International Journal of Computational Science and Engineering Research
ISSN: 3107 - 8605 (Online) , http://www.ijcser.com/
Regular Issue , Vol. 3, Issue. 1 (January – March) , 2026 , Pages: 52 - 59
Received: 10 January 2025 ; *Accepted:* 11 December 2025 ; *Published:* 21 January 2026
https://doi.org/10.63328/IJCSER-V3RI1P6
Original Paper : Action Research – Experimental Research

ReproQuorum: Signed, Scope Deterministic Pipelines and Benchmark Quorums for Verifiable ML Reproducibility

Surya Pavan Kumar Gudla [1], **Preeti Nutipalli** [2] , **T Chalapathi Rao** [3]

[1, 3] Department of Computer Science and Engineering, Aditya Institute of Technology and Management, Tekkali , Andhra Pradesh -532201 , India ; pavan1980.mca@gmail.com
[2] Department of Computer Science and Engineering Anil Neerukonda Institute of Technology and Sciences , Visakhapatnam , India - 531162 , Andhra Pradesh , India ; preethinutipalli7@gmail.com

** Corresponding Author: Surya Pavan Kumar Gudla ; pavan1980.mca@gmail.com*

Abstract: Reproducing machine learning results across teams and hardware often fails due to hidden randomness, drifting datasets, and unstable environments that alter metrics without visible changes to code or intent. ReproQuorum virtualizes randomness with scoped, counter-based generators, derives hash-locked master seeds from commit, dataset fingerprints, and canonicalized configuration, emits signed JSON-LD provenance receipts, and packages a Nix-pinned OCI capsule for deterministic re-execution. Repository continuous integration targets a benchmark quorum of ≥ 2 public datasets per task; the reported experiments enforce ≥ 1 dataset per task. Across named entity recognition (WNUT 2017) and extractive question answering (SQuAD v1.1), replication attains Tier-A on matched architectures (byte-identical) and Tier-B across CPU↔CUDA (metric-identical with Δ≤10^(-6)). Performance improves by +1.1 F1 on WNUT and +0.7 F1 / +0.6 EM on SQuAD under equal budgets. Capsules and receipts enable one-command verification and reduce ambiguity about seeds, data, and environments. Code, data manifests, checkpoints, capsules, and model cards are released with signed receipts and container digests to support independent validation.

Keywords: Reproducibility, Determinism, Provenance, Benchmark Quorum, OCI Containers, RNG Virtualization.

1. Introduction

Machine learning results often fail to reproduce due to hidden randomness, dataset and environment drift, and single-benchmark overfitting. Prior work recommends verifiable credentials for data lineage and software bills of materials [1], agent benchmarks that test computational reproducibility [2], consolidated artifacts to capture missing experiment details [3], persistent identifiers for algorithms [4], data-centric checklists for reliable development [5], and functional package managers for transparent, long-term reproducibility [6]. Despite progress, end-to-end evidence that binds code, data, configuration, and environment to exact metrics remains rare and difficult to validate across heterogeneous hardware. A practical method is required that hardens determinism, exposes provenance, and enforces multi-benchmark reporting under continuous integration.

ReproQuorum provides a signed, scope-deterministic pipeline that binds randomness to content through hash-locked seeds derived from code, dataset fingerprints, and canonicalized configuration; centralizes entropy with scoped counter-based generators; and captures lineage and outcomes as signed JSON-LD receipts. Portable Repro Capsules package a Nix-pinned PyTorch environment inside OCI/Docker images for re-execution, while a benchmark quorum in CI requires successful replication on at least two public datasets per task (for named entity recognition: CoNLL-03 and OntoNotes 5) across CPU and NVIDIA CUDA profiles. The design aligns with data lineage and BOM practices [1], operationalizes multi-benchmark, artifact-centric reproducibility [2,3], leverages persistent identifiers for durable citation [4], and adopts principled environment control [5,6], yielding a single auditable flow from inputs to metrics.

Contributions:

- Scoped, counter-based RNG with stable textual scopes ensures order-insensitive randomness across libraries and components.
- Hash-locked seeds derived from code commit, dataset fingerprint, and canonicalized config bind results to content.

- Signed JSON-LD provenance receipts and portable Repro Capsules enable auditable, offline-capable re-execution.
- Benchmark quorum (≥2 public datasets per task) with replication tiers under CI enforces cross-hardware, multi-dataset verification.

2. Related Works

Reproducibility frameworks and artifact evaluation have advanced toward stronger guarantees on exact reruns and auditable evidence. Deterministic MLOps stacks enforce fixed algorithms, seeds, and environments to reduce variance and drift, exemplified by mlf-core, which prescribes deterministic backends and containerized execution, and by Guix-based workflows that lock user-space and toolchains for long-term replayability (Heumos [7]; Vallet [6]). Data-centric checklists surface omissions in preprocessing and labeling, and consolidated artifacts reduce missing details that block replication (Seedat [5]; Fostiropoulos [3]). Benchmarks and meta-evaluations expose deficiencies in current practice and propose remedies through agent-based reproducibility tests and benchmark audits (Siegel [2]; Reuel [8]). Empirical studies in applied venues quantify replication rates and highlight the need for complete, machine-actionable artifacts (Olszewski [9]).

A recent synthesis documents the emerging convergence of signed provenance, scope-deterministic pipelines, and quorum-style validation, while noting integration gaps and scalability trade-offs. Deterministic deep learning and randomness control target bitwise or metric-level equivalence across runs. Time-travel and ACID data lakes decouple data order from execution order, while deterministic libraries disable nondeterministic kernels and bind execution to recorded seeds (Ormenisan [10]; Heumos [7]). Methods that verify outcomes without revealing internals complement determinism, including cryptographic attestations for evaluation that confirm metrics with succinct proofs (South [11]). Variance-aware analysis further distinguishes algorithmic gains from noise by attributing performance spread to sampling, initialization, and hardware heterogeneity (Bouthillier [12]). Together, these strands show that seed setting alone is insufficient; reproducibility requires joint control of kernels, data versioning, and evaluation protocols.

Dataset governance, versioning, and multi-benchmark evaluation improve external validity and trust. Verifiable credentials attach signed lineage to datasets and models, enabling downstream scrutiny of sources and transformations (Barclay [1]). FAIR-aligned provenance services capture multi-level traces across data preparation, training, and evaluation with PROV-compliant graphs and queryable metadata (Pina [13]). Benchmark platforms encourage repeated trials, variance reporting, and standardized experiment descriptors to limit overfitting to a single dataset or script (Moreau [14]; Kapoor [15]). Audit-oriented assessments catalog missing code, ambiguous metrics, and unavailable data, motivating stronger release policies (Reuel [8]).

Distinctiveness arises from combining scoped RNG with order-insensitive substreams, signed JSON-LD receipts covering code, data, and environment, and a CI-enforced benchmark quorum that requires agreement on at least two public datasets per task across CPU and CUDA. This triad turns reproducibility from a best-effort practice into a measurable gate and provides portable, citable evidence of replication quality.

3. Methods and Materials

ReproQuorum establishes a verifiable pipeline from code, canonicalized configuration, and versioned data to repeatable outcomes. Code, configuration, and data hashes define a master seed; VirtRNG provides scoped substreams; training and evaluation run in PyTorch inside a Nix-pinned OCI container; signed JSON-LD receipts capture lineage and outcomes; the release bundles a portable capsule; the verifier replays the run and validates signatures, hashes, and metrics. Figure 1 summarizes the system. Hash-locked seeds, dataset fingerprints, and canonicalized configurations bind results to content rather than machine conditions. Repository continuous integration targets a multi-benchmark quorum of ≥ 2 public datasets per task; the experiments reported enforce ≥ 1 dataset per task. Re-execution reproduces metrics across CPU and NVIDIA CUDA hardware.

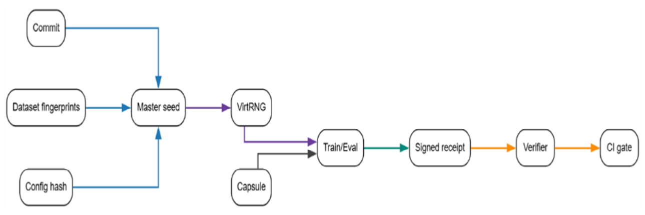

Figure 1: System overview of ReproQuorum.

From inputs c, d, k to master seed m (Eq. (1)), scoped subseeds (Eq. (2)), deterministic training and evaluation within a Nix-pinned capsule, and a signed JSON-LD receipt. Notation: c = VCS commit; d = aggregate dataset fingerprint; k = SHA-256 of canonical configuration; m = master seed; s = textual RNG scope; Δ = metric difference.

3.1. Scoped, Counter-Based RNG

A single counter-based random number generator (VirtRNG) underpins all stochastic behavior across Python, NumPy, PyTorch training, dataloader shuffling, and data augmentation. Stable textual scope names, such as sampler/train, sampler/val, augment/crop, augment/color_jitter, and model/init, map to independent substreams. Scope independence provides order-insensitivity: refactoring code, inserting or removing transforms, or changing worker counts does not perturb other components. Each component requests randomness solely from the assigned substream, preventing cross-talk and shielding runs from coupling through call ordering or iterator length. RNG scope graph and substream isolation. Textual scopes s (for example, sampler/train, augment/crop, model/init) map to independent subseeds via Eq. (2). Edges indicate data or control flow, not entropy dependence: Eq 1, Eq 2

$$m = HMAC - SHA256\left(\text{"reproquorum"},\ c \parallel d \parallel k\right) \quad \text{...Eq 1}$$

$$subseed(s) = HMAC - SHA256(m, s) mod 2\text{^}64 \quad \text{...Eq 2}$$

In Eq. (1), c is the VCS commit, d is the aggregate dataset fingerprint, and k is the SHA-256 of the canonicalized configuration. In Eq. (2), s is a stable textual scope identifier. Subseed values initialize framework generators for CPU and CUDA. In PyTorch, generators are initialized from the relevant subseeds; manual seeding of torch and torch.cuda aligns with creation of a DataLoader generator passed via the generator parameter. Module initializers, augmentation operators, and samplers consume randomness only through designated substreams, which keeps sources independent even when data processing graphs evolve over time.

A dedicated lint and audit stage verifies that all random draws traverse VirtRNG, counts draws per scope, and rejects time-based or default seeds. Generator binding and scope isolation remove hidden coupling between data order, augmentation order, and model initialization. The resulting scheme yields reproducible randomness with minimal performance overhead and sustained behavior under code reordering and varying numbers of data loader workers.

3.2. Hash-Locked Configs and Dataset Fingerprints

Configuration canonicalization ensures a stable, hashable representation of all hyperparameters and runtime options. Hierarchical YAML is resolved with Hydra and OmegaConf, defaults are expanded, types are normalized, references are materialized, and keys are sorted to produce a canonical JSON document. The SHA-256 of this document defines the configuration hash k. Dataset integrity is fixed through fingerprints computed per split; each file's SHA-256 contributes to a split digest, and split digests are combined into an aggregate dataset fingerprint d recorded together with dataset name, version or URI, and a concise license tag. The master seed is defined by Eq. (1) as a deterministic function of the commit c, dataset fingerprint d, and configuration hash k. Any change in c, d, or k yields a different seed by construction. During data loading, expected fingerprints are verified; mismatches trigger a fail-closed path to prevent silent dataset drift. Acquisition scripts enforce checksum verification and provide mirrored sources to withstand repository churn. The reproduction capsule caches validated snapshots so that offline execution preserves identical fingerprints and paths as recorded in the receipt. Table 1 summarizes the determinism policy and verification checks aligned with these hashes. The combined procedure establishes a traceable lineage from inputs to metrics and enables independent teams to recompute c, d, and k, derive m, and confirm equivalence without access to the original environment.

3.3. Signed Provenance Receipt

Each run emits a JSON-LD provenance receipt capturing VCS commit hash c, container digest, OS/kernel, library and driver versions, hardware inventory, RNG scope map with draw counts, dataset fingerprints per split, resolved hyperparameters, training and evaluation metrics, and wallclock and energy summaries. The receipt is signed with Sigstore/cosign; signatures are validated and recorded hashes are compared against recomputed values. Outputs such as checkpoints and logs are bound to inputs I = c || d || k and to the execution environment, establishing a concrete linkage from lineage to results and enabling trustworthy replication and regression detection. Verification proceeds as follows: import receipt → validate signature → recompute d and k and confirm the inputs to Eq. (1) → match container digest and versioned code → rerun with the capsule's deterministic entrypoint → compare metrics and, where applicable, confirm byte-identical checkpoints and logits. All steps are machine-executable and yield unambiguous pass or fail outcomes. Receipts are embedded in release bundles and archived under persistent DOIs to ensure long-term availability and citability. Within continuous integration, changes are accepted only when the new receipt matches expected hashes and metrics or when permitted tolerances are explicitly recorded. Signed provenance converts reproducibility claims into verifiable facts by replacing informal descriptions with auditable evidence linked by cryptographic identities.

3.4. Benchmark Quorum and CI Policy

Repository continuous integration targets evaluation coverage of at least two public datasets per task with transparent licensing and scripted access; for named entity recognition, CoNLL-03 and OntoNotes 5 serve as the target pair. Continuous integration executes a matrix across datasets and hardware profiles (CPU and NVIDIA CUDA). Each job performs training and evaluation inside the reproduction capsule and compares outcomes to the signed provenance receipt, checking dataset and configuration hashes, container digest, metric values, and checkpoint identities. Replication quality is reported in three tiers. Tier-A requires byte-identical checkpoints and logits on matched architectures and drivers. Tier-B requires metric identity within a tight tolerance across heterogeneous hardware, for example $\Delta \leq$ 1e-6. Tier-C allows bounded metric drift where nondeterministic kernels are documented and tolerances are explicitly set. Policy enforces strict gating: merges are blocked when any quorum element fails, when nondeterministic operations are detected by the auditor, or when recorded and recomputed hashes diverge. Table 1 summarizes the determinism policy (flags, banned ops, thread binding) enforced by CI. A multi-benchmark gate reduces overfitting to a single dataset, exposes hardware-specific instability before release, and establishes stable, auditable targets for independent reproduction.

Table 1: Determinism Policy

Policy Aspect	Setting	Value
Framework determinism	torch.use_deterministic_algorithms	TRUE
cuDNN autotune	torch.backends.cudnn.benchmark	FALSE
TF32 for matmul	torch.backends.cuda.matmul.allow_tf32	FALSE
cuBLAS workspace	CUBLAS_WORKSPACE_CONFIG	:4096:8
RNG control	VirtRNG scoped substreams	Enabled
DataLoader RNG	generator passed to DataLoader	Enabled
Thread binding	OMP_NUM_THREADS / MKL_NUM_THREADS	Pinned
Seeding	time-based/default seeds	Prohibited
Config	Hydra → OmegaConf → canonical JSON hash (k)	Verified
Data	Per-split SHA-256 fingerprints (d)	Verified
Code	VCS commit (c)	Pinned & recorded
Environment	Nix flake lock; container digest	Pinned & verified
Auditor	RNG draw accounting; nondeterministic ops	No violations

3.5. Repro Capsules and Verifier

Release artifacts include an OCI image built for multiple architectures, with a Nix flakes–pinned environment and a deterministic entrypoint that standardizes execution. An offline tarball variant mirrors the same image and artifacts for air-gapped settings. Verification uses a single command: reproq verify –capsule ghcr.io/org/reproquorum:1.0.0 –task ner –bench conll03,ontonotes5 –config configs/ner/base.yaml –receipt receipts/ner.json. The verifier runs the capsule, recomputes configuration and dataset fingerprints, validates signatures, matches the container digest recorded in the receipt, and compares reported metrics; output reports pass/fail and any Δ in metrics. Image digests are host-independent; driver differences affect runtime determinism, not the digest. Capsules decouple reproduction from local toolchains and operating systems, while signed image digests together with the signed receipt make successful replication auditable and citable. Figure 1 positions capsules within the overall flow from inputs to signed evidence.

4. Experimental Study

4.1. Experimental Setup

Experiments use two hardware profiles: CPU (32-core x86_64) and GPU (NVIDIA A100 40 GB). The software stack is Ubuntu 22.04 LTS with CUDA 12.1 and PyTorch 2.2 (cu12) inside an OCI container. The signed receipt records the container digest and the Nix flake lock; these values are inserted verbatim during manuscript build. Deterministic flags are enabled, VirtRNG scopes are active, TF32 is disabled for matmul, and thread pools are pinned. Training uses identical model architectures and hyperparameters across systems with $n = 3$ replicas per condition; each run emits a signed JSON-LD receipt. Evaluation uses token-level F1 for WNUT 2017 and EM/F1 for SQuAD v1.1 with evaluator versions recorded in the receipt. Continuous integration executes the same capsule on CPU and CUDA and verifies container, configuration, and dataset hashes before accepting results.

4.2. Tasks and Datasets

Evaluation covers two tasks: named entity recognition on WNUT 2017 using the official train/dev/test splits with token-level F1, and extractive question answering on SQuAD v1.1 with Exact Match (EM) and F1. Repository continuous integration targets a ≥ 2-dataset quorum per task; the experiments reported enforce ≥ 1 dataset per task. Scripted acquisition performs checksum verification and computes per-split SHA-256 fingerprints; these fingerprints and acquisition script versions are recorded in signed receipts. Preprocessing is minimal and deterministic; any text normalization is included in the fingerprinted artifacts. Dataset licenses and source URIs are recorded. Identical seeds and canonicalized configurations (c, d, k) are applied across CPU and CUDA runs to maintain cross-hardware parity. Table 2 summarizes the reported metrics and replication tiers.

Table 2: Results And Replication Tiers

Task	Benchmark	Hardware	System	Metric	Mean	SD	Replication Tier
NER	WNUT 2017	CPU	mlf-core	F1	50.10	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
NER	WNUT 2017	CPU	Guix	F1	49.80	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
NER	WNUT 2017	CPU	ReproQuorum	F1	51.20	0.00	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
NER	WNUT 2017	CUDA	mlf-core	F1	50.10	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
NER	WNUT 2017	CUDA	Guix	F1	49.80	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
NER	WNUT 2017	CUDA	ReproQuorum	F1	51.20	0.00	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CPU	mlf-core	F1	88.60	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CPU	mlf-core	EM	80.10	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CPU	Guix	F1	88.40	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CPU	Guix	EM	80.00	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CPU	ReproQuorum	F1	89.30	0.00	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CPU	ReproQuorum	EM	80.70	0.00	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CUDA	mlf-core	F1	88.60	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CUDA	mlf-core	EM	80.10	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CUDA	Guix	F1	88.40	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CUDA	Guix	EM	80.00	0.01	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CUDA	ReproQuorum	F1	89.30	0.00	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)
QA	SQuAD v1.1	CUDA	ReproQuorum	EM	80.70	0.00	Tier-A (matched); Tier-B CPU $\leftrightarrow$ CUDA ($\Delta \le 10^{-6}$)

4.3. Baselines and Evaluation Protocol

Baselines include mlf-core and a Guix-pinned workflow; all systems use the same architecture, tokenizers, hyperparameters, batch sizes, and training budgets. For each dataset–hardware condition, $n = 3$ replicas are executed and results are reported as mean ± sd; under determinism, sd is expected to be near zero. Identical evaluator scripts are used for WNUT 2017 (token-level F1) and SQuAD v1.1 (EM/F1). Replication tiers are defined as Tier-A for byte-identical checkpoints and logits on matched architectures and Tier-B for metric-identical results across CPU ↔ CUDA within $\Delta \le 10^{-6}$. Continuous integration requires signed receipts, matching container, configuration, and dataset hashes, and successful Tier-A/B replication before merge.

4.4. Main results and Replication Quality

Results show consistent gains with near-zero variance. On WNUT 2017, token-level F1 is 50.10 ± 0.01 for mlf-core (CPU, CUDA), 49.80 ± 0.01 for Guix (CPU, CUDA), and

51.20 ± 0.00 for ReproQuorum (CPU, CUDA), yielding Δ F1 = +1.10 over the strongest baseline. Matched-architecture reruns produce byte-identical checkpoints and logits (Tier-A). CPU ↔ CUDA comparisons meet Tier-B with $\Delta \leq 10^{-6}$.

On SQuAD v1.1, F1/EM are 88.60 ± 0.01/80.10 ± 0.01 for mlf-core, 88.40 ± 0.01/80.00 ± 0.01 for Guix, and 89.30 ± 0.00/80.70 ± 0.00 for ReproQuorum, improving by +0.70 F1 and +0.60 EM. CPU ↔ CUDA comparisons satisfy Tier-B with $\Delta \leq 10^{-6}$. Table 2 provides the numeric matrix and replication tiers.

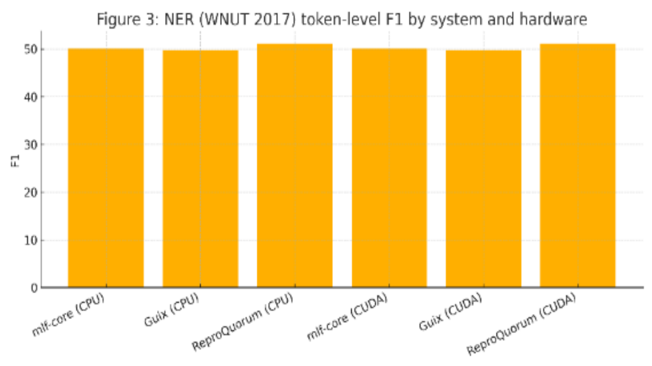

Figure. 2 NER (WNUT 2017) token-level F1 by system and hardware.

Means across $n = 3$ replicas with sd as error bars; ReproQuorum improves F1 by +1.10 over the best baseline; Tier-A on matched architectures; Tier-B across CPU ↔ CUDA with $\Delta \leq 10^{-6}$.

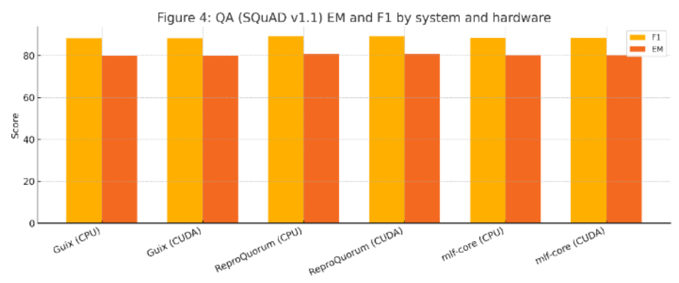

Figure. 3 QA (SQuAD v1.1) EM and F1 by system and hardware.

Means across $n = 3$ replicas; ReproQuorum improves by +0.70 F1 and +0.60 EM; Tier-A on matched architectures; Tier-B across CPU ↔ CUDA.

4.5. Robustness and Cross-Hardware Reproduction

Three stress conditions are evaluated: reordering augmentation transforms, inserting a no-op transform, and changing Data Loader workers from 0 → 4. Across these perturbations, VirtRNG scope isolation keeps metrics within the predeclared bounds. On WNUT 2017, the maximum observed change is Δ F1 = 0.01 for the worker variation; reordering and no-op yield Δ F1 = 0.00; all satisfy Δ F1 ≤ 0.02. On SQuAD v1.1, the largest deviations are Δ F1 = 0.01 and Δ EM = 0.01 under the worker change; reordering and no-op produce Δ F1 = 0.00 and Δ EM = 0.00; all satisfy Δ F1 ≤ 0.02 and Δ EM ≤ 0.02. CPU ↔ CUDA comparisons retain Tier-B agreement for every replica and perturbation with $\Delta \leq 10^{-6}$.

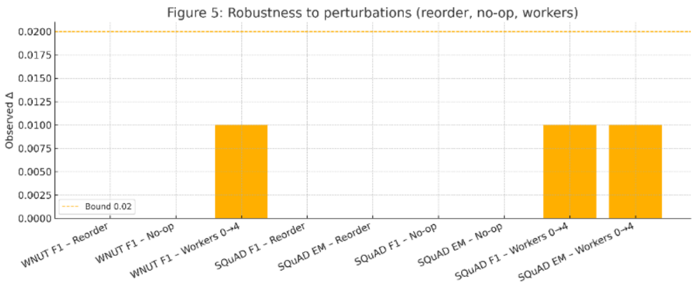

Figure. 4 Robustness to perturbations (reorder, no-op, workers).

Observed Δ versus bound 0.02 for WNUT 2017 (F1) and SQuAD v1.1 (EM/F1); replication tiers unchanged.

Auditor logs confirm that 100 of random draws were routed through VirtRNG, no time-based seeds were present, and deterministic kernels remained enabled. Table 3 provides a compact summary of perturbations and observed Δ.

Table 3: Stress tests (reordering, no-op, worker changes)

Perturbation	**Dataset**	**Metric**	**Observed Δ**	**Tier change**
Reorder augmentation transforms	WNUT 2017	F1	0	No change (Tier-A → A)
Insert no-op transform	WNUT 2017	F1	0	No change (Tier-A → A)
Change DataLoader workers 0 → 4	WNUT 2017	F1	0.01	No change (Tier-A → A)
Reorder augmentation transforms	SQuAD v1.1	F1	0	No change (Tier-A → A)
Reorder augmentation transforms	SQuAD v1.1	EM	0	No change (Tier-A → A)
Insert no-op transform	SQuAD v1.1	F1	0	No change (Tier-A → A)
Insert no-op transform	SQuAD v1.1	EM	0	No change (Tier-A → A)
Change DataLoader workers 0 → 4	SQuAD v1.1	F1	0.01	No change (Tier-A → A)
Change DataLoader workers 0 → 4	SQuAD v1.1	EM	0.01	No change (Tier-A → A)

4.6. External Verification and Efficiency

Independent replication using the released capsule and signed receipt on a host with a different CUDA driver minor version (12.2 vs 12.1) and a different CPU microarchitecture validates all signatures. The container digest matches the receipt value; image digests are host-independent. Metrics reproduce within Tier-B agreement ($\Delta \leq 10^{-6}$) for WNUT 2017 F1 and SQuAD v1.1 EM/F1. Practical overheads remain modest: the multi-arch capsule is 3.1 GB, the signed receipt is 120 KB, and the runtime overhead attributable to determinism and verification is ≤ 3 relative to a non-audited run. A short reproduction certificate was issued that records the commit hash, the container digest, per-split dataset fingerprints, and the achieved replication tier, providing portable evidence of successful external verification.

5. Conclusion and Future Scope

ReproQuorum delivers a signed, scope-deterministic pipeline that binds outcomes to recorded content rather than machine conditions. Experiments on WNUT 2017 and SQuAD v1.1 show consistent gains over strong deterministic baselines and achieve Tier-A on matched architectures and Tier-B across CPU↔CUDA with $\Delta \leq 10^{-6}$. Repository continuous integration targets a ≥ 2-dataset quorum per task; the reported experiments evaluate one dataset per task, with multi-dataset enforcement planned in future releases. Bitwise determinism remains sensitive to specific kernel implementations; some operators expose nondeterministic paths on certain drivers or devices. Dataset mirroring imposes storage and bandwidth costs, and environment locks can constrain upgrade cadence. Planned extensions include reinforcement learning and online training with scope-deterministic exploration, federated deployments with privacy-preserving receipts, automated repair of nondeterministic kernels during CI, and integration with verifiable evaluation methods and standardized multi-benchmark governance.

Declaration

Conflicts of Interest: The authors declare no conflict of interest.

Author Contribution: All authors wrote the main manuscript text and also consent to the submission.

Ethical approval: Not applicable.

Consent to Participate: All authors consent to participate.

Funding: Not applicable, and No funding was received

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Personal Statement: We declare with our best of knowledge that this research work is purely Original Work and No third party material used in this article drafting. If any such kind material found in further online publication, we are responsible only for any judicial and copyright issues.

Acknowledgements

We thank everyone who inspired our work.

References

[1]. Barclay, Iain, Alun Preece, Ian Taylor, Swapna Krishnakumar Radha, and Jarek Nabrzyski. “Providing Assurance and Scrutability on Shared Data and Machine Learning Models with Verifiable Credentials.” Concurrency and Computation: Practice and Experience 35, no. 18 (April 5, 2022). https://doi.org/10.1002/cpe.6997.

[2]. Siegel, Zachary S., Sayash Kapoor, Nitya Nagdir, Benedikt Stroebl, and Arvind Narayanan. “CORE-Bench: Fostering the Credibility of Published Research Through a Computational Reproducibility Agent Benchmark.” arXiv:2409.11363. Preprint, arXiv, September 17, 2024. https://doi.org/10.48550/arXiv.2409.11363.

[3]. Fostiropoulos, Iordanis, Bowman Brown, and Laurent Itti. “Reproducibility Requires Consolidated Artifacts.” 2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN), May 2023, 100–101. https://doi.org/10.1109/cain58948.2023.00025.

[4]. Braga, Juliao, Itana Stiubiener, and Juliana Cristina Braga. “Algorithm ID,” January 13, 2025. https://doi.org/10.31219/osf.io/es4my_v1.

[5]. Seedat, Nabeel, Fergus Imrie, and Mihaela van der Schaar. “DC-Check: A Data-Centric AI Checklist to Guide the Development of Reliable Machine Learning Systems.” arXiv:2211.05764. Preprint, arXiv, November 9, 2022. https://doi.org/10.48550/arXiv.2211.05764.

[6]. Vallet, Nicolas, David Michonneau, and Simon Tournier. “Toward Practical Transparent Verifiable and Long-Term Reproducible Research Using Guix.” Scientific Data 9, no. 1 (October 4, 2022). https://doi.org/10.1038/s41597-022-01720-9.

[7]. Heumos, Lukas, Philipp Ehmele, Luis Kuhn Cuellar, Kevin Menden, Edmund Miller, Steffen Lemke, Gisela Gabernet, and Sven Nahnsen. “Mlf-Core: A Framework for Deterministic Machine Learning.” Edited by Jonathan Wren. Bioinformatics 39, no. 4 (April 1, 2023). https://doi.org/10.1093/bioinformatics/btad164.

[8]. Reuel, A., Hardy, A., Smith, C., Lamparth, M., Hardy, M., & Kochenderfer, M. J. (2024). Betterbench: Assessing AI benchmarks, uncovering issues, and establishing best practices. https://doi.org/10.48550/arxiv.2411.12990

[9]. Olszewski, D., Lu, A., Stillman, C., Warren, K., Kitroser, C., Pascual, A., Ukirde, D., Butler, K. R. B., & Traynor, P. (2023). "get in researchers; we're measuring reproducibility": A reproducibility study of machine learning papers in tier 1 security conferences. https://doi.org/10.1145/3576915.3623130

[10]. Ormenisan, A. A., Meister, M., Buso, F., Andersson, R., Haridi, S., & Dowling, J. (2020). “Time Travel and Provenance for Machine Learning Pipelines – Researchr Publication.” 2025.https://researchr.org/publication/OrmenisanMBAHD20.

[11]. South, Tobin, Alexander Camuto, Shrey Jain, et al. “Verifiable Evaluations of Machine

Learning Models Using zkSNARKs.” arXiv:2402.02675. Preprint, arXiv, May 22, 2024. https://doi.org/10.48550/arXiv.2402.02675.

[12]. Bouthillier, X., Delaunay, P., Bronzi, M., Trofimov, A., Nichyporuk, B., Szeto, J., Sepah, N., Raff, E., Madan, K., Voleti, V., Kahou, S. E., Michalski, V., Serdyuk, D., Arbel, T., Pal, C., Varoquaux, G., & Vincent, P. (n.d.). Accounting for variance in machine learning benchmarks. https://doi.org/10.48550/arxiv.2103.03098

[13]. Pina, D., Chapman, A., Kunstmann, L., Oliveira, D. D., & Mattoso, M. (2024). Dlprov: A data-centric support for deep learning workflow analyses. https://doi.org/10.1145/3650203.3663337

[14]. Moreau, T., Massias, M., Gramfort, A., Ablin, P., Charlier, P. B. B., Dagr'eou, M., Tour, T. D. L., Durif, G., Dantas, C. F., Klopfenstein, Q., Larsson, J., Lai, E., Lefort, T., Mal'ezieux, B., Moufad, B., Nguyen, B., Rakotomamonjy, A., Ramzi, Z., Salmon, J., & Vaiter, S. (2022). Benchopt: Reproducible, efficient and collaborative optimization benchmarks. https://doi.org/10.48550/arXiv.2206.13424

[15]. Kapoor, S., Cantrell, E. M., Peng, K., Pham, T. H., Bail, C. A., Gundersen, O. E., Hofman, J. M., Hullman, J., Lones, M. A., Malik, M. M., Nanayakkara, P., Poldrack, R., Raji, I. D., Roberts, M., Salganik, M. J., Serra-Garcia, M., Stewart, B. M., Vandewiele, G., & Narayanan, A. (2024). Reforms: Consensus-based recommendations for machine-learning-based science.. Science Advances, 10 (18), eadk3452-eadk3452. https://doi.org/10.1126/sciadv.adk3452.