NeuroShard: Native Settlement for Continual, Verifiable Language Model Evolution

Protocol working draft and two-host experiments

LZ

September 11, 2026

Abstract

NeuroShard aims to maintain a shared language model through openly contributed computation, new training data, and native token incentives. This requires four executable mechanisms: continual data and evaluation, distributed training of the complete model, verifiable execution without repeating the whole model in every block, and architecture growth. We implement experimental versions of these mechanisms. A tied-weight Llama seed with 134,515,008 parameters trains across three workers on two CPU hosts. A deterministic expansion adds four residual blocks, producing 148,675,392 parameters and requiring a fourth worker under a declared 48-million-parameter limit per worker. Initial expansion preserves the tested logits exactly. A separate four-validator native chain rejects a forged update through a bounded-stage replay dispute and settles a valid update with exactly one experimental token of issuance. The dispute publishes 170,169,856 bytes; it is a coarse executable upper bound, not a succinct proof. We also identify a misleading evaluation regime dominated by prompt tokens and repeat the comparison with assistant-response targets. Neither candidate in the paired response-training comparison passes the predefined improvement gate, and the larger candidate has no demonstrated advantage under the same budget. A further run from the original seed with new examples and a higher learning rate also fails, increasing fresh response loss. These results establish concrete components, not perpetual quality improvement or a production permissionless deployment. Native promotion of evolving models, rolling data activation, sustainable audit incentives, and public worker admission remain integration and research obligations.

1 Objective and scope

NeuroShard's objective is an ongoing public learning service. Participants contribute training, inference, verification, or storage; its own blockchain settles duties and payments. The model should be able to consume fresh licensed data, improve on measured tasks, and gain useful capacity when additional resources become available. A particular seed checkpoint is an initial condition, not a permanent size ceiling.

Four properties must be distinguished. Execution correctness asks whether the prescribed computation occurred. Consensus asks which state transition and payment became final. Learning quality asks whether a candidate improves a chosen task distribution. Resource scaling asks whether a larger computation can run at an acceptable cost. A signature proves none of the latter three, and lower training loss does not establish general intelligence.

This draft supersedes the earlier five-page manuscript, preserved in `docs/archive`. It specifies the target protocol separately from the implemented experiments. The released 0.4.0 public testnet remains a bounded adapter-training and paid-inference demonstrator. The new code resides in

`neuroshard.evolution`; its experiments do not silently migrate public balances or replace the public serving model.

2 Related foundations

DiLoCo reduces synchronization between complete model replicas; it does not eliminate communication between the layers of a model-parallel replica or establish Byzantine correctness [1]. SWARM studies training pipelines over heterogeneous, unreliable resources [2]. We use these as distributed-systems foundations, while treating adversarial execution as a separate problem.

Function-preserving model expansion follows the general motivation of Net2Net [3]. Our implemented transformation is specifically an identity residual extension for a tied-weight Llama decoder. It neither reproduces every Net2Net transformation nor establishes that extra depth improves downstream quality.

Verde studies refereed delegation of machine-learning computation under an at-least-one-honest-provider assumption and uses reproducible operators to address hardware-dependent arithmetic [4]. Our current referee is much coarser: it replays an entire bounded worker stage. The verifier's dilemma is directly relevant: when correct verification is costly, rational participants can prefer to skip it [5]. Fraud bounties alone are not a demonstrated sustainable audit market.

3 Roles, assumptions, and state

3.1 Roles

The protocol has validators, task coordinators, compute workers, auditors, data contributors, storage replicas, inference providers, and customers. An account may perform several roles. The number of keys is never treated as evidence of independent operators. Coordinators assemble a job and may be replaced after an objective lease deadline. A public implementation must support competing coordinators and worker discovery without a mandatory operator-owned endpoint.

3.2 Assumptions

Native consensus uses bonded BFT membership and assumes less than one third Byzantine voting weight, with eventual synchrony for progress [6]. Signatures and content hashes are assumed secure. An optimistic execution claim is sound only if an honest, online observer checks the relevant graph, obtains its inputs, and can complete a dispute before its deadline. Sampling fewer stages gives a probabilistic condition, not full coverage by definition.

Learning evaluation additionally assumes sufficiently representative, uncontaminated data. Publicly available tests can be rehearsed off-chain. Committing a model before choosing a public test subset does not prevent a participant from training on the entire public pool. Independent curation, changing tasks, exposure accounting, and external evaluation are therefore part of the quality mechanism.

3.3 State commitments

The target state contains accounts, validator bonds, active task reservations, work commitments, disputes, data manifests, model revisions, evaluation tickets, serving aliases, inference escrows, and emission counters. A model revision commits architecture, component tensor roots, numerical semantics, tokenizer identity, and ancestry. A semantic computation identifier should commit weights,

architecture, data, objective, and optimizer settings independently of arbitrary history or job nonces; it prevents rewarding a cached transition repeatedly after a rollback.

The learning head and serving head are separate. Correctly executed training may advance a learning candidate without changing customer-facing inference. Only the quality-promotion transition changes the serving alias. Rejected candidates and unsuccessful work remain inspectable in history. Rollback changes the alias; it does not delete blocks, reverse paid valid work, or authorize paying again for the same semantic computation.

The implemented paid-task identifier commits parameter-component bytes, architecture, effective token inputs and targets, and optimizer settings. It normalizes equivalent explicit/implicit target masks and numeric configuration representations, and omits model ancestry, allocation layout and job nonces. The ledger rejects a task already paid under that identifier. This matters even without rollback: at an exact fixed point, unchanged weights can otherwise be wrapped in endlessly new model-manifest ancestry and rewarded again. A regression test executes this case and permits only its first payment. The identifier is an explicit protocol equivalence rule, not a solution to arbitrary program equivalence or proof of fresh physical energy expenditure.

4 Data as a versioned protocol input

A dataset identity is a content digest, not a mutable S3 key or repository branch name. Source manifests name the repository, full revision, split, license, parser/tokenizer profile, and cursor range. Object replicas may use S3, ordinary HTTP, or peer storage; none is the definition of the dataset.

The implemented collector uses pinned Parquet inputs with checksum verification, a durable cursor, immutable document and sequence objects, normalized document identifiers, and a SimHash near-duplicate screen. Exact document grouping precedes token-window assignment. The SimHash screen is a heuristic; it does not prove absence of paraphrases or semantic overlap. Training and protected evaluation roles share an exclusion registry within a corpus.

Each epoch collects a bounded new window, mixes it with historical replay, and commits the resulting batches. The current replay fraction is 25% when historical data exists. A daily resource budget can renew without erasing the cursor, model ancestry, or ledger. Exhausting a source causes a pause or a separately authorized source addition, not fabrication of fresh examples. A new source must supply provenance and an admissible license. Permissionless submission does not imply automatic acceptance of arbitrary text.

4.1 Assistant-response objectives

A first experiment used 64-token conversation prefixes and scored every next token. In its three evaluation groups, only 17, 18, and 10 of 64 examples contained any first-response tokens. This was an inadequate basis for a claim about improved assistant answers.

The preparation used for the reported paired quality experiments keeps up to 64 preceding prompt tokens and 64 tokens of the first assistant response. It pads only after the response. Labels equal the actual next token for response positions and the explicit ignore marker elsewhere. Every row must contain at least one scored target; invented target tokens and all-ignored rows are rejected. This remains a truncated-context response-loss experiment, not a general dialogue benchmark.

For token sequence x and target mask m, the loss is

$$L(\theta; x, m) = -\frac{\sum_{t=1}^{T-1} m_t \log p_\theta(x_{t+1} \mid x_{\le t})}{\sum_{t=1}^{T-1} m_t}. \tag{1}$$

Both x and m are committed task inputs. A worker cannot choose an easier mask after observing its outputs.

4.2 Content-addressed text semantics

The maintained epoch implementation now binds a model to a serialized fast-tokenizer backend, exact vocabulary and special-token definitions, chat template, bounded rendering rule, and tokenizer-library versions. A SHA-256 codec root appears in the model manifest, corpus settings and prepared windows. Equal vocabulary size does not establish equal token meaning. Training and identity depth growth preserve this root; mixed-codec batches and an implicit vocabulary replacement are rejected. Loading a codec reconstructs and reserializes its backend, and verifies runtime and canonical identity.

The supported template subset excludes clock access, helper calls, filters, nested or recursive loops and object introspection. Message loops and concatenation are bounded, and rendering streams through an output-size limit. Message content cannot contain reserved chat-control tokens. Encoding adds no second set of special tokens and performs no implicit truncation. These rules establish reproducible framing for the supported seed, not compatibility with every template or resistance to semantic prompt injection.

For each nonempty assistant turn, the new preparation requires a stable generation prefix and an actual end-of-response token. It divides real response targets into bounded windows, retaining preceding conversation/answer tokens as context and masking prompt and padding positions. Chunk boundaries do not invent end-of-response targets. Within a document's prepared windows, each retained target appears once. A finite budget rotates through first chunks of all turns before later chunks and records omitted targets. Training may retain such a partial document; evaluation ingestion rejects one whose response targets cannot all fit within the declared budget. This produces a measurable length-selection bias, and complete target coverage still uses truncated context.

Evaluation now reserves whole documents, including all their windows. If window j in document d has n_{dj} scored causal targets and mean loss ℓ_{dj}, its document observation is

$$\ell_d = \frac{\sum_j n_{dj}\ell_{dj}}{\sum_j n_{dj}}.$$

The paired quality gate receives one observation per document rather than treating correlated chunks as independent examples. Parent and candidate must use the same codec for this token-loss comparison. A future vocabulary migration requires explicit embedding/output-head conversion, retokenization of retained raw data, compatible clients, and evaluation on common raw-text or task outcomes; per-token losses across different tokenizations are not directly comparable. Continuous learning of new concepts does not itself require a growing vocabulary.

These are implemented text and data rules. Native settlement still checks prescribed numerical batches; it does not independently certify their raw-text provenance or usefulness. The quality measurements later in this paper retain their historical first-response objective and are not reinterpreted as results of this newer document-window policy.

5 Full-model training across workers

5.1 Ownership and schedule

The seed is the Apache-2.0 SmolLM2-135M-Instruct checkpoint at revision `12fd25f77366fa6b3b4b768ec3050bf629380bac` [7]. It contains 134,515,008 trainable parameters. The implementation imports its pinned files into float32 tensor components: one embedding, one final norm, and thirty

transformer blocks. All parameters participate in training; there is no frozen backbone plus a small adapter in this experiment.

The first worker owns the tied embedding and output head, the final norm, and an initial contiguous block range. Later workers own successive contiguous block ranges. Forward activations travel through the workers; the final hidden state returns to the first worker for the language-model loss. Its output-head gradient travels backward through the stages. The first worker accumulates its output-head and input-embedding contributions into the same parameter tensor.

Keeping the tied head with the embedding avoids sending its dense weight gradient across the network every step. It also creates an asymmetric first stage and a lower bound on that worker's capacity. Placement must account for this asymmetry; summing advertised memory alone is insufficient.

5.2 Optimizer and numerical contract

The current optimizer is clipped SGD, chosen to make state and replay explicit. For worker gradients g_j, the norm and update are

$$n^2 = \mathrm{fsum}_j \, \|g_j\|_2^2, \tag{2}$$

$$a = \min\left(1, \frac{c}{\sqrt{n^2} + 10^{-6}}\right), \qquad \theta'_j = \theta_j - \eta a g_j. \tag{3}$$

Each stage sums squared gradients in float64 with an explicit parameter order; the resulting scalar is committed as a hexadecimal float. The whole optimizer state is the weights and prescribed step policy; Adam moments are not silently omitted from an Adam implementation.

The tested profile uses CPU float32, single-thread execution, disabled MKLDNN, deterministic algorithms, a default ATen CPU path, and MKL SSE4.2 instructions. Package startup establishes this profile before helper modules initialize PyTorch. An early native-chain test exposed the importance of this order: all validators agreed with each other but disagreed with correctly configured workers. The corrected entry point reproduces the reference trace. Agreement alone is not numerical conformance.

The current bounds are 48 million resident parameters per stage, at most 64 stages, at most four rows and 512 total batch tokens, and at most 256 tokens in a row. These are execution-profile bounds, not proofs of a universal memory peak. The participant still needs room for gradients, activations, serialization, and runtime overhead. Other hardware and kernels require an explicit compatible numerical profile and conformance tests.

5.3 Recovery

Workers durably record operation intents and results. An operation identity binds session, step, phase, parent, and inputs. A repeated request returns the committed result; a conflicting request cannot occupy the same operation position. After interruption, a worker reloads the last complete parameter checkpoint and rebuilds only the unfinished graph. Coordinator journals retain the pending batch so a retry cannot silently train a different example. Tests cover a lost acknowledgment after backward computation and resumption through the next step.

Replacement workers need the same committed components and numerical semantics. Data replication is therefore a prerequisite for recovery, not an optional optimization. Current cross-host experiments use authenticated loopback RPC through SSH. They do not constitute a public worker-discovery protocol.

6 Verification and native settlement

6.1 Reservation and compact claim

A coordinator first reserves the current parent and training round on-chain, naming the reward identities. The reservation locks collateral and expires objectively. A submitted claim includes a model transition, a compact execution transcript, the assigned batch, and signed stage receipts that bind the reservation. Relaying a published transcript cannot replace the preassigned reward identities. Replay proves a result, not which physical processor performed it; subcontracting is compatible with the contract.

The ordinary state-machine check verifies exact component coverage, layer order, parent and step agreement, activation and gradient links, output-head dependencies, gradient-norm inputs, global clipping, optimizer settings, and candidate assembly. It also checks the assigned data and signatures. It does not load neural weights or repeat the model computation to accept an ordinary claim.

6.2 Why local replay can expose a global forgery

The transcript fixes all data dependencies. If every local transition is valid on its committed inputs and all edges connect to their prescribed predecessors, their composition is the prescribed graph execution. Conversely, for a forged graph with correctly bound initial inputs, either a structural dependency fails or at least one local transition is invalid. Checking one arbitrary stage does not establish that all stages are valid; an observer must locate an invalid transition or provide the stated coverage guarantee.

The implemented referee loads only one stage's parameters, re-executes its forward pass, its head operation when applicable, backward pass, and optimizer update, and compares the exact committed outputs. Incorrect shapes, nonfinite values, and invalid target masks are execution failures. The first-stage replay includes both tied-weight gradient contributions.

6.3 Availability and disputes

An availability challenger names an input or output object associated with the pending claim. The responsible publisher must place its chunks in native transactions and seal them against the requested digest before the deadline. A missing response rejects the claim. Third parties cannot poison an authorized publisher's upload sequence.

For a computation dispute, the challenger publishes all required replay inputs through finalized native transactions. Only then may the native referee execute. Validators do not fetch arbitrary off-chain URLs while deciding a block. Missing local durable block data is a node-recovery fault, not a permission to derive a different verdict from network timing.

Uploads are bounded and sequential. False or abandoned computation challenges burn their collateral; returning all of it to a colluding claimant would make repeated self-challenges nearly free. A claim also has an absolute lifetime so challenges cannot lock the single work slot forever. These mechanisms bound specific abuse paths. They do not establish that the chosen token prices fund real network and verification costs.

The terminal replay is deliberately coarse. The two-host dispute published 170,169,856 bytes in 184 upload/seal transactions, requiring 310.05 seconds. Resolution took 10.14 seconds including native transaction processing. This is usable for an experimental objective reference, but expensive for an adversarial public market. Finer authenticated operator transcripts and dispute narrowing should be evaluated against this measured bound; they are not already implemented by calling a stage a succinct proof.

6.4 Total verification cost and sampling limits

The compact check is not the entire verification budget. An honest observer that covers every stage still performs roughly another graph execution outside consensus. If V validators perform compact checks of cost c_s, observers spend c_a per accepted graph, and a fraction q of claims requires a dispute with replay cost c_r and publication cost c_d, an indicative budget is

$$C_{\text{verification}} = Vc_s + c_a + q(Vc_r + c_d). \tag{4}$$

The terms must use consistent resource prices; token amounts alone do not measure CPU, bandwidth, storage, or the opportunity cost of a stalled queue. Our experiments measure components of this expression, not its equilibrium under adversarial demand.

For f invalid stages among S, uniformly sampling k distinct stages after a binding commitment detects at least one with probability

$$p = 1 - \frac{\binom{S-f}{k}}{\binom{S}{k}}. \tag{5}$$

This formula requires the stated sampling conditions and does not establish coverage for the implemented optimistic claims. In a simplified one-shot comparison, if cheating saves compute worth ΔC, forfeits reward R and bond B when detected, its advantage over honest execution is $\Delta C - p(R + B)$. A deterrence inequality therefore depends on detection, actual resource prices and exposure of collateral, not merely on naming a slashing percentage. Collusion, adaptive faults, repeated play and attacks on availability require additional analysis. We do not replace complete observer coverage with a small random sample and then retain the same correctness claim.

6.5 Finality, rewards, and serving state

A disputed invalid transition is rejected before reward issuance. An unchallenged eligible transition settles after its challenge window, relying on the explicit observer assumption. The model transition and rewards commit atomically. Native membership remains independent of which worker computed a gradient; newly issued stake affects voting only through the prescribed bonding and activation rules.

The implemented experiment renews a per-period task budget and issues one experimental token per settled training step. It pays the reserved workers. It does not yet implement a sustainable positive-payment audit market. A production policy must allocate audit and availability funding as explicit expenses, including payment when no fraud is found. Rewarding only fraud discoveries creates the verifier's dilemma. The inflation schedule, long-term fee support, collateral pricing, and initial allocation require public review before an economic launch.

7 Growing the model

For a residual decoder block, write

$$u = h + W_O\ \mathrm{Attn}(\mathrm{RMS}(h)), \tag{6}$$

$$B(h) = u + W_D\left(\mathrm{SiLU}(W_G\,\mathrm{RMS}(u)) \odot W_U\,\mathrm{RMS}(u)\right). \tag{7}$$

The growth transformation copies an existing block's compatible parameters and initializes its attention output projection W_O and MLP down projection W_D to zero. In exact arithmetic, this yields $B(h) = h$. In the implemented floating-point test, finite intermediate activations yield identical

tested logits. New output projections receive gradients immediately; inner copied projections can receive gradients once those outputs move away from zero. The added parameters are independent trainable tensors even when initial component bytes coincide.

Four added blocks increase the model to 148,675,392 parameters. The three-worker assignment fails under the declared 48-million-parameter cap, whereas four workers admit it. This demonstrates resource-aware admission and actual parameter growth. It does not establish that either physical host could not hold the whole model under a different allocation.

A growth candidate must preserve old components and tokenizer identity, respect the execution bounds, obtain a feasible resource assignment, train, and pass evaluation before serving. Growth is not a periodic instruction to increase parameter count regardless of value. The evaluation should include a same-training-budget control and measured inference/storage costs. Wider models, expert routing, vocabulary expansion, and a change in numerical profile are distinct transformations requiring their own replay and migration rules.

The native application now admits a separate bonded `grow` claim. Its compact check preserves old tensor roots and architecture fields, restricts added depth, and verifies placement under declared capacities. A growth dispute recomputes the new block bytes using only the last parent block. Successful growth changes the learning head without issuing a training reward or changing the serving alias. A per-period growth budget limits frequency. Subsequent worker sessions begin at the ledger's existing training round, so architecture changes do not reset issuance history. Declared capacities still do not prove that suitable independent workers are online; admission and recovery must make that resource condition operational before public use.

8 Quality promotion and continued operation

The target protocol treats quality evaluation as a separately committed, challengeable computation. A candidate is fixed before the evaluation ticket resolves. The ticket names the baseline, data-selection rule, exposure limits, target construction, metrics, and decision thresholds. Each loss report must link to its model, protected examples, and verifiable forward/head operations. A signed scalar claiming that a model is better is insufficient.

For paired per-document losses, let $d_i = L_i(\theta') - L_i(\theta)$. The research controller computes $\bar{d} + 2.576\, s_d/\sqrt{n}$ as an approximate 99% normal upper bound. Fresh-data improvement must exceed 0.001 nats per scored token; retention allows an upper bound below a 0.02-nat regression margin. At least 32 paired examples are required, and the main experiment uses 64 per group. These choices are explicit research thresholds, not a universal significance or safety guarantee. Dependence, multiple model comparisons, and distribution shift limit their interpretation.

A separate untouched test group is reported after candidate construction. The initial experiment's prompt-dominated metric is retained as a negative methodological result, not repackaged as response improvement. The corrected experiment scores actual response positions on subsequent held-out source rows.

The retention, fresh and test groups here are disjoint hash partitions of the same pinned upstream test split. "Fresh" means previously unused examples in this experiment; it does not establish acquisition of newly published facts or retention across all previous domains. The seed's prior training contamination is not measured by these tests.

An indefinite deployment also needs an error budget across decisions and long-term reference checkpoints. A fixed false-positive allowance reused forever cannot provide a fixed whole-history guarantee. For example, a policy could allocate $\alpha_t = \alpha/[t(t+1)]$ to epoch t and subdivide it among its metrics. If each epoch's tests have the required conditional validity, the union bound gives

Pr(any false approval) $\leq \sum_t \alpha_t = \alpha$. This is a design rule for valid tests, not a guarantee supplied by our small normal-approximation experiment. Similarly, comparing retention only to the last model can accumulate small tolerated regressions; a deployment must also enforce domain-specific floors against durable reference checkpoints. These online statistical and long-term retention policies are not implemented by the current two-group research gate.

The local epoch controller persists data collection, training, candidate commitment, evaluation selection, results, and the accept/reject decision. A rejection keeps the previous accepted model. Restart does not train twice, resample an already reserved evaluation ticket, or consume a second daily epoch allowance for the same active run. This local research registry is not yet the native serving registry.

The target native transitions are: propose a data window; challenge and activate its provenance/task commitments; reserve and execute work; settle or reject the execution; open an evaluation ticket; settle its verifiable metrics; and atomically promote or retain the serving alias. Growth uses the same candidate/evaluation path with an additional architecture certificate. A rollback points to a previously eligible serving revision and prohibits duplicate semantic-work rewards. These transitions can occur without replacing the chain's consensus history or balances.

9 Inference paid in native tokens

A customer request must bind the finalized serving model, tokenizer, prompt commitment or explicitly public prompt, decoding rule, output limit, provider assignment, expiry, and maximum price. It escrows existing tokens rather than minting a new training reward. A result transcript binds every generated token to the same checkpoint and decoding state. Settlement transfers only the agreed fee, once; expiry returns unused escrow. Model promotion cannot change the checkpoint of an in-flight request.

Distributed greedy inference is exercised by the evolution pipeline. Paid inference is deployed in the separate 0.4.0 network. Connecting paid inference to an evolving model registry, including challengeable response transcripts, availability, and versioned cache recovery, is not complete in the new native application. These two facts must not be combined into an unsupported claim of a deployed evolving paid-inference network.

An assistant resembling a general-purpose personal agent also needs retrieval, user-controlled memory, tools, authorization, privacy, and task-level evaluation. Growing a public LM does not supply those properties by itself. Public blockchain execution does not provide confidential prompts or confidential memories.

10 Experimental evidence

Experiments use two authorized CPU hosts, each with four virtual CPUs and 16 GiB RAM, under one operator. Each guest reports two cores with two threads per core. The processors are Intel Xeon Platinum 8259CL and 8175M, respectively. They test real distributed processes and a native BFT ledger, but not independent economic participation or an adversarial population.

Check	Observed result
Seed implementation vs. pinned reference	Exactly equal tested float32 logits
Identity expansion	134,515,008 to 148,675,392; equal initial tested logits
Distributed whole-model step	Three workers; 8.95 seconds in the conformance run
Compact graph check	0.0246 seconds in that run
Independent stage replay	All three stages reproduced exactly
Native forged-update dispute	Rejected for optimizer-update mismatch after the startup fix
Native valid update	Accepted; exactly 1,000,000 atoms issued
Native state agreement	Matching app hash on all four validators at a common height
Recovery tests	Lost acknowledgment and coordinator/worker restart tests pass
Evaluation correction	Prompt-heavy prefix scoring replaced by explicit response targets

10.1 Response-only quality comparison

Each value is mean teacher-forced response loss in nats per scored token over 64 paired examples. The two response candidates start from the same parent weights and receive the same 32 response-masked batches; the larger candidate first adds four identity-initialized blocks.

Checkpoint	Retention	Fresh	Untouched test
Original seed (135M)	1.325133	1.281919	1.285737
Prefix-trained parent (135M)	1.325136	1.284188	1.285587
Response-trained candidate (135M)	1.324262	1.282754	1.285286
Response-trained candidate (149M)	1.324338	1.282783	1.285116

Neither response candidate passes the predefined promotion gate. Relative to their common parent, fresh-loss changes are -0.001434 and -0.001405 nats, but their approximate 99% upper bounds are +0.000639 and +0.000596; the required bound is below -0.001. The larger candidate has no demonstrated fresh-loss advantage over the smaller candidate under the same training budget. Relative to the original seed, both candidates also fail the improvement gate. These are negative promotion results, not evidence of continual capability improvement. All candidates produced the same tested 32-token greedy answer prefix to the single solar-panel prompt; that anecdotal output is not a quality benchmark. The chosen thresholds were not relaxed after observing these results.

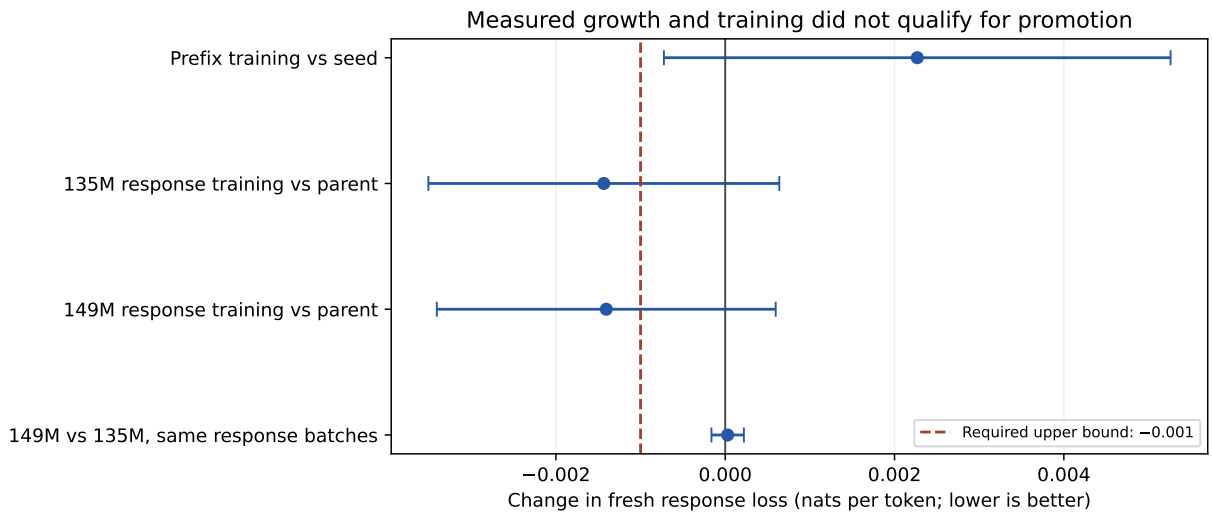

Figure 1: Paired fresh-response loss changes with approximate 99% normal intervals, 64 examples per comparison. Lower is better. Every interval crosses zero; none establishes the required improvement. Intervals are not corrected for multiple comparisons.

10.2 Exploratory response training from the seed

A subsequent plan, recorded before its training began, starts from the original 135M seed and uses 32 batches of two response-masked examples, SGD learning rate 0.03 and the same norm cap. Training uses subsequent source rows 1536–1919; protected examples come from test rows 1536–2303. Each evaluation group contains 128 previously unused paired examples, selected after the candidate commitment. This is an exploratory follow-up motivated by earlier failures, not an independently preregistered confirmatory experiment. The promotion thresholds are unchanged.

Group	Seed	Candidate	Change	99% upper bound
Retention	1.244962	1.249179	+0.004217	+0.011743
Fresh	1.129396	1.138708	+0.009312	+0.017282
Untouched test	1.311664	1.313363	+0.001700	+0.009115

Promotion is rejected: retention remains within the chosen tolerance, but fresh loss increases and fails the required improvement bound. All three stages of the final training step replay exactly. The run therefore provides another concrete separation between correct execution and useful learning. It does not test a larger-model control or establish improved capability. The repository preserves the pre-training plan, source snapshot, candidate, ordered batches, evaluation selection, individual losses and decision, with a bounded reproduction script. None of the quality candidates qualifies for promotion, and no threshold was relaxed to produce a positive result.

10.3 Native architecture transition and resumed training

A further four-validator experiment uses the real model on one physical host. It rejects a forged optimizer update, settles a valid update, rejects a forged identity-growth claim, settles valid growth to 148,675,392 parameters, and settles a subsequent training step from a new worker session at the existing round. The growth referee needs 14,161,224 bytes from one parent block; this smaller figure concerns the expansion transformation, not verification of an ordinary full training stage. Growth

itself issues no tokens. The two valid training steps issue exactly 2,000,000 atoms. After restarting the same four nodes, a shared block header at height 514 commits the post-payment application state at height 513. This checks agreement after the final settlement rather than accidentally comparing a header that precedes it.

The timings are measurements under these hosts, code, profiles, and workloads, not throughput forecasts. Concurrent experiments affect wall-clock time. A declared worker parameter budget is distinct from measured process RSS. The single-operator native test does not validate decentralization, an equilibrium incentive mechanism, or public Sybil resistance.

10.4 Text-profile conformance across machines

A subsequent run uses the same 134,515,008-parameter seed and three worker processes across two CPU hosts under one operator. Both hosts independently produce the same codec root and identical token sequences for eight English, Hebrew, Arabic, Chinese, emoji/accent, code, whitespace and mixed-direction probes. All probes round-trip exactly; this establishes text conformance, not multilingual answer quality. The codec preserves the existing 49,152-token vocabulary.

A manually written two-answer fixture contains 121 response targets in three windows, compared with 64 retained by the legacy first-response preparation. The actual training batch contains only the first two windows, totaling 101 targets. One full-model update takes 18.05 seconds; the three stage replays pass in 7.32, 6.41 and 5.76 seconds. Eight generated tokens decode to "The capital of France is Paris." The updated model and a 148,675,392-parameter growth candidate retain the codec identity. This is a functionality check after one update of a pretrained model, not evidence of improved capability. The public source includes the text conformance script and its bounded runtime profile.

An initial native run of this text-bound record stopped during dispute resolution when the synchronous RPC response expired while the referee was executing. Transport failure did not establish transaction rejection. The experiment driver was corrected to submit a signed envelope once and query its exact hash for a finalized outcome, with an explicit confirmation deadline. Lost acknowledgments, final rejection and an unresolved deadline are separately tested. This failure also emphasizes that the operational cost of a referee includes its effect on transaction and consensus latency.

The corrected run completes the real-model lifecycle with four validators on one host under one operator. It rejects forged training, pays valid training, rejects forged growth, accepts growth, and pays the following training step. Training replay publishes 171,114,496 bytes through 184 upload/seal transactions in 363.57 seconds; resolution takes 19.47 seconds. Growth replay publishes 14,161,224 bytes. The two training steps issue 2,000,000 atoms in total; growth issues none. All validators agree on the application hash in block 539, which commits the final paid transition at height 538. Tokenizer identity is preserved, while the serving head remains on the original model. This is an isolated settlement result, not a demonstration of independent ownership, economical verification or improved serving quality.

11 Implementation boundary and public-release requirements

The executable components include full-backbone pipeline training, durable worker operations, identity depth growth, fresh/replay data windows, response-target masks, paired evaluation, a local continual-epoch controller, bounded-stage replay, native reservation/reward settlement, collateralized availability/fraud challenges, and distributed generation. Their scopes and tests are recorded with the source and experiment artifacts.

The experimental ABCI application currently activates training updates from a pinned batch list, settles bonded growth claims, and keeps its serving root separate. It does not implement native rolling-data activation, quality-ticket adjudication, or paid inference for the evolving registry. Its worker RPC uses operator-managed authentication over SSH. Public discovery, independent worker collateral, sustainable audit payments, durable model replication and garbage collection, efficient long-lived ledger state, and an explicit 0.4.0 migration need implementation and testing before public cutover. The current source commitment covers all repository Python modules and validators check numerical-library versions at startup. Further runtime conformance must cover platform and dependency behavior beyond version labels.

A responsible release gate is therefore concrete: a new participant joins through the public client; contributes prescribed full-model work; survives a replacement worker or coordinator; a corrupt update and withheld artifact fail under independent observers; new data and a larger eligible model activate through native transitions; an old balance buys inference from the promoted checkpoint; and migration preserves history without duplicate token claims. None of these missing integration steps is made true by calling an off-chain research registry decentralized.

12 What the protocol can promise

NeuroShard can specify an ongoing procedure for proposing, executing, checking, comparing, and serving model revisions. It can preserve a usable model while rejecting unsuccessful experiments. It can add capacity when resources and measured benefit justify it. Under explicit consensus, observer, availability, and evaluation assumptions, it can settle those actions and their payments.

It cannot promise that every batch, every additional parameter, or every calendar day makes an LM universally better. Data can become exhausted, a task can saturate, participants can leave, and metrics can be gamed. The defensible objective is continual verified experimentation with demonstrated improvements and controlled promotion. The next implementation milestone is an integrated native epoch, not a claim that these component results already constitute an autonomous general intelligence.

References

[1] A. Douillard et al., "DiLoCo: Distributed Low-Communication Training of Language Models," 2023. https://arxiv.org/abs/2311.08105.

[2] M. Ryabinin et al., "SWARM Parallelism: Training Large Models Can Be Surprisingly Communication-Efficient," ICML, 2023. https://proceedings.mlr.press/v202/ryabinin23a.html.

[3] T. Chen, I. Goodfellow, and J. Shlens, "Net2Net: Accelerating Learning via Knowledge Transfer," ICLR, 2016. https://arxiv.org/abs/1511.05641.

[4] A. Arun et al., "Verde: Verification via Refereed Delegation for Machine Learning Programs," 2025. https://arxiv.org/abs/2502.19405.

[5] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena, "Demystifying Incentives in the Consensus Computer," CCS, 2015. https://eprint.iacr.org/2015/702.

[6] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on BFT consensus," 2018. https://arxiv.org/abs/1807.04938.

[7] Hugging Face, "SmolLM2-135M-Instruct," model card and pinned artifacts. https://huggingface.co/HuggingFaceTB/SmolLM2-135M-Instruct.