

Integration of nested cross-validation, automated hyperparameter optimization, high-performance computing to reduce and quantify the variance of test performance estimation of deep learning models

Paul Calle¹, Averi Bates¹, Justin Reynolds¹, Yunlong Liu¹, Haoyang Cui¹, Sinaro Ly¹, Chen Wang², Qinghao Zhang², Alberto J. de Armendi³, Shashank S. Shettar³, Kar-Ming Fung^{4,5}, Qinggong Tang^{2,5*}, Chongle Pan^{1,2*}

¹School of Computer Science, University of Oklahoma, Norman, OK

²Stephenson School of Biomedical Engineering, University of Oklahoma, Norman, OK

³Department of Anesthesiology, University of Oklahoma Health Sciences Center, Oklahoma City, OK, 73104, USA.

⁴Department of Pathology, University of Oklahoma Health Sciences Center, Oklahoma City, OK 73104, USA.

⁵Stephenson Cancer Center, University of Oklahoma Health Sciences Center, Oklahoma City, OK 73104, USA.

*Correspondence: please contact cpan@ou.edu for questions on deep learning and contact qtang@ou.edu for questions on medical imaging.

Abstract

The variability and biases in the real-world performance benchmarking of deep learning models for medical imaging compromise their trustworthiness for real-world deployment. The common approach of holding out a single fixed test set fails to quantify the variance in the estimation of test performance metrics. This study introduces NACHOS (Nested and Automated Cross-validation and Hyperparameter Optimization using Supercomputing) to reduce and quantify the variance of test performance metrics of deep learning models. NACHOS integrates Nested Cross-Validation (NCV) and Automated Hyperparameter Optimization (AHPO) within a parallelized high-performance computing (HPC) framework. NACHOS was demonstrated on a chest X-ray repository and an Optical Coherence Tomography (OCT) dataset under multiple data partitioning schemes. Beyond performance estimation, DACHOS (Deployment with Automated Cross-validation and Hyperparameter Optimization using Supercomputing) is

introduced to leverage AHPO and cross-validation to build the final model on the full dataset, improving expected deployment performance. The findings underscore the importance of NCV in quantifying and reducing estimation variance, AHPO in optimizing hyperparameters consistently across test folds, and HPC in ensuring computational feasibility. By integrating these methodologies, NACHOS and DACHOS provide a scalable, reproducible, and trustworthy framework for DL model evaluation and deployment in medical imaging.

1. Introduction

Deep learning (DL) has matched or surpassed human experts in performance across many medical applications ¹⁻³. However, the deployment of DL models to automate real-world diagnosis and medical procedures remains limited ⁴. A key concern is the unknown magnitude of the variances and biases in the estimation of the model performance in research studies, which undermines the trustworthiness of deep learning model ⁵⁻⁸. In a typical deep learning study, a small fraction (i.e. 10%-20%) of the labeled data is held out in the test set for benchmarking the performance of the final model in unseen data, while the majority of the labeled data is used in the training and validation sets for model development. In particular, the variance of the test performance metrics is typically unknown (because there is only one test set) and large (because only a small fraction of the labeled data is allocated to the test set). To make deep learning models trustworthy for medical decision-making, it is essential to estimate their performance metrics with low variance using more test data and measure the variance of the obtained estimates ^{9,10}.

Nested cross-validation (NCV) is an effective procedure to meet this requirement. Briefly, the entire dataset is partitioned into k folds that are rotated through a cross-testing loop. A model development procedure with a $(k-1)$ -fold cross-validation loop is nested within the cross-testing loop. The output of NCV is k estimates of the test performance metrics of k models. The average and variance of these k estimates reflect the expected performance and variability of this model development procedure across the entire dataset.

NCV has been used in a few medical machine learning studies. Nawabi et al. ¹¹ employed NCV to benchmark the performance of a random forest classifier for prediction of survival for acute intracerebral hemorrhage using extracted radiomic features obtained from non-enhanced computed tomography images. Their random forest classifier achieved an average test accuracy of 72% with a 95% confidence interval between 70% and 74%. We utilized NCV to benchmark the performance of convolutional neural networks (CNNs) for analysis of Optical Coherence Tomography (OCT) images in multiple endoscopic applications ¹²⁻¹⁵. For example, the average test

classification accuracy of CNN was found to be 82.6% with 3.0% standard error for detecting three different renal tissues from their OCT images. However, NCV is still under-utilized in the medical field. Roberts et al.¹⁶ conducted an analysis of COVID-19 research papers published between January and October 2020 and found a notable lack of NCV utilization, highlighting a gap in methodological rigor in the field.

A challenge of using NCV in a study is the need to implement automated hyperparameter optimization (AHPO) between the cross-testing loop and the cross-validation loop. Most medical deep learning studies perform manual hyperparameter optimization (MHPO). Practitioners can manually evaluate various model architectures, learning rates, regularization methods, and other hyperparameters based on cross-validation performance and select the configurations with the best validation performance to build the final model. However, it is impractical to perform MHPO independently and consistently in every test fold of NCV. Instead, AHPO needs to be performed during each testing iteration of the k -fold cross-testing loop in NCV to automatically identify the model configuration with the best cross-validation performance. AHPO within NCV provides reproducible model optimization and prevents inadvertent information leakage from the test set to the validation set during MHPO.

A second challenge of using NCV with AHPO is the need for significantly more computing than cross-validation with MHPO. Fortunately, the computation in NCV and AHPO can be readily partitioned by data folds for the cross-testing loop or the cross-validation loop and by model configurations for the AHPO loop. The folds can be distributed across many GPUs to compute in parallel. Thus, high-performance computing (HPC) can be used to complete NCV and AHPO within a reasonable amount of wall-clock time.

While many deep learning pipelines, including NiftyNet¹⁷, TorchIO¹⁸, DeepNeuro¹⁹, and GaNDLF²⁰, have been developed for medical imaging, they have not integrated NCV, AHPO, and HPC. To address these limitations, we developed NACHOS (**N**ested and **A**utomated **C**ross-validation and **H**yperparameter **O**ptimization using **S**upercomputing) to integrate NCV and AHPO into a parallelized computational workflow on HPC. A repository of chest X-ray datasets from the TorchXRayVision

library²¹, along with an kidney OCT dataset, derived from Wang et al.¹⁴, were used to demonstrate NACHOS. We compared different strategies for partitioning the two datasets into k folds in NCV. The results showed the significance of partitioning in benchmarking the test performance of deep learning models^{22,23}.

The outcome of the NACHOS algorithm is a reduced-variance and uncertainty-quantified estimation of test performance of the models generated by this computational procedure. To build the final model for production use, we developed an algorithm named **D**eployment with **A**utomated **C**ross-validation and **H**yperparameter **O**ptimization using **S**upercomputing (DACHOS). DACHOS identifies the overall best model configuration using all the data for the AHPO and cross-validation and then uses this configuration to train a model using all the data. Because the final model for deployment was hyperparameter-optimized and trained using more data than the k models generated in the k -fold NCV, its test performance, although unknown, is expected to be better than the average test performance of the k NCV models. We also demonstrated DACHOS using the chest X-ray repository and kidney OCT dataset.

2. Methodology

2.1 Nested and Automated Cross-validation and Hyperparameter Optimization using Supercomputing (**NACHOS**) algorithm

The NACHOS algorithm comprises three nested loops: the cross-testing (CT) loop, the AHPO loop, and the cross-validation (CV) loop. First, the dataset D is divided into k folds: F_0, F_1, \dots, F_{k-1} . The CT loop iterates over $i \in I = \{0, 1, 2, \dots, k-1\}$, where the fold, F_i , is held out as the test set, and the remaining folds are used for training and validation. The AHPO loop then iterates over $j \in J = \{0, 1, 2, \dots, n-1\}$, where n is the number of hyperparameter configurations to be tried and each h_j denotes the j^{th} hyperparameter configuration. Within the CV loop, the index $m \in I - \{i\}$ is used to reserve the fold, F_m , for validation while the model is trained on the remaining $k-2$ folds. The model's performance on the validation fold F_m is recorded as v_m^j . After completing cross-validation, the average validation performance—i.e. cross-validation

performance—for each hyperparameter h_j , denoted as \bar{v}^j , is calculated. Once the AHPO loop is completed, the best-performing hyperparameter h_{j^*} is selected based on the highest cross-validation performance. The model is then trained using h_{j^*} on all folds except the test fold F_i and evaluated on the test fold F_i with the result recorded as t_i . Finally, after all iterations of the cross-testing loop are completed, the benchmarking results across all test folds are used to calculate the average performance metric and its standard error.

In the current implementation of NACHOS, the AHPO loop used a random search algorithm²⁴ that randomly samples n combinations of values from a set of predefined hyperparameters choices. The choices of the batch size were powers of 2, ranging from 16 (2^4) to 128 (2^7). The choices of the learning rate and the decay were powers of 10, ranging from 0.01 (10^{-2}) to 0.0001 (10^{-4}). The choices of the momentum included 0.5, 0.9, and 0.99 with or without Nesterov acceleration. Three choices of model architectures, ResNet50²⁵, InceptionV3²⁶, and Xception²⁷, are available. A hyperparameter configuration was created by randomly selecting one of the choices for each hyperparameter. The AHPO loop iterates over n hyperparameter configurations to find the best one based on their cross-validation performance. In this study, 9 hyperparameter configurations were randomly generated for AHPO (Table 1).

2.2. Deployment with Automated Cross-validation and Hyperparameter Optimization using Supercomputing (**DACHOS**) algorithm

The DACHOS algorithm generates a production model, M , for deployment using AHPO and cross-validation. The dataset D is split into k folds for cross-validation. The AHPO loop iterates through n hyperparameter configurations $h_j, j \in J = \{0, 1, 2, \dots, n - 1\}$, which should be the same as those used by NACHOS. The cross-validation loop iterates through $m \in I = \{0, 1, 2, \dots, k - 1\}$ to select the fold F_m for validation and train the model on the remaining folds. The validation performance is recorded as v_m^j . After the k -fold cross-validation is completed for the hyperparameter configuration h_j , its average validation performance—i.e. cross-validation

performance—is calculated as \bar{v}^j . Once cross-validation for all hyperparameter configurations is completed, the best-performing hyperparameter h_{j^*} is selected based on its cross-validation performance. Finally, the production model, M , is trained using h_{j^*} with the entire dataset D . The DACHOS algorithm maximizes the performance of the production model, M , for deployment by using the entire dataset for AHPO and then using the entire dataset for model training.

2.3 Parallelization of NACHOS and DACHOS

The NACHOS and DACHOS algorithms were parallelized using a Python implementation of the Message Passing Interface (MPI) standard provided in the `mpi4py`²⁸ library. Both algorithms employed MPI point-to-point communication to enable direct interaction between parallel processes. The NACHOS algorithm distributes a total of $k*(k-1)*n$ training tasks over g GPUs, where k is the number of folds for NCV, n is the number of hyperparameter configurations for AHPO, and g is the number of GPUs. The DACHOS algorithm parallelizes $k*n$ training tasks over g GPUs. When launched, the two algorithms create a manager process along with g worker processes, with each worker process assigned to a separate GPU. The manager process is responsible for assigning the tasks and sending their hyperparameter configurations, test folds, and validation folds to the worker processes for computing on their assigned GPUs. When a worker process completes a training task, it requests a new task from the manager process until all the tasks are completed. The dynamic scheduling provides effective load balancing and ensures linear scalability.

2.4 Fault tolerance in NACHOS and DACHOS

To manage unexpected failures of training tasks in a job, NACHOS and DACHOS implement fault tolerance using a checkpointing system that includes two types of checkpoints: a metadata checkpoint and a model checkpoint. During training, the system continuously records the hyperparameter configuration h_j , the test fold F_i ,

the validation fold F_m , and the epoch number in the metadata checkpoint. After each epoch, a model checkpoint is saved while the previous one is deleted to conserve space. In the event of a failure, NACHOS or DACHOS needs to be rerun. The manager process resends all training tasks. Each worker process then consults the metadata checkpoint to determine if its assigned task has already been completed; if it has, the task is skipped. For unfinished tasks, the worker process resumes training by loading the corresponding model checkpoint.

2.5 Platform and dependencies

NACHOS and DACHOS were implemented using Python 3 and TensorFlow 2. They use dill²⁹ for saving and loading configuration checkpoints, mpi4py²⁸ for parallelization, fasteners for process locking and unlocking, NumPy³⁰ for working with arrays, scikit-learn³¹ for computing performance metrics, SciPy³² for statistical analysis of the results, and termcolor for color-coded standard output messages. NACHOS and DACHOS can are capable of generating learning curves, confusion matrices, and Receiver Operating Characteristic (ROC) curves for result visualization. They can also generate saliency maps for instance-wide prediction interpretation or feature importance^{33,34} using GradCAM³⁵ which requires Matplotlib³⁶ and seaborn³¹.

NACHOS and DACHOS were designed to operate on both a supercomputer with GPU nodes and a Beowulf cluster of GPU workstations connected via a local Ethernet network. The algorithms were tested on the Schooner supercomputer, utilizing GPU nodes equipped with NVIDIA A100 GPUs. Jobs on the supercomputer were managed through the SLURM system, with the GPU count per node specified. Additional experiments were performed on a Beowulf cluster comprising GPU workstations with NVIDIA RTX A6000 and NVIDIA RTX 4090 GPUs, running Ubuntu 20.04.6 LTS. In the Beowulf cluster, data were distributed across all workstations, and jobs were executed using a configuration file that specified the GPU count and the IP address of each workstation.

2.6 Medical imaging datasets for performance benchmarking

A chest X-ray repository was built using the ChestX-ray8 dataset ³⁷, the CheXpert dataset ³⁸, the MIMIC-CXR dataset ³⁹, and the PadChest dataset ⁴⁰ from the TorchXRayVision library ²¹. The NACHOS and DACHOS algorithms were evaluated on a binary classification task, in which deep learning models were trained to classify Posterior-Anterior (PA) chest X-ray images as either cardiomegaly or no finding. In the MIMIC-CXR and PadChest datasets, some lateral images were mistakenly labeled as PA. These incorrectly labeled images were identified through manual inspection and removed from our chest X-ray repository. All images were resized to a resolution of 224x224 pixels through interpolation. To create balanced data for benchmarking, we randomly selected 620 images with cardiomegaly and 620 images with no finding from each of the four datasets. These images are combined to build the chest X-ray repository with a total of 4,960 images (4 datasets X 2 classes X 620 images per class per dataset). The chest X-ray repository was partitioned into four folds using three different partitioning levels. In the image-level partitioning, images were randomly distributed across four folds. In the patient-level partitioning, all images from the same patient were assigned to the same fold. Finally, in the dataset-level partitioning, each dataset was exclusively allocated to a separate fold.

An OCT dataset was derived from our previous study ¹³ for a renal tissue classification task. The OCT images were originally captured as 3D volumes, each containing multiple 2D cross-sectional B-scan images. These 2D cross-sectional images with a resolution of 185x210 pixels were used as the input data in this study. The OCT dataset contains 600 images of the cortex tissue, 600 images of the medulla tissue, and 600 images of the pelvis tissue from each kidney. A total of 10 kidneys were included, yielding 18,000 images (10 kidneys X 3 tissue types X 600 images per tissue type per kidney). The OCT dataset was partitioned into 10 folds at three levels: image, volume, and kidney. In the image-level partitioning, all 18,000 images were randomly split into 10 folds. In the volume-level partitioning, images from the same volume were assigned to the same fold. In kidney-level partitioning, the 10 kidneys were divided into 10 folds, with each fold containing all images from a respective kidney.

3. Results

3.1 Reduced-variance estimation of the test performance with uncertainty quantification using the NACHOS algorithm.

Table 2 presents the benchmarking results generated by the NACHOS algorithm on the chest X-ray repository for the cardiomegaly detection task. The data was partitioned into four folds corresponding to the four datasets. NACHOS included three nested loops: the CT loop, the AHPO loop, and the CV loop. For each test fold, the AHPO loop evaluated the cross-validation performance of nine hyperparameter configurations shown in Table 1. When F_0 was reserved as the test fold, configuration h_2 had the highest cross-validation accuracy of $\bar{v}^2 = 0.72$, which was the average of the validation accuracies, $v_1^2 = 0.69$ using F_1 as the validation set, $v_2^2 = 0.70$ using F_2 as the validation set, and $v_3^2 = 0.76$ using F_3 as the validation set. After finding h_2 as the configuration with the best cross-validation accuracy, a model was trained on F_1 , F_2 , and F_3 using h_2 and then tested on F_0 to generate a test accuracy of $t_0 = 0.79$. This process was repeated for the remaining test folds— F_1 , F_2 , and F_3 —which generated the test accuracies of 0.71, 0.69, and 0.82, respectively. The standard deviation of the test accuracies was 0.06. The variability of the test performance stemmed from the different allocations of data folds for training, validation and testing, as well as the stochastic nature of stochastic gradient descend in model training and random search in AHPO.

The average test accuracy across all four folds was 0.75 with a standard error of 0.03. This average test accuracy, derived from four model instances, reflected the expected performance of model instances that can be generated by our development procedure. The standard deviation of this average test accuracy (0.03) was half ($\sqrt{4}$) of the standard deviation of the individual test accuracies (0.06) from different test folds, because the average test accuracy represented the test performance from all four folds.

The test accuracies were lower than the cross-validation accuracies in test folds F_1 and F_2 and higher in test folds F_0 and F_3 . Although the AHPO was expected to make the cross-validation accuracies higher than the test accuracies, the results were inconsistent probably because of the data variability.

The validation accuracies averaged across all hyperparameter configurations in each test fold were 0.65 for test fold F_0 , 0.70 for F_1 , 0.67 for F_2 , and 0.62 for F_3 . The cross-validation accuracy of the optimal hyperparameter configuration exceeded the average for fold F_0 by 0.07, for F_1 by 0.09, for F_2 by 0.10, and for F_3 by 0.08. The improvements underscored the significance of selecting the appropriate hyperparameters using AHPO.

The results from the NACHOS algorithm on the kidney OCT dataset are summarized in Table 3. The OCT images were partitioned into 10 folds with each fold containing all the images from one kidney. When F_0 was reserved as the test set, hyperparameter configuration h_5 yielded the best cross-validation accuracy of 0.93, which was the average of 9 validation accuracies from the 9-fold cross-validation from F_1 to F_9 . Then, a model trained on the nine folds from F_1 to F_9 using configuration h_5 achieved a test accuracy of $t_0 = 0.73$ on the reserved test fold F_0 . The cross-testing loop in NACHOS repeated this process for the remaining nine folds and generated the test accuracies of 0.71, 0.79, 0.75, 0.88, 0.87, 0.94, 0.90, 0.96, and 0.86 from folds F_1 to F_9 . The test accuracies had a wide range from 0.71 to 0.96 with a standard deviation of 0.09, which reflected the significant data variability from kidney to kidney in this dataset.

The average test accuracy from the 10 test folds was 0.84 with a standard error of 0.03. The cross-testing procedure reduced the standard deviation of the average test accuracy estimation by approximately three folds ($\sqrt{10}$) from 0.09 to 0.03 by averaging across 10 individual test accuracies from separate kidney samples. The standard deviation of the individual folds' test accuracy was higher in the kidney OCT data (0.09) than the chest X-ray data (0.06), but the standard error of the average test accuracy was 0.03 in both datasets because of the higher number of test folds used in the kidney OCT data than the chest X-ray data.

The validation accuracies averaged across all hyperparameter configurations for test folds F_0-F_9 were 0.90, 0.87, 0.87, 0.87, 0.85, 0.85, 0.87, 0.84, 0.86, and 0.85, respectively. The cross-validation accuracies of the optimal hyperparameter configurations for these folds exceeded the average by 0.03, 0.03, 0.02, 0.03, 0.05,

0.03, 0.03, 0.04, 0.04, and 0.05, which demonstrated the effect of AHPO for improving model performance.

3.2 Development of production model for deployment using the DACHOS algorithm.

The DACHOS algorithm was used to produce a final model for deployment. The results of applying DACHOS to the chest X-ray repository are shown in Table 4. When hyperparameter configuration h_0 was used, the validation accuracies were $v_0^0 = 0.71$ for fold F_0 , $v_1^0 = 0.71$ for fold F_1 , $v_2^0 = 0.69$ for fold F_2 , and $v_3^0 = 0.70$ for fold F_3 , which yielded a cross-validation accuracy of $\bar{v}^0 = 0.71$ for h_0 . DACHOS and NACHOS produced different validation accuracies for the same configuration, because DACHOS used 3 data folds for training while NACHOS used only 2. Four-fold cross-validation was performed for all remaining hyperparameter configurations; configuration h_5 achieved the highest cross-validation accuracy of $\bar{v}^5 = 0.75$. This configuration was then used to train a final model on all four data folds.

For the same configuration, 78% of the cross-validations accuracies for NACHOS were lower than the cross-validation accuracy for DACHOS. For example, with hyperparameter configuration h_5 , NACHOS achieved cross-validation accuracies of 0.72, 0.73, 0.71, and 0.69 for test folds F_0 to F_3 using 2 data folds for training. In contrast, DACHOS achieved a cross-validation accuracy of 0.75 with h_5 using 3 data folds for training. An additional data fold available for training in DACHOS may have contributed to its increased cross-validation accuracy. .

The DACHOS algorithm was also applied to the kidney OCT dataset (Table 5). Here, hyperparameter configuration h_0 achieved validation accuracies of 0.63, 0.88, 0.85, 0.76, 0.91, 0.97, 0.92, 0.87, 0.94, 0.88 for folds F_0 to F_9 , which was averaged to a cross-validation accuracy of $\bar{v}^0 = 0.86$. This process was repeated for the remaining hyperparameter configurations. Hyperparameter configuration h_2 had the highest average accuracy $\bar{v}^2 = 0.90$ and h_3 had the lowest $\bar{v}^3 = 0.84$. The production model was trained using h_2 on the entire dataset.

3.3 Evaluation and interpretation of model performance across different partitioning levels.

NACHOS and DACHOS required data to be partitioned into multiple folds for NCV and CV. An appropriate partitioning design was essential for evaluating a model's ability to generalize to unseen data from a new measurement, a new patient or a new location. The input data in the chest X-ray repository was organized into three partitioning levels, including the image level, the patient level, and the dataset level, for testing to evaluate different partitioning designs (Figure 1A). The chest X-ray repository comprised a total of 4,960 images from 4,678 patients across four datasets. There were 1,187 patients in the CheXpert dataset, 1,165 patients in the MIMIC-CXR dataset, 1,105 patients in the ChestX-ray8 dataset, and 1,221 patients in the PadChest dataset. The dataset-level partitioning was used in the previous Results sections to evaluate the model performance on unseen data from a different location. Here, we compared the test performance benchmarked using NACHOS across image-level, patient-level, and dataset-level partitioning (Figure 1B). Because each patient had approximately 1.06 images on average in the chest X-ray repository, the image-level and the patient-level partitioning yielded similar average test accuracies of 0.811 and 0.809, respectively, which reflected the test performance of the models on unseen data from new images or new patients within the 4 datasets. These were much higher than the average test accuracy of 0.750 from the dataset-level partitioning (Figure 1B). The variability in test accuracies across data folds also increased from 0.008 for the image-level partitioning and 0.009 for the patient-level partitioning to 0.061 for the dataset-level stratification.

The test performances were also benchmarked using NACHOS on the kidney OCT dataset at three partitioning levels: image, volume, and kidney (Figure 2A). The kidney-level partitioning was used in the previous Results sections to evaluate the model performance on unseen data from a new kidney. Here, we compared the test performance benchmarked by NACHOS using the image-level, volume-level, and kidney-level partitioning (Figure 2B). The image-level partitioning resulted in a perfect test accuracy of 1.00 across all data folds owing to nearly complete data redundancy among contiguous images from the same volume. The volume-level partitioning

generated an average test accuracy of 0.97 and a standard deviation of 0.01 across different data folds, suggesting still substantial data redundancy among volumes from the same kidney. In contrast, the kidney-level partitioning yielded an average test accuracy of 0.84 and a standard deviation of 0.09. The reduced accuracy and increased variability reflected the real-world variability of the OCT data from different kidneys and the need for the models to generalize well across kidneys.

3.4 Parallelization of NACHOS and DACHOS over multiple GPUs

We compared the execution time of the NACHOS algorithm on various GPU systems using the kidney OCT dataset with kidney-level partitioning using a single hyperparameter configuration. These systems included a Beowulf cluster of GPU workstations with Nvidia GeForce RTX 4090 or Nvidia RTX A6000 GPUs on a local Ethernet network, as well as the OSCER supercomputer with Nvidia A100 GPUs. The execution time was 21.9 hours on a single RTX A6000, 13.8 hours on a single RTX 4090, and 11.7 hours on a single A100 (Figure 3A). RTX A6000 has 336 tensor cores and memory bandwidth of 112.5 GB/s, RTX 4090 has 512 tensor cores and memory bandwidth ~1000 GB/s, and A100 has 432 tensor cores and memory bandwidth of ~2000 GB/s. RTX 4090 performed better than RTX probably due to higher number of tensor cores. A100 performed better than RTX 4090 probably due to higher memory bandwidth as well as optimizations for deep learning applications. The peak memory usages on these GPUs were approximately all 18.5 GB. This demonstrated the portability of NACHOS and DACHOS across a variety of computing systems.

NACHOS and DACHOS can distribute the training across multiple GPUs with fault tolerance to reduce the wall-clock time. Figure 3B presents the speedup ratios by the number of GPUs using the execution time of a single RTX A6000 GPU (21.9 hours) as the baseline. NACHOS reached linear scalability on RTX A6000s. The execution time was reduced to 11.1 hours on 2 RTX A6000s with a 2.0X speedup, further to 6.9 hours on 3 RTX A6000s with a 3.2X speedup, and finally to 5.4 hours on 4 RTX 6000s with a 4.1X speedup. NACHOS achieved super-linear speedups on RTX 4090s and A100s. The execution time was 13.8 hours on 1 RTX 4090, 7.0 hours on 2 RTX 4090s (3.1X speedup), and 4.7 hours on 3 RTX 4090s (4.6X speedup), and 3.6 hours on 4

RTX 4090s (6.1X speedup). Super-linear speedups were also achieved on A100s: 3.7X on 2 A100s, 7.4X on 4 A100s, and 14.1X on 8 A100s. The execution time was reduced to only 1.6 hours using 8 A100s.

4. Discussion

Accurate and robust benchmarking of the performance of machine learning models has been a challenge in the field of medical imaging ⁴¹. A commonly used approach is to split a labeled dataset into a training set for learning model parameters, a validation set for optimizing model hyperparameters, and a test set for benchmarking the performance of the obtained model. Although cross-validation is often used to rotate data partitions between the training set and the validation set, most machine learning studies split out a single fixed test set for performance benchmarking. NACHOS features NCV in an automated and user-friendly machine learning workflow for medical imaging applications. *k*-fold NCV offers two key advantages over a single test split for test performance benchmarking. First, NCV reduces the variance of the test performance estimation by rotating all data partitions through the test set. Specifically, the variance of the average performance score from *k*-fold NCV should be *k* times lower than the variance of the point estimate of the performance score from a single test split. Second, and more importantly, the variance of the performance estimation is not quantified using a single test split, but is quantified during cross-testing over *k* test partitions. For example, if a user holds out only the last partition for the test set, they would estimate the classification accuracy to be 0.82 in the chest X-ray repository and 0.86 in the kidney OCT dataset with large (± 0.06 and ± 0.09 , respectively) variabilities that are unknown to the user. If the user uses NVC, they would be able to better estimate the accuracy as 0.75 ± 0.03 in the chest X-ray repository and as 0.84 ± 0.03 in the kidney OCT dataset. These use cases demonstrated that NCV reduced and measured the variance of the performance benchmarking of deep learning models.

The partitioning level used by NCV is also important for accurate and robust performance benchmarking. The Checklist for Artificial Intelligence in Medical Imaging (CLAIM) ⁴² emphasizes transparent reporting of data partitioning and recommends partitioning at the patient level or higher. The classification accuracy in the chest X-ray

repository decreased from 0.809 with the patient-level partitioning to 0.750 with the dataset-level partitioning (Figure 1B). This means that it is more difficult for models to generalize to a new dataset acquired by other institutions than to generalize to new patients from a previously seen dataset. Similarly, Zech et al.⁴³ found that pneumonia classification models often perform better on internal test datasets originated from the same institution as the training data than on external test datasets from institutions different than the ones used for training. The choice of the partitioning level for NCV performance benchmarking should match the intended use scenario of the models. For instance, the patient-level partitioning can be used to evaluate models designed for use within the same hospitals that produced the training data, as it mimics the scenario of encountering new patients within these hospitals. The institution-level partitioning should be used to evaluate models intended to be deployed to new hospitals.

NACHOS benchmarks the average test performance of models generated by a reproducible workflow using a specific dataset. AHPO is needed in NACHOS because it is impractical to perform laborious manual hyperparameter optimization consistently across all test folds in NCV. Random search²⁴ is a simple, yet highly effective, AHPO method. In the chest X-ray repository (Table 2), the best hyperparameter configuration achieved improvements ranging from 0.07 to 0.10 over the average cross-validation accuracy of all configurations. In the kidney OCT dataset (Table 3), AHPO delivered performance gains between 0.02 and 0.05 above the overall average. In future, NACHOS can offer users multiple AHPO methods, such as Hyperband⁴⁴ and its Bayesian counterpart, Bayesian Optimization Hyperband (BOHB)⁴⁵.

NCV and AHPO in NACHOS incur a significant computational cost that needs to be distributed across multiple GPUs to shorten the wall-clock time of model development. The parallelization in NACHOS achieved linear and super-linear speedups on different kinds of GPUs (Figure 3A and 3B). Super-linear speedup, observed in certain cases, suggests additional efficiency gains arising from improved cache utilization, reduced memory bottlenecks, or synergistic effects in GPU parallelism. After NACHOS measures the test performance of models from a model development workflow, DACHOS is used to generate a production model for

deployment using this workflow. The actual test performance of this production model is unknown but should be higher than the test performance of the models benchmarked by NACHOS. This is because the AHPO in DACHOS can use an extra data fold than the AHPO in NACHOS and the final model training in DACHOS can use two additional data folds than the training in NACHOS.

In conclusion, NACHOS integrates NCV, AHPO and HPC into an automated workflow. NCV reduces and measures variance in test performance estimation by rotating all data folds through the test set. AHPO enhances model performance by searching for optimal hyperparameters and avoids the impracticality of manual tuning across multiple test folds in NCV. To mitigate substantial computational costs of NCV and AHPO, NACHOS can distribute computation across multiple GPUs with linear or super-linear speed-up. After NACHOS completes model benchmarking, DACHOS can be used to generate final production models with potentially higher performance. Together, these frameworks provide a robust, reproducible, and scalable approach to developing and evaluating machine learning models in medical imaging.

Data availability

The repository for NACHOS and DACHOS is available at
<https://github.com/theplanlab/NACHOS>.

Code availability

The kidney OCT dataset and chest X-ray repository are available at
<https://doi.org/10.5281/zenodo.14847200>.

References

- 1 Liu, X. *et al.* A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis. *The Lancet Digital Health* **1**, e271-e297 (2019). [https://doi.org/10.1016/S2589-7500\(19\)30123-2](https://doi.org/10.1016/S2589-7500(19)30123-2)

2 Brown, J. M. *et al.* Automated Diagnosis of Plus Disease in Retinopathy of Prematurity Using Deep Convolutional Neural Networks. *JAMA Ophthalmology* **136**, 803-810 (2018).
<https://doi.org:10.1001/jamaophthalmol.2018.1934>

3 Hwang, D. K. *et al.* Artificial intelligence-based decision-making for age-related macular degeneration. *Theranostics* **9**, 232-245 (2019).
<https://doi.org:10.7150/thno.28447>

4 He, M., Li, Z., Liu, C., Shi, D. & Tan, Z. Deployment of Artificial Intelligence in Real-World Practice: Opportunity and Challenge. *Asia-Pacific Journal of Ophthalmology* **9**, 299-307 (2020).
<https://doi.org:https://doi.org/10.1097/APO.0000000000000301>

5 Stanley, E. A. M. *et al.* Towards objective and systematic evaluation of bias in artificial intelligence for medical imaging. *Journal of the American Medical Informatics Association* (2024).
<https://doi.org:10.1093/jamia/ocae165>

6 Tejani, A. S., Ng, Y. S., Xi, Y. & Rayan, J. C. Understanding and Mitigating Bias in Imaging Artificial Intelligence. *RadioGraphics* **44**, e230067 (2024). <https://doi.org:10.1148/rg.230067>

7 Vrudhula, A., Kwan, A. C., Ouyang, D. & Cheng, S. Machine Learning and Bias in Medical Imaging: Opportunities and Challenges. *Circulation: Cardiovascular Imaging* **17**, e015495 (2024).
<https://doi.org:doi:10.1161/CIRCIMAGING.123.015495>

8 Yang, Z., Yu, Y., You, C., Steinhardt, J. & Ma, Y. in *International Conference on Machine Learning*. 10767-10777 (PMLR).

9 Berrar, D. Cross-Validation. *Encyclopedia of Bioinformatics and Computational Biology* **1**, 542-545 (2019).

10 Bates, S., Hastie, T. & Tibshirani, R. Cross-Validation: What Does It Estimate and How Well Does It Do It? *Journal of the American Statistical Association* **119**, 1434-1445 (2024).
<https://doi.org:10.1080/01621459.2023.2197686>

11 Nawabi, J. *et al.* Imaging-Based Outcome Prediction of Acute Intracerebral Hemorrhage. *Translational Stroke Research* **12**, 958-967 (2021). <https://doi.org:10.1007/s12975-021-00891-8>

12 Chen Wang, Y. L., Paul Calle, Xinwei Li, Ronghao Liu, Qinghao Zhang, Feng Yan, Kar-ming Fung, Andrew K. Conner, Sixia Chen, Chongle Pan, Qinggong Tang. Enhancing Epidural Needle Guidance using a Polarization-Sensitive Optical Coherence Tomography Probe with Convolutional Neural Networks.

13 Wang, C. *et al.* Epidural anesthesia needle guidance by forward-view endoscopic optical coherence tomography and deep learning. *Scientific Reports* **12**, 9057 (2022). <https://doi.org/10.1038/s41598-022-12950-7>

14 Wang, C. *et al.* Deep-learning-aided forward optical coherence tomography endoscope for percutaneous nephrostomy guidance. *Biomedical Optics Express* **12**, 2404-2418 (2021). <https://doi.org/10.1364/BOE.421299>

15 Wang, C. *et al.* Computer-aided Veress needle guidance using endoscopic optical coherence tomography and convolutional neural networks. *Journal of Biophotonics* **15**, e202100347 (2022).

16 Roberts, M. *et al.* Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans. *Nature Machine Intelligence* **3**, 199-217 (2021). <https://doi.org/10.1038/s42256-021-00307-0>

17 Gibson, E. *et al.* NiftyNet: a deep-learning platform for medical imaging. *Computer Methods and Programs in Biomedicine* **158**, 113-122 (2018). <https://doi.org/https://doi.org/10.1016/j.cmpb.2018.01.025>

18 Pérez-García, F., Sparks, R. & Ourselin, S. TorchIO: A Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. *Computer Methods and Programs in Biomedicine* **208**, 106236 (2021). <https://doi.org/https://doi.org/10.1016/j.cmpb.2021.106236>

19 Beers, A. *et al.* DeepNeuro: an open-source deep learning toolbox for neuroimaging. *Neuroinformatics* **19**, 127-140 (2021). <https://doi.org/10.1007/s12021-020-09477-5>

20 Pati, S. *et al.* GaNDLF: the generally nuanced deep learning framework for scalable end-to-end clinical workflows. *Communications Engineering* **2**, 23 (2023). <https://doi.org/10.1038/s44172-023-00066-3>

21 Cohen, J. P. *et al.* in *International Conference on Medical Imaging with Deep Learning*. 231-249 (PMLR).

22 Yagis, E. *et al.* Effect of data leakage in brain MRI classification using 2D convolutional neural networks. *Scientific Reports* **11**, 22544 (2021). <https://doi.org/10.1038/s41598-021-01681-w>

23 Tampu, I. E., Eklund, A. & Haj-Hosseini, N. Inflation of test accuracy due to data leakage in deep learning-based classification of OCT images. *Scientific Data* **9**, 580 (2022). <https://doi.org/10.1038/s41597-022-01618-6>

24 Bergstra, J. & Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research* **13** (2012).

25 He, K., Zhang, X., Ren, S. & Sun, J. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770-778.

26 Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818-2826.

27 Chollet, F. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1251-1258.

28 Dalcin, L. & Fang, Y.-L. L. mpi4py: Status update after 12 years of development. *Computing in Science & Engineering* **23**, 47-54 (2021).

29 McKerns, M. M., Strand, L., Sullivan, T., Fang, A. & Aivazis, M. A. Building a framework for predictive science. *arXiv preprint arXiv:1202.1056* (2012).

30 Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585**, 357-362 (2020). <https://doi.org/10.1038/s41586-020-2649-2>

31 Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* **12**, 2825-2830 (2011).

32 Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods* **17**, 261-272 (2020).

33 Badré, A. & Pan, C. LINA: A Linearizing Neural Network Architecture for Accurate First-Order and Second-Order Interpretations. *IEEE Access* **10**, 36166-36176 (2022).
<https://doi.org:10.1109/ACCESS.2022.3163257>

34 Badré, A. & Pan, C. Explainable multi-task learning improves the parallel estimation of polygenic risk scores for many diseases through shared genetic basis. *PLOS Computational Biology* **19**, e1011211 (2023). <https://doi.org:10.1371/journal.pcbi.1011211>

35 Selvaraju, R. R. *et al.* in *Proceedings of the IEEE international conference on computer vision*. 618-626.

36 Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in science & engineering* **9**, 90-95 (2007).

37 Wang, X. *et al.* in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2097-2106.

38 Irvin, J. *et al.* CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. *Proceedings of the AAAI Conference on Artificial Intelligence* **33**, 590-597 (2019).
<https://doi.org:10.1609/aaai.v33i01.3301590>

39 Johnson, A. E. W. *et al.* MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports. *Scientific Data* **6**, 317 (2019). <https://doi.org:10.1038/s41597-019-0322-0>

40 Bustos, A., Pertusa, A., Salinas, J.-M. & de la Iglesia-Vayá, M. PadChest: A large chest x-ray image dataset with multi-label annotated reports. *Medical Image Analysis* **66**, 101797 (2020).
<https://doi.org:https://doi.org/10.1016/j.media.2020.101797>

41 Varoquaux, G. & Cheplygina, V. Machine learning for medical imaging: methodological failures and recommendations for the future. *npj Digital Medicine* **5**, 48 (2022).
<https://doi.org:10.1038/s41746-022-00592-y>

42 Mongan, J., Moy, L. & Kahn Jr, C. E. (Radiological Society of North America, 2020).

43 Zech, J. R. *et al.* Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study. *PLOS Medicine* **15**, e1002683 (2018).
<https://doi.org/10.1371/journal.pmed.1002683>

44 Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. & Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* **18**, 6765-6816 (2017).

45 Falkner, S., Klein, A. & Hutter, F. in *Proceedings of the 35th International Conference on Machine Learning* Vol. 80 (eds Dy Jennifer & Krause Andreas) 1437--1446 (PMLR, Proceedings of Machine Learning Research, 2018).

Acknowledgements

Special thanks to Jessica Shaw for her assistance with the coding aspects of this project. This work was supported by grants from the University of Oklahoma Health Sciences Center (3P30CA225520), National Science Foundation (OIA-2132161, 2238648, 2331409), National Institute of Health (R01DK133717), Oklahoma Center for the Advancement of Science and Technology (HR23-071), the medical imaging COBRE (P20 GM135009), the Prevent Cancer Foundation, and the Midwest Biomedical Accelerator Consortium (MBArC), an NIH Research Evaluation and Commercialization Hub (REACH). Histology service provided by the Tissue Pathology Shared Resource was supported in part by the National Institute of General Medical Sciences COBRE Grant P20GM103639 and National Cancer Institute Grant P30CA225520 of the National Institutes of Health. Financial support was provided by the OU Libraries' Open Access Fund.

Algorithm 1: NACHOS

Required k : number of folds
 n : number of set of hyperparameter configurations from random search
 D : Dataset
 $I = \{0, 1, \dots, k - 1\}$
 $J = \{0, 1, \dots, n - 1\}$
 $H = \{h_0, h_1, \dots, h_n\}$

Defined v_m^j : Validation performance for hyperparameter configuration h_j on
 the validation fold F_m
 \bar{v}^j : Average validation performance of hyperparameter configuration h_j
 across all validation folds
 t_i : Test performance on the test fold F_i

Result average and standard error for performance metric
Split D into k folds: F_0, F_1, \dots, F_{k-1}

/ Cross-testing loop */*

for $i \in I$ **do**

 Set F_i as test dataset

/ AHPO loop */*

for $j \in J$ **do**

 Set h_j as hyperparameter configuration

/ Cross-validation loop */*

for $m \in I - \{i\}$ **do**

 Set F_m as validation dataset

// Distribution of tasks

 Train model on remaining folds $(I - \{i, m\})$ with h_j

$v_m^j \leftarrow$ Evaluate model for validation performance on F_m

end

$\bar{v}^j \leftarrow$ Average values $v_m^j, m \in I - \{i\}$

end

$j^* \leftarrow \arg \max_j \{\bar{v}^j : j \in J\}$

 Train model on folds: $I - \{i\}$ with h_{j^*}

$t_i \leftarrow$ Evaluate model for test performance on F_i

end

return average and standard error for values $t_i, i \in I$

Algorithm 2: DACHOS

Required k : number of folds

D : Dataset

$I = \{0, 1, \dots, k - 1\}$

$J = \{0, 1, \dots, n - 1\}$

$H = \{h_0, h_1, \dots, h_n\}$

Defined v_m^j : Validation performance for hyperparameter configuration h_j on the validation fold F_m

\bar{v}^j : Average validation performance of hyperparameter configuration h_j across all validation folds

Result M : Model ready to be deployed

Split D into k folds: F_0, F_1, \dots, F_{k-1}

/ AHP0 loop*

**/*

for $j \in J$ **do**

 Set h_j as hyperparameter configuration

/ Cross-validation loop*

**/*

for $m \in I$ **do**

 Set F_m as validation dataset

// Distribution of tasks

 Train model on remaining folds $(I - \{m\})$ with h_j

$v_m^j \leftarrow$ Evaluate model for validation performance on F_m

end

$\bar{v}^j \leftarrow$ Average values $v_m^j, m \in I$

end

$j^* \leftarrow \arg \max_j \{\bar{v}^j : j \in J\}$

$M \leftarrow$ Train model on D with h_{j^*}

return M

Table 1: Randomly generated hyperparameter configurations for AHPO

Index	Architecture	Batch size	Learning rate	Decay	Momentum	Nesterov
h_0	ResNet50	128	0.01	0.01	0.9	Enabled
h_1	InceptionV3	16	0.001	0.001	0.9	Disabled
h_2	ResNet50	64	0.01	0.01	0.99	Enabled
h_3	Xception	16	0.001	0.001	0.5	Disabled
h_4	ResNet50	64	0.01	0.01	0.5	Disabled
h_5	ResNet50	32	0.01	0.01	0.99	Enabled
h_6	ResNet50	32	0.0001	0.0001	0.99	Disabled
h_7	ResNet50	32	0.01	0.01	0.9	Enabled
h_8	InceptionV3	64	0.01	0.01	0.5	Disabled

Table 2: NACHOS accuracy results for chest X-ray repository

		Fold reserved for Test			
		F_0	F_1	F_2	F_3
AHPO/Cross-Validation	h_0	$v_1^0: 0.53$	$v_0^0: 0.50$	$v_0^0: 0.50$	$v_0^0: 0.54$
		$v_2^0: 0.61$	$v_2^0: 0.50$	$v_1^0: 0.49$	$v_1^0: 0.58$
		$v_3^0: 0.75$	$v_3^0: 0.77$	$v_3^0: 0.50$	$v_2^0: 0.50$
	h_1	$v_1^1: 0.66$	$v_0^1: 0.80$	$v_0^1: 0.81$	$v_0^1: 0.80$
		$v_2^1: 0.68$	$v_2^1: 0.74$	$v_1^1: 0.76$	$v_1^1: 0.72$
		$v_3^1: 0.80$	$v_3^1: 0.84$	$v_3^1: 0.71$	$v_2^1: 0.50$
	h_2	$v_1^2: 0.69$	$v_0^2: 0.60$	$v_0^2: 0.70$	$v_0^2: 0.67$
		$v_2^2: 0.70$	$v_2^2: 0.67$	$v_1^2: 0.64$	$v_1^2: 0.68$
		$v_3^2: 0.76$	$v_3^2: 0.71$	$v_3^2: 0.66$	$v_2^2: 0.61$
	h_3	$v_1^3: 0.67$	$v_0^3: 0.67$	$v_0^3: 0.71$	$v_0^3: 0.69$
		$v_2^3: 0.63$	$v_2^3: 0.65$	$v_1^3: 0.63$	$v_1^3: 0.66$
		$v_3^3: 0.82$	$v_3^3: 0.79$	$v_3^3: 0.76$	$v_2^3: 0.56$
	h_4	$v_1^4: 0.50$	$v_0^4: 0.51$	$v_0^4: 0.50$	$v_0^4: 0.50$
		$v_2^4: 0.50$	$v_2^4: 0.50$	$v_1^4: 0.50$	$v_1^4: 0.50$
		$v_3^4: 0.50$	$v_3^4: 0.78$	$v_3^4: 0.67$	$v_2^4: 0.50$
	h_5	$v_1^5: 0.69$	$v_0^5: 0.76$	$v_0^5: 0.70$	$v_0^5: 0.73$
		$v_2^5: 0.69$	$v_2^5: 0.67$	$v_1^5: 0.69$	$v_1^5: 0.72$
		$v_3^5: 0.76$	$v_3^5: 0.76$	$v_3^5: 0.74$	$v_2^5: 0.64$
	h_6	$v_1^6: 0.50$	$v_0^6: 0.77$	$v_0^6: 0.75$	$v_0^6: 0.73$
		$v_2^6: 0.51$	$v_2^6: 0.66$	$v_1^6: 0.50$	$v_1^6: 0.50$
		$v_3^6: 0.77$	$v_3^6: 0.76$	$v_3^6: 0.78$	$v_2^6: 0.50$
	h_7	$v_1^7: 0.70$	$v_0^7: 0.77$	$v_0^7: 0.74$	$v_0^7: 0.51$
		$v_2^7: 0.49$	$v_2^7: 0.69$	$v_1^7: 0.67$	$v_1^7: 0.69$
		$v_3^7: 0.78$	$v_3^7: 0.80$	$v_3^7: 0.74$	$v_2^7: 0.56$
	h_8	$v_1^8: 0.50$	$v_0^8: 0.80$	$v_0^8: 0.81$	$v_0^8: 0.81$
		$v_2^8: 0.61$	$v_2^8: 0.71$	$v_1^8: 0.74$	$v_1^8: 0.73$
		$v_3^8: 0.78$	$v_3^8: 0.80$	$v_3^8: 0.77$	$v_2^8: 0.50$
	best: h_j^*	h_2	h_1	h_8	h_5
	Average	$\bar{v}^2: 0.72$	$\bar{v}^1: 0.79$	$\bar{v}^8: 0.77$	$\bar{v}^5: 0.69$
Cross-Testing	Accuracy	$t_0: 0.79$	$t_1: 0.71$	$t_2: 0.69$	$t_3: 0.82$
	Average and standard error 0.75 ± 0.03				

Note: F_i represents the test fold i . h_j represents the hyperparameter configuration j . In the AHPO/CV loop, inside each cell, the validation accuracy v_m^j for hyperparameter configuration j and validation fold m is located. For each test fold, the best hyperparameter configuration is selected by comparing the average validation accuracy and the whole cell is highlighted in blue. For instance, for F_0 , the hyperparameter configuration h_2 has the highest average accuracy. $\bar{v}^2 = (0.69 + 0.70 + 0.76) / 3 = 0.72$. In the cross-testing loop, the selected hyperparameter configuration is used to calculate the test accuracy t_i for test fold i . For test fold F_0 , test accuracy is $t_0 = 0.79$.

Table 3: NACHOS accuracy results for kidney OCT dataset

Hyperparameter configuration	Fold reserved for Test									
	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
Cross-Validation/AHPO/	h_0	0.91±0.02	0.87±0.04	0.83±0.06	0.85±0.04	0.86±0.03	0.86±0.03	0.87±0.03	0.85±0.03	0.85±0.03
	h_1	0.92±0.01	0.86±0.04	0.87±0.03	0.83±0.06	0.85±0.06	0.88±0.04	0.83±0.04	0.87±0.03	0.87±0.04
	h_2	0.89±0.02	0.87±0.03	0.91±0.02	0.88±0.02	0.85±0.04	0.89±0.02	0.82±0.04	0.90±0.02	0.88±0.02
	h_3	0.89±0.03	0.84±0.05	0.85±0.05	0.82±0.06	0.83±0.06	0.80±0.06	0.81±0.06	0.83±0.04	0.84±0.05
	h_4	0.86±0.03	0.88±0.04	0.85±0.04	0.81±0.04	0.83±0.04	0.85±0.04	0.84±0.03	0.83±0.03	0.83±0.04
	h_5	0.93±0.01	0.89±0.03	0.89±0.02	0.90±0.02	0.88±0.03	0.89±0.02	0.82±0.03	0.90±0.02	0.87±0.03
	h_6	0.88±0.02	0.86±0.04	0.89±0.03	0.85±0.06	0.82±0.05	0.88±0.03	0.85±0.04	0.83±0.03	0.85±0.03
	h_7	0.89±0.01	0.87±0.04	0.87±0.03	0.86±0.03	0.87±0.03	0.88±0.03	0.87±0.03	0.83±0.02	0.80±0.03
	h_8	0.91±0.02	0.89±0.04	0.89±0.03	0.86±0.04	0.86±0.04	0.88±0.03	0.85±0.05	0.88±0.04	0.89±0.03
best: h_j^*	h_5	h_5	h_2	h_5	h_5	h_2	h_0	h_2	h_8	h_2

Cross-Testing	Average and standard error 0.84±0.03									
	t_0 : 0.73	t_1 : 0.71	t_2 : 0.79	t_3 : 0.75	t_4 : 0.88	t_5 : 0.87	t_6 : 0.94	t_7 : 0.90	t_8 : 0.96	t_9 : 0.86
Accuracy	0.73	0.71	0.79	0.75	0.88	0.87	0.94	0.90	0.96	0.86

Note: F_i : represents the test fold i . h_j : represents the hyperparameter configuration j . In the AHPO/CV loop, inside each cell, the average and standard error of the validation accuracies v_m^j for hyperparameter configuration j and validation fold m are located. The cells highlighted in blue have the highest average accuracy for a test fold. For instance, for F_0 , the hyperparameter configuration h_5 has the highest average accuracy. $\bar{v}^5=0.93$. In the cross-testing loop, the selected hyperparameter configuration is used to calculate the test accuracy t_i .

Table 4: DACHOS accuracy results for chest X-ray repository

Hyperparameter configuration	F_0	F_1	F_2	F_3	Average
h_0	$v_0^0: 0.71$	$v_1^0: 0.71$	$v_2^0: 0.69$	$v_3^0: 0.70$	$\bar{v}^0: 0.71$
h_1	$v_0^1: 0.79$	$v_1^1: 0.63$	$v_2^1: 0.71$	$v_3^1: 0.71$	$\bar{v}^1: 0.71$
h_2	$v_0^2: 0.78$	$v_1^2: 0.70$	$v_2^2: 0.69$	$v_3^2: 0.65$	$\bar{v}^2: 0.71$
h_3	$v_0^3: 0.75$	$v_1^3: 0.69$	$v_2^3: 0.66$	$v_3^3: 0.80$	$\bar{v}^3: 0.72$
h_4	$v_0^4: 0.50$	$v_1^4: 0.50$	$v_2^4: 0.59$	$v_3^4: 0.78$	$\bar{v}^4: 0.59$
h_5	$v_0^5: 0.75$	$v_1^5: 0.74$	$v_2^5: 0.71$	$v_3^5: 0.80$	$\bar{v}^5: 0.75$
h_6	$v_0^6: 0.73$	$v_1^6: 0.71$	$v_2^6: 0.73$	$v_3^6: 0.77$	$\bar{v}^6: 0.73$
h_7	$v_0^7: 0.77$	$v_1^7: 0.72$	$v_2^7: 0.63$	$v_3^7: 0.75$	$\bar{v}^7: 0.72$
h_8	$v_0^8: 0.81$	$v_1^8: 0.74$	$v_2^8: 0.63$	$v_3^8: 0.81$	$\bar{v}^8: 0.75$
best: h_{j^*}	h_5				

Note: F_m : represents the validation fold m . h_j : represents the hyperparameter configuration j . Inside each cell, the validation performance v_m^j for hyperparameter configuration j and validation fold m is located. In order to select the configuration hyperparameter, the highest average accuracy \bar{v}^j is selected. Among them, h_5 has the highest average validation accuracy $\bar{v}^5=0.75$, highlighted in blue. Despite apparent equality due to rounding, h_5 's average validation accuracy is higher than h_8 .

Table 5: DACHOS accuracy results for kidney OCT dataset

Fold reserved for Validation											
Hyperparameter configuration	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	Average
h_0	0.63	0.88	0.85	0.76	0.91	0.97	0.92	0.87	0.94	0.88	\bar{v}^0 : 0.86
h_1	0.51	0.91	0.79	0.93	0.90	0.91	0.98	0.84	0.97	0.90	\bar{v}^1 : 0.86
h_2	0.77	0.88	0.81	0.91	0.96	0.93	0.92	0.91	0.97	0.92	\bar{v}^2 : 0.90
h_3	0.37	0.90	0.82	0.83	0.87	0.91	0.91	0.92	0.98	0.87	\bar{v}^3 : 0.84
h_4	0.68	0.89	0.86	0.84	0.78	0.99	0.88	0.86	0.93	0.91	\bar{v}^4 : 0.86
h_5	0.77	0.88	0.89	0.79	0.94	0.97	0.92	0.91	0.96	0.93	\bar{v}^5 : 0.90
h_6	0.66	0.88	0.81	0.91	0.84	0.95	0.91	0.90	0.97	0.92	\bar{v}^6 : 0.88
h_7	0.69	0.87	0.84	0.93	0.85	0.79	0.95	0.86	0.97	0.91	\bar{v}^7 : 0.87
h_8	0.64	0.89	0.69	0.94	0.92	0.97	0.90	0.97	0.90	0.97	\bar{v}^8 : 0.88

Note: F_m : represents the validation fold m . h_j : represents the hyperparameter configuration j . Inside each cell, the validation accuracy v_m^j for hyperparameter configuration j and validation fold m is located. In order to select the configuration hyperparameter, the highest average accuracy \bar{v}^j is selected. Among them, h_2 has the highest average validation accuracy $\bar{v}^2=0.90$, highlighted in blue. Despite apparent equality due to rounding, h_2 's average validation accuracy is higher than h_5 .

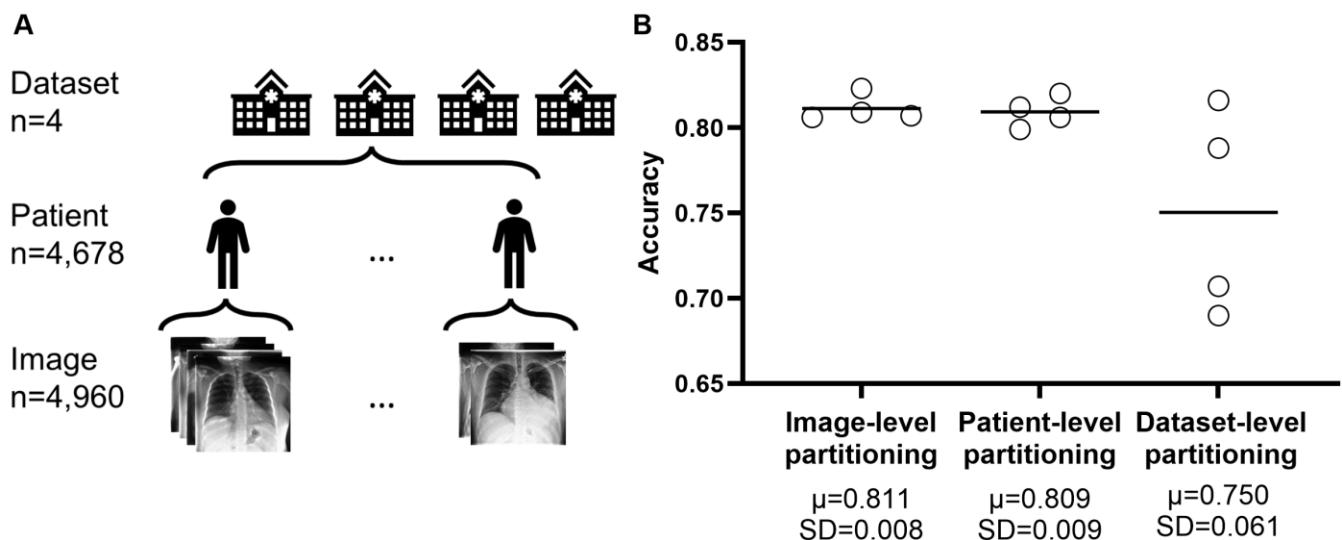


Figure 1: Data partitioning schemes of the chest X-ray repository. [A] Data structure of the chest X-ray repository. The repository includes four datasets, each originating from a different set of institutions, capturing variations in imaging protocols and patient populations. **[B]** Mean (μ) and standard deviation (SD) of the test accuracy from data partition on the image level, the patient level, and the dataset level. The four open circles represent the test accuracies of four partitions of mixed images regardless of patients, four partitions of mixed patients regardless of their source datasets, and four partitions corresponding to the four datasets. The dataset-level partitioning had significantly lower mean and higher variability in the test accuracies.

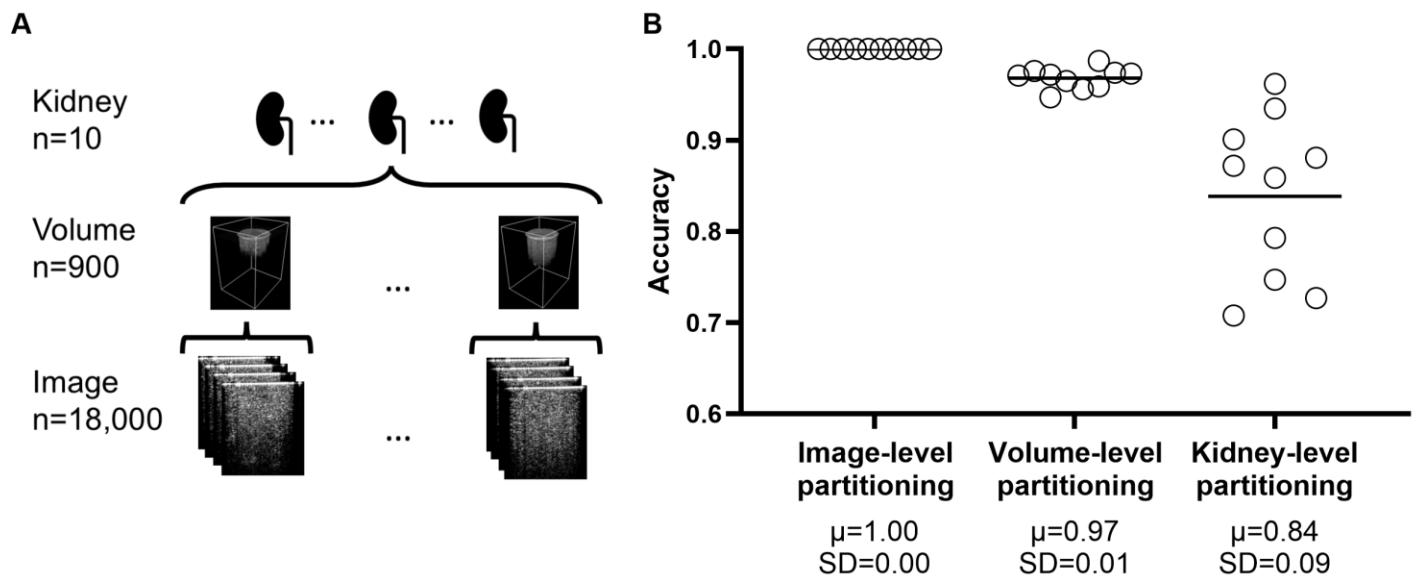


Figure 2: Data partitioning schemes of the kidney OCT dataset. [A] Data structure of the kidney OCT dataset. The dataset includes 10 kidneys, each generating 90 OCT volumes. 20 B-scan images are extracted from each volume. **[B]** Mean (μ) and standard deviation (SD) of the test accuracy from data partition on the image level, the volume level, and the kidney level. The 10 open circles represent the test accuracies of 10 partitions of mixed images regardless of volume, 10 partitions of mixed volume regardless of their source kidney, and 10 partitions corresponding to the 10 kidneys. The kidney-level partitioning had significantly lower mean and higher variability in the test accuracies.

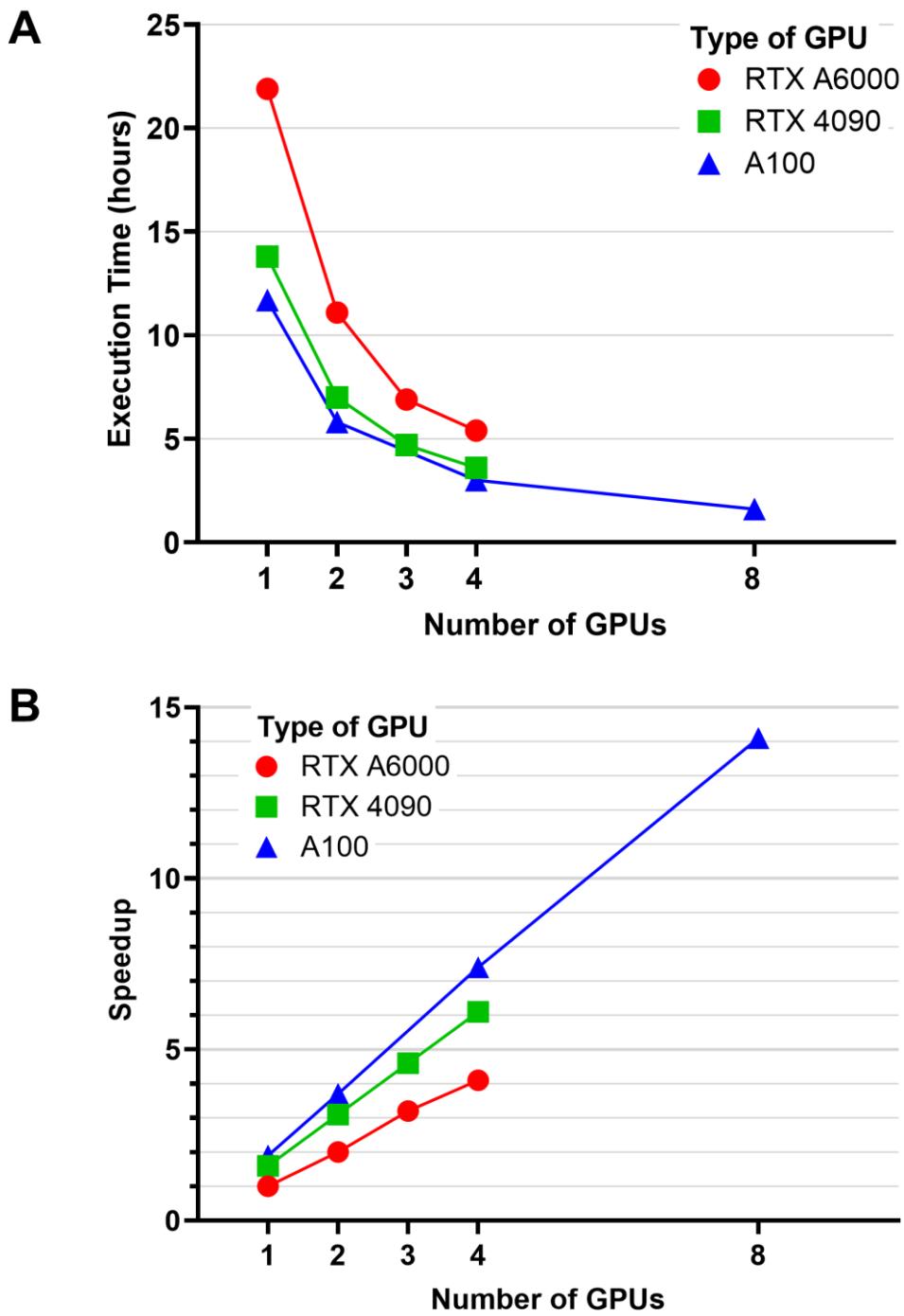


Figure 3: Scalability of NACHOS parallelization.

[A] Execution time as a function of the number of GPUs for three GPU types: RTX A6000 (Beowulf cluster), RTX 4090 (Beowulf cluster), and A100 (supercomputer). **[B]** Speedup relative to the number of GPUs. The speedup of all GPU types is calculated using the execution time of a single RTX A6000 GPU as the baseline (1X). NACHOS achieved linear speedup on the A6000 GPUs and super-linear speedup on the RTX 4090 GPUs and the A100 GPUs.