arXiv:2302.03845v2 [cs.LG] 8 Sep 2023

两步超参数优化方法：通过使用部分训练数据集加速超参数搜索 *

Sungduk Yu
加利福尼亚大学欧文分校地球系统科学系，加利福尼亚州欧文市，
sungduk@uci.edu

Mike Pritchard
加利福尼亚大学欧文分校地球系统科学系，加利福尼亚州欧文市

Po-Lun Ma
太平洋西北国家实验室，华盛顿州里奇兰

Balwinder Singh
太平洋西北国家实验室，华盛顿州里奇兰

Sam Silva
地球科学系，南加州大学，洛杉矶，加利福尼亚州

摘要

超参数优化（HPO）是机器学习（ML）模型开发中的重要步骤，但常见做法仍然过时，主要依赖手动搜索或网格搜索。这在一定程度上是因为采用先进的超参数优化算法会增加工作流程的复杂性，从而延长计算时间。这给机器学习应用带来了显著挑战，因为不理想的超参数选择会限制机器学习模型性能的潜力，最终妨碍充分发挥机器学习技术的作用。本文结合应用机器学习参数化工作的实践经验，提出一种两步超参数优化方法，作为减少计算需求和等待时间的策略。第一阶段是在训练数据集的一个小子集上对超参数进行初步评估，然后对表现最佳的候选项进行重新评估

*本作品已提交至地球系统人工智能（ISSN:2769-7525）。本作品的版权可能会在不另行通知的情况下转让。

模型使用整个训练数据集重新训练后，再对其进行评估。这种两步式超参数优化方法适用于各种超参数优化搜索算法，我们认为它能显著提升效率。

作为一项案例研究，我们介绍了近期将两步超参数优化方法应用于气溶胶活化神经网络模拟器开发的工作。尽管我们的主要应用场景是数据丰富的情形，拥有数百万个样本，但我们也发现，在初始步骤中使用最多仅占数据量 0.0025% 的数据（数千个样本），便足以从规模大得多的采样中找到最优超参数配置，实现最高达 135× 的加速。通过评估超参数和模型性能，该方法的优势得以显现，并揭示了实现最佳性能所需的最低模型复杂度。超参数优化过程中筛选出的各类顶尖模型，使我们能够选择推理成本低且性能出色的模型，以便高效应用于全球气候模式（GCMs）。

关**键词** 气候模式 · 优化 · 深度学习 · 神经网络 · 人工智能 · 机器学习

1 引言

近年来，人工智能/机器学习（AI/ML）技术在地球系统科学中的应用日益普及。然而，我们这一领域对现代超参数优化（HPO）或调参技术的采用——即先进的搜索算法和广阔的搜索空间——进展缓慢。例如，尽管已有大量超参数优化软件包和图形处理器资源可用，已发表的研究仍普遍采用人工搜索或网格搜索。于是自然会产生这样一个问题：我们的机器学习应用是否充分发挥了机器学习模型的潜力？随着机器学习架构日益复杂，需要优化的超参数也越来越多，这一问题愈发突出，也就是所谓的“维数灾难”。

我们认为，人们感知到的巨大时间和计算资源投入，是阻碍现代超参数优化实践广泛采用的一大障碍。当然，现今有多种方法可以缩短计算时间：例如，分布式搜索（即同时评估多个超参数配置）、自适应搜索算法（即根据先前超参数评估的结果选择超参数配置，例如贝叶斯方法（Snoek 等，2012）），以及自适应资源分配（即通过锦标赛选择，将更多资源分配给有前景的超参数配置，例如 Hyperband（Li 等，2016））。尽管如此，超参数优化任务仍可能耗费大量时间，因为训练数据集规模和机器学习模型复杂度都在不断增长。

除了并行计算和自适应算法等传统策略外，我们还探索基于核心集（也称为实例选择）方法来加速超参数优化任务。该方法通过选取能够代表原始数据集的样本来缩小数据集规模。过去十年间，核心集方法在计算机科学领域备受关注，常用于缩短大数据的模型训练和数据处理时间（Feldman 2020年；Mirzasoleiman 等，2020年；Killamsetty 等，2021a、b，2022）。具体而言，针对超参数优化，Wendt 等（2020年）提出了一种有趣的两阶段方法：先使用一个小子集进行范围广、粒度较粗的搜索，再使用完整数据集，在广泛搜索所确定的缩小范围内进行最终搜索。他们表明，使用 10% 的子集结合随机搜索算法，能够可靠地获得与单阶段随机搜索相同的结果，且速度快 7 倍。此外，Visalpara 等（2021年）表明，使用 5% 的子集进行超参数优化，能够取得与使用完整数据集进行超参数优化相当的结果。尽管这些报告令人鼓舞，但核心集方法是否适用于现实问题仍有待验证，因为这些研究基于的都是仅包含数万个实例的玩具数据集（例如 CIFAR-100 和 MNIST）。在与混合人工智能气候模拟相关的实际机器学习参数化问题中检验类似思路，尚属首次。

本文针对一个实际的地球系统建模问题测试了核心集方法，并提出了一种两步超参数优化方法。该方法可视为 Wendt 等人（2020年）和 Li 等人（2016年）方法的一种变体，但适用性要简单得多。我们的方法旨在构建一个通用框架，不受所用搜索算法或计算环境的具体限制。该方法的核心思想是，先在训练数据集的一个小子集上进行超参数优化，即可有效确定最优超参数配置，从而降低总体计算成本。我们通过一个案例研究展示了该方法的有效性：在研究中，我们应用两步超参数优化方法优化了一个用于气溶胶活化过程的神经网络模拟器。结果表明，仅使用训练数据集的一个小子集（在本案例中低至 0.025%），两步超参数优化方法便能有效找到最优超参数配置。

本文其余部分安排如下。第 2 节介绍我们的两步超参数优化方法，包括对成本节省的估算。第3节通过案例研究介绍将该方法用于优化神经网络模拟器。第4节介绍案例研究的计算设置。第5节展示结果，第6节总结研究发现及其潜在影响。

2 通过两步超参数优化方法降低计算成本

我们提出的两步式超参数优化方法旨在通过使用训练数据集的一个小子集，降低传统超参数优化的计算负担。在第一步中，将进行大量试验

仅使用训练数据集的一小部分。在第二步中，将第一步筛选出的顶尖候选者放在完整训练数据集上重新训练，以进行最终选择（图 1a）。第二步还有助于使用完整训练数据集重新校准模型权重，从而得到可直接部署的最终模型。该方法假设，在数据丰富极限下，使用子集进行的超参数评估能够反映使用整个数据集时的情况。使用小数据集训练的模型预计不如使用大数据集训练的模型准确。然而，如果这一假设成立，那么在步骤2中重新训练后，顶尖模型的准确率应趋于一致，而不受步骤1所用子集大小的影响。以下各节介绍的案例研究将对此进行实证检验。

与传统单步方法相比，我们的两步超参数优化方法可以显著降低超参数优化所需的计算负担。使用此方法必须预先设置两个参数：步骤1中使用的完整训练数据集的比例（p_{subset}），以及从步骤1的超参数评估试验总数中选取并在步骤2中重新训练的比例（$p_{retrain}$）。因此，两步方法的计算成本占比约为 $p_{subset} + p_{retrain}$，即

$$\begin{aligned}\frac{\text{Two step HPO}}{\text{(Traditional) one step HPO}} &= \frac{\text{Step 1 HPO + Step 2 HPO}}{\text{(Traditional) one step HPO}} \\ &\approx \frac{p_{subset} \cdot X + (p_{retrain} \cdot n_{trials}) \cdot \bar{x}}{X} \\ &= p_{subset} + p_{retrain},\end{aligned}$$

其中，X 表示传统单步超参数优化在评估n_{trials} 个超参数配置时的总计算成本（例如 GPU 小时），而 $\bar{x}$ 表示使用完整训练数据集评估一次试验的平均计算成本（即 $\bar{x} = X/n_{trials}$）。该估算假设，在给定的软硬件环境下，训练时间随训练数据集规模线性变化。

3 案例研究：气溶胶活化模拟器

为展示两步式超参数优化的有效性，我们介绍近期针对气溶胶活化模拟器开展的超参数优化工作。气溶胶活化（或“液滴成核”）是指在上升空气团中，气溶胶粒子自发增长为云滴的过程。该过程发生在 10–100 μm 尺度上，因此在网格分辨率大出数个数量级的天气和气候模型中，需要在不同程度的假设下对其进行参数化（Abdul-Razzak and Ghan 2000；Ghan 等人 2011）。与依赖此类参数化方案不同，基于神经网络的模型能够直接模拟详细的云团模型；后者以最少的假设，更明确地模拟上升空气团中的液滴成核过程。

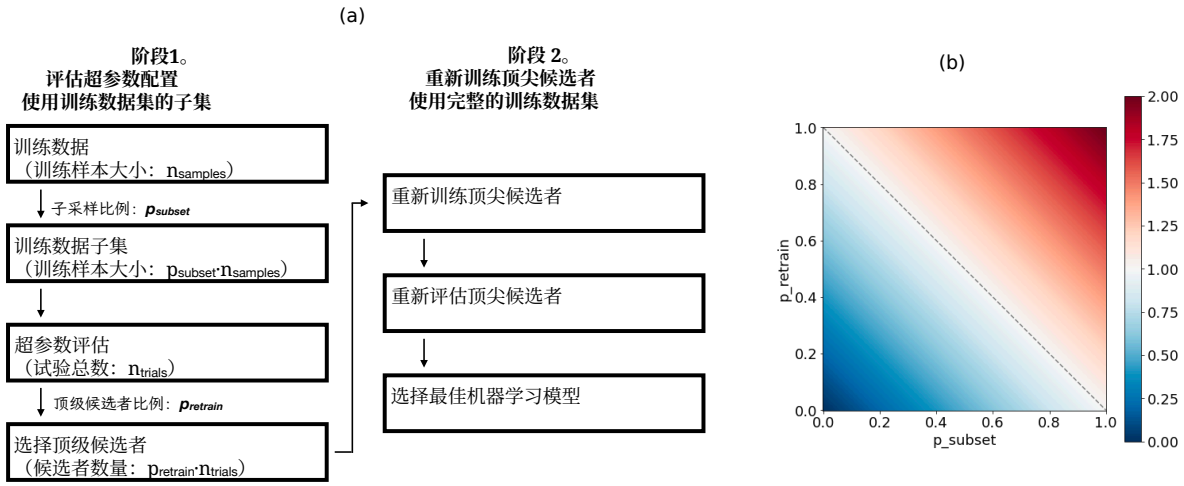


图 1：（a）两步超参数优化方法示意图。步骤1中，使用完整训练数据集的 p_{subset} 部分（共 $n_{samples}$ 个样本）进行超参数优化，试验次数为 n_{trials} 次。步骤2中，将 n_{trials} 次试验中的 $p_{retrain}$部分使用完整训练数据集重新训练，以进行最终评估并选出最佳超参数配置。（b）两步超参数优化方法相较于传统单步方法（即 $p_{subset} = 1$ 和 $p_{retrain} = 0$）所需计算资源（例如，GPU 小时）比例的近似值。计算成本的节省取决于 p_{subset} 和 $p_{retrain}$的取值。例如，如果步骤1使用数据集的 5%，并在步骤2中重新训练排名前 5% 的候选模型（即 p_{subset}= 0.05 和 $p_{retrain} = 0.05$），计算成本可降至 10%。值得注意的是，传统方法是两步超参数优化方法的一种特殊情况，其中 $p_{subset} = 1$ 和 $p_{retrain} = 0$。更多信息请参见第 2 节。

我们基于 Ghan 等人（2011）的云团模型，通过对气溶胶活化进行显式数值计算，生成用于开发深度神经网络模拟器的数据集。云团模型模拟的初始条件取自美国能源部能源百亿亿次级地球系统模型（E3SM）（Golaz 等人 2019）大气模式（EAM）1.0 版本（Rasch 等人 2019）为期一年的模拟，从中提取每小时的瞬时大气状态变量。共获得 9860 万个训练样本，并将其划分为用于超参数优化的训练/验证数据集（1970 万个样本；20%）和留出测试数据集（7880 万个样本；80%）。模拟器的输入变量和输出变量如下所列。请注意，气溶胶相关变量的维度为四（“nmodes”），这是因为 E3SM 和云团模型均设置了四种气溶胶模态，对应于模态气溶胶模块（MAM4）的四模态版本（Liu 等人 2016；Wang 等人 2020）。

输入：

- “气温”：温度，K

- “气压”：气压，hPa
- “rh”：相对湿度
- “wbar”：垂直速度，cm/s
- “num_aer [nmodes]”：气溶胶数浓度，1/cm3
- “r_aer [nmodes]”：气溶胶干半径，μm
- “kappa [nmodes]”：吸湿性

输出：

- “fn [nmodes]”：活化分数

输入变量使用 z 分数进行标准化，但输出变量的取值范围本身限定在零和一之间。

4 超参数优化

4.1 两步式超参数优化设置

为实现良好的采样，我们采用相同设置，开展计算量较大的超参数优化项目，并在步骤1中将子集大小依次缩小至 100%、50%、25%、5%、0.5%、0.05% 和 0.025%（即分别为 p_{subset} = 1.00、0.50、0.25、0.05、0.005、0.0005 和 0.00025），对应约 19.7M、9.9M、4.9M、1.0M、98.6K、9.9K 和 4.9k 个样本。下文将这些项目分别称为 P1.00、P0.50、P0.25、P0.05、P0.005、P0.0005 和 P0.00025 项目。P1.00 项目作为对照实验，用于评估其余子集超参数优化项目。每个项目均包含数量异常庞大的 10,000 次超参数评估试验（即 n_{trials} =10000）。P1.00、P0.50、P0.25 和 P0.05 项目的训练集/验证集划分比例为 80:20；P0.005、P0.0005 和 P0.00025 项目则为 50:50。步骤2中，从每个项目中选取排名前 50 的试验，并使用完整训练数据集（即 $p_{retrain}$ = 0.005）重新训练。

4.2 超参数搜索空间

我们采用多层感知机（MLP）作为模拟器的机器学习架构。该架构已应用于多种场景下的物理过程模拟，包括气溶胶活化和其他微物理过程（Silva 等人 2021年；Gettelman 等人 2021 年；Chiu 等人 2021 年；Alfonso 和 Zamora 2021 年）。Silva 等人（ 2021 年）采用了现代化的超参数优化工作流，但调参试验次数有限（∼400）。为确保对超参数搜索空间进行稳健采样，我们仅关注两个关键超参数（隐藏层的数量和

每层的节点数），它们定义了多层感知器神经网络的骨干结构（即复杂度）。不过，任何超参数组合（例如批大小、学习率、优化器、正则化等）都可以纳入超参数优化任务。每个超参数的搜索空间如下：

- 隐藏层的数量（N_{Layers}）： [1, 2, 3, 4, 5, 6, 7, 8, 9, 10 , 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]；以及
- 每层中的节点数量（N_{Nodes}）： [8, 16, 32, 64, 128, 256, 512, 1024, 2048]。

请注意， N_{Nodes} 会针对每个隐藏层独立选择。也就是说，对于 $N_{Layers} = k$， N_{Nodes}会从搜索空间中抽取k次。其余超参数固定如下：隐藏层使用ReLU激活函数；输出层使用sigmoid激活函数；损失函数使用均方误差；使用Adam优化器（Kingma和Ba，2014）作为梯度下降算法，批大小为1024，学习率为0.001。我们采用提前停止规则，耐心值为5个训练轮次，最大训练轮次为100。在触发提前停止时，记录神经网络权重和验证均方误差。

4.3 搜索算法

我们采用随机搜索算法，该算法易于并行化，并且在设计上具备容错性。尽管随机搜索算法较为简单，但已知其效率高于网格搜索，或至少与网格搜索一样好（Bergstra and Bengio 2012）。需要指出的是，我们的两步式超参数优化可应用于任何超参数优化算法，因为它只涉及训练数据集的大小。然而，我们的两步式超参数优化方法可能不适用于自适应搜索算法，因为这类算法依赖前序超参数评估的结果，为后续评估分配超参数配置。与随机（或非自适应）算法不同，后者会在给定分布下均匀采样超参数，而不考虑训练数据集的大小；自适应算法则可能会根据训练数据集（即训练数据集的信息量）收敛到不同的最优解。

4.4 分布式搜索的计算设置

我们采用分布式搜索设置来加快这一过程，使用匹兹堡超级计算中心（Towns et al. 2014）的Bridge-2超级计算机上的 GPU 节点。为此，我们选择了KerasTuner，因为它能够与TensorFlow/Keras库无缝集成（https://github.com/keras- team/keras-tuner）；不过，我们的两步法也应当能够轻松应用于其他流行的HPO软件框架（例如Hyperopt、Optuna和Ray Tune）。KerasTuner的分布式模式采用管理器-工作器模型。尽管KerasTuner的界面十分友好，但在具有作业调度器的HPC上设置分布式搜索模式可能会因动态

计算环境而变得复杂。供读者参考，补充材料中提供了我们如何在Bridge-2上设置分布式超参数优化的代码示例。

我们使用两个 GPU 节点，共计 16 个 GPU（每个节点 8 个 GPU；NVIDIA Tesla V100-32GB SXM2）。每个工作进程分配一个 GPU，因此可并行评估 16 组超参数配置。计算所需的实际运行时间随训练数据集的大小线性增长，例如，P0.05、P0.25、P0.50 和 P1.00 投影分别需要 5.2、23.4、48.6 和 94.3 小时。另一方面，较小子集的超参数优化项目（例如 P0.00025、P0.0005 和 P0.005）仅使用半个 GPU 节点（4 个 GPU），分别需要 2.8、3.0 和 5.0 小时的实际运行时间。请注意，P1.00 所需的计算资源（例如 GPU 小时）约为 P0.00025 的 135 倍。P0.00025–P0.005 未呈现线性缩放，是因为随着训练集规模缩小，计算开销所占的相对比例不断增加（即与神经网络训练没有直接关系的任务，例如文件 I/O）。

5 个结果

训练数据集较大的超参数优化项目，在步骤1后通常能得到验证均方误差较低的试验模型，但表现明显不佳的模型除外，例如验证均方误差 $>1\times10^{-2}$（图2a和图S1）。虽然人们很容易认为数据量就是决定一切的因素，但我们的工作假设是：在第二阶段接触完整数据后，小规模子集上的超参数优化项目所揭示的少数顶尖架构，其表现可能与在P1.00中花费大得多的代价采样出的最佳架构同样出色。也可以设想其他观点，例如，使用较小数据集进行超参数优化会受到自身限制，因为较小的数据集可能会产生与较大数据集不同的最优解。不过，我们认为这种情况不太可能发生，因为我们使用的是随机搜索算法，试验超参数的选择独立于此前各次试验的评估结果。

如果我们的假设成立，那么从训练数据集较小的超参数优化项目（P0.00025至P0.50）中筛选出的顶尖架构，其表现应与使用完整训练数据集的超参数优化项目（P1.00）所筛选出的架构相当。我们在步骤2中验证这一假设：对每个超参数优化项目中排名前50的试验模型，使用完整训练数据集重新训练，然后比较步骤1（使用子集）和步骤2（使用完整数据集）得到的验证均方误差，以重新评估其表现。

结果证实了我们的假设：研究发现，使用训练数据集的小子集进行超参数优化，能够识别出性能良好的超参数配置（图2b和S2）。尽管步骤1中的四个超参数优化项目之间存在显著的性能差距（图2b中横轴方向的分布范围），但重新训练后，小子集超参数优化项目（P0.00025至P0.50）的顶级试验模型性能通常会与同一数量级内的P1.00项目结果趋于一致

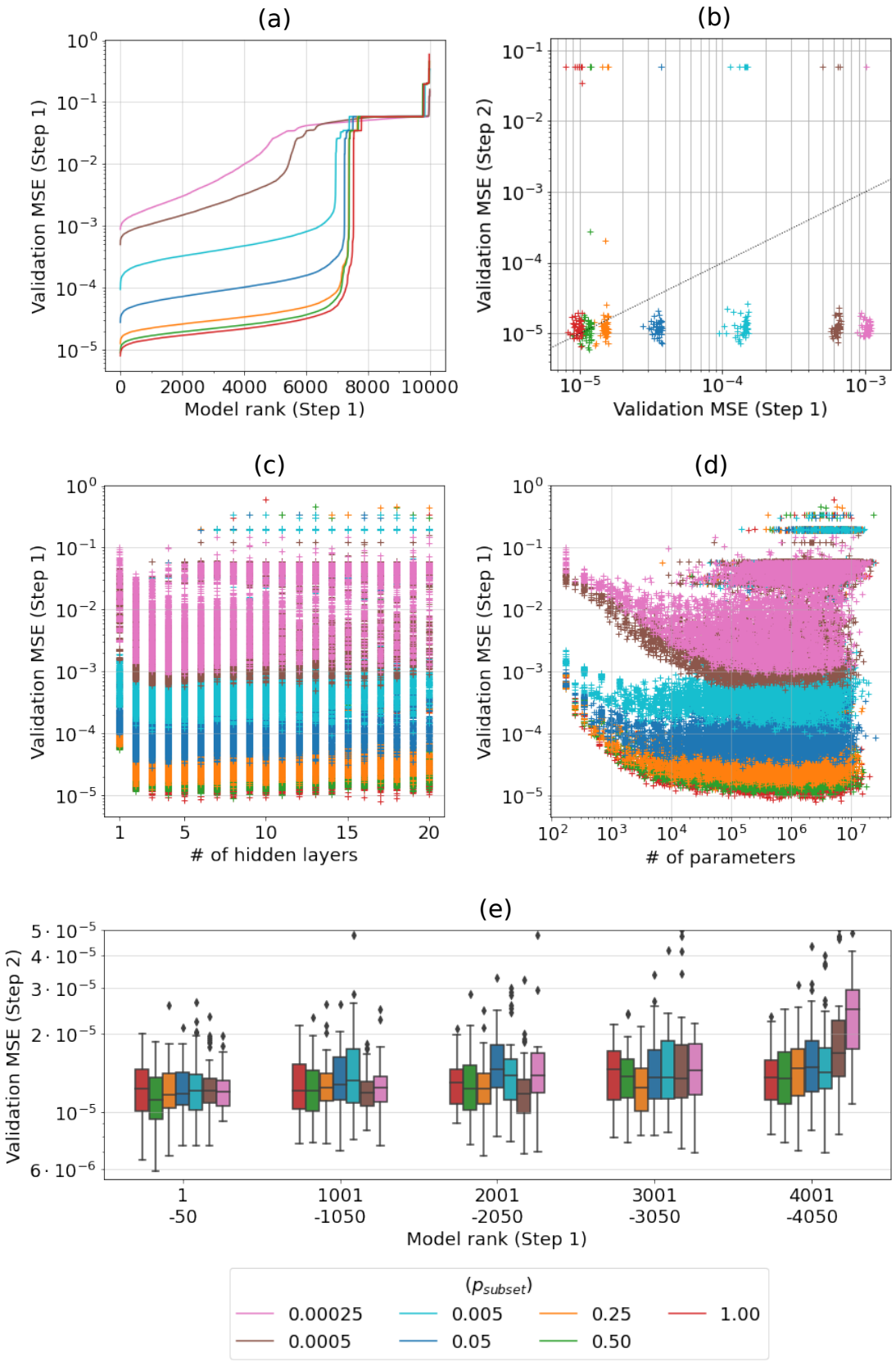

(a)
Validation MSE (Step 1)
Model rank (Step 1)
(b)
Validation MSE (Step 2)
Validation MSE (Step 1)
(c)
Validation MSE (Step 1)
# of hidden layers
(d)
Validation MSE (Step 1)
# of parameters
(e)
Validation MSE (Step 2)
1 -50
1001 -1050
2001 -2050
3001 -3050
4001 -4050
Model rank (Step 1)
(p_{subset})
0.00025
0.0005
0.005
0.05
0.25
0.50
1.00

图 2: （续见下页。）

图 2：（a）将所有评估试验按照步骤1中的最小验证均方误差排序后，显示各试验的最小验证均方误差。（b）针对步骤1中排名靠前的候选模型（每个超参数优化项目50个），比较步骤2与步骤1的最小验证均方误差。（c和d）最小验证均方误差分别与隐藏层数量和可学习参数数量之间的关系。（e）对步骤1中不同秩组的模型（例如秩1–50、1001–1050、2001–2050、3001–3050和4001–4050）使用完整训练数据集重新训练，然后将其最小验证均方误差以箱线图显示。每个秩组包含50个模型。箱体表示四分位距（IQR），其中的线表示中位数，须线范围为四分位距（IQR）的1.5倍。离群点表示异常值，即位于须线之外的点。图S4、S5和S6分别展示了与图c、d和e类似的图，但其中每个超参数优化项目都绘制在单独的子面板中。

（图 2b 中纵轴上的收敛情况）。图2b和S2似乎表现出较弱但明显的敏感性，例如，随着子集大小减小，模型误差增大；然而，这被发现是训练过程中引入的随机性所造成的假象（详见第6节）。诚然，少数模型在重新训练后性能有所下降，这种情况无法避免（即除 P0005 和 P00025 外，所有均方误差大于 1×10^{-2} 的模型）；模型过拟合或训练中的随机因素，例如权重初始化或每个训练轮次开始前打乱样本，都可能导致此类问题。这种情况似乎在复杂度较高的模型中更为普遍，例如具有更多隐藏层或更多可学习参数的模型（图 S3）。不过，重点在于图 2b 中所有模型在重新训练（步骤2）后表现出的技能相近，这令人安心，也证实了两步超参数优化方法的有效性。尤其值得注意的是，仅使用 0.025% 的子集或 ∼5K 个样本（P0.00025；相比之下，P1.00 使用 ∼20M 个样本）进行超参数优化，就能够找出表现出色的架构。

此外，此处开展的规模异常庞大 的超参数优化项目，为考察超参数对模型性能的影响提供了机会。图 2c 和 2d 揭示了忠实模拟给定气溶胶活化数据集所需的最低模型复杂度：要实现最佳性能，至少需要两到三层隐藏层以及 10^4–10^5 个可学习参数。有趣的是，实现最高性能所需的最少可学习参数似乎取决于子集大小，例如，p_{subset} 越大，所需参数越少（P0.00025 时为$\sim10^5$，P1.00 时为 $\sim10^4$）。虽然这并非超参数优化的主要目的，但分析超参数与模型性能之间的关系，能够为理解模型架构以及在下一轮迭代中开发更完善的模型提供有用信息；不过，我们承认，这显然取决于具体问题。

步骤2之后所选机器学习模型的预测准确率如图 3a–e 所示：利用留出测试数据集，将真实值（使用云团模型计算的气溶胶活化比例）与预测值（由神经网络模拟器预测）绘制成图。所选模型来自步骤2的全部模型池（共350个模型），包括表现最好的三个模型

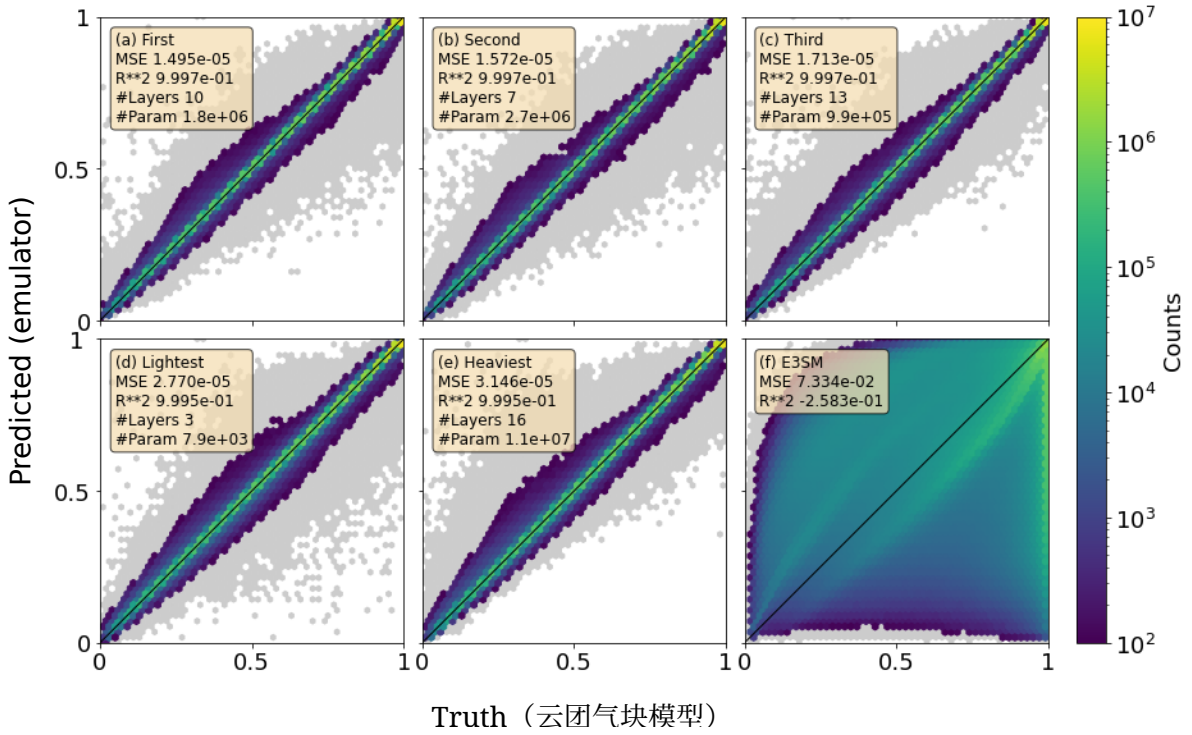


图 3：六边形分箱直方图，展示由云团模型计算（横轴）并由不同神经网络模拟器估算（纵轴）的气溶胶活化分数（仅第一模态）。各子图分别对应：(a) 最佳模拟器，(b) 次佳模拟器，(c) 第三佳模拟器，(d) 最轻量的模拟器，(e) 最重量级的模拟器，以及 (f) E3SM 当前的参数化方案（Abdul-Razzak 和 Ghan，2000年）。该留出测试数据集的样本总数为 7.9×10^7；样本数少于 100（占总样本数的 0.0001%）的分箱以灰色显示。每个子图左上角标出了均方误差（MSE）和决定系数（R**2）、隐藏层数量（#Layers）以及可学习参数总数（#Param）。

以及按可学习参数数量衡量的最轻和最重模型。请注意，均方误差大于 1×10^{-4} 的错误模型不纳入筛选。五种神经网络模型的表现均远胜于 E3SM 当前使用的气溶胶活化参数化方案（图 3f）。然而，尽管模型复杂度差异很大，它们之间的差别几乎难以察觉——例如，五种模型的可学习参数数量相差三个数量级，从 7.9×10^3 到 1.1×10^7。值得注意的是，最轻的模型（图 3d）的表现几乎与最佳模型（图 3a）不相上下，而后者的可学习参数约为前者的 1,400 倍。这项广泛的 超参数优化是找到这一幸运架构的关键。

大规模超参数优化的另一项关键优势，是由此产生的顶尖模型具有多样性。如表S1所示，顶尖模型的模型架构差异显著，为

用户根据自身具体的应用需求提供了多种选择。例如，对于我们的气溶胶活化模拟器，尽管最轻量的模型并非性能最佳，但由于计算效率较高且能够保持性能，因此更适合在气候模型中进行推理（图 S7）。

6 两步超参数优化的参数设置

与任何超参数优化方法一样，我们的两步式超参数优化方法需要预先设定若干参数：p_{subset}（步骤1评估所用训练集的比例）和 $p_{retrain}$（步骤2重新评估所用步骤1试验的比例），以及 n_{trials}（试验次数；这是任何超参数优化方法普遍适用的参数）。在此，我们尝试根据案例研究提供有关如何选择这些参数的见解。

图 2a 显示，模型性能在步骤1后渐近收敛。例如，对于 P0.005–P1.00，秩低于 6000 的模型（“tier-1”）的验证均方误差处于同一数量级。鉴于这一结果，接下来要探究的是，使用完整训练数据集重新训练后，tier-1 模型之间的模型性能差异有多大。如果 tier-1 模型在重新训练后的性能一致，那么在我们的案例研究中，确定步骤2的顶尖候选者（例如 50 个模型）并不需要较大的 n_{trials} 值（例如 10,000）。这一问题源于我们的发现：在每个类别的前 50 个模型中，步骤1的秩无法决定步骤2的秩（图 S8）——这表明，超参数配置同样具有竞争力的模型，其性能主要受训练期间随机性的影响（例如随机权重初始化和小批量效应）。

为了解决这一问题，我们将秩组不同的步骤1模型分组（例如，秩 1–50、1001–1050、2001–2050、3001–3050 和 4001–4050），然后使用完整训练数据集重新训练这些模型（步骤2）。接着，我们绘制每个步骤1秩组对应的步骤2验证均方误差箱线图（见图 2e 和图 S6）。在箱线图中，属于不同秩组的模型，其步骤2均方误差差异很小；但 P0.0005 和 P0.00025 除外，因为步骤1模型性能未表现出明确的收敛趋势（图 2a）。这一结果表明，在步骤1中使用规模小得多的 n_{trials}，也足以筛选出步骤2的前 50 名候选者。例如，由于我们需要渐近收敛区域中的 50 名顶尖候选者，根据对现有试验进行的随机抽样（图 S9），对于 P0.005–P1.00，使用n_{trials}=100，对于 P0.0005–P0.00025，使用 n_{trials}=200，便已足够。不过，为了充分发挥这种两步超参数优化方法的计算效率，使用更多的 n_{trials} 仍有益处，尤其是在 p_{subset} 值较小时。在我们的案例研究中，要确定超参数与模型性能之间的关系，大约需要 1,000 n_{trials}（图 S10 和图 S11）。

这项回顾性分析指出，模型性能与排名图（类似于图 2a）可能有助于判断两步超参数方法中的三个参数是否设置得当。有效的参数配置应使性能–排名图呈现渐近收敛；同时，n_{trials} 应足够大，以确保步骤2的顶尖候选者数量（$p_{retrain} \cdot n_{trials}$）落在渐近收敛区域内。然而，要为两步超参数优化参数的选择提供更全面的指导，仍需进一步研究并开展更多案例研究。

7 总结

我们分享了经验：两步式方法是高效优化大规模架构搜索中超参数的一种有效途径。第一步，使用训练数据集的一个小子集进行超参数优化，以确定一组超参数组合，即可行架构。第二步，使用完整训练数据集重新训练表现最佳的配置，以最终确定模型并重新校准模型权重。我们针对气溶胶活化神经网络模拟器优化开展的案例研究表明，仅使用训练数据集的 0.025% 进行两步超参数优化，就能有效找到最优超参数配置。据我们所知，我们的案例研究是首次将计算机科学文献（例如 Feldman（2020））中广受关注的核心集方法，深入应用于实际气候建模参数化研究问题；研究结果证实，核心集方法适用于数据量范围广泛的问题（例如，我们的案例研究中样本量从 5K 到 20M）。此外，我们还展示了大规模超参数优化（即在较大的超参数搜索域内进行大量试验）的其他优势。例如，大规模超参数优化有助于了解超参数与模型性能之间的关系。此外，在混合机器学习—物理气候建模中，需要权衡拟合精度和推理效率；在若干表现相近的选项中，它还提供了选择合适的超参数配置的机会。

我们希望，这种两步式超参数优化方法以及展示其在 GPU 集群上实际应用的附录，能够通过减少所需时间和计算资源，为渴望享受大规模超参数优化优势的机器学习从业者提供切实可行的解决方案。此外，该方法还可以轻松地根据具体需求进行调整。例如，可以修改第二步中顶尖候选者的筛选方式（如若希望获得适合提升推理效率的紧凑架构，只选择具有两层或三层隐藏层的候选者），也可以将第二步改为搜索空间更集中的另一轮超参数优化。

我们注意到，两步超参数优化方法目前仅通过随机搜索算法进行了验证，其与其他搜索算法的适用性仍有待探索。尽管存在这一局限，

随机搜索所采用的两步方法所带来的潜在成本节约，可能超过使用自适应算法的优势。随机搜索具有良好的并行性，并且可以在多个 GPU 节点上进行缩放。确定两步超参数优化方法所需参数（n_{trials}、 p_{subset} 和 $p_{retrain}$）的最优值同样是一个有待进一步研究的开放问题。在我们的案例研究中， n_{trials}=10000、 p_{subset}= 0.00025 和 $p_{retrain}$=0.005 均表现有效，但最优值可能会因训练数据集的复杂程度和超参数搜索空间的广度而有所不同。

我们希望其他人在其他场景中应用类似技术时，也能分享经验教训。机器学习参数化是一门经验艺术，密集经验采样还有许多值得学习之处。与此同时，对该方法的独立测试已在此处讨论的气溶胶成核应用之外取得了颇具竞争力的结果，具体是在一个新的基准项目中，该项目聚焦于对流的机器学习参数化和辐射参数化（Yu 等，2023年）。

致谢

我们感谢 Elizabeth A. Barnes 和两位匿名审稿人提供宝贵意见，显著改进了我们的稿件；也感谢 UCI 的同事 Savannah Ferretti、Jerry Lin 和 Yan Xia 提供有益的评论。本研究得到“实现全球允许对流尺度的气溶胶-云相互作用”（EAGLES）项目（项目编号：74358）的支持。该项目由美国能源部（DOE）科学办公室生物与环境研究办公室的地球系统模型开发（ESMD）项目领域资助。本研究使用了国家能源研究科学计算中心（NERSC）的资源。该中心是美国 DOE 科学办公室的用户设施，位于劳伦斯伯克利国家实验室，由该实验室根据合同编号 DE-AC02-05CH11231 运营；本研究使用了 NERSC 资助额度 ALCC-ERCAP0016315、BER-ERCAP0015329、BER-ERCAP0018473 和 BER-ERCAP0020990。本研究还使用了阿贡领导计算设施的资源。该设施是 DOE 科学办公室的用户设施，根据合同 DE-AC02-06CH11357 获得支持；本研究使用了由高级科学计算研究（ASCR）领导计算挑战赛（ALCC）计划提供的计算时间额度。太平洋西北国家实验室由巴特尔纪念研究所根据合同 DE-AC05-76RL01830 为美国 DOE 运营。此外，本研究使用了 XSEDE 的 PSC Bridges-2 系统（美国国家科学基金会资助项目 ACI-1548562 和 ACI-1928147 ）。我们感谢 David Walling 通过 XSEDE 扩展协作支持服务计划协助提供高性能计算支持。

数据可用性声明

本研究使用的气溶胶活化数据集可通过 Zenodo 数据存储库公开获取（https://doi.org/10.5281/zenodo.7627577）。本研究使用的两步超参数优化代码可通过 GitHub 存储库公开获取（https://github.com/ sungdukyu/Two-step-HPO），相关说明见补充材料。

参考文献

Abdul-Razzak, H., and S. J. Ghan, 2000年: 气溶胶活化参数化：2. 多种气溶胶类型。J Geophys Res Atmospheres, 105, 6837–6844, https://doi.org/10.1029/1999jd901161.

Alfonso, L., and J. M.Zamora, 2021年: 云滴自动转化过程的双矩机器学习参数化方案。Atmos Res, 249, 105269, https://doi.org/10.1016/j.atmosres.2020.105269.

Bergstra, J., and Y. Bengio, 2012: 用于超参数优化的随机搜索。Journal of Machine Learning Research, 13, 281–305.

Chiu, J. C., C. K. Yang, P. J. van Leeuwen, G. Feingold, R. Wood, Y. Blanchard, F. Mei, 和 J. Wang，2021年：利用机器学习和优化技术对暖云微物理过程进行观测约束。Geophys Res Lett，48，e2020GL091236，https://doi.org/10.1029/2020gl091236。

Feldman, D.，2020年：核心集：最新综述。arXiv，10，https://doi.org/10.48550/arxiv.2011.09384。Gettelman, A., D. J. Gagne, C. -C. Chen, M. W. Christensen, Z. J. Lebo, H. Morrison, 和 G. Gantos，2021年：机器学习暖雨过程。J Adv Model Earth Sy，13，https://doi.org/10.1029/2020ms002268。

Ghan, S. J., 等，2011年：液滴成核：基于物理的参数化方案及比较评估。J Adv Model Earth Sy, 3，https://doi.org/10.1029/2011ms000074。

Golaz, J., 等，2019年：美国能源部 E3SM 耦合模式第 1 版：标准分辨率下的概述与评估。J Adv Model Earth Sy, 11，2089–2129，https://doi.org/10.1029/2018ms001603。

Killamsetty, K.、D. Sivasubramanian、G. Ramakrishnan、A. De 和 R. Iyer, 2021a：GRAD-MATCH：基于梯度匹配的数据子集选择，用于高效的深度模型训练。arXiv，https://doi.org/10.48550/arxiv.2103.00123。

——、——、—— 和 R. Iyer，2021b：GLISTER：基于泛化的数据子集选择，用于高效稳健学习。AAAI 人工智能会议论文集，35，8110–8118，https://doi.org/10.1609/aaai.v35i9.16988。

——, G. S. Abhishek, Aakriti, A. V. Evfimievski, L. Popa, G. Ramakrishnan 和 R. Iyer，2022：AUTOMATA：用于计算高效超参数调优的基于梯度的数据子集选择。arXiv，https://doi.org/10.48550/arxiv.2203.08212。Kingma, D. P. 和 J. Ba，2014：Adam：一种随机优化方法。Arxiv，https://doi.org/10.48550/arxiv.1412.6980。

Li, L., K. Jamieson, G. DeSalvo, A. Rostamizadeh 和 A. Talwalkar，2016：Hyperband：一种用于超参数优化的新型基于多臂老虎机的方法。Arxiv，https://arxiv.org/abs/1603.06560。

Liu, X., P.-L. Ma, H. Wang, S. Tilmes, B. Singh, R. C. Easter, S. J. Ghan 和 P. J. Rasch，2016：社区大气模式 5.3版中模态气溶胶模块（MAM4）四模态版本的描述与评估。《地球系统建模进展》，9，505–522，https://doi.org/10.5194/gmd-9-505-2016。

Mirzasoleiman, B., J. Bilmes 和 J. Leskovec，2020年：用于机器学习模型数据高效训练的核心集。arXiv，https://doi.org/10.48550/arxiv.1906.01827。

Rasch, P. J. 和合著者，2019：能源百亿亿次级地球系统模型大气分量概述。《地球系统建模进展》，11，2377–2411，https://doi.org/10.1029/2019ms001629。

Silva, S. J., P.-L. Ma, J. C. Hardin, 和 D. Rothenberg，2021年：气溶胶活化的物理正则化机器学习模拟器。Geosci Model Dev，14，3067–3077，https://doi.org/10.5194/gmd-14-3067-2021。

Snoek, J., H. Larochelle, 和 R. P. Adams，2012：机器学习算法的实用贝叶斯优化。arXiv，https://doi.org/10.48550/arxiv.1206.2944。

Towns, J. 和合著者，2014：XSEDE：加速科学发现。Comput Sci Eng，16，62–74，https://doi.org/10.1109/mcse.2014.80。

S. Visalpara、K. Killamsetty 和 R. Iyer，2021年：用于高效超参数调优和自动机器学习的数据子集选择框架。国际机器学习大会上的 SubSetML 2021研讨会。

Wang, H. 和合作者，2020年：E3SM 第 1 版中的气溶胶：新进展及其对辐射强迫的影响。《地球系统建模进展》，12，https://doi.org/10.1029/2019ms001851。

Wendt, A.、M. Wuschnig 和 M. Lechner，2020年：通过两阶段搜索加速常见的超参数优化方法。2020年IEEE工业电子学会第46届年会，00，517–522，https://doi.org/10.1109/iecon43393.2020.9254801。

Yu, S. 等，2023年：ClimSim：用于在混合多尺度气候模拟器中训练高分辨率物理模拟器的开放式大规模数据集。arXiv，https://doi.org/10.48550/arxiv.2306.08754。

补充材料

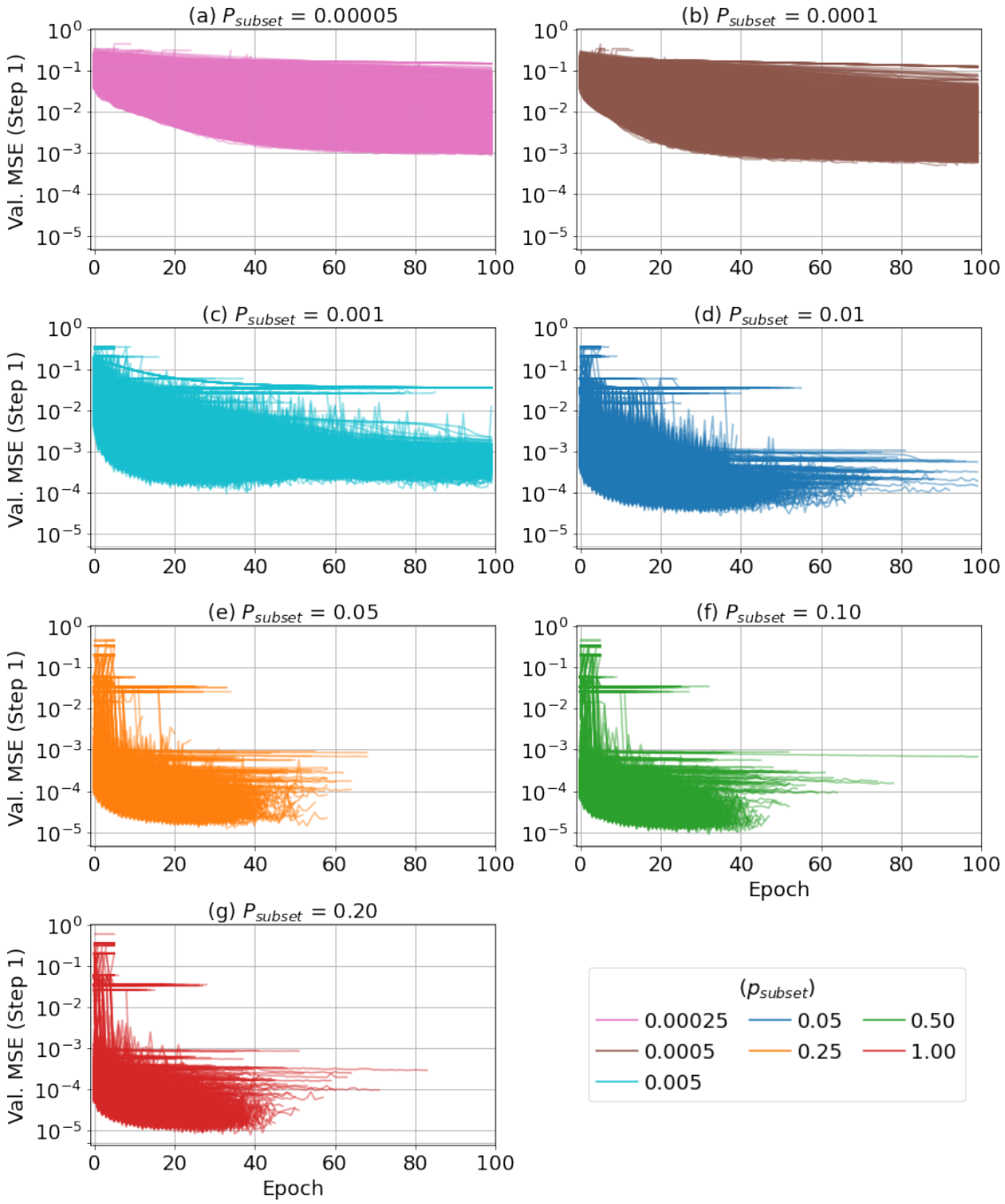


图 S1：步骤1中各训练轮次的验证均方误差。每条线代表一次独立的调参试验。请注意，采用了耐心值为 5 个训练轮次的提前停止规则，而训练轮次的最大数量为 100。

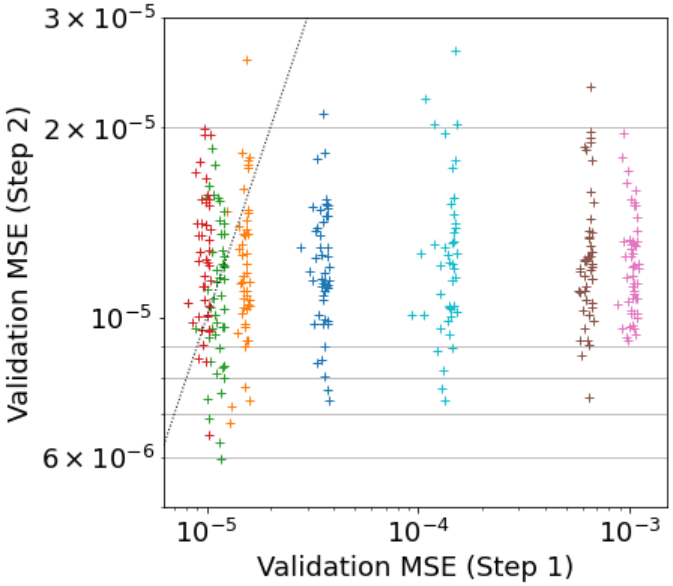


图 S2：除纵轴视图范围外，其余与图 2b 相同。

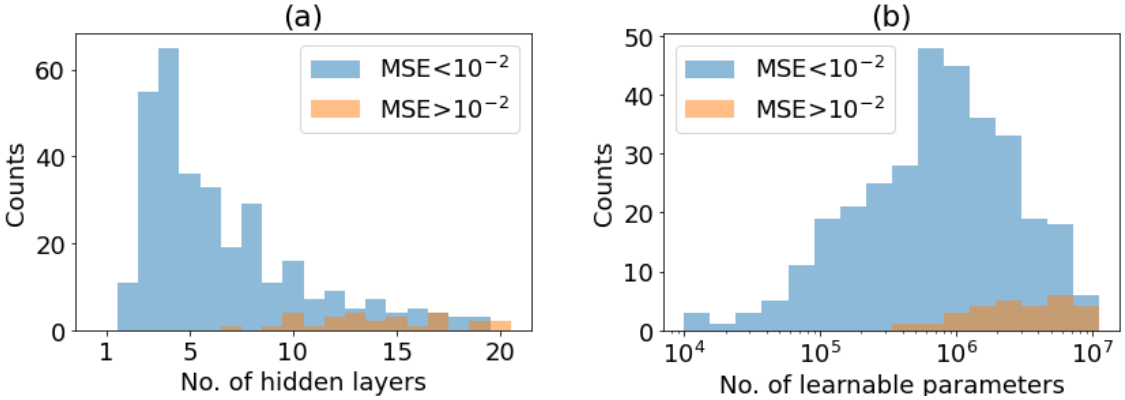


图 S3：步骤2重新训练后，350 个试验模型（来自七个超参数优化项目的排名前 50 的模型）的 (a) 隐藏层数量和 (b) 可学习参数数量的直方图。蓝色（橙色）柱表示测试均方误差低于（高于） 1×10^{-2} 的模型。

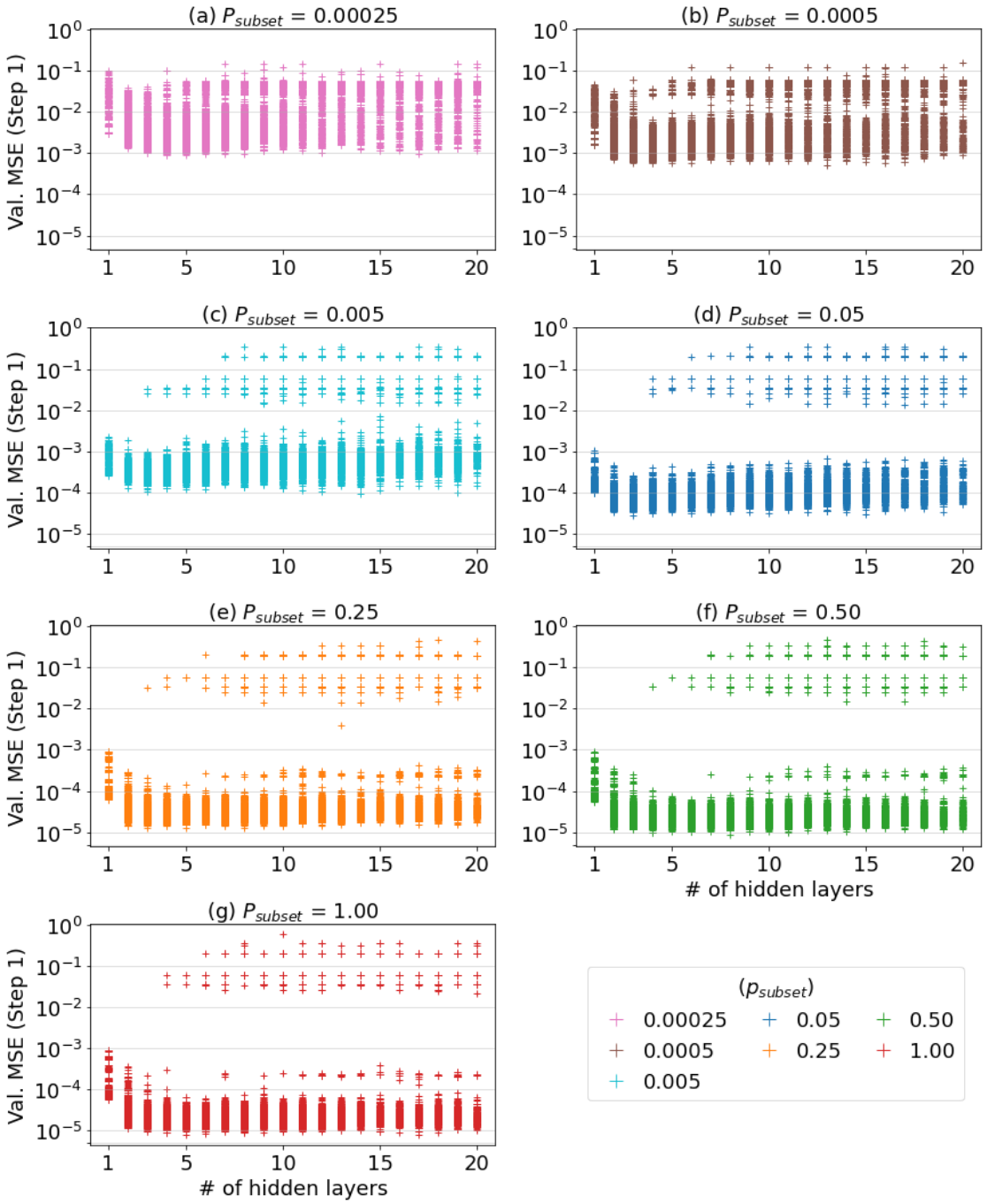


图 S4：与图 2c 相同，不同之处在于每个超参数优化项目都绘制在单独的子面板中。

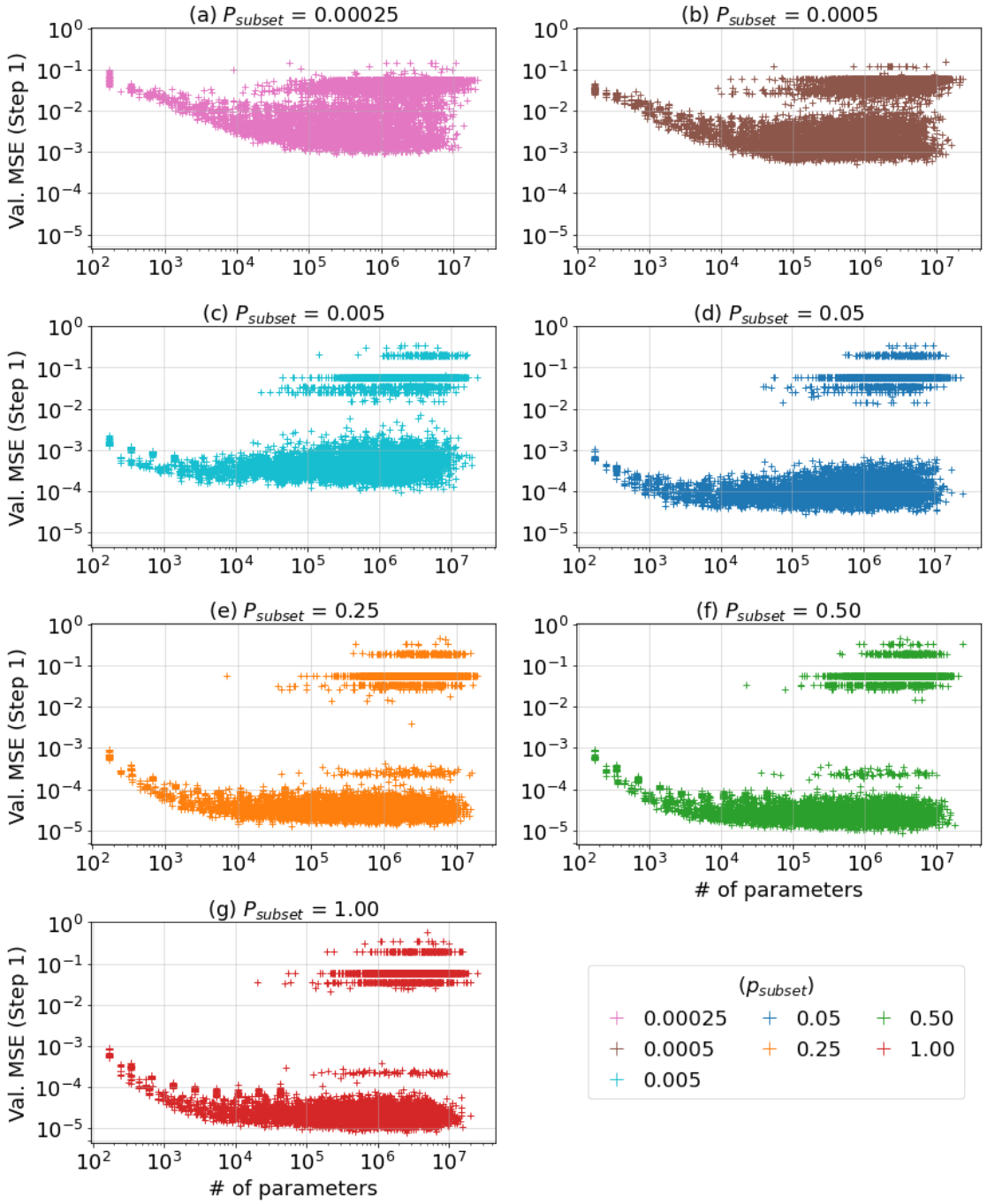


图 S5：与图 2d 相同，不同之处在于每个超参数优化项目均绘制在单独的子面板中。

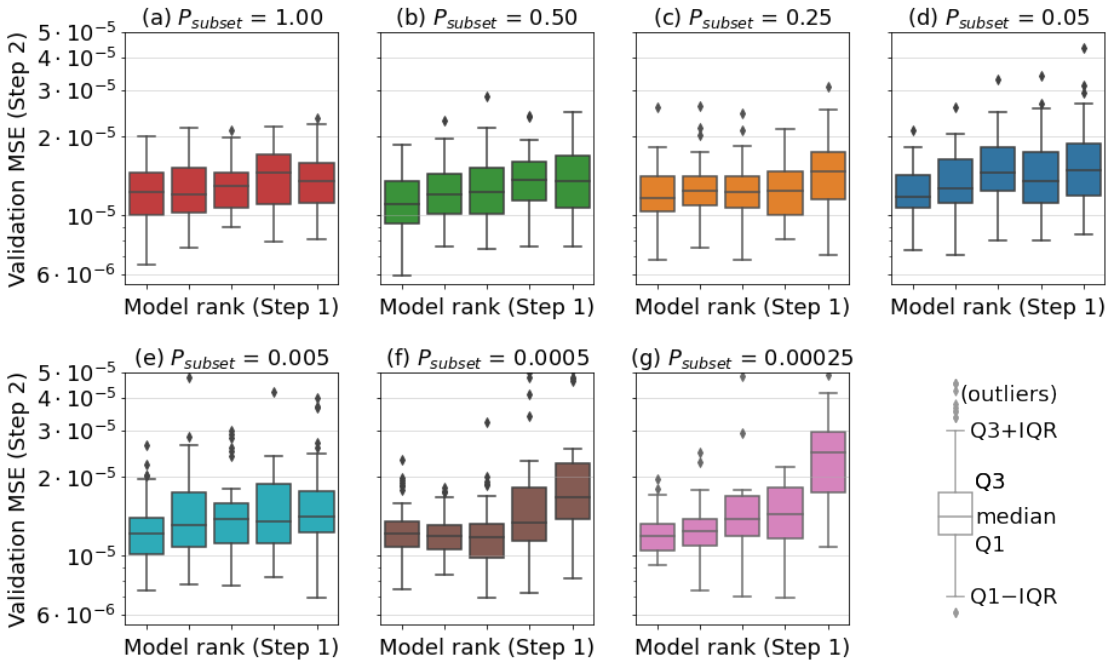


图 S6：与 图 2e 相同，但每个超参数优化项目分别绘制在独立子面板中 l.

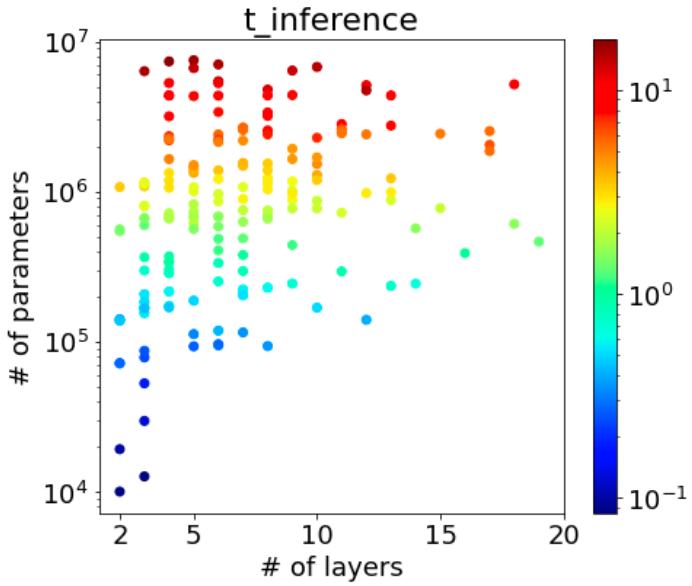


图 S7：批大小为 1,024 时 100,000 个样本的推理时间。重复推理 10 次，然后对推理时间取平均值。使用一个 CPU 核心（AMD EPYC 7742）。单位：秒。

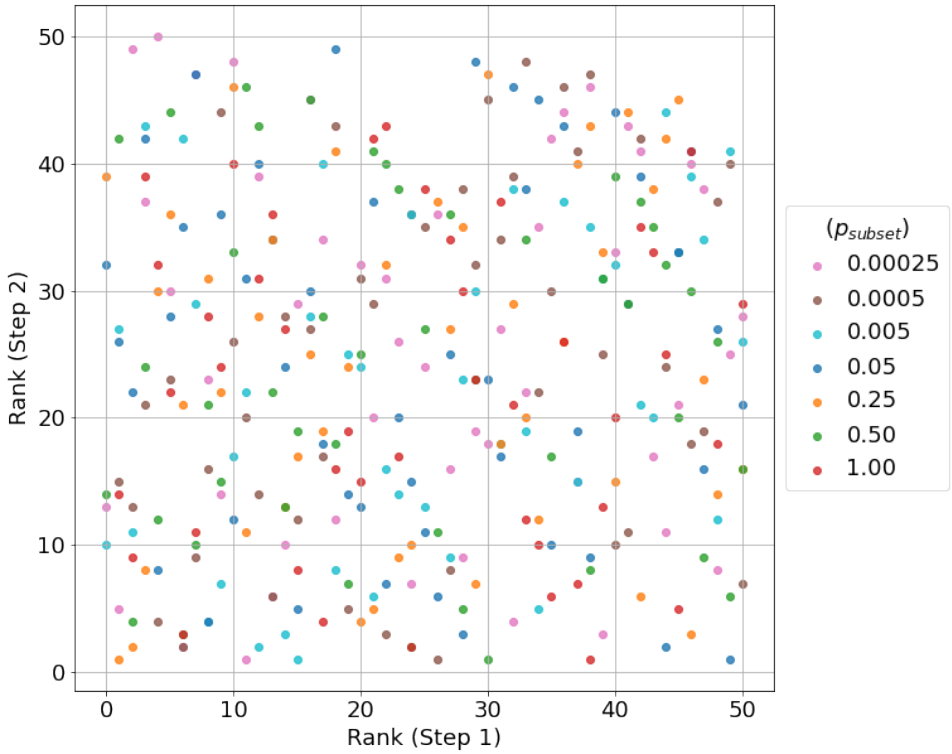


图 S8：每个超参数优化项目中排名前 50 的模型的第一步秩（横轴）与第二步秩（纵轴）之间的散点图。

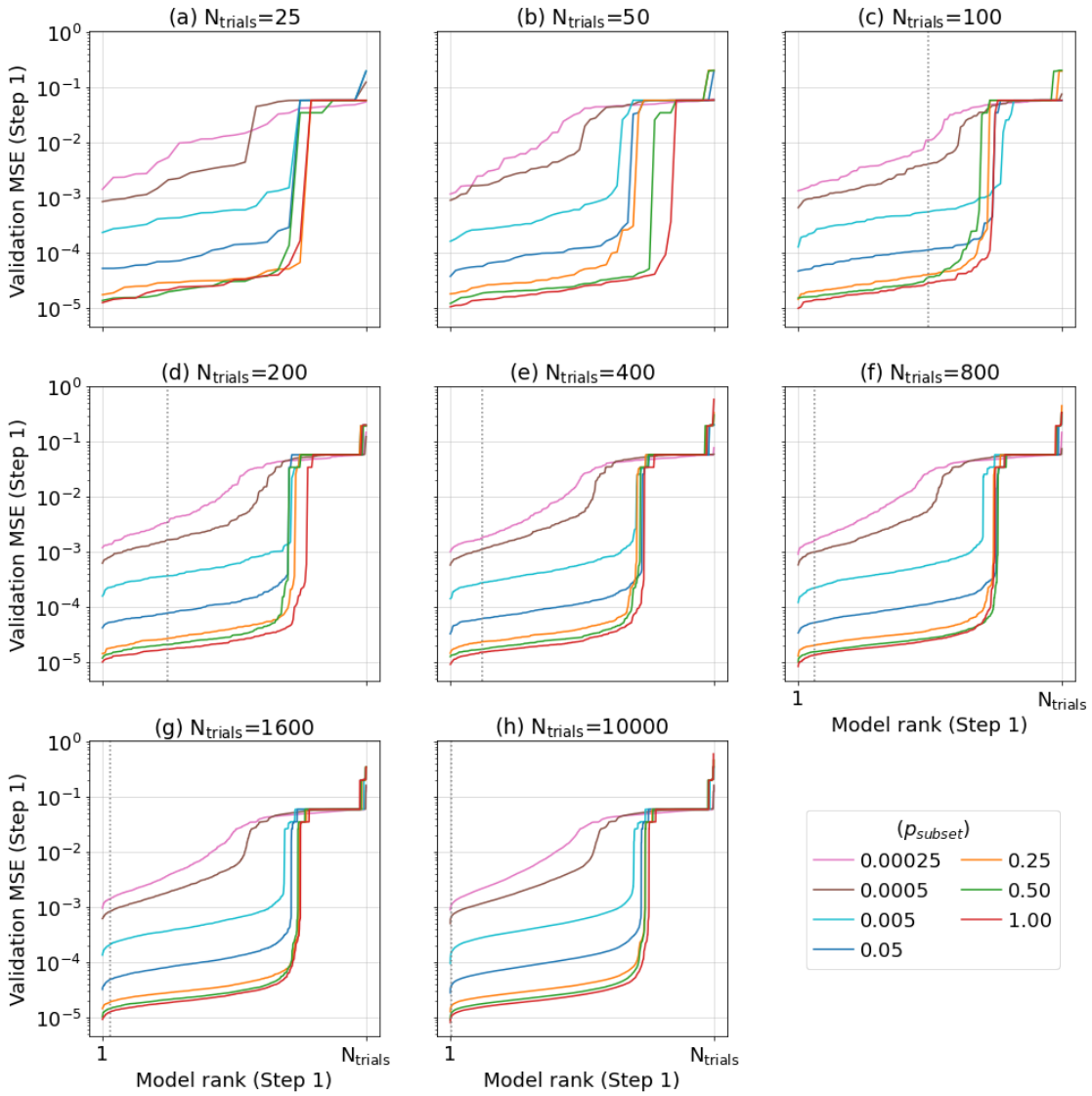
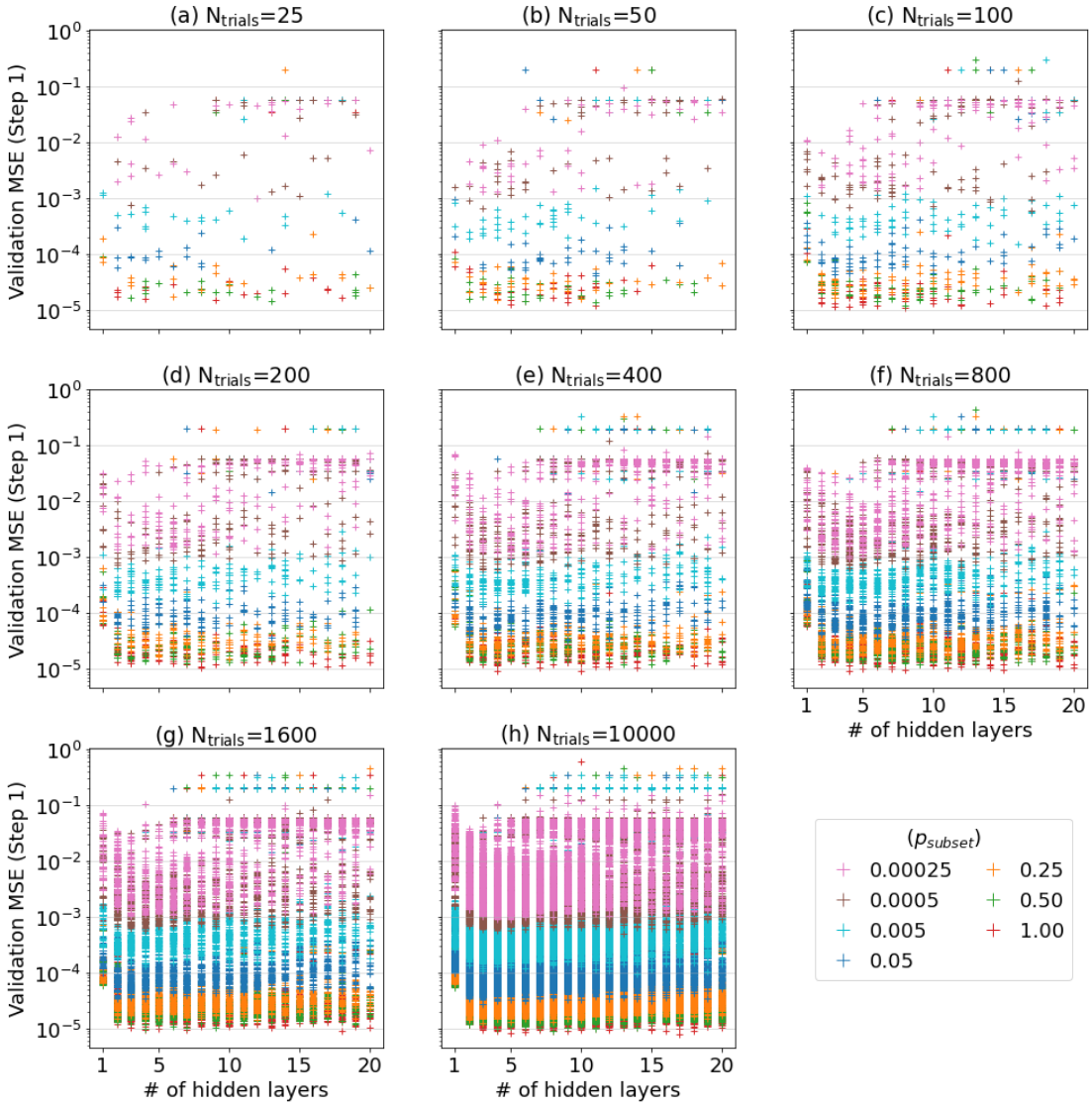


图 S9：与图 2a 相同，但绘制时采用了随机抽取且不放回的试验。各面板使用的试验数分别为 25（a）、50（b）、100（c）、200（d）、400（e）、800（f）、1600（g）和 10000（h）。面板（h）与图 2a 完全相同。虚线表示 x 值为 50。



图S10：与图 2c 相同，但绘制时使用的是随机抽取且不放回的试验。各面板使用的试验数量分别为：25（a）、50（b）、100（c）、200（d）、400（e）、800（f）、1600（g）和 10000（h）。面板（h）与图 2c 相同。

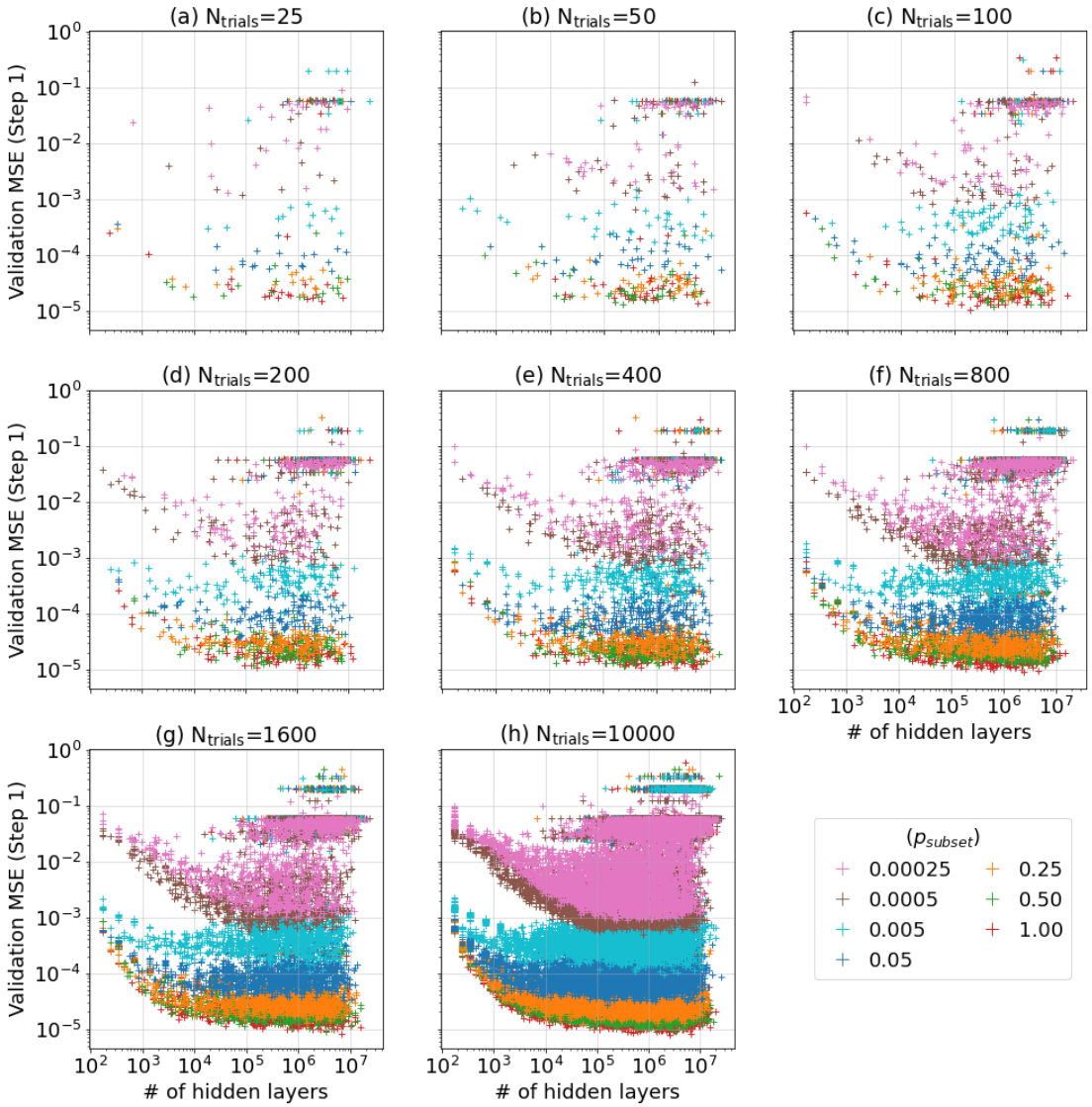


图 S11：与图 2d 相同，但本图是在不放回地随机抽取试验后绘制的。各面板中使用的试验次数分别为 25 次（a）、50 次（b）、100 次（c）、200 次（d）、400 次（e）、800 次（f）、1600 次（g）和 10000 次（h）。面板（h）与图 2d 完全相同。

Rank (步骤2)	投影	Rank (步骤1)	MSE (步骤2)	#	# 参	隐藏层节点数
1	0.5	30	5.937E-06	10	1.777E+06	1024, 512, 256, 1024, 64, 256, 512, 64, 1024, 512
2	0.5	24	6.307E-06	7	2.660E+06	1024、2048、256、64、16、8、128
3	1	38	6.499E-06	13	9.887E+05	64, 32, 512, 512, 32, 256, 1024, 128, 1024, 16, 16, 256, 512
4	0.25	1	6.798E-06	8	4.775E+06	1024, 2048, 1024, 512, 32, 8, 64, 256
5	0.5	6	6.887E-06	9	9.712E+05	512, 512, 1024, 8, 32, 8, 32, 1024, 128
6	0.25	2	7.218E-06	5	1.462E+06	1024, 128, 1024, 1024, 128
7	0.25	46	7.387E-06	7	2.656E+06	512、256、1024、2048、8、128、1024
8	0.05	49	7.393E-06	11	2.443E+06	1024、2048、128、64、16、64、32、16、256、16、2048
9	0.005	15	7.393E-06	10	2.683E+06	2048, 1024, 512, 32, 64, 16, 16, 8, 16, 256
10	0.5	2	7.406E-06	11	2.817E+06	2048, 1024, 8, 512, 8, 32, 8, 512, 1024, 64, 1024
11	0.0005	26	7.440E-06	12	9.145E+05	2048, 64, 1024, 64, 32, 128, 64, 128, 256, 2048, 16, 64
12	0.5	28	7.574E-06	6	2.139E+06	256, 1024, 512, 2048, 128, 256
13	0.05	44	7.656E-06	6	1.384E+06	2048, 32, 1024, 64, 2048, 512
14	0.005	12	7.716E-06	10	8.498E+05	1024, 512, 512, 16, 512, 16, 512, 16, 32, 64
15	0.25	20	7.760E-06	6	1.183E+05	256, 256, 16, 32, 16, 2048
16	0.5	49	8.025E-06	9	1.169E+06	256、2048、32、16、8、32、128、256、2048
17	0.05	28	8.052E-06	6	9.641E+05	256, 2048, 16, 256, 512, 512
18	0.5	19	8.139E-06	8	2.289E+05	256, 256, 32, 32, 1024, 64, 256, 128
19	0.005	14	8.227E-06	12	5.111E+06	2048, 2048, 32, 256, 16, 8, 256, 2048, 128, 64, 32, 16
20	0.5	38	8.338E-06	5	7.333E+05	128, 1024, 512, 128, 64
21	0.5	47	8.352E-06	9	1.646E+06	2048, 64, 2048, 512, 16, 64, 64, 256, 1024
22	0.05	8	8.481E-06	9	8.778E+05	512、1024、256、8、8、1024、64、32、64
23	0.5	7	8.489E-06	14	5.690E+05	1024, 64, 16, 1024, 64, 1024, 8, 64, 8, 2048, 128, 16, 16, 2048
24	1	24	8.501E-06	7	8.911E+05	128, 512, 512, 1024, 8, 8, 2048
25	0.05	15	8.549E-06	19	1.052E+07	1024, 2048, 512, 256, 2048, 1024, 32, 512, 256, 2048, 128, 2048, 64, 128, 8, 32, 2048, 1024, 1024
26	1	6	8.607E-06	8	1.386E+06	1024, 256, 128, 2048, 128, 32, 256, 2048
27	0.0005	6	8.683E-06	6	7.118E+06	2048, 2048, 1024, 256, 1024, 256
28	0.005	8	8.848E-06	5	6.305E+05	2048, 256, 256, 8, 256
29	0.005	34	8.926E-06	8	5.743E+06	2048, 2048, 512, 64, 512, 128, 512, 512
30	0.5	26	8.955E-06	4	7.164E+05	1024, 512, 64, 2048.

表 S1：基于测试均方误差，步骤2超参数优化后的前30个模型。第三列中的验证排名，是每个超参数优化项目内根据第一步超参数优化所得最小验证均方误差计算的排名。

文本 S1。在由 SLURM 管理的高性能计算（HPC）集群中设置 Keras Tuner 分布式模式的代码示例。

Keras Tuner 提供了一个高级接口，只需设置四个环境变量即可启用分布式搜索模式。其中包括三个 KerasTuner 专用变量（1–3）和一个与 CUDA 相关的变量（4）：

1. KERASTUNER_TUNER_ID：分配给管理进程和工作进程的唯一 ID，其中管理进程使用“chief”。2. KERASTUNER_ORACLE_IP：管理进程（chief）的 IP 地址或主机名。3. KERASTUNER_ORACLE_PORT：管理进程（chief）的端口号。4. CUDA VISIBLE DEVICES：本地 GPU 设备 ID，用于控制分配给单个进程的 GPU 数量及其选择。

这四个环境变量会根据高性能计算环境中计算资源的可用情况动态分配。因此，编写能够自动设置上述四个环境变量正确值的脚本，在技术上可能颇具挑战。为此，我们提供了基于案例研究中所用脚本的代码示例。

我们的案例研究在 PSC Bridges-2 (https://www.psc.edu/resources/bridges-2/) 上进行，该系统的 GPU 分区中每个节点配有 8 个 GPU。我们使用了两个节点（共 16 个 GPU），并为每个工作进程分配一个 GPU。为了自动设置上述四个环境变量，我们编写了三个脚本：

- 脚本 1. <sbatch-keras-tuner.sh>：SLRUM 作业提交脚本，用于通过 srun 启动并行作业（在我们的案例中为“run-dynamic.sh”）。需要根据用户的需求设置以下选项：

- partition：GPU 分区的名称
- nodes：GPU 节点的数量
- gpus：GPU 进程总数，即 nodes x GPUs/node
- ntasks：任务数，即 gpus +1。额外的一个进程用于管理器进程。

- 脚本 2。 <run-dynamic.sh>：用于连接 SLURM 脚本（‘sbatch-kears- tuner.sh’）和使用 Keras Tuner 的 Python 脚本（‘keras-tuner-dynamic.py’）的中间脚本。它会与每个作业步骤并行启动，每个步骤都有自己的 SLURM 生成的环境变量，例如 SLURM LOCALID 和 SLURMD NODENAME。_ _

- 脚本 3。 <keras-tuner-dyanamic.py>： 使用 Keras Tuner 进行超参数调优的 Python 脚本，其环境变量 (1-4) 会根据 SLURM 变量自动设置：

  • num_gpus_per_node（第 1 行）： 每个节点上的 GPU 数量。

脚本 1. sbatch-keras-tuner.sh

```
#!/bin/sh

## 以下选项应由用户提供 #SBATCH --job-name=#SBATCH --
output=#SBATCH --account=#SBATCH --time=


## 以下选项也应由用户提供。## 不过，我们提供了一个在 PSC Bridges-2 上使用 2 个节
点的示例

#SBATCH --partition=GPU # Bridges-2 的 GPU 分区名称 #SBATCH --nodes=2  # 节点数量 #
SBATCH --gpus=16  # 图形处理器总数：# 即  <节点数量>  乘以  <每个节点的 GPU 数量>#
SBATCH --ntasks=17  # 任务总数：# 应为 GPU 总数加一，# 即  <节点数量>  乘以  <每个节点的
GPU 数量>  +  1


srun --等待=0bash run-dynamic.sh
```

脚本 2. run-dynamic.sh

```
#!/bin/sh

# 以下三行用于生成日志文件，可选。# SLURM_LOCALID 和 SLURMD_NODENAME 是由 SLURM 自动设置的
环境变量。echo "--- run-dynamic.sh ---"echo SLURM_LOCALID $SLURM_LOCALIDecho SLURMD_NODENAME $
SLURMD_NODENAME


# 运行 keras-tuner 脚本。# 标准输出和错误输出会写入单独的日志文件。python
keras-tuner-dynamic.py  > log-$SLURM_JOBID-$SLURMD_NODENAME-$SLURM_LOCALID.log  2>&1
```

脚本 3. keras-tuner-dyanamic.py

```
def set_environment(num_gpus_per_node="8"): #’num_gpus_per_node’ 应根据高性能计算规格进行设置。
import os 节点名  =os.environ[’SLURMD_NODENAME’]进程 ID  =  os.environ[’SLURM_LOCALID’]print(
节点名) print(进程 ID) 流  =  os.popen(’scontrol show hostname $SLURM_NODELIST’) 输出  =  流.read()
Oracle  =  输出.split("\n")[0]
```

打印(Oracle)
如果进程 ID==数量_GPU_每个_节点：os.environ["KERASTUNER_
调优器_ID"] = "主节点"os.environ["CUDA_可见_设备"] = "0"

否则：
操作系统.environ["KERASTUNER_TUNER_ID"] = "tuner-" + str(nodename) + "-" + str(procid) 操作系统.environ["CUDA_VISIBLE_DEVICES"] = procid

操作系统.environ["KERASTUNER_ORACLE_IP"] = oracle + ".ib.bridges2.psc.edu" # 使用完整主机名 操作系统.environ["KERASTUNER_ORACLE_PORT"] = "8000" 打印("KERASTUNER_TUNER_ID: %s"%操作系统.environ["KERASTUNER_TUNER_ID"]) 打印("KERASTUNER_ORACLE_IP: %s"%操作系统.environ["KERASTUNER_ORACLE_IP"]) 打印("KERASTUNER_ORACLE_PORT: %s"%操作系统.environ["KERASTUNER_ORACLE_PORT"])

定义 主函数():
在此编写用户的调优代码，例如，# - 导入必要的程序包，包括 Keras Tuner # - 读取数据集 # - 根据需要进行归一化/缩放 # - 定义“超模型”（在此指定搜索空间）# - 定义“调优器”（在此指定搜索算法）# - 执行超参数搜索

如果 __名称__ == '__主函数__': 设置_环境() 主函数()
