ASCS ADAPTIVE & SECURE COMPUTING SYSTEMS LABORATORY

# Towards Generalizable Models of I/O Throughput

Mihailo Isakov, Eliakin del Rosario, Michel A. Kinsy

Adaptive and Secure Computing Systems (ASCS) Laboratory

Texas A&M University

Sandeep Madireddy, Prasanna Balaprakash, Philip Carns, Robert B. Ross

Mathematics and Computer Science Division

Argonne National Laboratory

ELECTRICAL & COMPUTER ENGINEERING TEXAS A&M UNIVERSITY

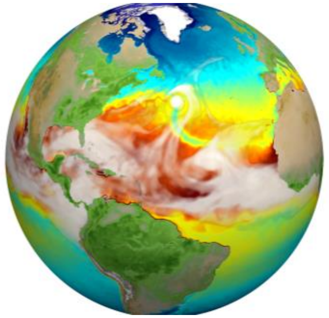
Argonne NATIONAL LABORATORY

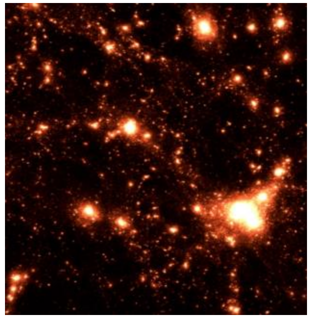
# Presentation Outline

- **Modelling HPC I/O using machine learning (ML)**
- Diagnosing lack of ML model generalization
- Robust test set generation
- Limits of I/O throughput prediction
- Increasing prediction accuracy on out-of-sample HPC jobs
- Conclusion

# Accelerating Scientific Workloads

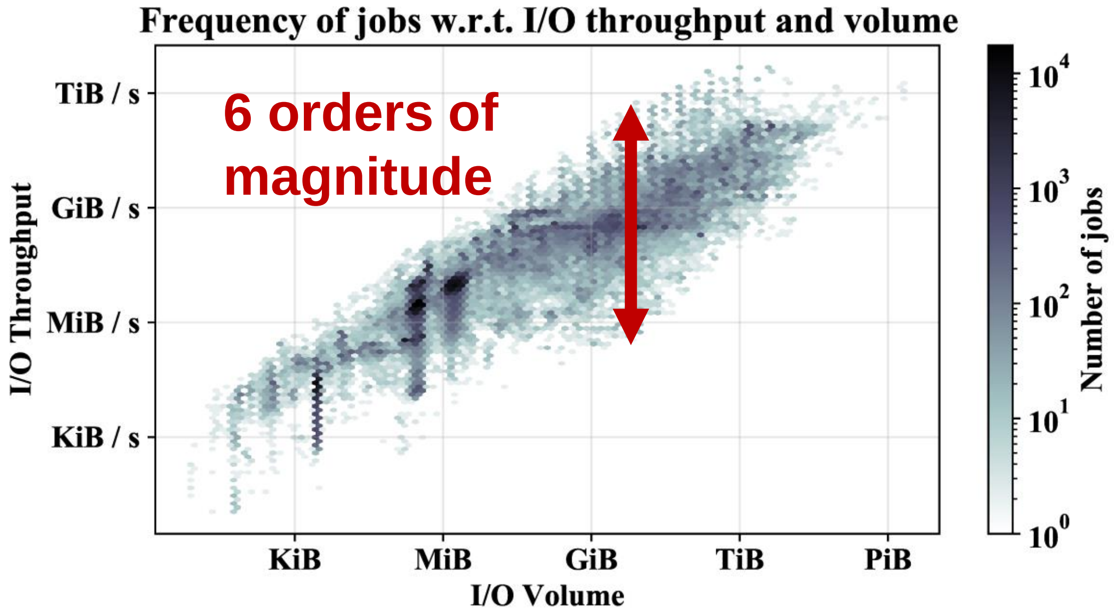
Climate science [1] Cosmology [2]





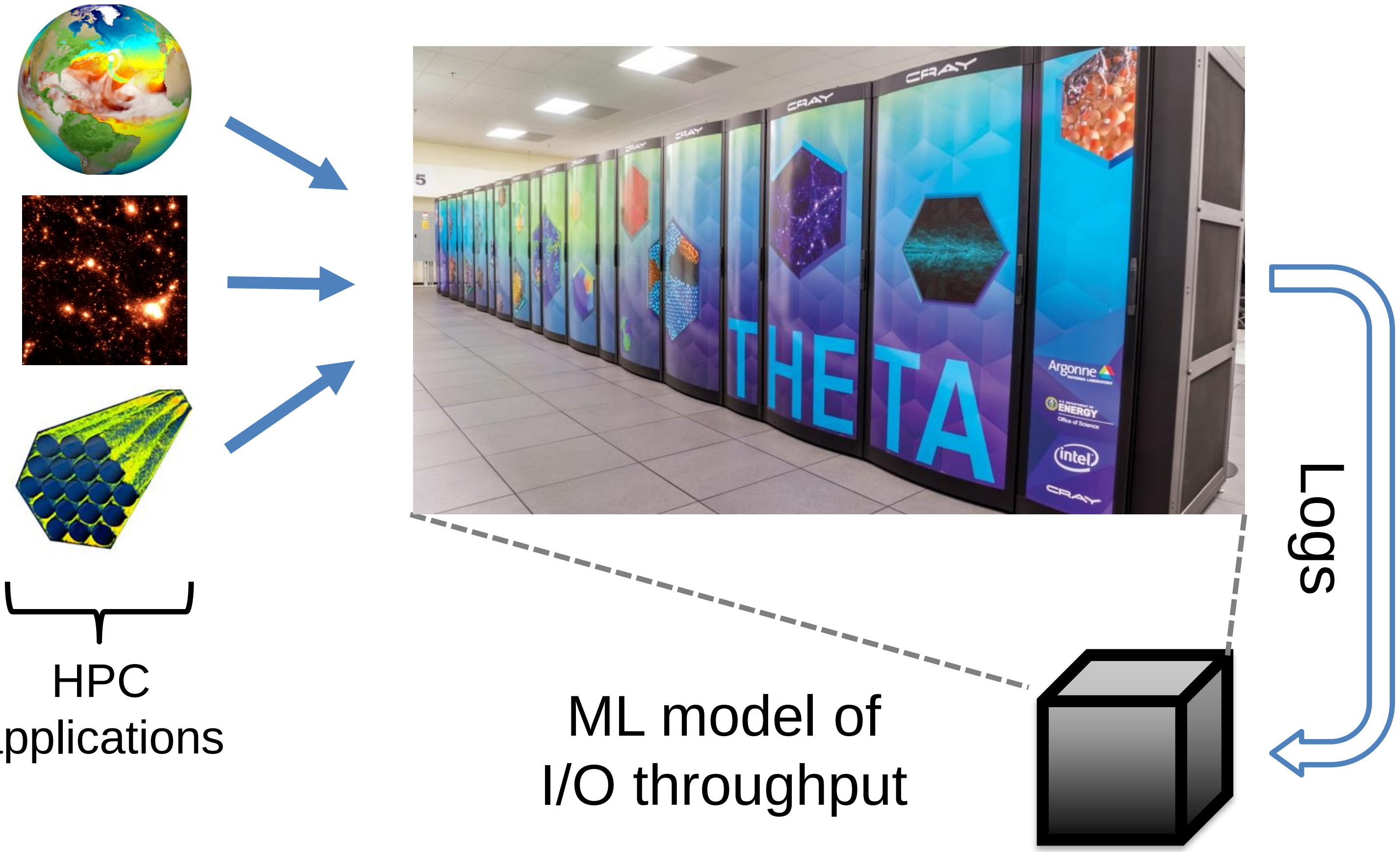

- Complex, general-purpose system
- Many diverse co-located workloads
- Shared hardware, memory & I/O bandwidth
- Debugging performance bottlenecks is hard!

- I/O problems can cause 10-100× degradation in performance
- Some jobs are very susceptible to I/O contention
- Debugging I/O performance issues is hard: the problem can hide in any of the layers!



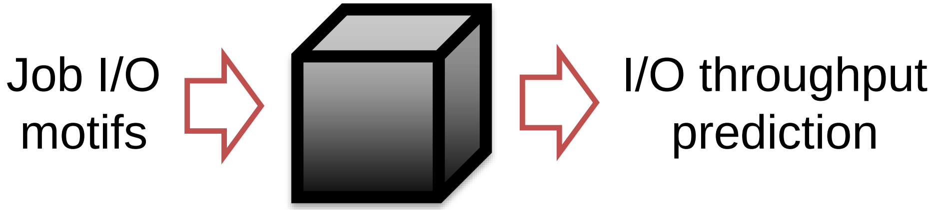

[1] Image by Mat Maltrud / Los Alamos National Laboratory
[2] John Spizzirri, Cartography of the cosmos

# Modelling an HPC System Using ML



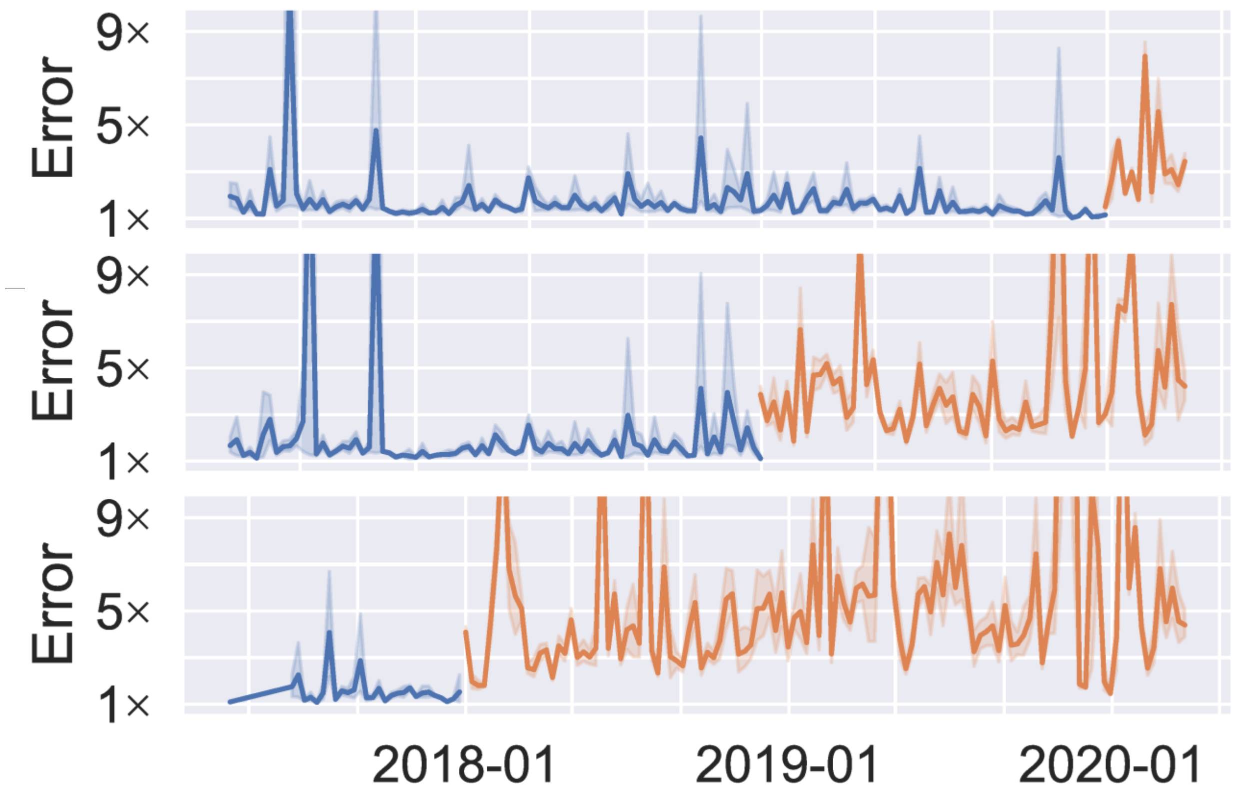

Many reasons why we want ML models of HPC systems:

- Can use them to predict runtime or I/O throughput of future jobs
- Can use them as an “early warning system” for wasteful jobs
- Can help better schedule jobs that are e.g., sensitive to I/O contention or that negatively impact other jobs
- **Can interpret the model to better understand the HPC system**



# Real-World Usage of I/O Throughput Models

- Our models are trained on data from 2017 to 2020
  - Dataset split 80/20 into training and test sets
- We evaluate our models on new real-world data
  - Collected after training has ended
  - Blue line represents model errors on data test data
  - Orange line represents errors on newly collected data
- **Blue line is supposed to be representative of real-world performance – what went wrong?**
- Possible that the system or applications changed
  - Repeated experiments at different cutoffs show this is not the case





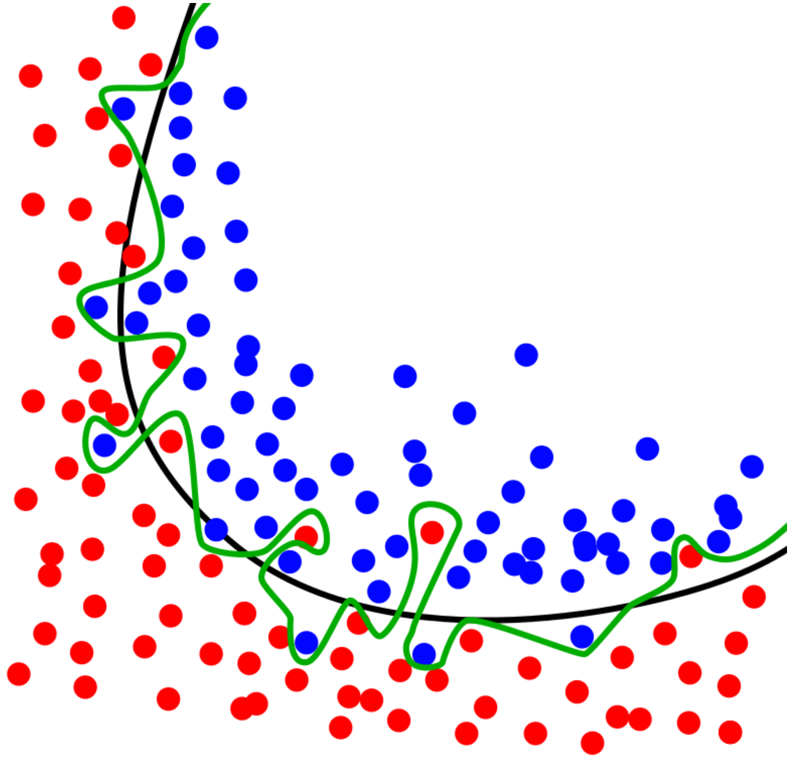

For more information about modelling HPC systems, check out our SC20 paper “HPC I/O Throughput Bottleneck Analysis with Explainable Local Models”

# Presentation Outline

- Modelling HPC I/O using machine learning (ML)
- **Diagnosing lack of ML model generalization**
- Robust test set generation
- Limits of I/O throughput prediction
- Increasing prediction accuracy on out-of-sample HPC jobs
- Conclusion

# Diagnosing Lack of Generalization

- “Generalization refers to your model's ability to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model.” [1]
- Good accuracy on training sets but bad accuracy on real tasks hints at lack of generalization
- We *do* test on unseen data
  - Our test set is built specifically for this purpose
  - But it doesn’t seem to work!



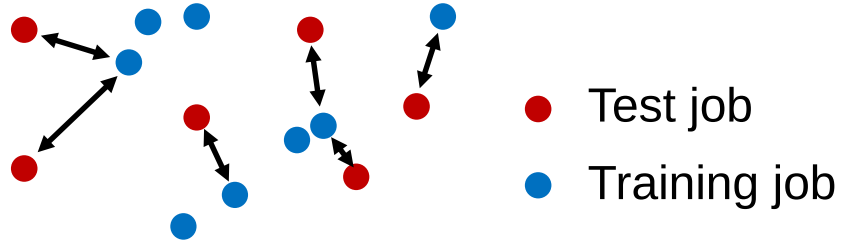
[2]

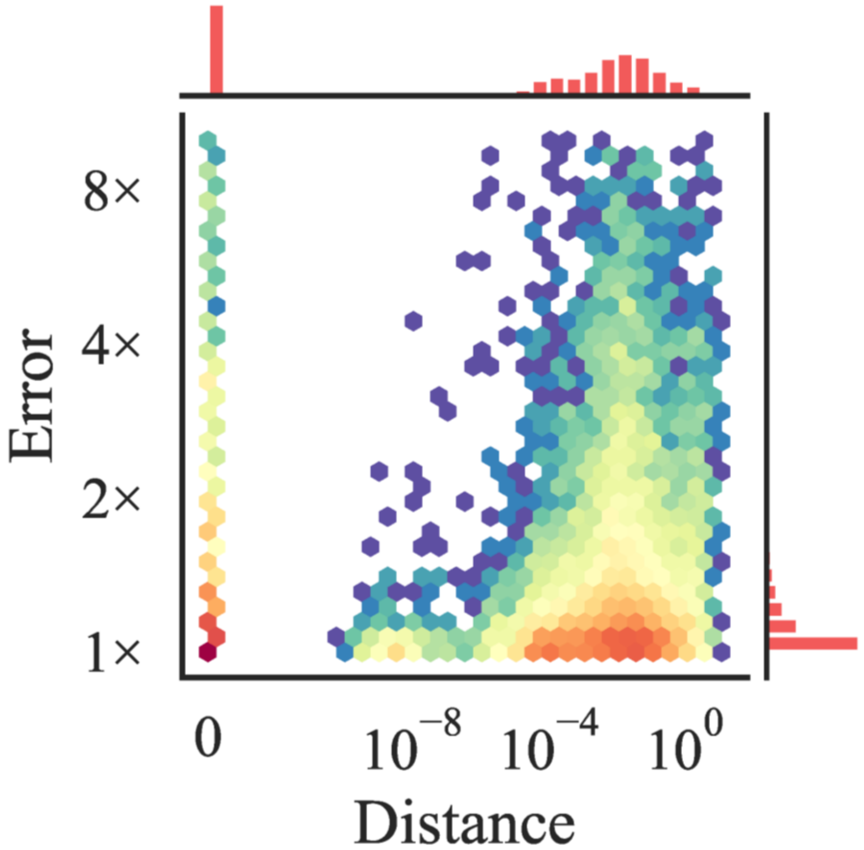


[1] https://developers.google.com/machine-learning/crash-course/generalization/video-lecture
[2] https://en.wikipedia.org/wiki/Overfitting

# Training-Test Set Distances

- **Hypothesis: our test set doesn't work because it is too similar to the training set**
- We measure the nearest neighbor distance between pairs of jobs where one job is in the training set, and the other is in the test set
  - Figure on the right shows a 2D histogram of training-test nearest neighbor distances & I/O throughput diff.
- Some conclusions:
  - Very similar nearest neighbors in the training set
  - Plenty of jobs have identical neighbors (distance = 0)
  - Nearest neighbor predictions are surprisingly good?



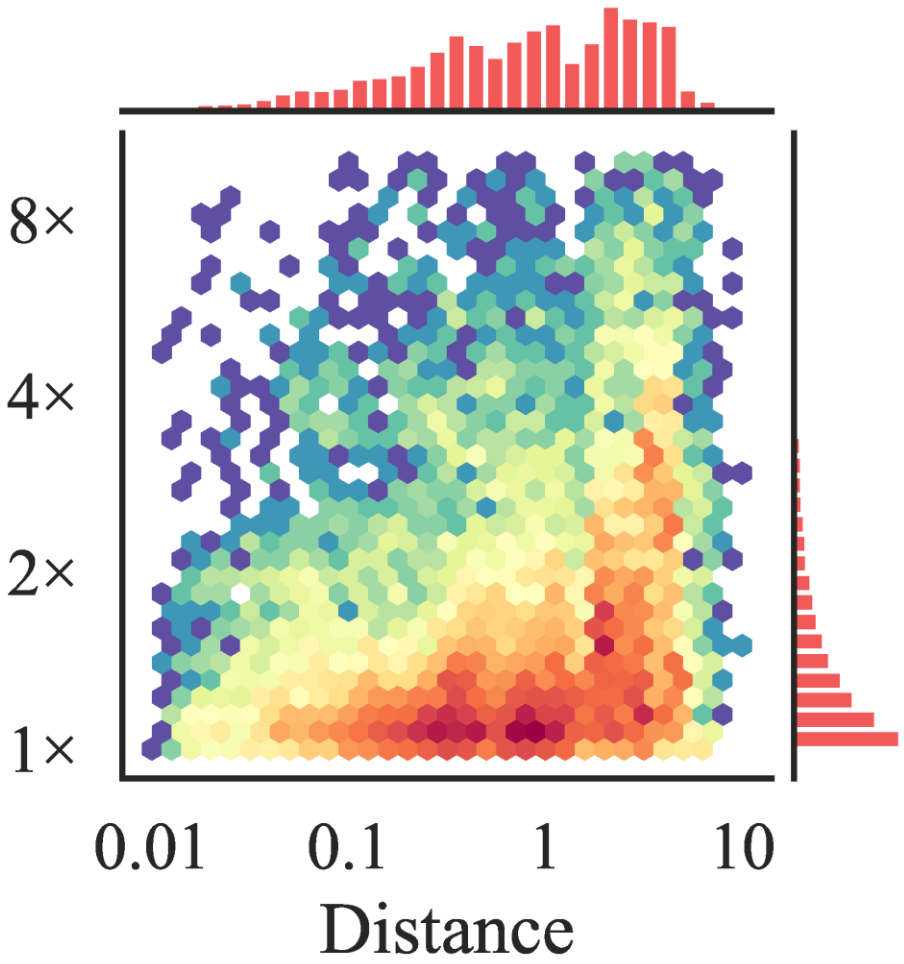
# Presentation Outline

- Modelling HPC I/O using machine learning (ML)
- Diagnosing lack of ML model generalization
- **Robust test set generation**
- Limits of I/O throughput prediction
- Increasing prediction accuracy on out-of-sample HPC jobs
- Conclusion

# Robust Test Sets

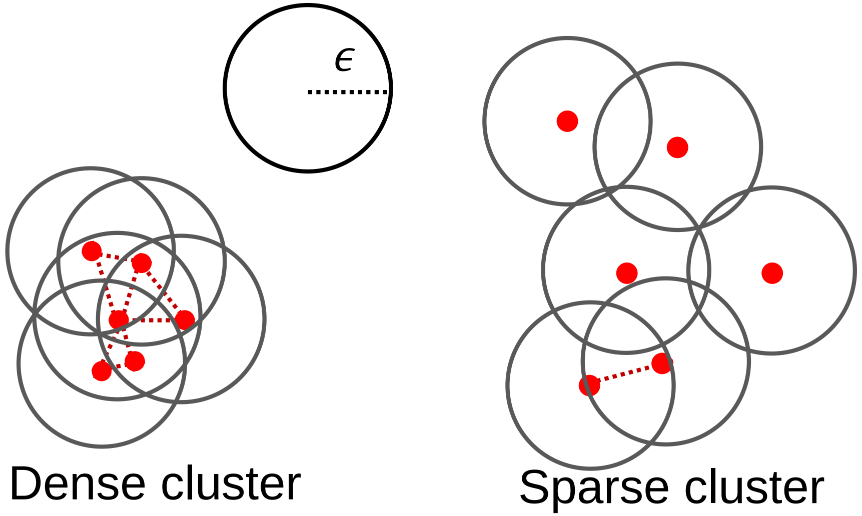
- How can we build test sets that enforce greater separation from the training set?
- **Idea: hold-out all jobs of a single application to test generalization**
- On the right we see the training-test nearest neighbor distribution for a held out climate application
- Problems with holding out apps:
  - Some apps are a lot harder to predict than others
  - Can't try each one – we have 600+ apps

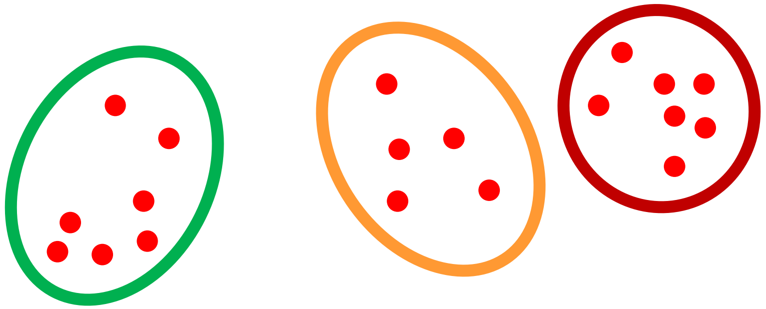
Climate application



# DBSCAN-based Test Sets

- DBSCAN is an agglomerative clustering method that iteratively groups together points or clusters closer than some $\epsilon$ distance
- Benefits:
  - We can guarantee minimum distance of $\epsilon$ between the training and test sets
- We can use DBSCAN to cluster the dataset, and hold-out some set of clusters at random
- Problems:
  - Some clusters are a lot harder to predict than others
- We solve this by adapting K-fold crossvalidation:
  - We split clusters into $n$ groups of about the same size
  - Each group of clusters acts as a test set once
  - We have to train and evaluate $n$ models

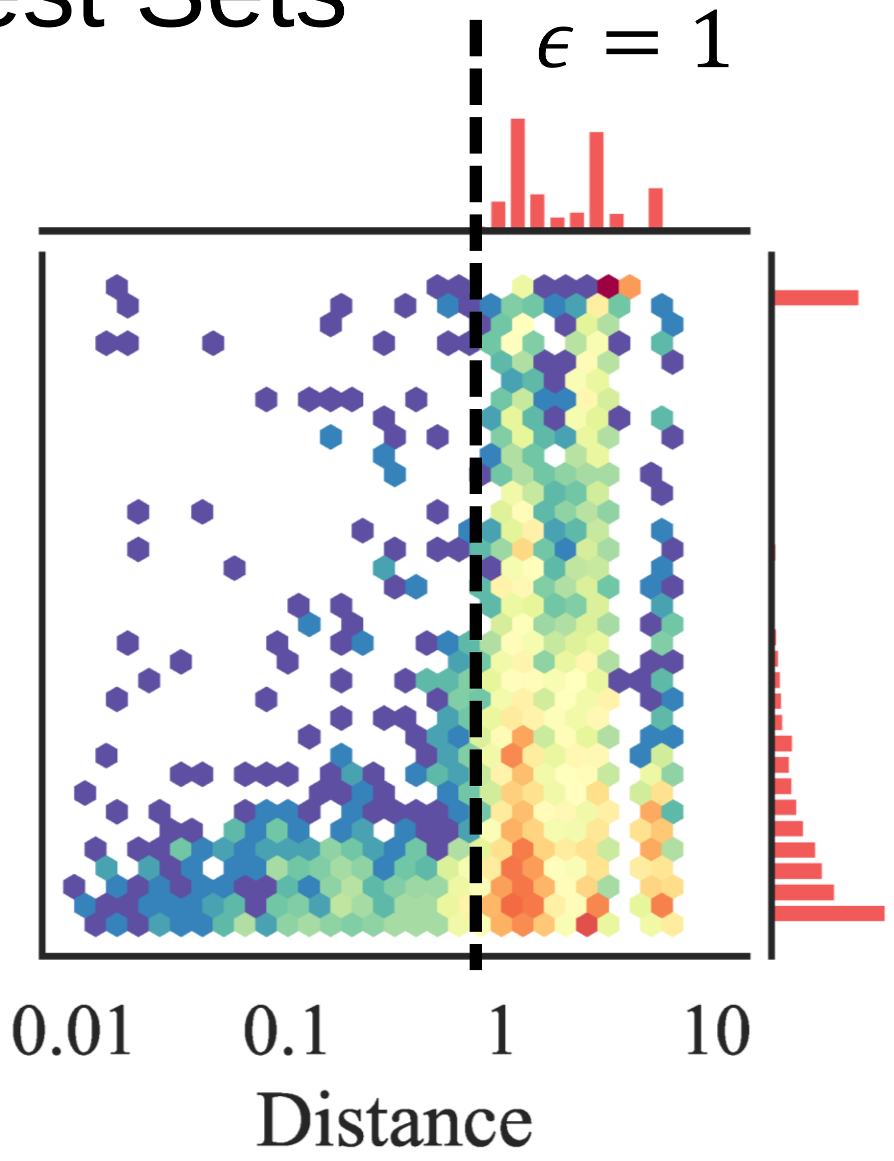


Which cluster ends up in the test set strongly affects ML model test accuracy!

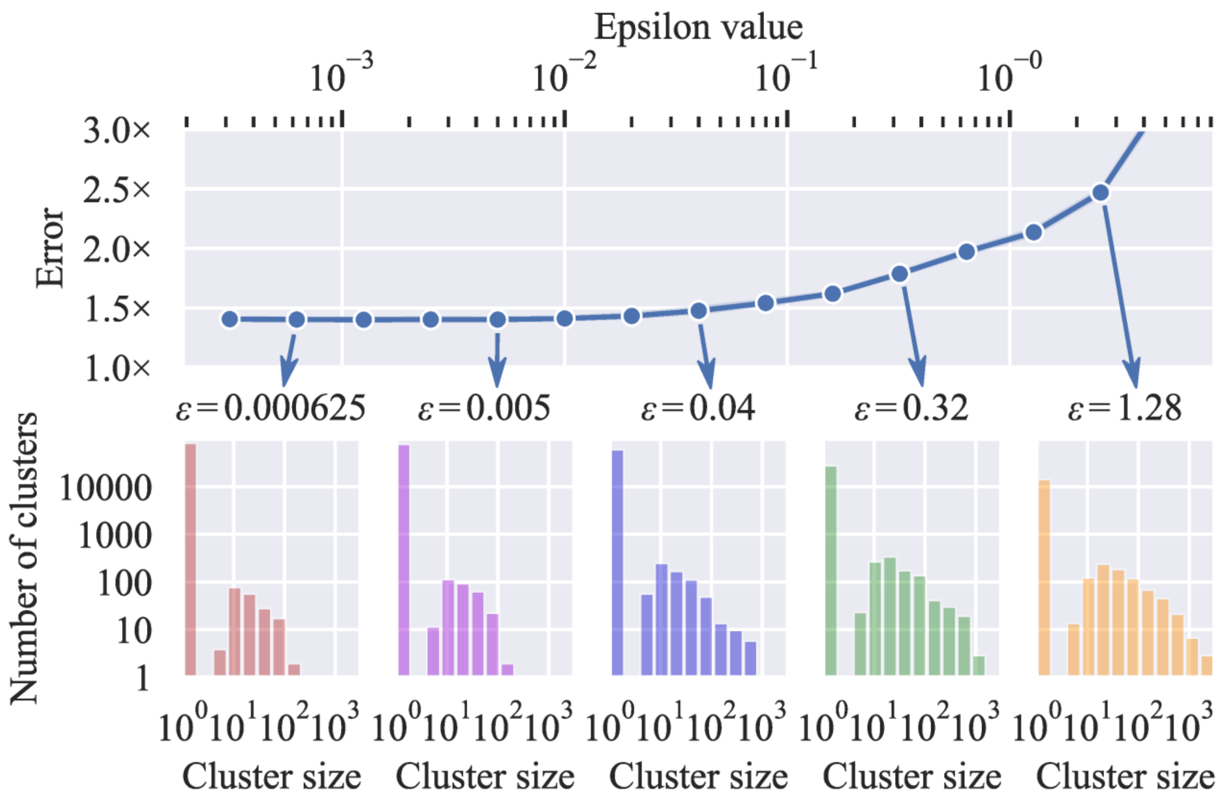
# DBSCAN-based Test Sets

- DBSCAN is an agglomerative clustering method that iteratively groups together points or clusters closer than some $\epsilon$ distance
- Benefits:
  - We can guarantee minimum distance of $\epsilon$ between the training and test sets
- We can use DBSCAN to cluster the dataset, and hold-out some set of clusters at random
- Problems:
  - Some clusters are a lot harder to predict than others
- We solve this by adapting K-fold crossvalidation:
  - We split clusters into *n* groups of about the same size
  - Each group of clusters acts as a test set once
  - We have to train and evaluate *n* models



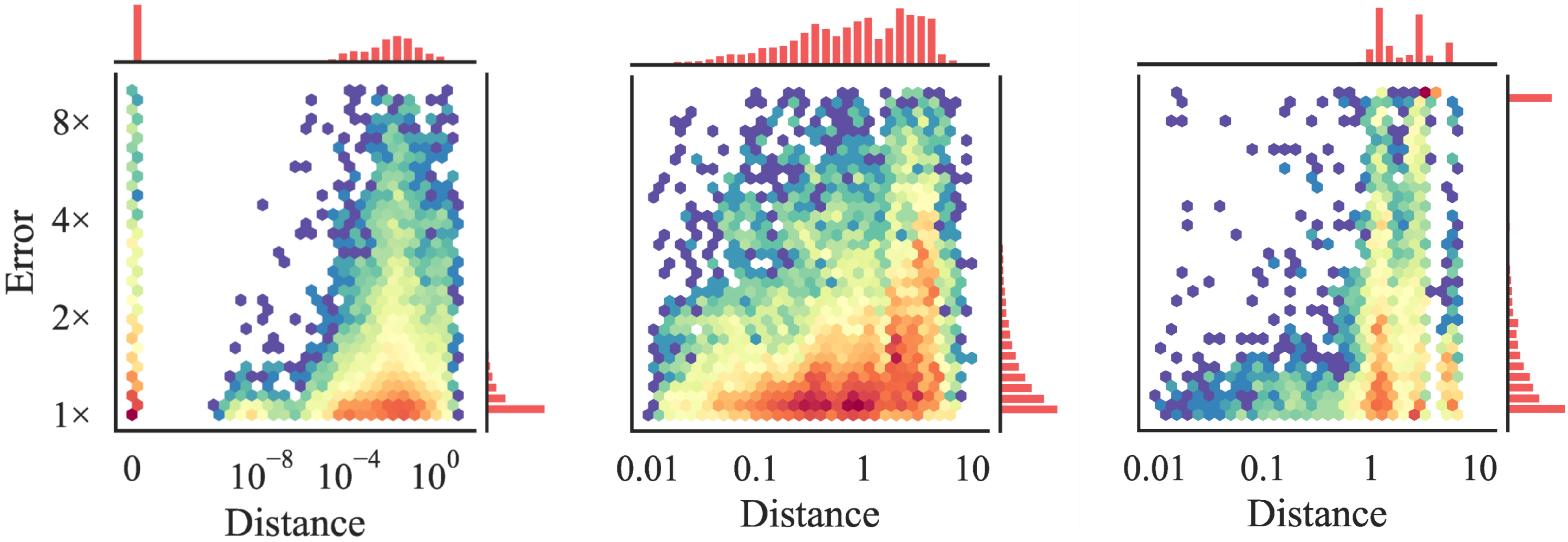
# Optimizing Models for Exploitation vs. Generalization

- We still have to select the $\epsilon$ value!
- For $\epsilon \rightarrow 0$, the DBSCAN-based test set approximates the random one
  - Good for testing how model will perform on previously seen data
- For large $\epsilon$, the DBSCAN-based test set is similar to app-based ones
  - Good for testing how model will perform on completely new applications
- There is no perfect value – it is up to the user to select what the model's goal is



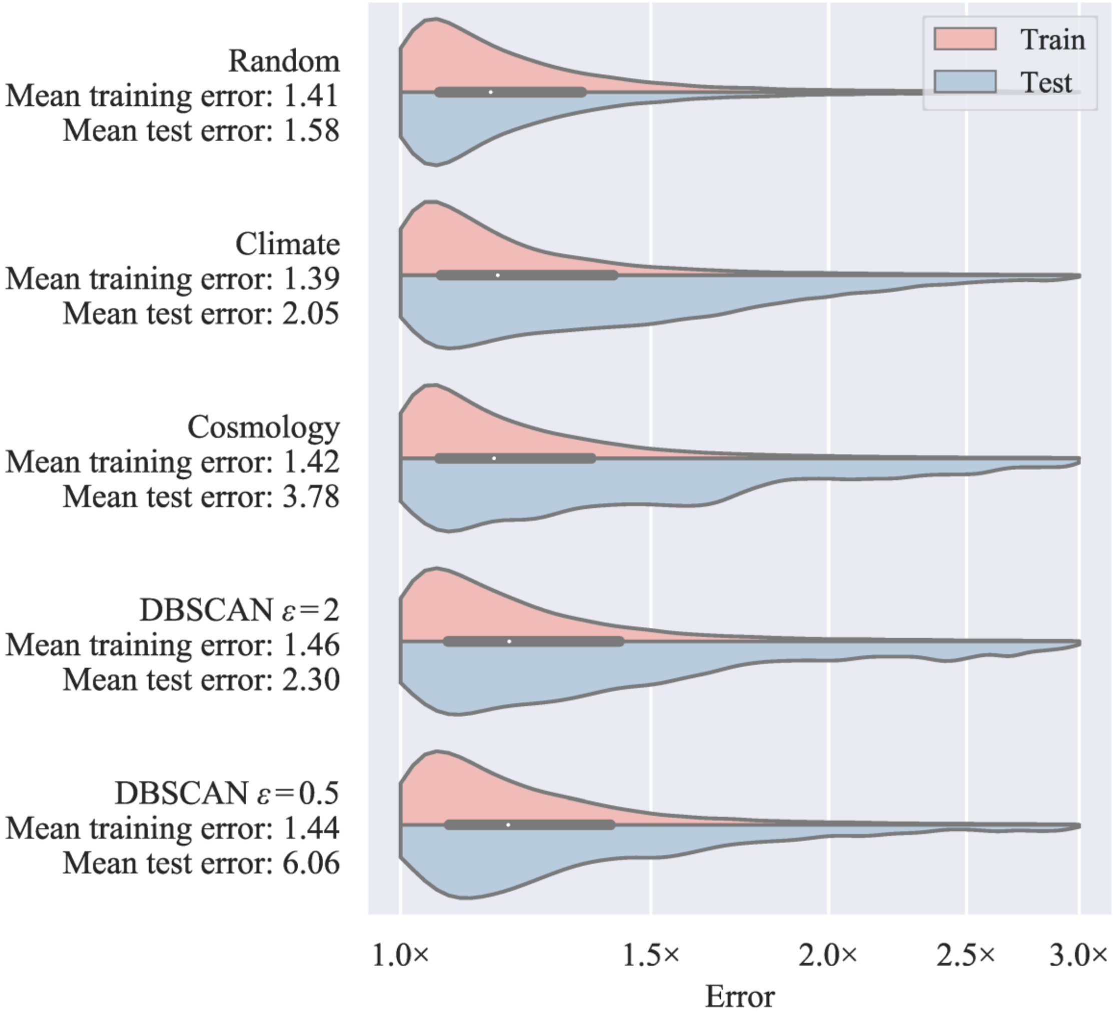
# Test Set Comparison

Randomly Selected Test Set | App-based Test Set | DBSCAN-based Test Set



# Evaluating ML Models on Each Test Set

- We train and test an ML model using each of the proposed test set generation methods:
  - Randomly split training / test set
  - Climate science / cosmology applications held out as test sets
  - DBSCAN-based test sets for $\epsilon = 2$ and $\epsilon = 0.5$
- We present both training and test error distributions
  - All training sets have similar error distributions
  - Test sets have very different distributions

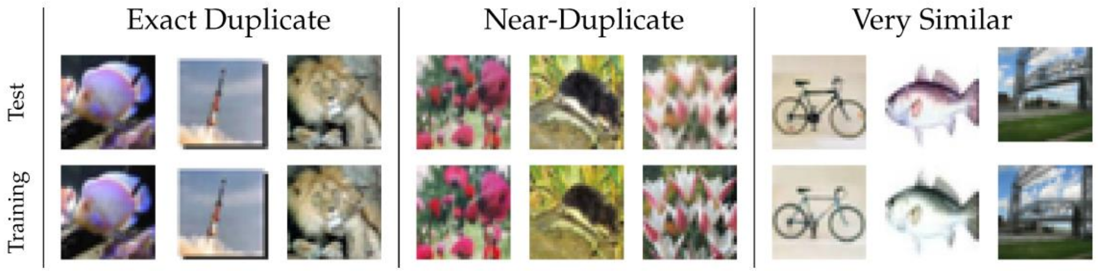


# Presentation Outline

- Modelling HPC I/O using machine learning (ML)
- Diagnosing lack of ML model generalization
- Robust test set generation
- **Limits of I/O throughput prediction**
- Increasing prediction accuracy on out-of-sample HPC jobs
- Conclusion

# Limits of I/O Throughput Prediction

- Comparing our models to those of previous works is hard:
  - Different datasets, collected at different points
    - E.g., some works have access to I/O contention logs, we don't
  - Lack of open datasets & reproducible code
  - Different goals, different metrics
- Instead of comparing our I/O throughput prediction models to some baseline, can we establish the best case scenario?
  - What is the upper bound on accuracy, given access to this data?



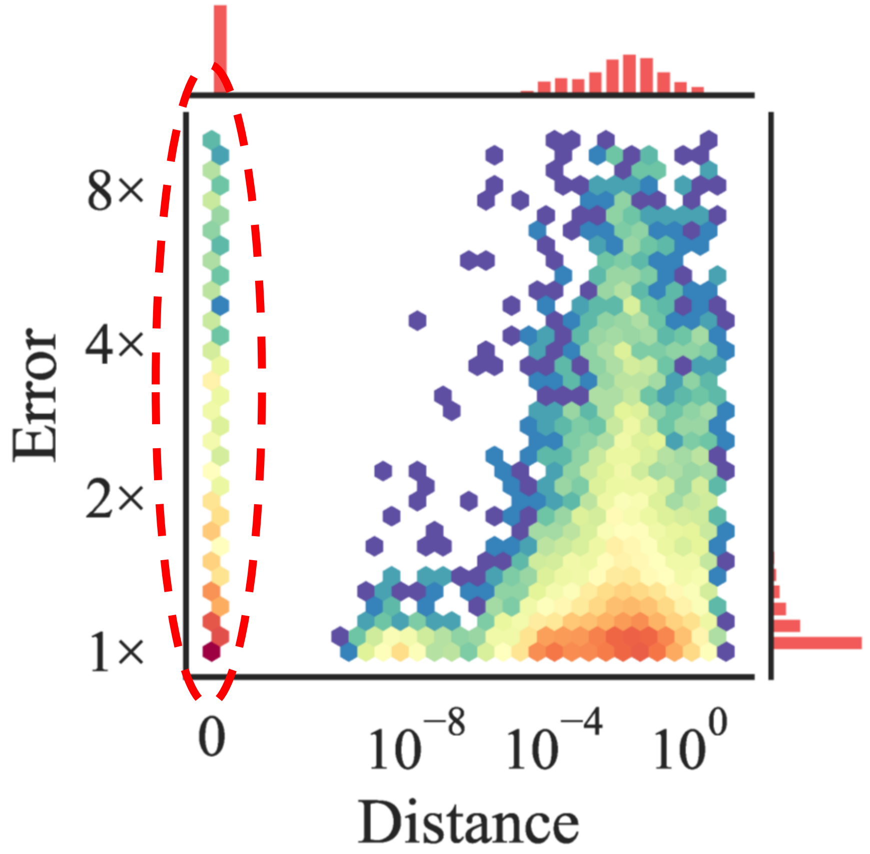

Björn Barz, Joachim Denzler, **Do We Train on Test Data? Purging CIFAR of Near-Duplicates,** *Journal of Imaging,* 2020

- If there is noise in the labels (I/O throughput measurements in our case) there is a fundamental upper bound to accuracy we can achieve when predicting I/O throughput
  - We simply can't predict noise

# Using Duplicate Jobs to Probe I/O Contention

- Duplicate jobs are jobs with identical input features:
  - Same number of bytes, files, accesses, same I/O access patterns, etc.
  - Typically runs of the same application, on data of the same size & format
- Duplicate jobs differ on system-sensitive features:
  - Runtime, I/O throughput, file open & close timestamps
- We've already seen duplicate jobs!
- Duplicate jobs look identical to our ML models:
  - The only thing that changes is the target output (I/O throughput)
  - Since duplicates are identical, we can't predict better than average
- ML models can typically achieve 100% accuracy on the training set
  - That is assuming that there are no inconsistent samples (e.g., identical jobs with different I/O throughputs)
- We use duplicate jobs to estimate the best possible (training set) accuracy achievable

Training – test set distances for a randomly selected test set

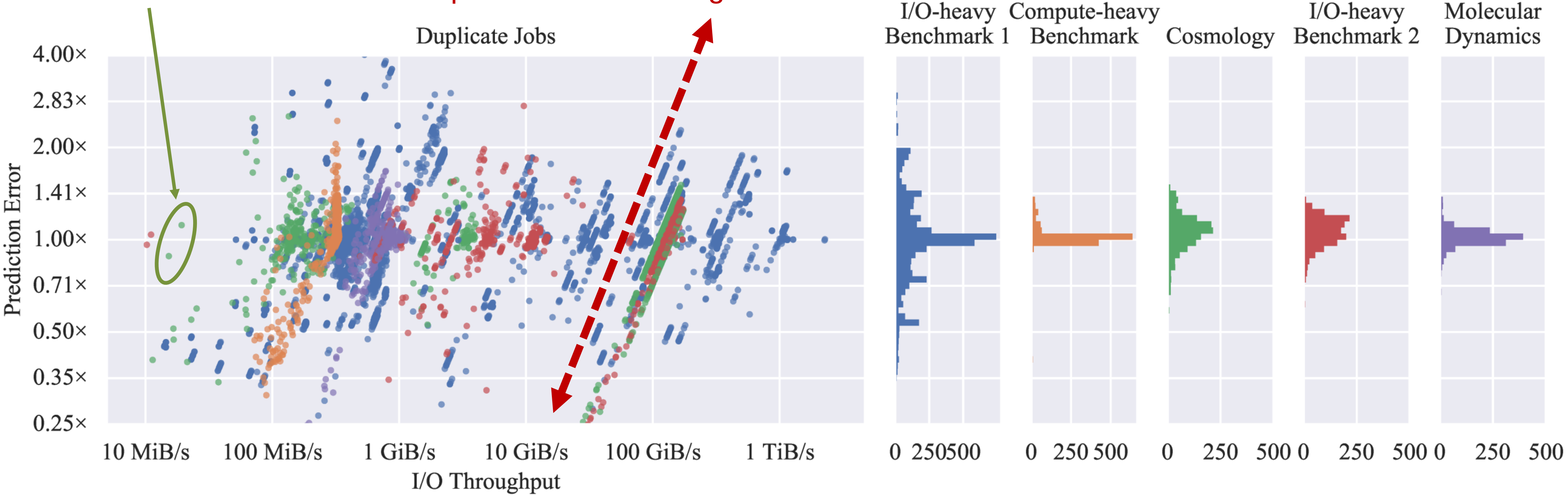


# Using Duplicate Jobs to Probe I/O Contention

Pair of duplicate jobs

Faster jobs have both higher I/O throughput & larger prediction error, so duplicates lie on a diagonal

I/O-intensive applications' duplicates can vary by 4x in I/O throughput




Less I/O-intensive applications have less variance

# Using Duplicates to Estimate Best-Case Accuracy

- Predicting I/O throughput of duplicate jobs is easy
  - Given an input, take the average I/O throughput of all other duplicates you have
- We can use duplicates to estimate the upper bound on accuracy
  - We use k-nearest neighbors (kNN) to predict I/O throughput of non-duplicate jobs, and compare results to duplicate predictions

| Type | Duplicates | $k = 1$ | $k = 2$ | $k = 5$ | $k = 10$ | $k = 20$ |
|---|---|---|---|---|---|---|
| $R^2$ | 0.974 | 0.966 | 0.972 | 0.973 | 0.970 | 0.967 |

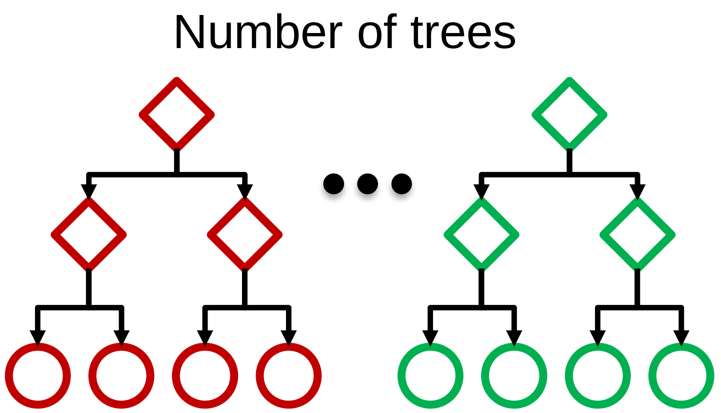
- We see that $R^2$ of 0.974 is as far as we could push our models

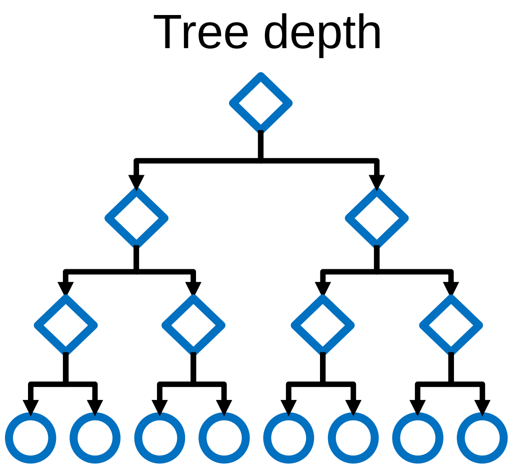
# Presentation Outline

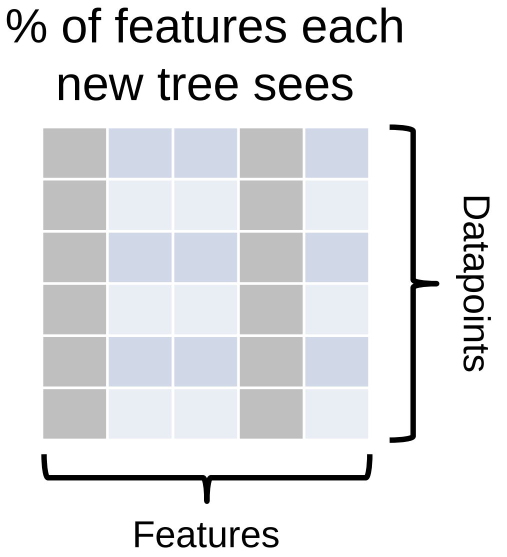
- Modelling HPC I/O using machine learning (ML)
- Diagnosing lack of ML model generalization
- Robust test set generation
- Limits of I/O throughput prediction
- **Increasing prediction accuracy on out-of-sample HPC jobs**
- Conclusion

# Increasing Prediction Accuracy on Out-of-Sample HPC jobs

- We now have both test sets that can reveal generalization (or lack of thereof), as well as an estimate of best-case accuracy
- We now metaoptimize our ML models on DBSCAN & random test sets:
- We metaoptimize XGBoost gradient boosting trees on 4 parameters
  - We evaluate 240 different configurations, each on two test sets

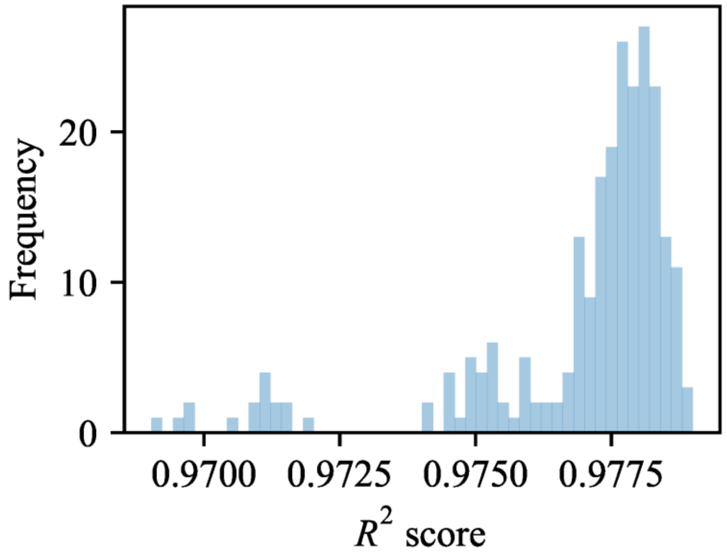


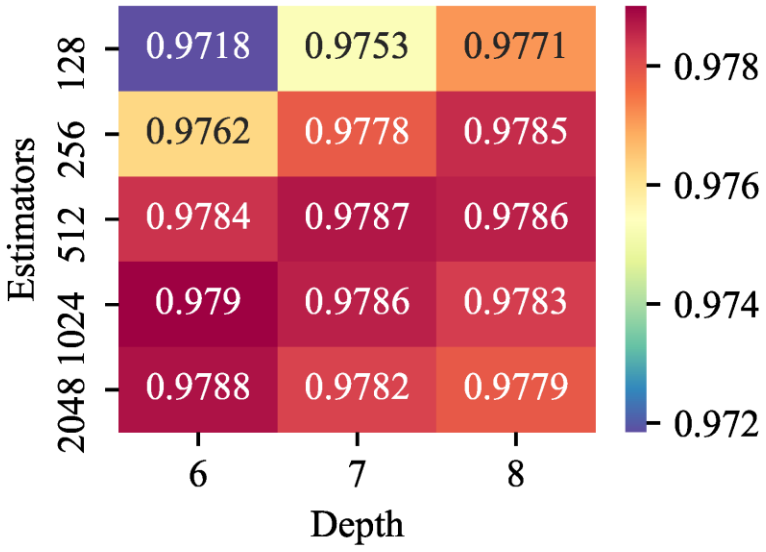



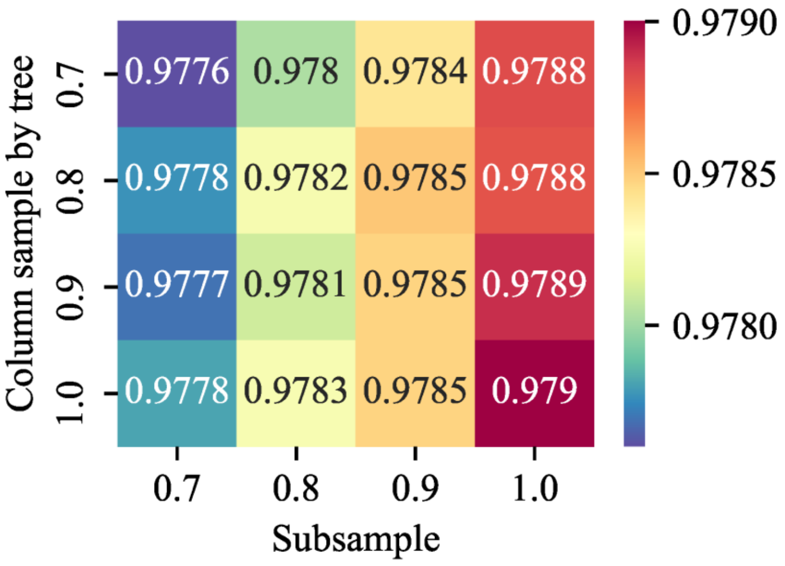




# Random Test Set Grid Search

- Relatively small $R^2$ variance (0.97 – 0.98)
- More capacity (either number of trees, or tree depth) is better
- Trees perform better when they can see all features & datapoints

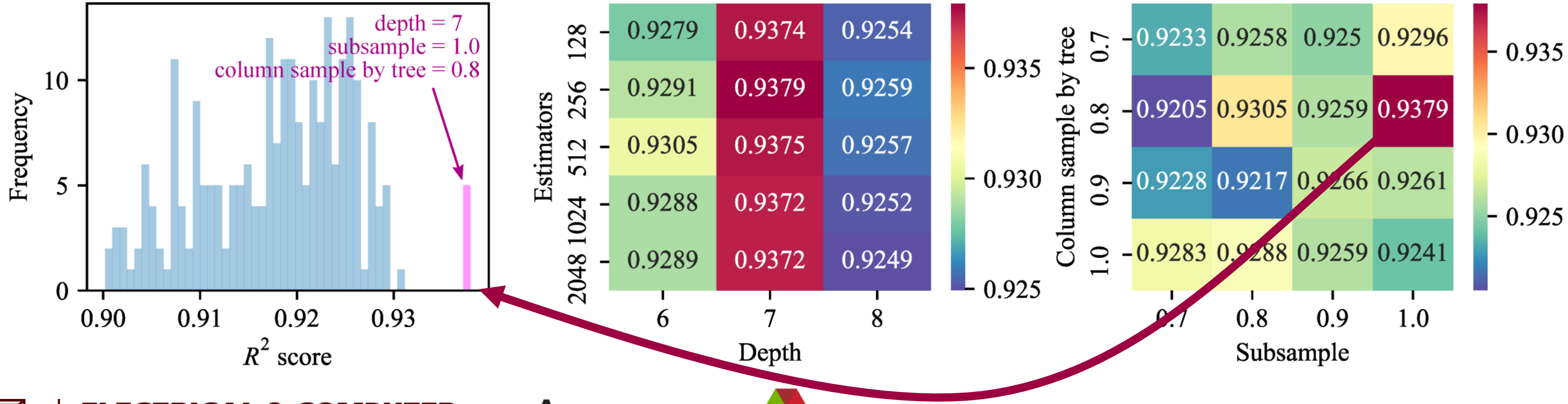


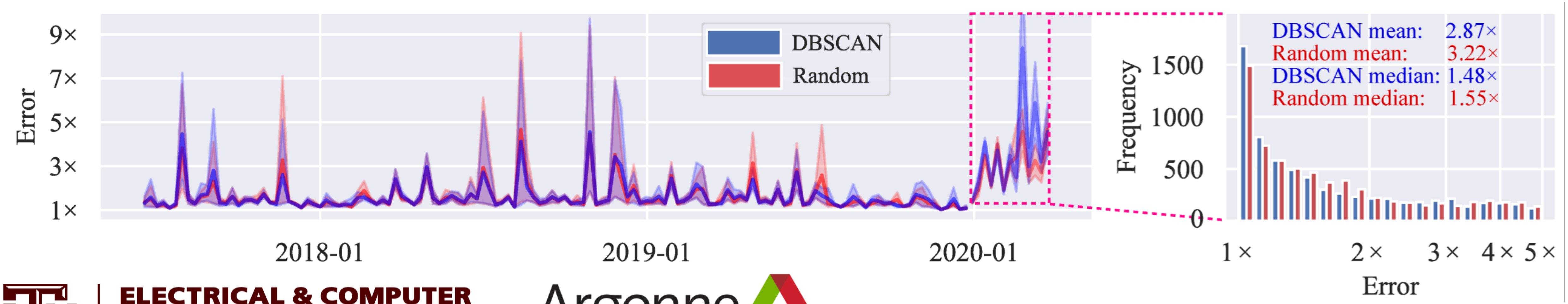
# DBSCAN-based Test Set Grid Search

- Far greater, but also lower $R^2$ range (0.90 – 0.94)
  - Actually makes sense to metaoptimize – we can discriminate between experiments
- $R^2$ histogram reveals a set of configurations much better than avg.
- Models very sensitive to depth! Depth of 7 better than either 6 or 8
  - No longer encouraged to overfit, so more capacity is not always better?
- A specific configuration of sampling params (1, 0.8) yields best results



# Evaluating Models in Production

- Let's evaluate a model with the best metaparameters on real-world data
- We compare two models:
  - One metaoptimized on the randomly-sampled test set
  - One metaoptimized on a DBSCAN-based test set
- We plot the error distribution on the right
  - The DBSCAN model achieves 11% lower mean and 5.5% lower median error

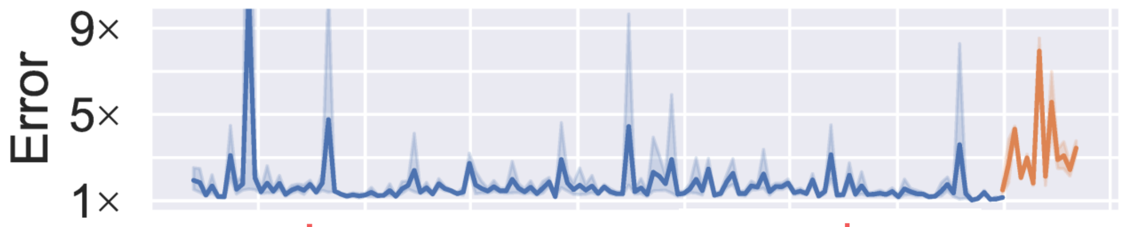


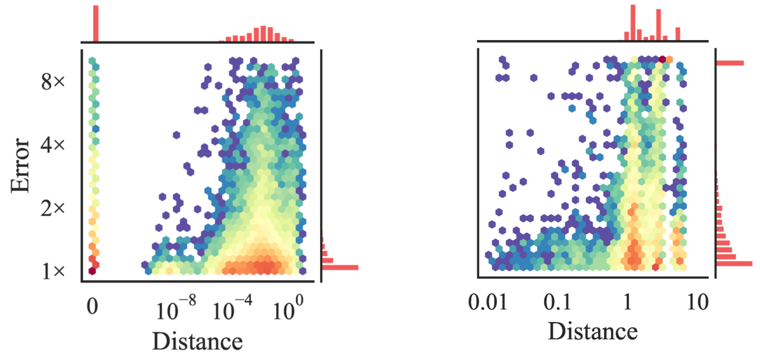
# Presentation Outline

- Modelling HPC I/O using machine learning (ML)
- Diagnosing lack of ML model generalization
- Robust test set generation
- Limits of I/O throughput prediction
- Increasing prediction accuracy on out-of-sample HPC jobs
- **Conclusion**

# Conclusion

- In this work we:
  - Presented difficulties in deploying I/O throughput prediction models
  - Diagnosed training-test set similarity as the cause of the problem
  - Proposed a DBSCAN-based test set generation method
  - Estimated the upper bound on I/O throughput prediction accuracy
  - Showed that using the new test sets, we can better meta-optimize models




| Type | Duplicates | $k = 1$ | $k = 2$ | $k = 5$ | $k = 10$ | $k = 20$ |
|---|---|---|---|---|---|---|
| $R^2$ | 0.974 | 0.966 | 0.972 | 0.973 | 0.970 | 0.967 |