that HPO with a 5% subset can achieve a result comparable to HPO with a full dataset. Despite these promising reports, the applicability of a core-set approach still needs verification for a real-world problem since these studies are based on toy datasets that contain only several 10,000 instances [e.g., Canadian Institute for Advanced Research, 100 classes (CIFAR-100) and Modified National Institute of Standards and Technology (MNIST)]. Testing comparable ideas in the context of an applied machine-learning parameterization problem relevant to hybrid AI climate simulation is new.

In this article, we test a core-set approach for a real-world Earth system modeling problem and propose a two-step HPO method, which could be viewed as a variant of Wendt et al. (2020) and Li et al. (2016) but with much simpler applicability. Our method is designed to be a general framework that is independent of the specific search algorithms or computing environments used. The central idea behind our method is that HPO applied initially to a small subset of a training dataset can effectively identify optimal hyperparameter configurations, thus reducing the overall computational cost. We demonstrate the effectiveness of our approach through a case study in which we apply our two-step HPO method to optimize a neural network emulator for aerosol activation processes. Our results show that the two-step HPO method can effectively discover optimal hyperparameter configurations using only a small subset of the training dataset (as low as 0.025% in our case).

The rest of the article is organized as follows. In section 2, we present our two-step HPO method, including an estimation of cost savings. Section 3 is a case study of our method applied to optimize a neural network emulator. Section 4 covers the computational setup of our case study. Results are presented in section 5, and section 6 concludes with a summary of findings and potential implications.

2. Two-step HPO method and estimated computational cost saving

Our two-step HPO method is designed to reduce the computational burden of traditional HPO by using a small subset of a training dataset. In the first step, a large number of trials are conducted using only a small portion of the dataset. In the second step, the top candidates are retrained with a full training dataset for the final selection (Fig. 1a). The second step also serves the purpose of recalibrating model weights with a full training dataset, resulting in final models that are ready for deployment. This approach assumes that hyperparameter evaluations using a subset are indicative of using the entire dataset in data-rich limits. A model trained with a small dataset is expected to be less accurate than one trained with a large dataset. However, if this assumption is valid, the accuracy of top models after retraining in step 2 should converge regardless of the size of the subset used in step 1. This will be empirically tested in the case study presented in the following sections.

Our two-step HPO method can meaningfully reduce the computational burden required for HPO, compared to the traditional one-step approach. Two parameters must be preset
to use this method: a portion of an entire training dataset to be used in step 1 \(p_{\text{subset}}\) and a portion of total hyperparameter evaluation trials from step 1 to be retrained in step 2 \(p_{\text{retrain}}\). Then, the fractional computational cost of the two-step approach is approximately equal to \(p_{\text{subset}} + p_{\text{retrain}}\), that is,

\[
\text{two-step HPO} \Rightarrow \text{(traditional) one-step HPO} = \frac{\text{step 1 HPO} + \text{step 2 HPO}}{\text{(traditional) one-step HPO}} = \frac{p_{\text{subset}}X + (p_{\text{retrain}})^n}{X} = p_{\text{subset}} + p_{\text{retrain}},
\]

where \(X\) is the total computation cost [e.g., graphical processing unit (GPU) hour] of a traditional one-step HPO for evaluating \(n\) trials, number of hyperparameter configurations and \(\tau\) is an average computational cost for evaluating one trial with a full training dataset (i.e., \(\tau = X/n\)). This estimation assumes the training time linearly scales to the size of a training dataset for a given software and hardware environment.

3. Case study: Aerosol activation emulator

To demonstrate the effectiveness of the two-step HPO, we present our recent HPO work for aerosol activation emulators. Aerosol activation (or “droplet nucleation”) is the process that describes the spontaneous growth of aerosol particles into cloud droplets in an ascending air parcel. This process occurs at 10–100-μm scales and accordingly is parameterized with varying degrees of assumptions in weather and climate models of which the grid resolutions are orders of magnitude larger (Abdul-Razzak and Ghan 2000; Ghan et al. 2011). Instead of relying on such parameterizations, a neural network (Abdul-Razzak and Ghan 2000; Ghan et al. 2011). Instead of relying on such parameterizations, a neural network-based model enables direct emulation of detailed cloud parcel models that more explicitly simulate the droplet nucleation process in a rising air parcel with minimal assumptions.

We generate a dataset for developing deep neural network emulators using explicit numerical calculations of aerosol activation based on the Ghan et al. (2011) cloud parcel model. The initial conditions for the cloud parcel model simulations are populated by harvesting hourly instantaneous atmospheric state variables from a yearlong simulation of the U.S. Department of Energy’s Energy Exascale Earth System Model (E3SM) (Golaz et al. 2019). Atmosphere Model (EAM), version 1 (Rasch et al. 2019). A total of 98.6 million training samples are obtained and are divided into a train/validation dataset for HPO (19.7 million samples; 20%) and a holdout test dataset (78.8 million samples; 80%). The input and output variables for the emulators are listed below. Note that the aerosol-related variables have a dimension of four \(q\) as both E3SM and the cloud parcel model are set up with four aerosol modes corresponding to the four-mode version of the modal aerosol module (MAM4; Liu et al. 2016; Wang et al. 2020).

Inputs:
- “Tair”: Temperature (K)
- “Pressure”: Pressure (hPa)
- “rh”: relative humidity
- “wbar”: vertical velocity (cm s\(^{-1}\))

Outputs:
- “num_aer [nmodes]”: aerosol number (cm\(^{-3}\))
- “r_aer [nmodes]”: aerosol dry radius (μm)
- “kappa [nmodes]”: hygroscopicity

4. Hyperparameter optimization

a. Two-step HPO setup

For good sampling, a computationally ambitious HPO project with an identical setup is repeated four times with decreasing subset sizes for step 1: 100%, 50%, 25%, 5%, 0.5%, 0.05%, and 0.025% (i.e., \(p_{\text{subset}} = 1.00, 0.50, 0.25, 0.05, 0.005, 0.0005, 0.00025\), respectively) containing about 19.7 million, 9.9 million, 4.9 million, 1.0 million, 98 600, 9 900, and 4 900 samples, respectively. Hereinafter, these projects will be referred to as P1.00, P0.50, P0.25, P0.05, P0.005, P0.0005, and P0.00025 projects. The P1.00 serves as a control experiment that the rest of the subset HPO projects are evaluated against.

Each project includes an unusually expansive 10 000 trials of hyperparameter evaluation (i.e., \(n_{\text{trials}} = 10 000\)). The train/validation split ratio is 80:20 for the P1.00, P0.50, P0.25, and P0.05 and 50:50 for the P0.005, P0.0005, and P0.00025. For step 2, the top 50 trials from each project are retrained with a full training dataset (i.e., \(p_{\text{retrain}} = 0.005\)).

b. Hyperparameter search space

We use a multilayer perceptron (MLP) as a machine learning architecture for our emulator, which has been applied for emulating physical processes in various applications including aerosol activation and other microphysical processes (Silva et al. 2021; Gettelman et al. 2021; Chiu et al. 2021; Alfonso and Zamora 2021). Silva et al. (2021) employed a modern HPO workflow but only with a limited number of tuning trials (~400). To ensure the robust sampling of the hyperparameter search space, we focus on only two key hyperparameters (the number of hidden layers and the number of nodes in each layer) that define the backbone (i.e., complexity) of the MLP neural networks. However, any arbitrary combinations of hyperparameters (batch size, learning rate, optimizer, regularizer, etc.) can be included in an HPO task. The search space for each hyperparameter is as follows:

- The number of hidden layers \(N_{\text{Layers}}\): \([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]\)
- The number of nodes \(N_{\text{Nodes}}\) in each layer: \([8, 16, 32, 64, 128, 256, 512, 1024, 2048]\)

Note that \(N_{\text{Nodes}}\) is selected independently for each hidden layer. That is, for \(N_{\text{Layers}} = k\), \(N_{\text{Nodes}}\) is drawn from the search space \(k\) times. The rest of the hyperparameters are fixed as follows: we use a ReLU activation function for the hidden layers, a sigmoid activation for the output layer, MSE as a loss function, and an Adam optimizer (Kingma and Ba 2014) as the gradient descent algorithm with a batch size of 1024.
FIG. 2. (a) Minimum validation MSEs of all evaluation trials after sorted by their minimum validation MSE from step 1. (b) Comparison between minimum validation MSE from step 2 and minimum validation MSE from step 1 for the top step 1 candidate models (50 per HPO project). Relationship between the minimum validation MSE and the (c) number of hidden layers and (d) learnable parameters. (e) Step 1 models with different rank groups, e.g., ranks 1–50, 1001–1050, 2001–2050, 3001–3050, and 4001–4050, are retrained with a full training.
and a learning rate of 0.001. We enforce an early stopping rule with a patience of 5 epochs and maximum training epochs of 100. The neural network weights and validation MSE are recorded when early stopping is called.

c. A search algorithm

We use a random search algorithm, which is easy to parallelize and fault tolerant by design. Despite its simplicity, a random search algorithm is known to be more efficient than—or at least as good as—a grid search (Bergstra and Bengio 2012). We note that our two-step HPO can be applied to any HPO algorithm since it only concerns the size of a training dataset. However, our two-step HPO approach may not be suitable for adaptive search algorithms, which rely on the result of preceding hyperparameter evaluations to assign hyperparameter configurations for succeeding evaluations. Unlike a random (or nonadaptive) algorithm that uniformly samples hyperparameters under a given distribution regardless of a training dataset size, an adaptive algorithm may converge toward different optima depending on the training dataset (i.e., the information content of a training dataset).

d. A computational setup for distributed search

We use a distributed search setup to speed up the process, using GPU nodes on the Bridges-2 supercomputer at the Pittsburgh Supercomputing Center (Towns et al. 2014). For that, we choose KerasTuner for its seamless integration with Tensorflow/Keras libraries (https://github.com/keras-team/keras-tuner); however, our two-step method should be easily implemented in other popular HPO software frameworks (e.g., Hyperopt, Optuna, and Ray Tune). The distributed mode of KerasTuner uses a manager–worker model. Despite the user-friendly interface of KerasTuner, setting up a distributed search mode on a high-performance computing (HPC) cluster with a job scheduler can be complex due to dynamic computing environments. For readers’ reference, code examples of how we set up distributed HPO on Bridge-2 are included in the online supplemental material.

We use two GPU nodes, totaling 16 GPUs (8 GPUs/node; NVIDIA Tesla V100-32GB SXM2). One GPU is assigned for one worker, allowing for concurrent evaluations of 16 hyperparameter configurations. The elapsed wall-clock time for computation scales linearly with the size of the training dataset, for example, 5.2, 23.4, 48.6, and 94.3 h for P0.05, P0.25, P0.50, and P1.00 projects, respectively. On the other hand, the HPO projects with smaller subsets, for example, P0.00025, P0.0005, and P0.005, are conducted with only a half GPU node (4 GPUs) taking 2.8, 3.0, and 5.0 wall-clock hours, respectively. Note that the computational resources (e.g., GPU-hour) for P1.00 are about 135 times more than those for P0.00025. The absence of linear scaling in P0.00025–P0.005 is due to the increasing relative portion of computational overhead (tasks not directly related to training neural networks, e.g., file I/O) as the volume of the training set decreases.

5. Results

An HPO project with a larger training dataset consistently yields trial models with lower validation MSEs after step 1, except for ones that have markedly poor performance, for example, validation MSE > 1 × 10⁻² (Fig. 2a and Fig. S1 in the online supplemental material). While it can be tempting to assume that data volume is all that matters, our working hypothesis is that just the top few architectures revealed in HPO projects with smaller subsets could perform just as well as the best architectures sampled at much greater expense in P1.00, once exposed to the full data in a second stage. One can imagine alternative views, such as an HPO with a smaller dataset being self-limiting in the sense that using a smaller dataset may result in different optima than an HPO with a larger dataset. However, we view this as unlikely since we use a random search algorithm in which the selection of trial hyperparameters is independent of the evaluations of preceding trials.

If our hypothesis is valid, we expect that top architectures harvested from HPO projects with smaller training datasets (P0.00025–P0.50) shall have comparable performance to those from the HPO project that used the full training dataset (P1.00). We verify this hypothesis in step 2: the top 50 trial models from each HPO project are retrained with the full training dataset, and then, their performances are reexamined by comparing the validation MSE from step 1 (using a subset) and that from step 2 (using a full dataset).

The results affirm our hypothesis: HPO with a small subset of a training dataset is found to be capable of discerning competent hyperparameter configurations (Fig. 2b and Fig. S2). Despite the notable performance gaps across the four HPO projects in step 1 (spreads across the x axis in Fig. 2b), the performances of top trial models of the subset HPO projects (P0.00025–P0.50) after retraining generally converge to those from the P1.00 project within the same order of magnitude (convergence on the y axis in Fig. 2b). Figure 2b and Fig. S2 appear to exhibit a weak but noticeable sensitivity, for example, increasing model errors as the subset size decreases; however, this is found to be an artifact due to the randomness introduced during training (further discussed in section 6). Admittedly, one cannot escape a few models that suffer degraded performance after retraining (i.e., all models with MSE larger than 1 × 10⁻² except in P0.005 and P0.00025); model overfitting or stochastic aspects of training such as during weight initialization or shuffling samples before each epoch can cause such issues. This appears more prevalent in dataset, and then, their minimum validation MSEs are displayed as box diagrams. Each rank group contains 50 models. The box shows the interquartile range (IQR) with a line at the median, and the whiskers are 1.5 times the IQR. Filer points are outliers, i.e., those beyond the whiskers. Figures S4–S6 show similar figures as in (c)–(e), but each HPO project is plotted in a separate subpanel.
models with higher complexity, such as those with more hidden layers or more learnable parameters (Fig. S3). But the main point is the reassuringly similar skill of all fits after retraining (step 2) in Fig. 2b, which affirms the two-step HPO method. In particular, it is remarkable that HPO with only a 0.025% subset or 5000 samples (P0.00025; cf. 20 million samples in P1.00) is capable of identifying top-performing architectures.

Furthermore, the unusually extensive HPO project performed here provides an opportunity to examine the effect of hyperparameters on the model performance. Figures 2c and 2d reveal a minimum model complexity required for faithfully emulating the given aerosol activation dataset: at least two or three hidden layers and $10^4$–$10^5$ learnable parameters are required for optimal performance. Interestingly, the minimum learnable parameter for maximum performance seems to depend on the subset size, for example, smaller for larger $p_{\text{subset}}$ ($10^2$ for P0.00025 and $10^4$ for P1.00). While it is not the main purpose of HPO, analyzing the relationship between hyperparameters and model performance provides useful information to understand model architectures and develop further refined models in the next iteration, although we admit it is undoubtedly problem specific.

The accuracy of predictions by selected ML models after step 2 is illustrated in Figs. 3a–e by mapping the truth values (aerosol activation fractions calculated using the cloud parcel model) and the predicted values (predicted by a neural network emulator), using the holdout test dataset. The selected models from the entire step 2 pool (total of 350 models) include the top three performers and the lightest and the heaviest models in terms of the number of learnable parameters. Note that the erroneous models whose MSE is larger than $1310^2$ are excluded from the selection process. All five neural network models display stark improvement from the current aerosol activation parameterization used in E3SM (Fig. 3f). However, the difference among them is barely noticeable despite large differences in model complexity—for example, the number of learnable parameters of the five models spans three orders of magnitudes from $7.9310^3$ to $1.1310^7$. Remarkably, the lightest model (Fig. 3d) performs almost as well as the best model (Fig. 3a), which has about 1400 times more learnable parameters. Extensive HPO was key to finding this lucky architecture.

The diversity of resulting top-performing models is another key merit of an extensive HPO. As shown in Table S1, the model architectures of top-performing models vary significantly, providing users with a range of options depending on their specific application needs. For example, in the case of our aerosol activation emulators, the lightest model, although

![Image](Hexagonally binned histogram with aerosol activation fractions (first mode only) computed by the cloud parcel model (x axis) and estimated by different neural network emulators (y axis). The subplots correspond to (a) the best emulator, (b) the second-best emulator, (c) the third-best emulator, (d) the lightest emulator, (e) the heaviest emulator, and (f) E3SM’s current parameterization (Abdul-Razzak and Ghan 2000). The total number of samples in this held-out test dataset is $7.9 \times 10^7$, and bins less than 100 samples (0.0001% of the total sample) are shown in gray. The MSE and coefficient of determination ($R^2$), the number of hidden layers (#Layers), and the total number of learnable parameters (#Param) are displayed in the top-left corner of each subplot.)
not the best performer, is a more strategic choice for inference within a climate model due to its computational efficiency given that it maintains performance (Fig. S7).

6. Setting parameters for two-step HPO

Just like in any HPO method, our two-step HPO approach requires several preset parameters: p\textsubscript{subset} (the portion of the training set used for step 1 evaluation) and p\textsubscript{retrain} (the portion of step 1 trials used for step 2 reevaluation) in addition to n\textsubscript{trials} (the number of trials, a parameter universally applicable for any HPO method). Here, we attempt to provide insights into choosing these parameters based on our case study.

Figure 2a shows that model performance asymptotically converges after step 1. For instance, for P0.005–P1.00, models with ranks below 6000 (“tier-1”) have validation MSE within the same order of magnitude. Given this outcome, the subsequent query that arises is how much model performance differs across the tier-1 models after retraining with the full training dataset. If performance is consistent for the tier-1 models post-retraining, a substantial value of n\textsubscript{trials} (e.g., 10000) is not required to identify top candidates for step 2 (e.g., 50 models) in our case study. This inquiry arises from our discovery that, among the top 50 models in each category, step 1 ranks do not determine step 2 ranks (Fig. S8)—implying that the performance of models with equally competitive hyperparameter configurations is primarily governed by stochasticity during training (e.g., random weight initialization and minibatch effects).

To address this question, we cluster step 1 models with different rank groups (e.g., ranks 1–50, 1001–1050, 2001–2050, 3001–3050, and 4001–4050) and proceed to retrain them using the full training dataset (step 2). We then make a box chart for the step 2 validation MSEs per step 1 rank group (see Fig. 2c and Fig. S6). In the box chart, the step 2 MSEs exhibit minimal variation among models belonging to different rank groups, except for P0.0005 and P0.00025 where step 1 model performances do not show clear convergence (Fig. 2a). This result indicates that we could have employed a significantly smaller n\textsubscript{trials} in step 1 to identify the top 50 candidates for step 2. For example, since we needed 50 top candidates in the asymptotically converged region, employing n\textsubscript{trials} = 100 for P0.005–P1.00 and n\textsubscript{trials} = 100 for P0.0005–P0.00025 would have sufficed, based on a random sampling of existing trials (Fig. S9). However, to fully exploit the computational efficiency of this two-step HPO method, using a larger number of n\textsubscript{trials} is still beneficial, especially with smaller p\textsubscript{subset} values. In our case study, approximately 1000 n\textsubscript{trials} is necessary to identify the relationship between hyperparameters and model performance (Figs. S10 and S11).

This retrospective analysis points out the potential utility of a model performance versus a rank diagram (akin to Fig. 2a) to ascertain the adequacy of the three parameters within the two-step HP method. An effective parameter configuration should yield asymptotic convergence in the performance–rank diagram, and n\textsubscript{trials} should be large enough that the number of top candidates for step 2 (p\textsubscript{retrain} × n\textsubscript{trials}) falls within the region of asymptotic convergence. However, further investigation and additional case studies are required to attain more thorough guidance on the choice of two-step HPO parameters.

7. Summary

We have shared our lesson learned that a two-step approach is an attractive way to efficiently optimize hyperparameters in extensive architecture searches. In the first step, we perform HPO with a small subset of a training dataset to identify a set of hyperparameter combinations, that is, viable architectures. In the second step, we retrain the top-performing configurations with the entire training dataset for the final selection and the recalibration of model weights. Our case study of optimizing neural network emulators for aerosol activation showed that the two-step HPO with only 0.025% of the training dataset was effective in finding optimal hyperparameter configurations. Our case study, which is to our knowledge the first in-depth application of the core-set approach popularized in computer science literature (e.g., Feldman 2020) to an operational climate modeling parameterization research problem, confirms the usefulness of the core-set approach for problems with a wide range of data volumes (e.g., 5000–20 million samples as shown in our case study). Additionally, we demonstrated other benefits of extensive HPO (i.e., many trials with large hyperparameter search domains). For example, an extensive HPO provides an opportunity to learn the relationship between hyperparameters and model performance. Moreover, it offers the option to choose the right hyperparameter configuration among comparable options, in the balance between fit precision and inference efficiency that is relevant to hybrid ML–physics climate modeling.

We hope that this two-step HPO method, and appendixes illustrating its practical use on GPU clusters, will prove a practical solution for ML practitioners who crave the benefits of extensive HPO, by reducing the time and computation involved. Additionally, this method can be easily tailored to specific needs. For example, the selection of top candidates for the second step can be modified (e.g., choosing only ones with two or three hidden layers if a compact architecture is desired for inference efficiency), or the second step could be another HPO process but with a more focused search space.

We note that our two-step HPO method has only been demonstrated using a random search algorithm and its applicability with other search algorithms remains to be explored. Despite this limitation, the potential cost savings offered by the two-step approach with random search may outweigh the benefits of using an adaptive algorithm. Random searches are attractively parallel and can scale across GPU nodes. Determining the optimal values for the parameters required for the two-step HPO method (n\textsubscript{trials}, p\textsubscript{subset}, and p\textsubscript{retrain}) is also an open question that warrants further research. In our case study, n\textsubscript{trials} = 10000, p\textsubscript{subset} = 0.00025, and p\textsubscript{retrain} = 0.005 were effective, but the optimal values may vary depending on the complexity of a training dataset and the expansiveness of a hyperparameter search space.

We hope others may share lessons learned from applying similar techniques in other settings. Machine learning parameterization is an empirical art, and much is yet to be learned.
from dense empirical sampling. Meanwhile, independent testing of this method has enabled competitive results beyond the aerosol nucleation use case discussed here, in a new benchmark project focused on ML parameterization for convection and radiation parameterization (Yu et al. 2023).

Acknowledgments. We thank Elizabeth A. Barnes and two anonymous reviewers for providing valuable feedback that improved our manuscript significantly as well as our UC colleagues Savannah Ferretti, Jerry Lin, and Yan Xia for helpful comments. This study was supported by the Enabling Aerosol–Cloud Interactions at Global Convection-Permitting Scales (EAGLES) project (project 74358), funded by the U.S. Department of Energy (DOE), Office of Science, Office of Biological and Environmental Research, Earth System Model Development (ESMD) program area. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. DOE Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract DE-AC02-05CH11231 using NERSC Awards ALCC-ERCP0016315, BER-ERCP0015329, BER-ERCP0018473, and BER-ERCP0020990. This research also used resources of the Argonne Leadership Computing Facility, a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357, using an award of computer time provided by the Advanced Scientific Computing Research (ASCR) Leadership Computing Challenge (ALCC) program. The Pacific Northwest National Laboratory is operated for the U.S. DOE by the Battelle Memorial Institute under Contract DE-AC05-76RL01830. Additionally, this work used XSEDE’s PSC Bridges-2 system (National Science Foundation Grants ACI-1548562 and ACI-1928147). We thank David Walling for his assistance with HPC support through the XSEDE Extended Collaborative Support Service program.

Data availability statement. The aerosol activation dataset used in this study is openly available in the Zenodo data repository (https://doi.org/10.5281/zenodo.7627777). The two-step hyperparameter optimization codes used in this study are openly available in the GitHub repository (https://github.com/sungdukyu/Two-step-HPO) and described in the supplemental material.

REFERENCES


