ECRL: Elastic and Correct Recovery Lab (DDP)

Comprehensive Matrix Report for Failure-Recovery Tradeoffs

Haowen Chen
National Taiwan University
b11902156@ntu.edu.tw

Abstract

We present a comprehensive matrix evaluation of failure-recovery semantics and performance tradeoffs in PyTorch Distributed Data Parallel (DDP) training using ECRL. The project is a systems semantics and measurement effort, not a new checkpointing algorithm.

The evaluated matrix spans 8 suites (CIFAR-10/CIFAR-100 $\times$ ResNet-18/ResNet-50 $\times$ two deterministic failure schedules), each executed with three variants at constant global batch: reference, failure+blocking, and failure+overlapped.

Across all suites and variants, semantic correctness invariants hold exactly (pass rate $= 1.0$), and failure variants exhibit the expected restart behavior (mean restarts $= 2.0$). Relative to reference, average goodput drops are $-13.2466\,\%$ for blocking and $-11.5026\,\%$ for overlapped. Overlapped improves goodput over blocking by $1.9931\,\%$ on average (8/8 suites positive), with substantial reductions in checkpoint-induced stall ($-10.2205\,\mathrm{s}$ mean) and write time ($-92.2858\,\%$ mean).

Although the end-to-end gain is modest on these compute-heavy ResNet workloads, the checkpoint-path reduction indicates strong scaling headroom as checkpoint volume grows. All artifacts (logs, metrics, aggregate tables, and publication figures) are reproducible and included.

1 Introduction

Distributed training reliability requires both semantic correctness after faults and acceptable runtime overhead. Many reports emphasize throughput while underspecifying semantic guarantees, or prove semantic behavior while underreporting system cost. ECRLis designed to make both dimensions explicit, auditable, and reproducible.

This report targets three concrete questions:

1. **Correctness under crash-restart**: after injected failures, does training consume exactly the expected sample windows with no duplicate/missing semantics?

2. **Useful progress rate under faults**: what goodput remains after checkpointing and restart overheads?

3. **Strategy tradeoff**: for identical failure schedules and geometry, how does overlapped checkpointing compare to blocking checkpointing?

Unlike broad benchmark surveys, this report fixes recovery semantics and focuses on controlled matrix comparisons. The contribution is an end-to-end, artifact-grounded characterization of failure-recovery tradeoffs under strict invariants.

2 Background

2.1 PyTorch DDP Context

PyTorch has become a standard software substrate for large-scale deep learning systems [1]. Its DistributedDataParallel (DDP) execution model is widely used in production and research clusters, with detailed implementation experience reported in the systems literature [2] and API/runtime references maintained by the project [3]. This report studies recovery behavior directly in that operational context rather than via a toy simulator.

2.2 Why Checkpoint Policy Matters

In fail-stop training, checkpointing is the mechanism that bounds replay work after crash. However, synchronous persistence can stall useful compute progress. Overlapped persistence seeks to move filesystem I/O off the critical path while preserving synchronization semantics at step boundaries. The engineering question is therefore not "can we checkpoint?" but "how much useful progress is lost to checkpoint policy under realistic failures?"

3 Scope and Hard Constraints

3.1 Scope

ECRLevaluates DDP data-parallel training only. It does not claim a new optimizer, a new checkpoint format, or a new distributed runtime algorithm.

3.2 Non-Negotiable Constraints

The implementation enforces:

- Python 3.11 virtual-environment execution.
- DDP-only orchestration via `torch.distributed.run`.
- Constant `GLOBAL_BATCH` across fresh and resumed attempts.
- Runtime divisibility: `GLOBAL_BATCH` $\bmod W = 0$, where W is world size.

- `LOCAL_BATCH` $=$ `GLOBAL_BATCH`$/W$, `num_workers=0`, `drop_last=True`.
- Epoch geometry defined by fixed logical step count, not dataloader exhaustion.
- Rank-0-owned atomic checkpoint write protocol.
- Failure injection only at synchronized step boundaries.
- Overlapped path writes via bounded background pipeline, not asynchronous model mutation.

3.3 Epoch Geometry Contract

Given dataset size D and constant global batch B:

$$S = \left\lfloor \frac{D}{B} \right\rfloor, \qquad N = S \cdot B. \tag{1}$$

Each epoch executes exactly S logical steps.

3.4 Correctness Invariant

For each epoch, recovered runs must match the reference sample-ID multiset exactly:

$$\texttt{duplicates} = 0, \quad \texttt{missing} = 0, \quad \texttt{extra} = 0. \tag{2}$$

4 System Design

4.1 Data-Progress Instrumentation

Supported datasets emit triplets $(x, y, \texttt{sample\_id})$, enabling step-level semantic auditing. Sample-ID hash/count fields are persisted in rank logs to verify data-progress behavior at scale without storing full ID payloads per step.

4.2 Resumable Sampler

For epoch e, ECRLconstructs a deterministic permutation from $(\texttt{seed}, e)$, truncates to `epoch_samples`, and defines global step window s:

$$\texttt{window}_s = \texttt{perm}_e\big[s \cdot \texttt{GLOBAL\_BATCH} : (s+1) \cdot \texttt{GLOBAL\_BATCH}\big]. \tag{3}$$

Rank r receives contiguous local shard:

$$\texttt{window}_s\big[r \cdot \texttt{LOCAL\_BATCH} : (r+1) \cdot \texttt{LOCAL\_BATCH}\big]. \tag{4}$$

Checkpointed sampler state is minimal and explicit: $\{\texttt{epoch}, \texttt{cursor\_step}, \texttt{seed}\}$.

4.3 StatePack and Atomic Persistence

State capture includes model, optimizer, scheduler, sampler, global step-state, and RNG states (Python/NumPy/torch CPU/CUDA). Persistence follows a strict atomic sequence:

```
tmp write -> fsync -> rename -> latest pointer update
```

The intent is to avoid partial checkpoint visibility under interruption. Both model and optimizer states are persisted; optimizer state is often a dominant contributor to checkpoint size in practical training jobs.

4.4 Checkpoint Strategies

- **Blocking**: `barrier -> capture -> rank0 write -> barrier`
- **Overlapped**: `barrier -> capture -> enqueue -> barrier`

Overlapped uses bounded inflight writes (here `max_inflight=4`) and keeps model-step semantics synchronized at boundaries.

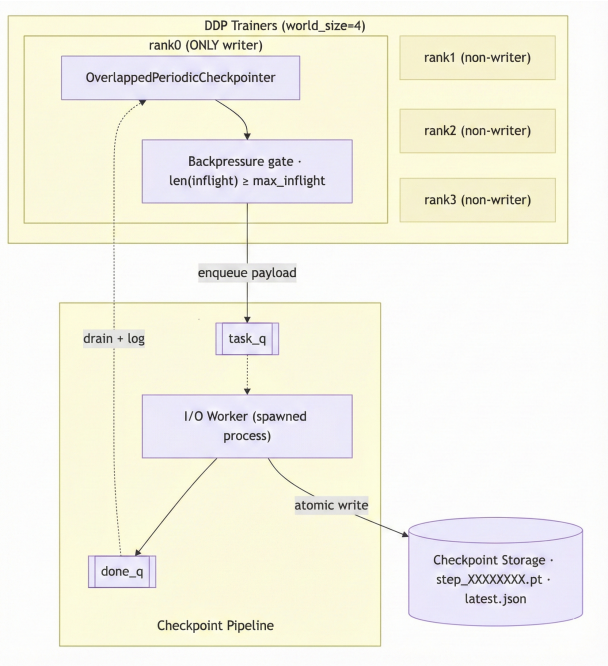

Figure 1: Overlapped checkpoint architecture used in Exp4.

4.5 Failure Model and Supervisor

Failure injection is deterministic (rank-0 exit 137) at configured global steps only after boundary-safe synchronization. A supervisor restarts from `latest.json` until target step completion or policy stop.

4.6 Deterministic Step Timeline

Each training step follows a strict ordering to avoid ambiguous kill boundaries:

1. materialize deterministic rank-local shard from global window,
2. execute forward/backward/update,
3. synchronize at checkpoint boundary if periodic trigger fires,
4. capture state and either write (blocking) or enqueue (overlapped),
5. synchronize post-capture boundary,
6. optionally inject failure after final barrier.

This timeline ensures restart points are always associated with a globally well-defined cursor state.

4.7 Supervisor State Machine

Supervisor behavior is intentionally deterministic:

- **Launch**: invoke `torch.distributed.run` with fresh or `--resume-latest`.
- **Monitor**: classify exit as completion vs failure.
- **Recover**: on failure, inspect `latest.json`, increment counters, relaunch.
- **Terminate**: stop only at target completion or policy limit.

The supervisor does not modify checkpoint payloads; it only selects latest committed state.

4.8 Formal Recovery Semantics

Let committed logical step g denote a step where all ranks finished update, sampler cursor advanced exactly once, and boundary protocol completed. Crash-restart may replay physical compute, but semantic progress is defined by committed logical sequence.

Let $E_{e,s}$ be expected sample multiset at epoch e, step s, and $O_{e,s}$ be observed merged multiset from rank logs. Correctness requires:

$$\forall e, s: \quad O_{e,s} = E_{e,s}. \tag{5}$$

4.9 Logging Schema

Table 1: Core logging fields used by the analysis pipeline.

Field	Purpose
`global_step`	Logical progress accounting and restart boundary checks
`epoch`, `cursor_step`	Deterministic window reconstruction
`sample_ids_hash/count`	Scalable semantic verification
`world_size`, `rank`	Cross-rank consistency checks
`loss`	Trajectory and divergence inspection
`checkpoint_rank0.jsonl`	Snapshot/write/enqueue/backpressure/stall decomposition
`supervisor.json`	Attempt/restart/terminal status accounting

5 Experimental Methodology

5.1 Methodology Goals

The goal is not peak throughput; it is auditable recovery semantics under realistic failures, followed by quantitative decomposition of overhead under controlled matrix variation.

5.2 Runtime Environment

All Exp4 matrix runs were executed on the following host environment.

Table 2: System environment for Exp4 execution.

Item	Value
OS	Arch Linux, kernel `6.12.53-1-lts`, `x86_64`
CPU	2x Intel Xeon Gold 5118 @ 2.30GHz
Cores / Threads	24 physical cores, 48 logical CPUs
System memory	125 GiB RAM, 19 GiB swap
GPU inventory	4x NVIDIA GeForce RTX 4090 (24564 MiB), 2x NVIDIA GeForce RTX 3090 (24576 MiB)
NVIDIA stack	Driver `580.95.05`, CUDA runtime `13.0` (`nvidia-smi`)
Python	`3.11.14`
PyTorch	`2.10.0+cu128`
Torch CUDA build	`12.8` (`torch.version.cuda`)
Torch visible GPUs	`cuda_available=True`, `cuda_count=6`
DDP launch shape	`nproc_per_node=4` for all reported runs

5.3 Matrix Definition

The matrix prefix is `exp4_matrix_20260215_151709`. We evaluate:

- Datasets: CIFAR-10, CIFAR-100 [4, 5]
- Models: ResNet-18, ResNet-50 [6]
- Failure schedules: `base` (400,1200), `late` (800,1400)
- Checkpoint periodicity $K = 50$, overlapped `max_inflight=4`
- Variants per suite: reference, failure+blocking, failure+overlapped

The matrix balances three axes (dataset, model scale, failure timing) while fixing world size, global batch, checkpoint cadence, and seed. Suites are indexed as `<dataset>-<model>-<failure>` (e.g., `CIFAR-10-ResNet-18-base`), and each suite runs the same three variants: `ref`, `blk`, and `ovl`.

Table 3: Exp4 run matrix definition.

Axis	Values	Count	Role
Dataset	CIFAR-10; CIFAR-100	2	Data complexity variation
Model	ResNet-18; ResNet-50	2	Compute/parameter scale variation
Failure schedule	`base`; `late`	2	Failure timing sensitivity
Variants	`ref`; `blk`; `ovl`	3	Method comparison
Total suites	–	8	Cartesian product over first three axes
Total run units	–	24	8 suites $\times$ 3 variants

5.4 Variables and Control Policy

Table 4: Controlled and varied factors for Exp4.

Factor	Role	Treatment
`GLOBAL_BATCH`	Controlled	Fixed within suite and across restart attempts
World size (W)	Controlled	Fixed at 4 for this matrix
Checkpoint strategy	Varied	Blocking vs overlapped
Failure schedule	Varied	`base` (400,1200), `late` (800,1400)
Dataset/model	Varied	CIFAR-10/100 and ResNet-18/50
Seed count	Controlled	One seed per suite in current report

5.5 Execution and Resume Policy

Matrix orchestration supports resume-safe execution. For this run set, manifest status is mixed because completed suites were reused:

- `completed`: 4 suites freshly executed in this session
- `skipped_exists`: 4 suites reused from existing artifact-complete outputs

All 8 suites have complete publishable artifacts and are included in aggregation.

5.6 Metrics

Primary metrics:

- Correctness pass rate.
- Goodput (steps/sec) and relative deltas.
- Restart counts.
- Checkpoint timing decomposition: snapshot, write, stall, wall.
- Divergence to reference via mean absolute loss difference and checkpoint L2 traces.

Goodput is:

$$\texttt{goodput} = \frac{\texttt{committed\_logical\_steps}}{\texttt{wall\_clock\_time}}. \tag{6}$$

We interpret wall time as:

$$t_{\text{wall}} = t_{\text{compute}} + t_{\text{snapshot}} + t_{\text{write}} + t_{\text{sync}} + t_{\text{recovery}}. \tag{7}$$

5.7 Statistical Treatment

This matrix has $n = 1$ seed per suite, so reported aggregates are descriptive means across suites, not inferential confidence claims. We use sliced means and full per-suite tables to expose spread and avoid over-interpreting a single scalar summary.

5.8 Threat Model

The evaluated failure model is fail-stop process termination at step boundaries. This study does not model Byzantine behavior, partial checkpoint corruption, or long-duration network partitions.

5.9 Quality Gates

A suite is accepted only if:

1. target steps complete for all three variants,
2. correctness checker passes,
3. publishable JSON/markdown artifacts exist,
4. supervisor and checkpoint metadata are internally consistent.

6 Results

6.1 Aggregate Outcomes Across 8 Suites

Table 5: Aggregate Exp4 outcomes (means across 8 suites).

Metric	Reference	Blocking	Overlapped	Ovl-Block
Pass rate	1.0000	1.0000	1.0000	–
Goodput (steps/s)	10.9284	9.4560	9.6433	+1.9931%
Drop vs ref	0.0000%	-13.2466%	-11.5026%	+1.7440pp
Restarts	0.0000	2.0000	2.0000	0.0000
Wall time (s)	157.1323	181.3431	177.7858	-3.5573
Stall time (s)	18.0629	18.0859	7.8654	-10.2205
Write time (s)	17.8939	17.9036	1.3811	-16.5225
Snapshot time (s)	0.0295	0.0324	0.0328	+0.0004

Two points are immediate: (i) semantic correctness remains exact under both failure strategies, and (ii) overlapped is consistently faster than blocking in this matrix while preserving those semantics.

6.2 Completion Accounting

The analyzed artifact set contains 8 suites and 24 variant-units (reference, blocking, overlapped per suite). All 24 variant-units pass correctness and are included in aggregation.

6.3 Suite-Level Ranking

The worst suite is still positive for overlapped (+0.2358%), so the directional result is unanimous (8/8).

Table 6: Per-suite ranking by overlapped gain over blocking. Positive $\Delta_{\text{ovl-blk}}$ means overlapped is faster.

Dataset	Model	Failure	Ref GP	Blk GP	Ovl GP	$\Delta_{\text{ovl-blk}}$	$\Delta_{\text{blk-ref}}$	$\Delta_{\text{ovl-ref}}$	Δ Stall (s)
CIFAR-100	ResNet-18	late (800,1400)	14.5657	12.2979	12.7471	+3.6527%	-15.5693%	-12.4854%	-6.6940
CIFAR-100	ResNet-50	base (400,1200)	8.6444	7.6277	7.8637	+3.0946%	-11.7612%	-9.0305%	-13.9375
CIFAR-10	ResNet-50	late (800,1400)	7.2259	6.6058	6.8030	+2.9853%	-8.5812%	-5.8521%	-14.6115
CIFAR-10	ResNet-18	base (400,1200)	11.7289	11.3023	11.5721	+2.3869%	-3.6372%	-1.3371%	-4.9425
CIFAR-10	ResNet-50	base (400,1200)	8.5743	6.7222	6.8576	+2.0142%	-21.6013%	-20.0222%	-14.5308
CIFAR-100	ResNet-18	base (400,1200)	14.9770	12.5790	12.7443	+1.3142%	-16.0113%	-14.9075%	-5.9794
CIFAR-100	ResNet-50	late (800,1400)	8.3879	7.2988	7.3179	+0.2609%	-12.9842%	-12.7571%	-16.1284
CIFAR-10	ResNet-18	late (800,1400)	13.3230	11.2143	11.2408	+0.2358%	-15.8274%	-15.6289%	-4.9403

6.4 Sliced Analysis

Table 7: Sliced means of overlapped relative improvement.

Slice	Group	Ovl vs Blk	Δ Stall (s)	Δ LossDiff
Dataset	CIFAR-10	+1.9055%	-9.7563	+0.0009
Dataset	CIFAR-100	+2.0806%	-10.6848	+0.0009
Model	ResNet-18	+1.8974%	-5.6390	+0.0018
Model	ResNet-50	+2.0888%	-14.8021	+0.0000
Failure	base	+2.2025%	-9.8476	+0.0012
Failure	late	+1.7837%	-10.5935	+0.0006

ResNet-50 suites show larger mean stall reduction than ResNet-18 suites in this matrix ($-14.8021\,\text{s}$ vs $-5.6390\,\text{s}$), consistent with greater overlap opportunity when write-path cost is more dominant.

6.5 Divergence Notes

Mean absolute loss-difference is close between methods:

$$\overline{\Delta\text{loss}}_{\text{blk}} = 0.0660, \quad \overline{\Delta\text{loss}}_{\text{ovl}} = 0.0669. \tag{8}$$

The cross-suite mean delta (ovl-blk) is only +0.000890. This is consistent with no obvious stability regression from overlap in the observed matrix; correctness invariants are satisfied in all suites regardless.

6.6 Visual Summary (Color Figures)

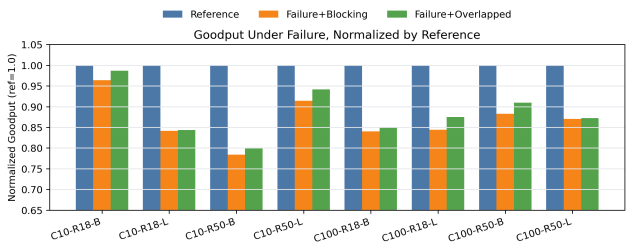

Figure 2: Normalized goodput by suite and strategy (reference normalized to 1.0).

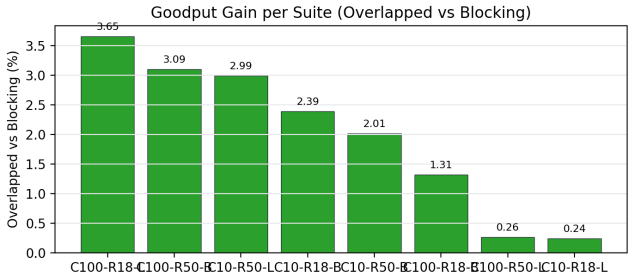

Figure 3: Overlapped goodput gain over blocking across all suites.

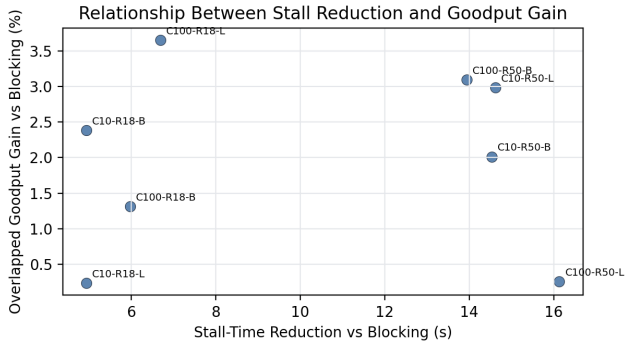

Figure 4: Relationship between stall reduction and overlapped gain by suite.

7 Discussion

7.1 What Is Strongly Supported

The matrix provides strong directional evidence that overlapped checkpointing is preferable to blocking for failure-injected DDP runs under these settings:

- 8/8 suites show positive overlapped gain vs blocking.
- Correctness remains exact for all variants.
- Mean stall reduction is substantial ($56.5110\,\%$ relative to blocking stall).
- Mean write reduction is very large ($92.2858\,\%$ relative to blocking write total).

7.2 Interpretation of Stall vs Goodput

Write-time reduction is necessary but not sufficient to explain end-to-end gain; barrier synchronization and restart cadence still contribute to wall-clock behavior. That is why the observed wall-clock gain ($1.9616\,\%$) is much smaller than write-time gain, despite consistent directionality.

7.3 Why This Report Uses Matrix Aggregation

Single-suite anecdotes are unstable for systems reporting. Matrix aggregation across data/model/failure axes shows whether directional claims survive configuration changes. Here they do, but with nontrivial spread ($0.2358\,\%$ to $3.6527\,\%$).

7.4 Scalability Implication

The key practical signal is not only the $1.9931\,\%$ mean goodput gain; it is the collapse of checkpoint write overhead itself ($-92.2858\,\%$ relative to blocking). On CIFAR-scale ResNet runs, compute still dominates wall-clock, capping end-to-end gains. As checkpoint payloads grow (larger models, richer optimizer states, denser save cadence), the same write-path reduction should translate into larger goodput impact.

8 Limitations and Threats to Validity

- **Single-seed per suite**: current matrix has `num_seeds=1`; confidence intervals are not meaningful.
- **Fixed checkpoint knobs**: `checkpoint_every=50` and `max_inflight=4` are held constant.
- **CIFAR-scale scope**: conclusions are strongest for small-image workloads and tested ResNet variants.
- **Fail-stop model**: this report does not cover Byzantine failure, network partition, or filesystem corruption models.

These limits do not invalidate the directional finding, but they constrain external generalization.

9 Future Work

The next phase should focus on turning directional evidence into stronger, publication-grade claims:

- **Statistical robustness**: rerun the same 8-suite matrix with multiple seeds (at least 2–3 per suite) and report confidence intervals for goodput and timing deltas.
- **Checkpoint-policy sensitivity**: sweep `checkpoint_every` and `max_inflight` to map where overlap helps most and where synchronization dominates.
- **Larger state footprints**: repeat evaluation on larger backbones and higher-resolution workloads where model+optimizer checkpoint payloads are much larger.
- **Storage-path variation**: compare local NVMe, network filesystems, and shared-cluster I/O paths to isolate bottlenecks that control overlap effectiveness.
- **Elasticity under failure**: extend the matrix to explicit $N \rightarrow M$ resume cases under injected failures, not only fixed-world-size restart.
- **Richer failure models**: add controlled fault types beyond fail-stop exit (e.g., transient I/O errors and delayed checkpoint completion) while preserving semantic invariants.

10 Reproducibility and Artifact Map

Primary artifact root for this report:

`results/results_exp4`

Key files:

- `exp4_matrix_20260215_151709_manifest.csv`
- `exp4_matrix_20260215_151709_summary.csv`
- `exp4_matrix_20260215_151709_detailed_table.csv`
- `exp4_matrix_20260215_151709_analysis.md`
- per-suite `reports/*_publishable.json`
- `overleaf/figures/exp4_*.png` and PDF companions

11 Conclusion

ECRLprovides an auditable DDP recovery framework with strict semantics and reproducible instrumentation. In the 8-suite Exp4 matrix, correctness invariants are exact across all failure and checkpoint strategies, and overlapped checkpointing consistently improves goodput over blocking. The measured gains are modest but robust in direction, supported by substantial stall/write reductions. The next publication-strength increment is straightforward: keep the same matrix and add more seeds (and optionally checkpoint-interval sweeps) under resume-safe execution.

Appendix

A Full Suite Table

Table 8: Complete Exp4 detailed table (selected fields).

Dataset	Model	Failure	Ref GP	Blk GP	Ovl GP	Ovl vs Blk	Blk Stall	Ovl Stall	Ovl-Blk Wall (s)
CIFAR-10	ResNet-18	base	11.7289	11.3023	11.5721	+2.3869%	11.0699	6.1273	-3.3002
CIFAR-10	ResNet-18	late	13.3230	11.2143	11.2408	+0.2358%	10.8221	5.8818	-0.3357
CIFAR-10	ResNet-50	base	8.5743	6.7222	6.8576	+2.0142%	24.2825	9.7517	-4.6996
CIFAR-10	ResNet-50	late	7.2259	6.6058	6.8030	+2.9853%	24.7192	10.1076	-7.0211
CIFAR-100	ResNet-18	base	14.9770	12.5790	12.7443	+1.3142%	11.1750	5.1956	-1.6500
CIFAR-100	ResNet-18	late	14.5657	12.2979	12.7471	+3.6527%	11.7959	5.1019	-4.5848
CIFAR-100	ResNet-50	base	8.6444	7.6277	7.8637	+3.0946%	24.2103	10.2728	-6.2965
CIFAR-100	ResNet-50	late	8.3879	7.2988	7.3179	+0.2609%	26.6126	10.4842	-0.5705

B Run Manifest Snapshot

- Matrix prefix: `exp4_matrix_20260215_151709`
- Suites: 8; variants per suite: 3; total run units: 24
- Manifest statuses: `completed`=4, `skipped_exists`=4
- Checkpoint policy: periodic every 50 steps
- Failure schedules: `base(400,1200)`, `late(800,1400)`

C Data and Model Sources

- CIFAR datasets [4, 5]
- ResNet family [6]
- PyTorch DDP API [3]

[]

References

[1] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1

[2] S. Li, Y. Zhao, R. Varma, J. J. Kossaifi, M. Xu, P. Damania, V. Or, V. R. R., K. Sivasubramaniam, G. Chen, B. Chiu, M. Khanuja, Y. Zou, J. He, A. Aggarwal, Z. DeVito, and S. Chintala, "Pytorch distributed: Experiences on accelerating data parallel training," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3005–3018, 2020. 1

[3] PyTorch Contributors, "Distributeddataparallel," https://pytorch.org/docs/stable/generated/torch.nn.parallel.DistributedDataParallel.html, 2026, accessed: 2026-02-13. 1, 7

[4] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009. 3, 7

[5] A. Krizhevsky and G. Hinton, "Cifar-100 dataset," https://www.cs.toronto.edu/~kriz/cifar.html, 2009, accessed: 2026-02-13. 3, 7

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3, 7

[7] J. W. Young, "A first order approximation to the optimum checkpoint interval," *Communications of the ACM*, vol. 17, no. 9, pp. 530–531, 1974.

[8] J. T. Daly, "A higher order estimate of the optimum checkpoint interval for restart dumps," *Future Generation Computer Systems*, vol. 22, no. 3, pp. 303–312, 2006.

[9] K. M. Chandy and L. Lamport, "Distributed snapshots: Determining global states of distributed systems," *ACM Transactions on Computer Systems*, vol. 3, no. 1, pp. 63–75, 1985.

[10] E. N. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Computing Surveys*, vol. 34, no. 3, pp. 375–408, 2002.

[11] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski, "Design, modeling, and evaluation of a scalable multi-level checkpointing system," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2010.

[12] A. Sergeev and M. D. Balso, "Horovod: Fast and easy distributed deep learning in tensorflow," https://arxiv.org/abs/1802.05799, 2018.

[13] J. Mohan, A. Phanishayee, and V. Chidambaram, "Checkfreq: Frequent, fine-grained dnn checkpointing," in *19th USENIX Conference on File and Storage Technologies (FAST)*, 2021.

[14] B. Nicolae, J. Li, J. Wozniak, G. Bosilca, M. Dorier, and F. Cappello, "Deepfreeze: Towards scalable asynchronous checkpointing of deep learning models," in *20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2020.

[15] T. Gupta, S. Krishnan, R. Kumar, A. Vijeev, B. Gulavani, N. Kwatra, R. Ramjee, and M. Sivathanu, "Just-in-time checkpointing: Low cost error recovery from deep learning training failures," in *Proceedings of the ACM European Conference on Computer Systems (EuroSys)*, 2024.