Transfer Learning Workflow for High-Quality I/O Bandwidth Prediction with Limited Data

Dmytro Povaliaiev*, Radita Liem*, Julian Kunkel‡, Jay Lofstead+, Philip Carns§

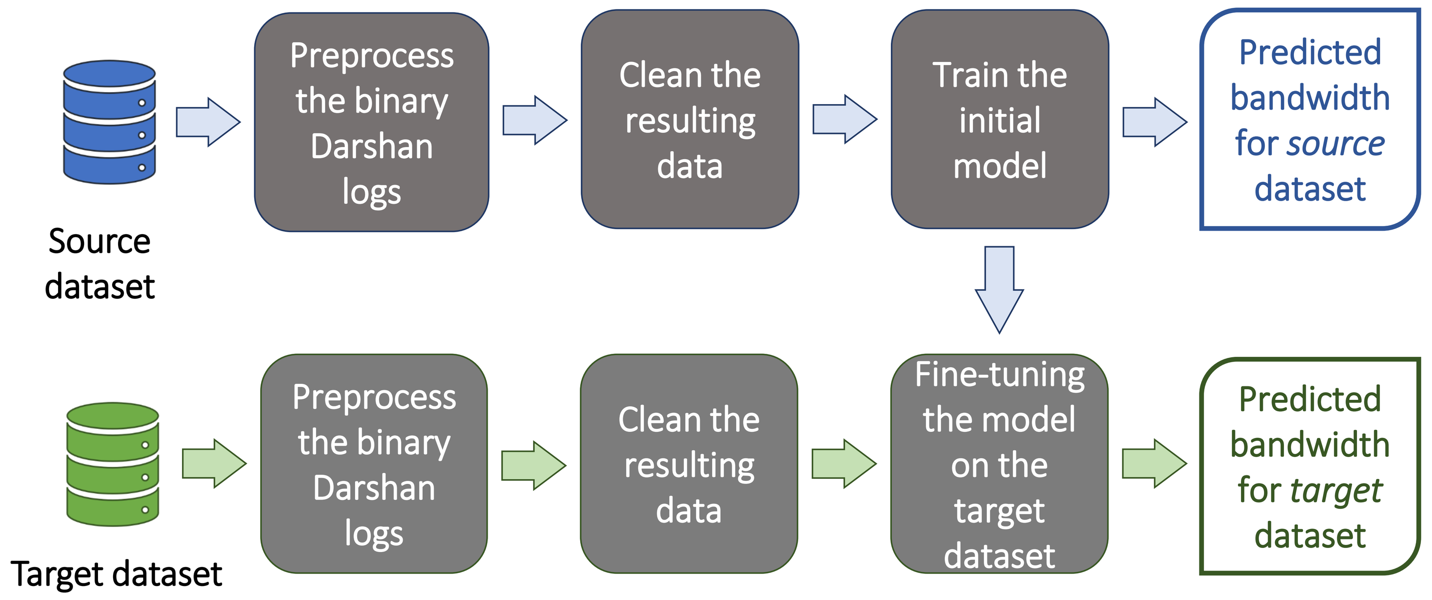
*RWTH Aachen University, ‡ Göttingen University/GDWG, + Sandia National Lab, § Argonne National Lab

Motivation

The I/O performance of a scientific application is difficult to predict due to multiple intertwined variables coming from the hardware, middleware, and the application layer[1]. This makes predicting I/O performance a good candidate problem for machine learning due to the complex relationships of the variables involved.

However, making a high-quality prediction requires a large amount of high-quality data, and collecting it is a big challenge for most data centers. Comprehensive I/O performance data from various types of applications can take years to gather and is rarely done in practice by small to medium data centers due to their limited resources.

To answer this problem, we propose to apply transfer learning to perform the I/O prediction. We use publicly available Darshan[2] logs from the Blue Waters cluster operated by NCSA from 2012-2021[3] for predicting I/O bandwidth of two target clusters: CLAIX18 from RWTH Aachen University and Theta from Argonne National Lab, using as few as <1% of the number of records compared to the Blue Waters.



Experiment Setup, Data & Target Prediction

Experiment Setup

The Darshan binary logs were processed on one node of CLAIX18 (2 Intel Skylake with 2.1 GHz and 48 cores in total and 192 GB of memory). For training of the deep learning models, we used the CLAIX16 GPU partition (NVIDIA P100-SXM2 16 GB GPU with 1 Intel Broadwell 2.2 GHz and 12 cores and 64 GB memory).

Data Source & Target Prediction

We use around 680,000 Darshan v3.21+ logs with POSIX records from Blue Waters for the initial training. Our target dataset for transfer learning is around 1,300 Darshan logs from scientific applications and benchmarks running on CLAIX18 and a random selection of around 60,000 Darshan logs collected at ALCF Theta.

Since we want to verify that the model is looking at the right thing, we calculate the bandwidth ourselves according to the formula used by Darshan[2] as follows:

$$MiB/s = \left(\frac{\sum_{rank=0}^{n-1} (bytes_r + bytes_w)}{\max_{rank=0}^{n-1} (t_{md} + t_r + t_w)} \right)$$

MiB/S = Calculated Bandwidth
$bytes$ = Total POSIX bytes read (r) & write (w)
t = Total time from metadata (md), read (r), and write (w)

References:

[1] Jay Lofstead, Milo Polte, Garth Gibson, Scott Klasky, Karsten Schwan, Ron Oldfield, Matthew Wolf, and Qing Liu. 2011. Six degrees of scientific data: reading patterns for extreme scale science IO. In Proceedings of the 20th international symposium on High performance distributed computing (HPDC '11). Association for Computing Machinery, New York, NY, USA, 49–60. doi: 10.1145/1996130.1996139

[2] S. Snyder, P. Carns, K. Harms, R. Ross, G. K. Lockwood, and N. J. Wright, “Modular HPC I/O Characterization with Darshan,” in 2016 5th Workshop on Extreme-Scale Programming Tools (ESPT), Nov. 2016, pp. 9–17. doi: 10.1109/ESPT.2016.006

[3] https://bluewaters.ncsa.illinois.edu/data-sets.

[4] Katharina Benkert, Edgar Gabriel, and Michael M. Resch. 2008. Outlier detection in performance data of parallel applications. In 2008 IEEE International Symposium on Parallel and Distributed Processing. (Apr. 2008), 1–8. doi: 10.1109/IPDPS.2008.4536463.

[5] Jean-Gabriel Attali and Gilles Pagès. 1997. Approximations of Functions by a Multilayer Perceptron: a New Approach. Neural Networks, 10, 6, (Aug. 1997), 1069–1081. doi: 10.1016/S0893-6080(97)00010-5.

[6] M. Isakov *et al.*, "HPC I/O Throughput Bottleneck Analysis with Explainable Local Models," *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, Atlanta, GA, USA, 2020, pp. 1-13, doi: 10.1109/SC41405.2020.00037.

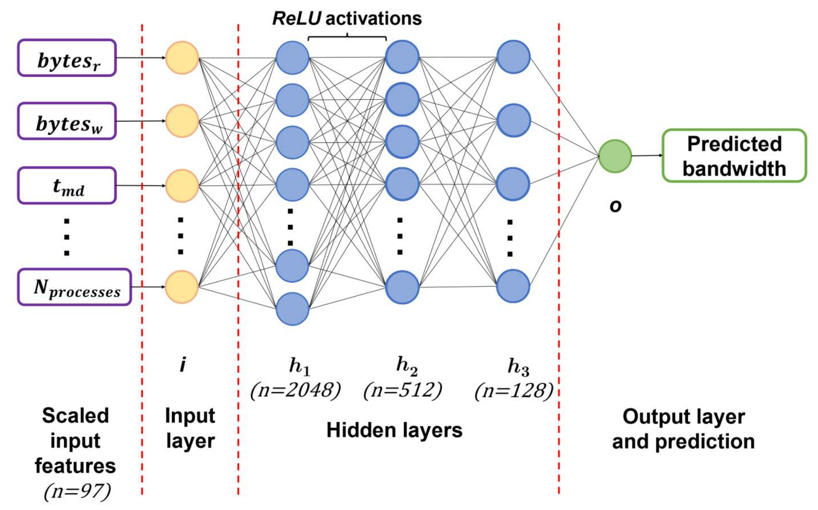
Transfer Learning Workflow

1. Data Preprocessing

We remove logs with erroneous data and drop several all-zero metrics. Bandwidth outliers are identified and eliminated using Interquartile Range (IQR) method[4]. In total we shed ~15% of the data from all datasets.

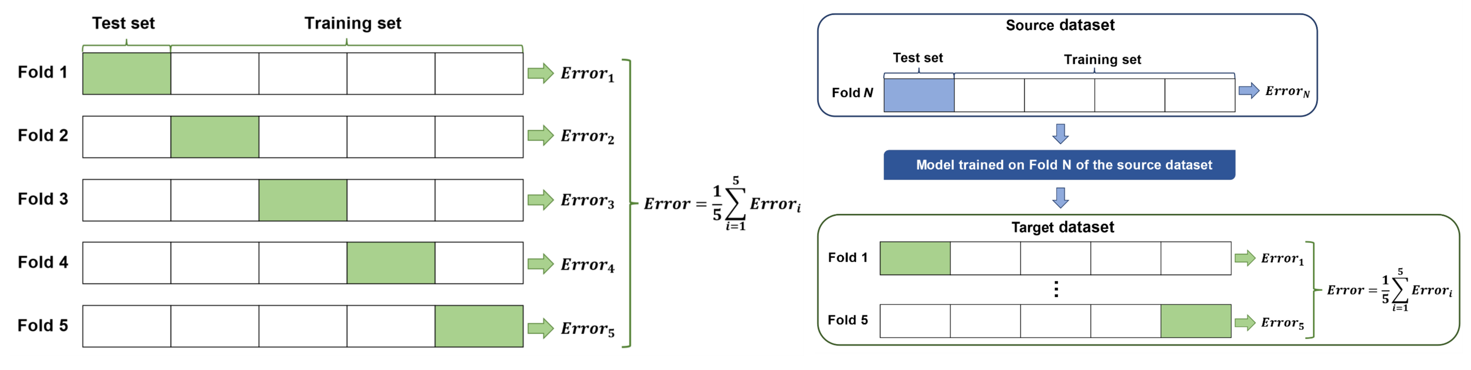
2. Building Neural Network Model

We use a Multi-Layer Perceptron[5] with 3 fully-connected hidden layers and ReLU activations. The model receives 96 Darshan POSIX counters and the # of processes as input and produces bandwidth in MB/s.



3. 5-Fold Cross-Validation During Both Stages

Our dataset is split into 5 folds. In each iteration, a different fold is used as a test set and model is trained on a merge of the other folds. We cross validate using 10 seeds resulting in 50 models in the initial state. Then, the process is repeated for each base model in transfer learning stage, producing 250 models in total

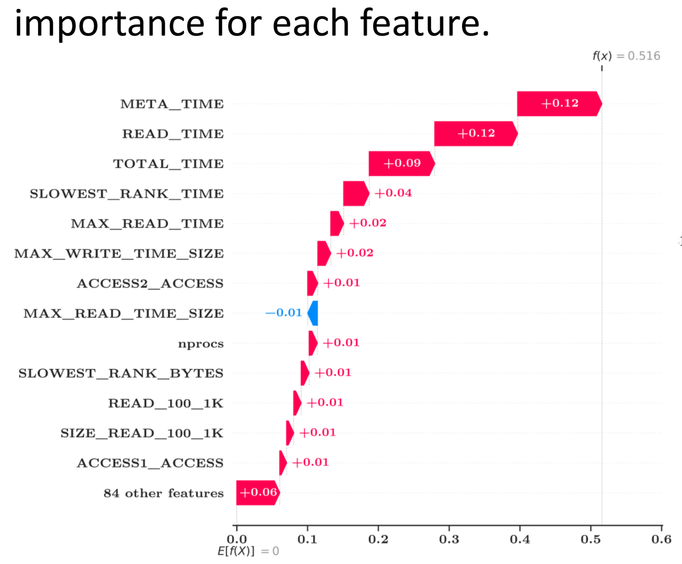


Initial Stage Cross Validation

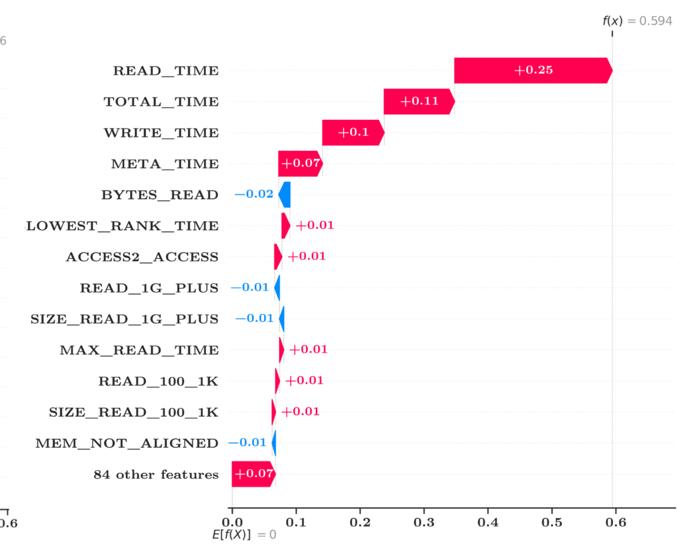
Transfer learning stage

4. Explainable AI

We use 9 approaches (Integrated Gradients, Integrated Gradients with Noise Tunnel, DeepLift, Feature Ablation, Shapley Value Sampling, Guided Propagation, Feature Permutation, InputXGrad, Saliency) and average the attributed importance for each feature.



Top features of full dataset variant

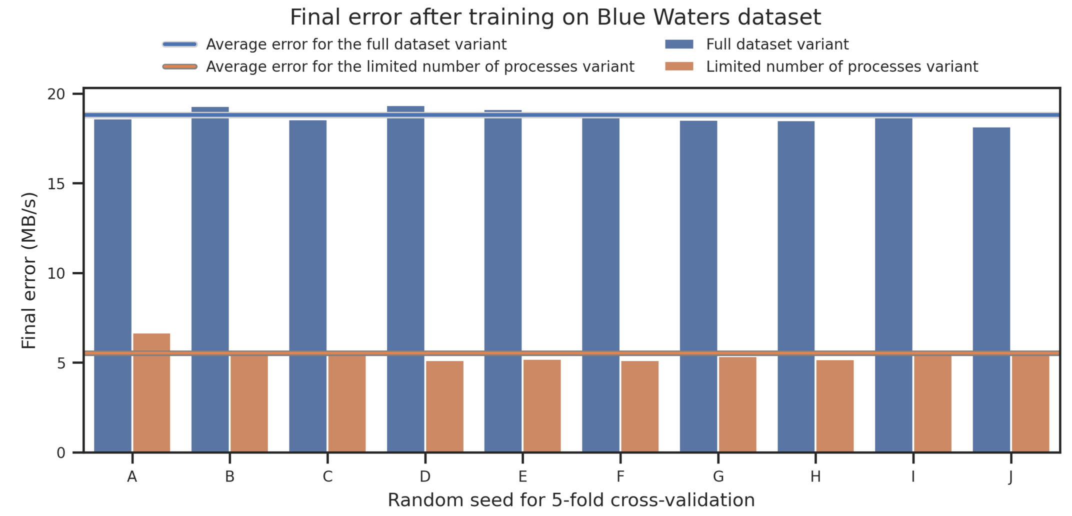


Top features of limited dataset variant

Our detailed work, references, findings, and analysis can be found in this QR code link.

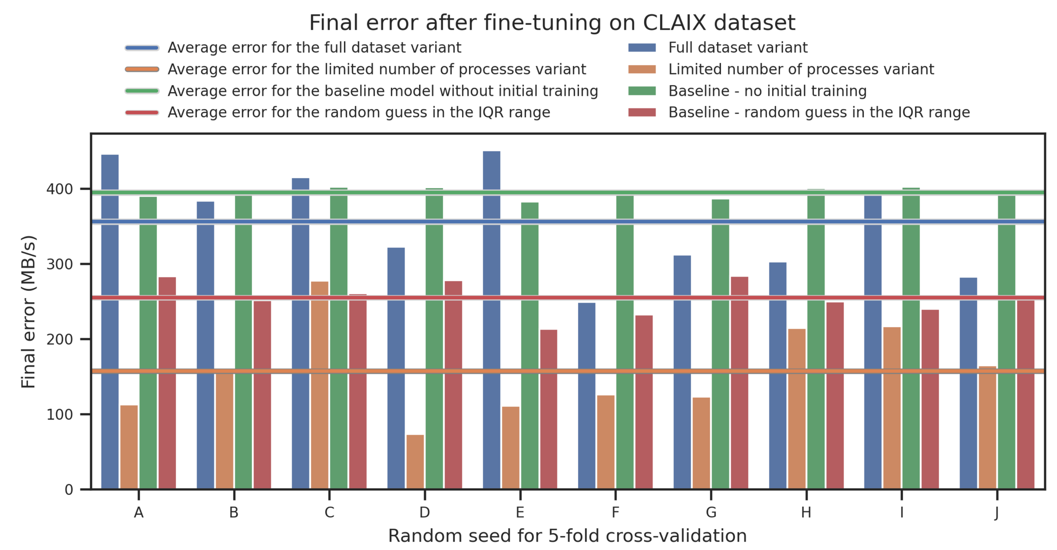
Results

Initial Training on the Blue Waters Dataset



Variant	MAE (n=50)
Full dataset	18.81 MB/s
Limited # of procs	5.53 MB/s
Random guess in IQR	155.4 MB/s

Transfer Learning on the CLAIX18 Dataset



Variant	MAE
Full dataset	355.92 MB/s
Limited # of procs	157.44 MB/s
No initial training	394.67 MB/s
Random guess in IQR	254.75 MB/s

Comparison Between Initial & Transfer Learning Stage Results

Transfer Learning on the limited number of processes performs better than the current state of the art[6], the random guess in the IQR range, and the model trained directly on the target dataset.

Variant	Initial training	Fine-tuning
Full dataset	11.6%	20.1%
Limited # of processes	3.4%	8.92%
Random guess in IQR	95.9%	14.4%
Without transfer learning (trained directly on the target dataset)	-	22.4%
Current state of the art (Isakov et al.)[6]	10%	10%

Explainable AI Outcome – The Model Found the Bandwidth Formula

Top 10 list of features with the highest impact according to the model contains the same metrics Darshan uses to calculate the bandwidth. This is an important finding since we know that the result is not a mere coincidence.

Besides learning the formula to calculate the bandwidth, it also identifies other components that can be used to deduce the bandwidth. Our future work will explore this features to deduce application’s runtime execution

$$MiB/s = \left(\frac{\sum_{rank=0}^{n-1} (bytes_r + \boxed{bytes_w})}{\boxed{\max_{rank=0}^{n-1} (t_{md} + t_r + t_w)}} \right)$$

Times:
- POSIX_F_READ_TIME
- POSIX_TOTAL_TIME
- POSIX_F_MAX_READ_TIME
- POSIX_F_META_TIME
- POSIX_F_WRITE_TIME
- POSIX_F_SLOWEST_RANK_TIME

Sizes:
- POSIX_ACCESS2_ACCESS
- POSIX_SLOWEST_RANK_BYTES
- POSIX_MAX_READ_TIME_SIZE
- POSIX_BYTES_WRITTEN

\# of processes

liem@itc.rwth-aachen.de