CHARACTERIZING REPRODUCIBILITY: RUN-TO-RUN NUMERICAL VARIATION IN A NEAR PRODUCTION EXASCALE COMPUTER

Anonymous authors
Paper under double-blind review

ABSTRACT

Reproducible training results require reproducible numerical accuracy across nodes and across time. It is useful in that context for the run-to-run variation of accuracy to be known and reasonably constrained. We characterize the empirical distribution of Top-1 accuracy for ResNet-50 trained on ImageNet-1k.

Across 340 instances drawn from 7 cluster wide runs, each using 2052 tiles (171 nodes), final top-1 accuracy spans 0.7062 to 0.7307, a spread of 2.45 percentage points, with relative standard deviation of 0.0056.

We show that this variation is not a consequence of scale alone: we show that variations exist even at single rank on a separate A100-based platform with all framework determinism controls enabled that were available at the time of experiment.

As a positive control, we report a pipeline change that moves image augmentation to the GPU that costs 3.6 to 8.3 percentage points of accuracy, an effect several times larger than the run-to-run spread.

We conclude that single-run accuracy measurements are statistically unsound and recommend adopting a variance-aware range when interpreting large-scale training results. This work also provides practical guidance for reasoning about reproducibility concerns in modern AI workloads like LLM, VLM, etc.

1 INTRODUCTION

Qualification workloads for large-scale deep learning systems are commonly expressed as time-to-quality targets: a model is trained until it reaches certain accuracy. The same idea is also used in MLPerf training categories. This convention implicitly treats the final accuracy value as a stable scalar measurement. In practice, however, training accuracy is the outcome of a long numerical trajectory influenced by floating-point reduction order, kernel scheduling, data ordering, library implementations, and system-level effects. On near-production exascale systems, where hardware, firmware, communication libraries, and software stacks are still tuned, the presumption is that the measured accuracy of otherwise identical runs can vary more.

This paper characterizes that variance empirically using ResNet-50 training on ImageNet-1k as a controlled and well-understood workload. We focus on the near-production Aurora exascale system. We collected 340 completed training instances across 7 cluster-wide executions, each instance using 2052 tiles or GPUs. Rather than reporting a single best or representative accuracy, we measured the distribution of final Top-1 accuracy and completion time. This allows the benchmark result to be interpreted statistically: as a distribution with a mean, spread, and confidence level, rather than a single deterministic value.

The results show that run-to-run numerical variation is measurable, repeatable, and relevant to benchmark interpretation. Across the Aurora runs, final Top-1 accuracy spans 0.7062 to 0.7307, a 2.45 percentage-point range, while the relative standard deviation remains small at 0.00506. We further show that scale is not a necessary condition for this behavior: variation appears in a single-rank control experiment on a single A100 GPU of Polaris with framework determinism controls enabled at the time of the experiment. As a positive control, we also evaluate a pipeline change that

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

moves image augmentation to the GPU and observed a much larger accuracy degradation, confirming that the methodology can distinguish intrinsic run-to-run variance from substantive workload changes.

The primary contributions are threefold. First, we report a large-sample empirical distribution of Top-1 accuracy for an exascale-scale ResNet-50/ImageNet workload, showing bounded but measurable accuracy variation. Second, we use isolation experiments to narrow the interpretation of that variation and separate it from large systematic effects such as altered augmentation semantics. Third, we argue for variance-aware benchmark criteria in which outcomes are ideally evaluated using repeated measurements or an explicitly stated uncertainty range. For example, a reported best accuracy of 0.90 should be interpreted with an accompanying expected range rather than as a deterministic scalar.

The remainder of the paper reviews related work, describes the Aurora and Polaris platforms, presents the experimental methodology and measured distributions, and discusses implications for reproducible deep learning benchmarking.

2 Prior Work

Large-batch optimization for ImageNet established the role of layer-wise adaptive methods, especially LARS, for preserving convergence at high global batch sizes where vanilla SGD often degrades (You et al., 2017; You et al.; Mikami et al., 2018). Early results showed that warmup and learning-rate scaling can recover baseline accuracy while substantially reducing time to train, motivating their adoption in system-scale acceptance workloads.

Benchmark practice in MLPerf further codified this space by defining reproducible training targets, time-to-quality metrics, and reference convergence behavior for ResNet-50 (MLCommons). These rules provide an external anchor for validating whether a platform-specific run is broadly in-family, even when framework versions, precision modes, and communication stacks differ across systems.

A parallel thread of work studies determinism and reproducibility in deep learning software (Shah et al., 2021; NVIDIA; PyTorch). Even with fixed seeds and deterministic operator settings, outcomes can vary because of floating-point non-associativity, kernel scheduling differences, data-order effects, and implementation details in fused or reduced-precision paths. In distributed settings, these sensitivities are amplified by collective reduction order and overlap strategies.

Communication-focused studies, including NCCL analyses and synthesis approaches such as TACCL (Shah et al., 2021; NVIDIA), show that collective-algorithm choices affect both performance and numerical trajectory (Sanders et al., 2009; com). Related work on straggler mitigation and hierarchical synchronization highlights that systems interventions intended to improve throughput can also perturb optimization dynamics (PyTorch). This literature motivates treating run-to-run variation as a first-class measurement target, not merely noise around a mean.

3 System Descriptions

3.1 Aurora

The Aurora supercomputer, housed at the Argonne Leadership Computing Facility (ALCF), was, at the time of the experiments, a near-production exascale system based on the HPE Cray EX platform (Argonne Leadership Computing Facility, a). It features a theoretical peak performance exceeding 2 exaflops and has achieved 1.012 exaflops on the High-Performance Linpack (HPL) benchmark. The system architecture consists of 10,624 compute nodes, distributing workloads across 21,248 Intel Xeon CPU Max Series processors and 63,744 Intel Data Center GPU Max Series accelerators. Each accelerator contains two GPU tiles. Intra-node communication is handled through dedicated scale-up links called XeLinks. For inter-node communication, Aurora relies on the HPE Slingshot interconnect network configured with 84,992 fabric endpoints.

3.2 Reference Platform: Polaris

Polaris, an NVIDIA A100-based supercomputer at Argonne, was used as the cross-platform reference (Argonne Leadership Computing Facility, b). Each of the 560 compute nodes in the Polaris supercomputer is built on an HPE Apollo 6500 Gen10+ architecture that features a single 32-core AMD EPYC Milan 7543P host processor paired with 512 GB of DDR4 system memory. This CPU drives four NVIDIA A100 Tensor Core GPUs, each equipped with 40 GB of HBM2 memory, which communicate peer-to-peer over a high-speed, node-local NVLink mesh delivering 600 GB/s of bidirectional bandwidth. While the host CPU connects to this GPU cluster via PCIe Gen4 lanes, external communication is handled by two HPE Slingshot 11 Cassini Network Interface Cards. These dual 200 Gbps fabric endpoints provide an aggregate injection bandwidth of 50 GB/s per node, linking the node directly into the system's global Dragonfly network topology.

Both Intel Data Center GPU Max Series accelerators and NVIDIA A100 GPUs have broadly comparable FP32 compute capability, at approximately 20 TFLOP/s per device.

3.3 Software Stacks

The software stack on Aurora consisted of SLES Linux, HP PBS, Level Zero driver, oneAPI libraries, and PyTorch.

The software stack on Polaris was similar and consisted of SLES Linux, PBS, the NVIDIA driver, CUDA libraries, and PyTorch.

4 Challenges of Near-Production Systems

4.1 Dataset Staging on Diskless Nodes

Reading ImageNet from the parallel filesystem made a 90-epoch run take hours. Moreover, the Lustre file system was not designed to scale to 10K nodes exercising high throughput data I/O. As no node-local storage is available, the 150 GB dataset was staged into /tmp ramdisk of each node. /tmp is backed by 1 TB DRAM of CPU memory on each node. Staging used a two-step scheme: first a single copy to a root node, followed by high-bandwidth MPI broadcast to the remaining nodes. Staging from /tmp eliminated I/O-related bottlenecks in two significant ways: parallelization of the dataset and ultrafast I/O of the RAM file system.

Some hardware instability during the near-production period required additional node filtering prior to data collection runs. Hence every data-collection run was preceded by a set of tests that identified deficient nodes and fabric endpoints. The deficient nodes were then removed from data collection runs.

5 Experiment Description

5.1 Model, Dataset, and Optimizer

We used ResNet-50 v1.5 on ImageNet-1k. Because of the large global batch size, the LARS optimizer was used instead of SGD.

5.2 Hyperparameters

The per-rank batch size was 16, base learning rate was 21.2, momentum was 0.94, warmup length was 16 epochs, weight decay was 2.5×10^{-5}, and label smoothing was 0.1. The hyperparameters were used as is from previously published MLPerf results. No hyperparameter search was conducted to find a better set.

5.3 Execution Instances and Data Collection

In the experiments, multiple instances were run to fill as much of the available cluster as possible. Each instance used 171 nodes or 2052 ranks. Seven runs contributed between 39 and 53 completed

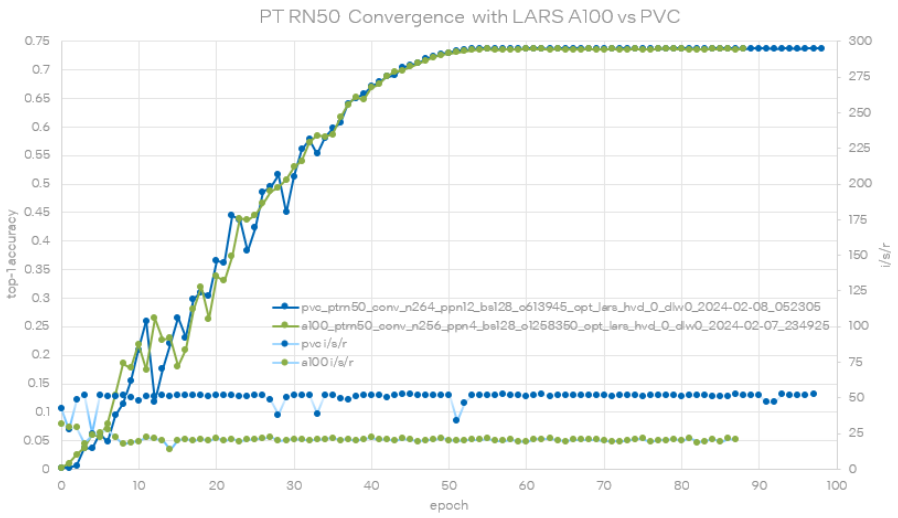

Figure 1: Convergence – LARS – Aurora (blue) vs. Polaris (green). The lower two curves show throughput. Aurora shows better throughput due to more scale-out NICs.

instances each, for 340 instances in total. Each instance logged output to the /tmp directory, and a post-processing chain later downloaded and consolidated the logs into the Lustre file system.

6 Establishing a Baseline

We compared ResNet-50 convergence of Aurora with Polaris, which has a similar software stack, using the same hyperparameters. Figure 1 shows that after approximately epoch 40, both systems have indistinguishably similar accuracy. Aurora's higher throughput is attributable to its higher scale-out network capacity. This experiment used a micro-batch size of 256 with 264 ranks on Aurora and 256 ranks on Polaris.

7 Experimental Results

7.1 Distribution of Top-1 Accuracy

Over the aggregated 340 instances, the basic statistics were: min 0.7062, max 0.7307, mean 0.7211, standard deviation 0.00395, giving a 0.0245 spread. Figure 2 shows a histogram of the aggregate of all instances. The peak histogram bucket is around 0.7221.

7.2 Per-run Top-1 Accuracy Variations

The max-minus-min varied from 0.0125 to 0.0024. Per-run relative standard deviation ranges from 0.0043 to 0.00636, indicating low variation. The chart for each of the runs is visually similar, with peaks at about 0.7225 (Figure 3).

7.3 Completion Time Variation

Variation was also observed in 90-epoch completion time across instances, as shown in Figure 4. The maximum-minus-minimum completion-time difference is 18.4 seconds, and the standard deviation is 3.8 seconds. Both are small relative to the mean completion time of 331 seconds.

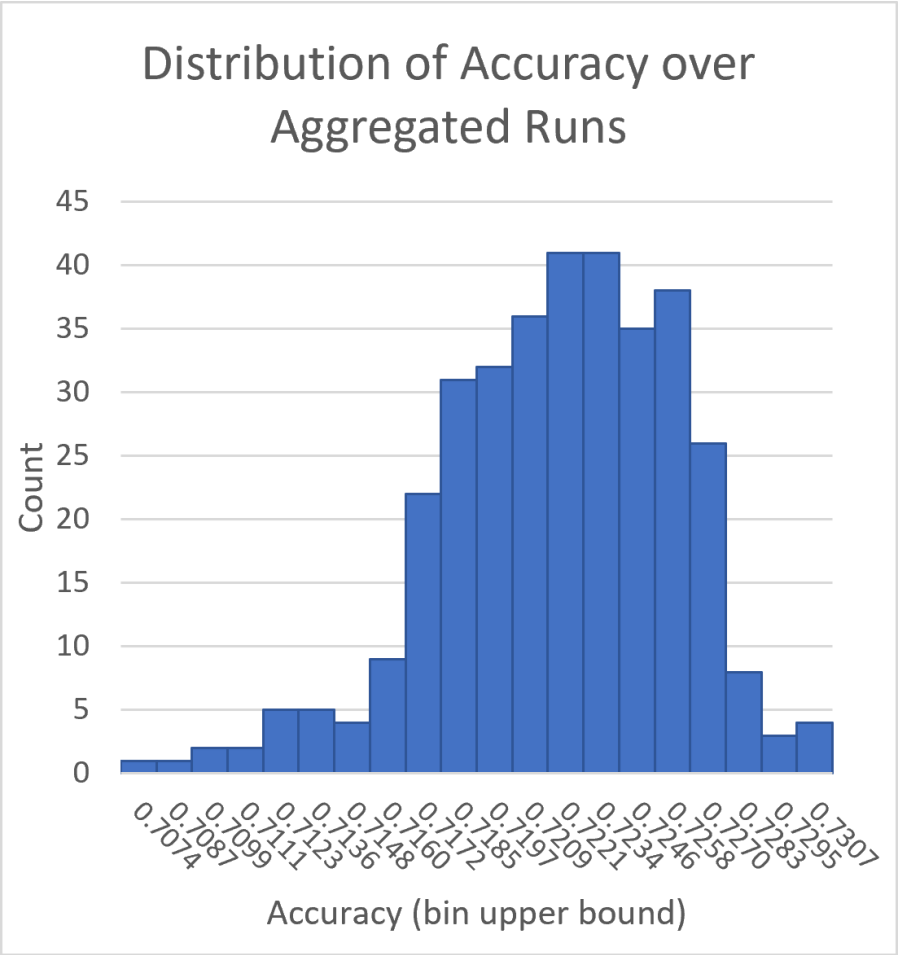

Figure 2: Distribution of Top-1 accuracy over 340 runs.

8 Isolating the Source of Variation

8.1 Single-Rank Control

Both variations persisted for a single-rank CIFAR-10 run on a single A100 on Polaris with all framework determinism settings enabled, as available at the time of experiments. Neither scale nor collective communication is therefore necessary to produce variations.

8.2 Positive Control: GPU-Side Augmentation

Moving RandomCrop, RandomFlip, and Normalize to the GPU reduced final Top-1 accuracy from 0.7231 to 0.6406 when images were resized to 224×224, and from 0.7235 to 0.6872 at 256×256: losses of 8.25 and 3.63 percentage points, respectively. The effect tracks the crop headroom left by the resize, indicating lost augmentation strength rather than a numerical fault. Throughput also fell by 3 to 6 percent. This provides a way to confirm that the setup resolves genuine differences.

8.3 Open Questions

The question of eliminating run-to-run variation in a large distributed cluster is an unsolved problem. In recent years, there have been efforts at all layers of the stack to bring more numerical determinism.

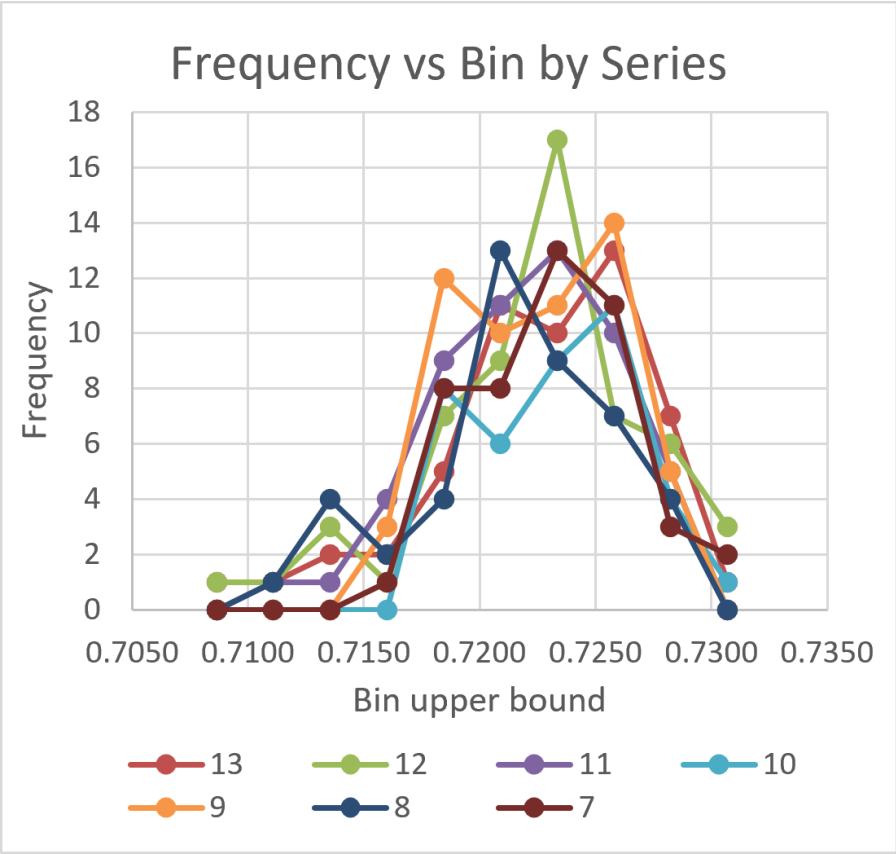

Figure 3: Per-instance distributions of Top-1 accuracy over 7 runs.

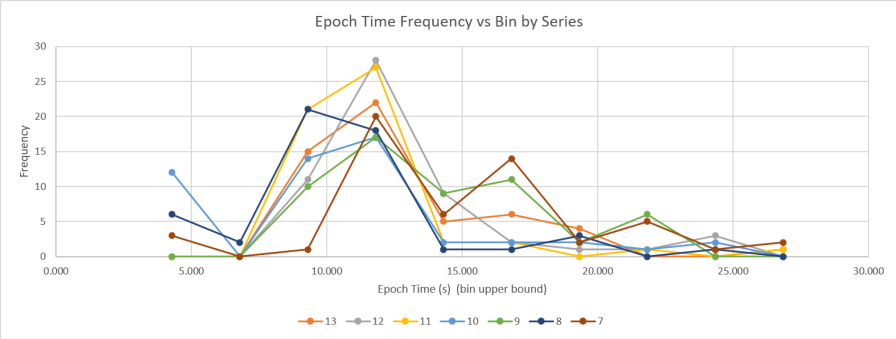

Figure 4: Completion-time variation across runs.

There are also many additional sources of variability in a large distributed system that are not present in a single process. These include chip-to-chip variation, node-to-node variation, collective associativity nondeterminism, and related system effects.

Variations at the single-process level include kernel reduction order, kernel launch order, and other low-level scheduling effects.

Given the variation observed in numerical accuracy, although on a near-production system, questions about reproducibility of AI results are not hard to understand. Claims of the best results on any benchmark suite are likely to be variable per this observation. Based on our measurements, we offer the rounded version of the observed RSD, 0.005, as an expected variation level for benchmark results.

Table 1: Summary statistics of Top-1 accuracy and completion-time variation.

Metric	Top-1	Time (s)
Mean	0.7213	331.726
Median	0.7216	329.803
Min	0.7062	328.412
Max	0.7307	346.851
Std-dev	0.00404	3.844
Rel. std-dev	0.56%	1.16%

Table 1 shows the aggregate variations for both Top-1 accuracy and completion time.

The observed completion-time variation was more than 2x the numerical variation. Completion-time variation is not as consequential as numerical variation, as long as the variation is less than some margin of error in percentage, for example below 5%. However, the question remains why completion-time variation is higher. Both are FP32 numbers. One explanation is that the precision available near 0 is much higher than at 320. At 0.72 and 331.7, the precisions are approximately 59.605×10^{-9} and 30.518×10^{-6}, respectively. Another contributor could be that accuracy measurements are over 50 samples for each class, whereas the time measurement is just 1 sample.

9 Conclusion

We characterized run-to-run numerical variation in large-scale ResNet-50 ImageNet training on a near-production exascale system. Across 340 instances across 7 runs on Aurora, final Top-1 accuracy remained fairly bounded but varied measurably, with values ranging from 0.7062 to 0.7307. A single-rank control on Polaris (A100) showed that scale and collective communication are not necessary conditions for such variations, while a GPU-side augmentation change produced a substantially larger and clearly distinguishable accuracy shift. These results support variance-aware interpretation of time-to-quality and best-accuracy claims: large-scale benchmark results should ideally be reported as distributions. However, due to the cost involved, it may not be feasible to run 30 or more experiments. At minimum, top benchmark-number claims should be viewed as a sample from a distribution (Gaussian?).

AI Use Disclosure

Generative AI tools were used to assist with editorial polishing, formatting review, and preparation of this manuscript for conference submission. The authors reviewed and take responsibility for all final text, claims, experimental descriptions, and reported results. Generative AI tools were not used to generate experimental results, fabricate data, or replace scientific validation.

Ethics Statement

This work studies reproducibility and numerical variation in large-scale deep learning benchmarks. It does not introduce human-subject data, personally identifiable information, or new deployed decision systems. The primary ethical consideration is transparent reporting: benchmark claims should communicate uncertainty so that downstream readers do not overinterpret single-run measurements.

Reproducibility Statement

The manuscript reports the model, datasets, platforms, optimizer, key hyperparameters, number of runs, and aggregate statistics used in the study. Before submission, the authors should verify whether code, exact software versions, launch scripts, random-seed policy, and log-processing scripts can be shared or described in an appendix or supplementary material without violating facility or organizational policies.

ACKNOWLEDGMENTS

The authors thank the facility and system teams who made the computing resources used in this study available. For double-blind review, identifying names and institutional acknowledgments have been omitted and should be restored only in the camera-ready version.

REFERENCES

Communication algorithm-architecture co-design for distributed deep learning. Available at https://par.nsf.gov/servlets/purl/10374122. Accessed 2026-09-11.

Argonne Leadership Computing Facility. Aurora. Available at https://www.alcf.anl.gov/aurora, a. Accessed 2026-09-11.

Argonne Leadership Computing Facility. Polaris. Available at https://www.alcf.anl.gov/polaris, b. Accessed 2026-09-11.

Hiroaki Mikami et al. Imagenet/resnet-50 training in 224 seconds. Available at https://nnabla.org/paper/imagenet_in_224sec.pdf, 2018. Accessed 2026-09-11.

MLCommons. Mlperf training benchmarks. Available at https://mlcommons.org/benchmarks/training/. Accessed 2026-09-11.

NVIDIA. Massively scale your deep learning training with nccl 2.4. Available at https://developer.nvidia.com/blog/massively-scale-deep-learning-training-nccl-2-4/. Accessed 2026-09-11.

PyTorch. Straggler mitigation on pytorch ddp by hierarchical sgd. Available at https://pytorch.org/blog/straggler-mitigation/. Accessed 2026-09-11.

Peter Sanders, Jochen Speck, and Jesper Larsson Träff. Two-tree algorithms for full bandwidth broadcast, reduction and scan. *Parallel Computing*, 35(12):581–594, 2009.

Aashaka Shah et al. Taccl: Guiding collective algorithm synthesis using communication sketches. *arXiv preprint arXiv:2111.04867*, 2021.

Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Yang You et al. Scaling sgd batch size to 32k for imagenet training. Available at https://www.comp.nus.edu.sg/~youy/publications/batch32k.pdf. Accessed 2026-09-11.