

A cost-engineered factorized $K=3$ cumulant chain with a transferred learned corrector

ARC Whitebox Estimation Challenge 2026 · Phase 2 algorithmic write-up · team **whest_winner** · designated submission #337389 · code: github.com/pranay172/ARC-WHEST-Challenge

3.82e-9

adjusted score (graded)

1.94e-8

final-layer raw MSE

19.75%

FLOP utilization of the 2^{41} budget

0.899×

corrector effect, graded (0.899× predicted before upload)

WHAT IS OURS, AND HOW TO CHECK ITS IMPACT

- **Not ours:** the factorized $K=3$ cumulant chain (Wu et al. 2026), its Phase 2 design (team 504aldo) and the corrector's design (Susupro's LT2). All three are MIT-licensed, and their notices ship with every file. §9 gives the split.
- **Ours, each with a graded or paired measurement:**
 - the cost engineering that took our port from $6.4e-9$ to $4.48e-9$ (§4);
 - the diagnosis and repair of a residual-time failure that blocked a -7% improvement (§5);
 - the corrector's port onto our chain, with a frozen held-out evaluation that predicted its graded effect to three digits (§6);
 - an oracle map of where the chain's error lives (§7).
- **Isolating the corrector:** the chain without the corrector was also graded ($4.25e-9$). The final submission differs from it only by `lt.py`, `lt_weights.py` and about 40 lines in `estimator.py` that feed the chain's statistics to the corrector and add its output to the final row. The propagated state is untouched.

1 Problem and scoring

For a random ReLU MLP of depth 16 and width 1024 with a standard Gaussian input, the estimator returns every neuron's mean activation as a $(16, 1024)$ float32 array. It sees only the weights, and all of its arithmetic runs through FlopScope, which bills FLOPs. Each network scores $\text{raw_MSE} \times \max(0.1, \text{FLOPs} / 2^{41})$ on the final layer, and the leaderboard averages over networks.

- Above the 10% floor, a 1% FLOP saving is worth exactly as much as a 1% raw-MSE reduction.
- FLOPs vary by only a few percent between networks. So for a change to a fixed chain we predict the graded effect as raw-MSE ratio $\times$ FLOP ratio. This is an approximation, because the grader averages per-network products.
- Besides FLOPs, the grader enforces a total wall-time limit and a 0.4 s cap on the client-side "residual" time (§5).

2 The estimator

2.1 The chain (method from Wu et al. and 504aldo; port and engineering ours)

At each layer the chain carries:

- the mean and the full 1024×1024 pre-activation covariance;
- the per-neuron third and fourth cumulants κ_3 and κ_4 ;
- the pairwise mixed third cumulants $\kappa(i,i,j)$, never stored as a tensor;
- regenerated, memoryless fourth-order pair terms: κ_{31} from the off-diagonal covariance with an adaptive coefficient λ , and $\kappa_{22} \approx (\kappa_{4i} + \kappa_{4j})/6$.

How the pairwise third cumulants are kept. Each ReLU layer creates third-order "sources".

- Each source is carried forward as two factor matrices, the "legs", which every later weight matrix multiplies.
- "Hub" contractions of the legs recover what the readout needs.
- Legs lose fidelity as they age: dense while young ($\text{age} < 4$), then projected onto a shared rank-384 basis, then a nested rank-192 basis.
- Rank-8 residual and rank-4 feedback channels carry what the bases miss.

The ReLU update uses the paper's Wick/Hermite tables, unchanged.

2.2 The corrector (design Susupro's LT2; implementation, training and evaluation ours)

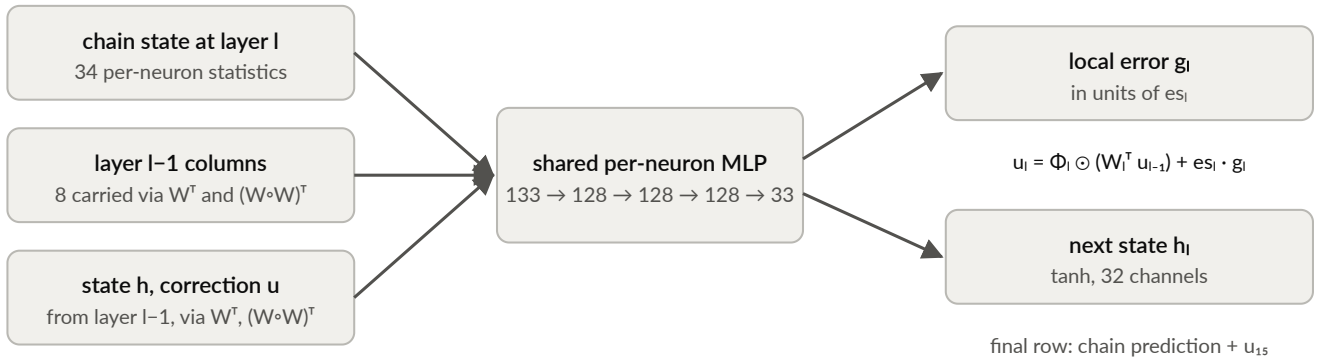


Figure 1. The corrector at layer l . One small GELU network is shared by every neuron and layer; a layer one-hot gives it the depth. Previous-layer quantities reach neuron i through column i of W^T and $(W \circ W)^T$. es_l is a fixed per-layer error scale from training.

Inputs. For each layer l and neuron i , the network reads:

- 34 per-neuron chain statistics, such as the mean, variance, $\alpha = \mu/\sigma$, φ , Φ , κ_3 , κ_4 , pair-cumulant row and column summaries, λ and the chain's prediction;
- 8 previous-layer columns, carried through W^T and $(W \circ W)^T$;
- its own 32-channel state and the correction so far, carried the same way.

Inputs pass through $\text{sign}(x) \cdot \log_{10}(10^3|x|)$ and are standardized.

Network and outputs. Three hidden layers of width 128 (54,433 parameters) output the layer's *local* error g_l , in units of es_l , and the next state.

How the correction is applied. It is carried linearly, as $u_l = \Phi_l \odot (W_l^T u_{l-1}) + es_l g_l$. This is the first-order response of a ReLU layer to an error in its input mean.

- Only the final row changes: the prediction plus u_{15} .
- Training minimizes every layer's local error with equal weight.
- The deployed form is a FlopScope port. The weights ship as a base64 string of float32 values, decoded with `struct`; there is no pickle and no NumPy.

3 How the score developed

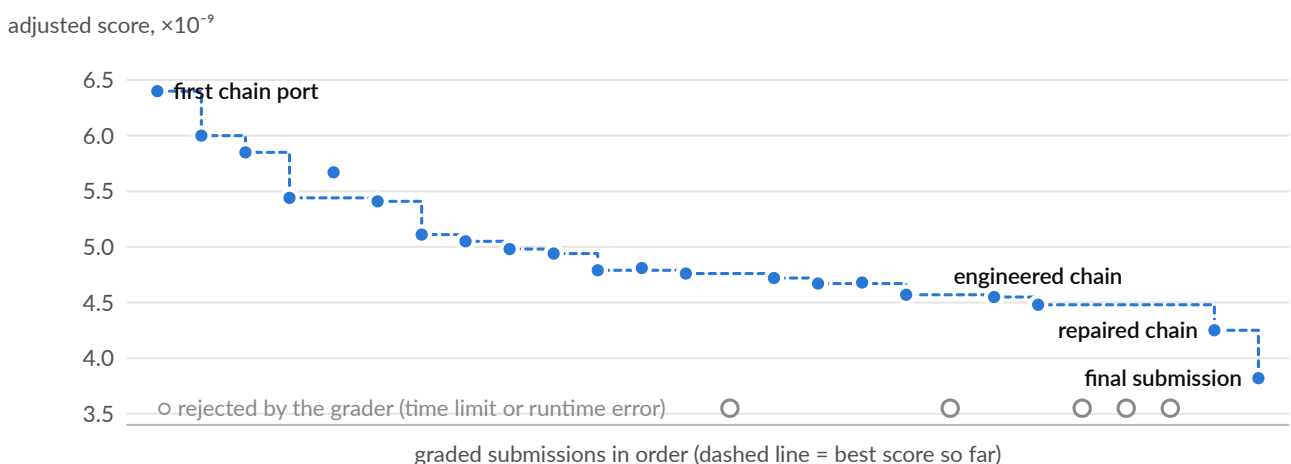


Figure 2. Every graded submission from our first chain port to the final one. Earlier sampling estimators ($1.3e-7$ and $9.4e-8$) are off-scale. Utilization fell from 31% to 19.5% along the way, while raw MSE stayed between $2.06e-8$ and $2.20e-8$ until the corrector cut it to $1.94e-8$.

Replacing Monte Carlo sampling with the cumulant chain was a $15\times$ gain. Every later step was worth 1–8%:

- **First chain port to engineered chain ($4.48e-9$):** cheaper arithmetic for essentially the same estimate, plus three small accuracy changes.
- **Engineered chain to repaired chain ($4.25e-9$):** hub pruning, made deployable by cutting client operations.

- **Repaired chain to final submission (3.82e-9):** the learned corrector.

Four of the five submissions the grader rejected were attempts to cut compute that broke a time limit. That is the lesson of §5.

4 Cost engineering of the chain (ours)

Change	FLOPs saved	Graded change
Strassen–Winograd kernel for forming and contracting old legs in the shared basis		–7.0%
Grouped QR joins in the conveyor; Strassen threshold 16	30.7B / network	–5.5%
Rectangular Strassen (level 3, then uncapped on the rank-384 tier, then the rank-192 tier)	$\approx 5B + 11.5B + 1.8B$	–1.2%, –2.1%, –0.4%
Residual/feedback ranks $16 \rightarrow 8/4$	33.4B	–3.0%
Conveyor compression schedule; metered symmetric weighted Grams; Strassen-routed basis updates	$\approx 0.4\text{--}1.3\%$ each	–0.8%, –1.1%, –1.5%
Accuracy: adaptive localized λ ; pairwise κ_4 coupling; exact residual-channel extraction	—	–0.6%, –1.4%, –0.8%

- **Strassen–Winograd under metering.** FlopScope bills a dense product at its textbook count, while a Winograd level costs 7 products plus 15 additions, billed exactly. One persistent kernel with pooled, pre-warmed buffers handles square, rectangular and batched products.
- **Symmetric weighted Grams.** $A \cdot \text{diag}(d) \cdot A^T$ is symmetric, and FlopScope's repeated-operand einsum bills only half of it. Its symmetry validation can, however, fail on float32 roundoff on the grader. Scaling both factors by the same exact power of two avoids this and is bitwise identical to the dense product.
- **Adaptive λ .** $\lambda = \lambda_{\text{base}}(l) \cdot \text{clip}(\text{mean } \kappa_4 / (\text{mean } \sigma^2 \cdot r_l), 0.5, 2)$, with r_l fitted once on development networks.
- **Dead-row hub pruning** (idea from koushik_rudra's V34 and Susupro's column cut; implementation ours). The pairwise hub contractions skip dead rows ($\alpha \leq -2.3$) and columns with $\Phi(\alpha) < 0.014$. Over 20 development networks this cut FLOPs by 7.06% and changed raw MSE by +0.001%: –7.07% adjusted [–7.55, –6.56], better on all 20. An earlier band-of-rows variant also dropped always-active rows, which pass κ_3 almost linearly, and lost 11–27% raw.

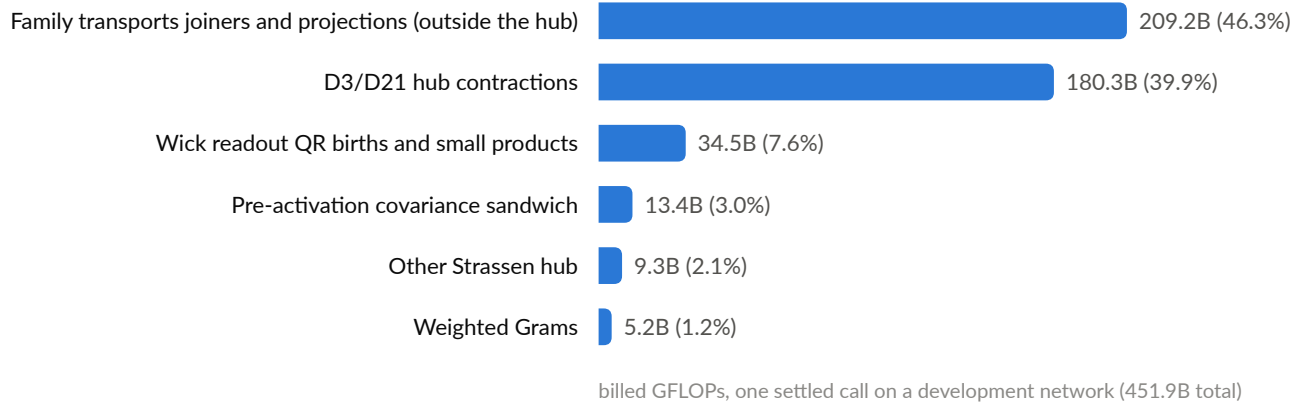


Figure 3. Where the engineered chain's FLOPs go, profiled per operation on one settled call. Transport and hub contraction of the young, dense legs are 95.4B each, together 42% of the call.

5 The residual-time failure and its repair (ours)

The failures. Level-6 Strassen cut a further 4.8% of FLOPs, but it ran 1.8–2.4× slower and hit the grader's wall-time limit. Two other cheaper variants, and then the pruned chain of §4, failed the 0.4 s residual cap; the pruned chain failed on 21 of 100 networks. In-process, its timings matched the engineered chain's, which had passed a month earlier.

Diagnosis. We ran both through the public flopscope 0.12.3 client and server, on separate physical cores and in ABCCBA order.

- The pruned chain was not slower: 0.440 s against 0.438 s per settled call. Both were close to the cap.
- A profile of one call shows about 65k client round trips, 36k of them array slices.
- A micro-benchmark puts the residual at 3–5 μs per short operation, rising to about 24 μs once a server operation lasts more than a few milliseconds.
- Each Strassen level adds a fixed set of combine and assembly operations, however small the product.

- We guess that the grader's move to flopscope 0.12.3 in October narrowed the margin; we cannot see grader-side timings.

Repair.

- **Slice cache:** every slice of a pool buffer is now created once. That cut slices from 36k to 7k, but saved only 0.007 s.
- **Work-floored Strassen level:** each product's top level is capped by its dense work (level 5 at $\geq 4 \cdot 10^9$ FLOPs, level 4 at $\geq 10^9$, else 3). Most level-5 calls had been on small products that saved almost nothing.

Result. The residual fell to 0.357 s, for +1.77% FLOPs. That is +1.96% adjusted [1.58, 2.33] against the pruned chain, with raw MSE unchanged, which predicts -5.2% against the engineered chain. The repaired chain graded at **4.25e-9 (-5.1%)**. The corrector later added 0.004–0.01 s of residual.

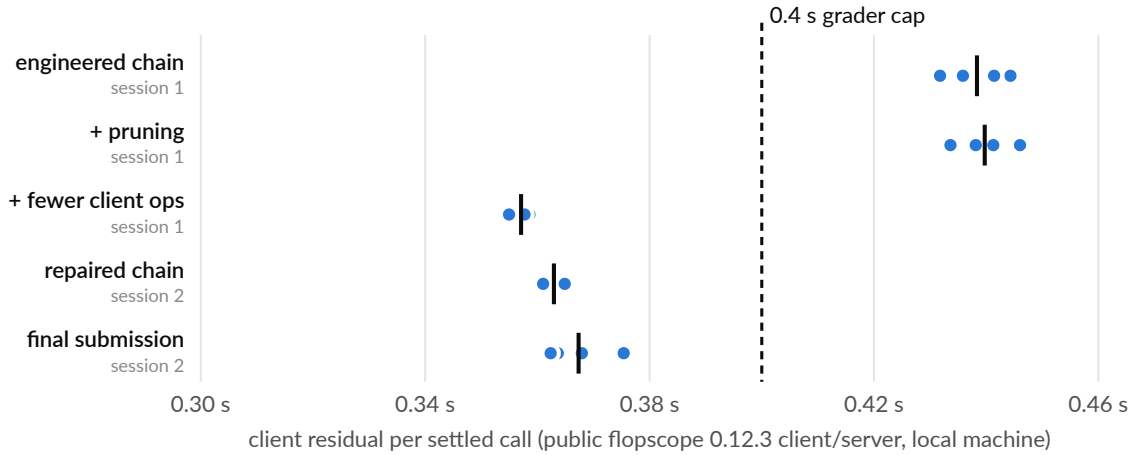


Figure 4. Client residual per settled call through the public client/server pair. Dots are individual calls; the tick is the mean. Session 2 checked the final package against the repaired chain and ran about 0.008 s slower overall. Local seconds are not grader seconds; the margin is the evidence.

6 Transferring the LT2 corrector to our chain

Protocol.

- **Capture.** A research copy of the repaired chain, with read-only hooks, recorded every corrector input and the per-layer prediction for 3,048 training networks: the public full split and the first 2,048 networks of Keenan Pepper's public corpus, which has 10^8 -sample reference means.
- **Training.** One GPU, 30 epochs per arm, Adam with OneCycle (maximum learning rate $2 \cdot 10^{-3}$), batch 4.
- **Selection.** We kept the best epoch on the public mini split: ratio 0.8704 at epoch 25.
- **Freeze.** Both arms' weights were committed, with hashes, before any truth was read for the score panel: 200 corpus networks never used in training.

Quantity	Value
raw ratio, 95% network bootstrap	0.8877 [0.8818, 0.8936]
networks improved	199 / 200
median (range)	0.889 (0.788–1.001)
FLOP ratio (433.80B / 428.48B)	1.0124
predicted adjusted ratio	0.899
graded: 3.82e-9 / 4.25e-9	0.899
graded raw ratio (1.94 / 2.18)	0.890

The panel's reference has Monte Carlo noise of about $9 \cdot 10^{-10}$ MSE, roughly 4% of the chain's $2.26 \cdot 10^{-8}$. This pulls every ratio slightly toward 1.

Susupro reports the same design giving 0.900× on their own chain, so the transfer is at full strength.

Our ablation: readout-weighted loss (negative). A unit of local error at layer l reaches the output with some gain κ_l . Weighting the local loss by $\kappa_l \cdot \epsilon_{s_l}^2$ should spend capacity where it pays, but it did not:

- worse on mini: 0.8721 against 0.8704;
- worse on the frozen panel by +0.21% [+0.07, +0.34], beating uniform weighting on only 89 of 200 networks.

Our reading: the carried state already transmits early errors forward, so up-weighting late layers mostly starves the early ones of signal.

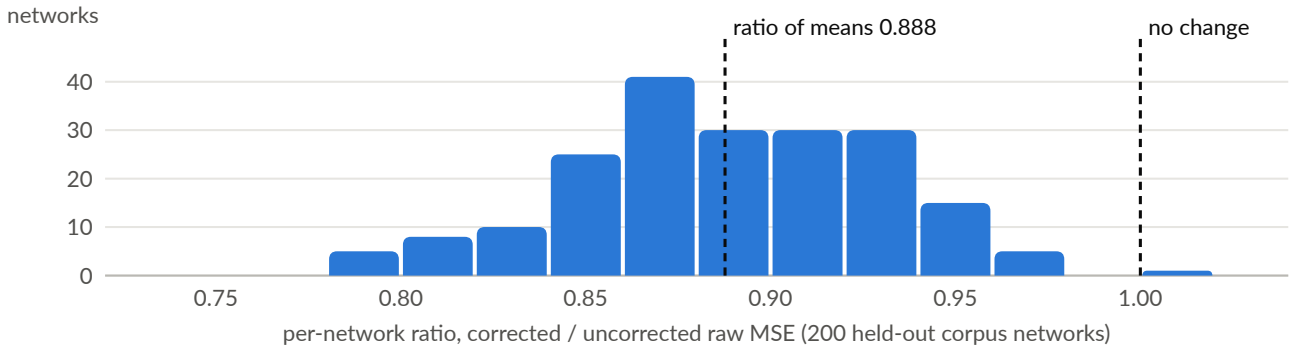


Figure 5. Per-network raw-MSE ratio of the frozen corrector against the uncorrected chain on the 200 held-out networks. The single network above 1 is worse by 0.09%.



Figure 6. Ratio on the public mini split (corrected / uncorrected raw MSE) after each training epoch for the two loss weightings. Markers are the selected epochs. Each epoch took about 270 s on one L4 GPU.

A deployment pitfall: float16 features, float32 inference. The first packaged build returned a non-finite final row.

- **Cause:** some training columns are exactly constant, such as $\Phi(0) = 0.5$ at layer 0. Stored as float16, they got a float16 mean and the 10^{-6} scale floor. The float32 port differs from that mean by $2 \cdot 10^{-3}$, so $z \approx 2000$, and the carried state overflowed.
- **Fix:** multiply by $\text{where}(\text{scale} > 1e-5, 1/\text{scale}, 0)$ instead of dividing by a floored scale.
- **After the fix:** the package matches the NumPy reference within $2.5 \cdot 10^{-4}$ of the largest correction; rows 0–14 are bit-identical to the chain; and the float32 inputs score marginally better than the float16 ones.

7 Where the chain's error lives (ours)

Method. A research copy of the engineered chain can overwrite one part of its state at every layer with reference values from 10^9 samples, letting everything downstream run unchanged. With no oracle it reproduces the chain bit for bit. The comparisons use 20 development networks, paired, with a network bootstrap.

What this shows:

- **Given exact marginals, the chain is nearly exact.** MSE stays at $4\text{--}6.5 \cdot 10^{-10}$ at every depth, so neither the readout closure nor truncation at $K=3$ is the bottleneck. Errors compound: they leak between channels and are inherited downstream.
- **Variance error comes from the off-diagonal post-activation covariance,** and about 71% of each layer's share is inherited from the layer before. The fresh part is the small residual of the non-Gaussian pair terms, which already capture 99.5–99.9% of the true non-Gaussian pair structure.
- **The state's biases:** variance is unbiased, κ_3 slightly low (slope 0.98 at layer 15) and κ_4 strongly low (slope 0.72), with a per-neuron correlation of about 0.8.

Repairing κ_4 alone made things worse (table below). The chain's fitted parts, namely the adaptive λ (which reads mean κ_4), the κ_4 coupling and the κ_{22} model, are calibrated to its κ_4 deficit. Raising κ_4 also raises λ by 39% at layer 15. *We believe this is why a learned corrector that sees the whole state works where hand-written slot repairs did not.*

Change to κ_4 (20 networks)	Effect on the κ_4 state	Adjusted	Wins
scale κ_3 and κ_4 per layer toward the reference level (two tables)	level correct	+25.7% / +54.8%	0/20
add $0.4 \cdot \kappa_3^2 / \sigma^2$ to κ_4	correlation 0.85 $\rightarrow$ 0.91; level overshoots	+66.6%	0/20
same term rescaled to keep each layer's κ_4 level ($b = 0.75$)	correlation 0.85 $\rightarrow$ 0.93; level unchanged	+0.08%	8/20

Change to κ_4 (20 networks)	Effect on the κ_4 state	Adjusted	Wins
same term rescaled to keep each layer's κ_4 level ($b = 1.5$)	correlation $0.85 \rightarrow 0.93$; level unchanged	+3.6%	3/20

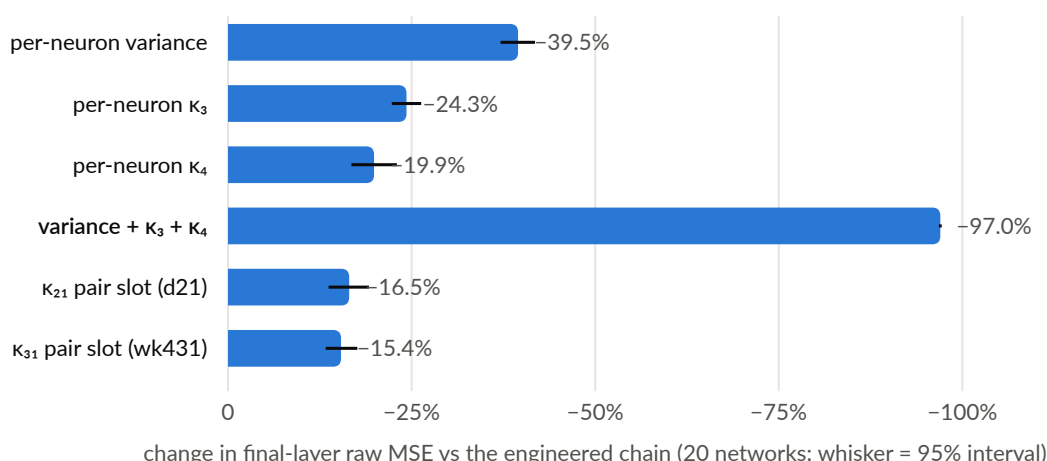


Figure 7. Final-layer raw-MSE change when one channel is replaced by its reference at every layer. The single-channel gains sum to 84%, but together the three marginals remove 97% ($2.17\text{e-}8 \rightarrow 6.5\text{e-}10$).

8 Gap to the best public result, dead ends, and grader pitfalls

The gap. Susupro's write-up reports their chain alone at $3.494\text{e-}9$ ($2.09\text{e-}8$ raw, 16.6% utilization) and with the corrector at $3.145\text{e-}9$. Ours are $4.25\text{e-}9$ ($2.18\text{e-}8$, 19.5%) and $3.82\text{e-}9$. The corrector factor is the same (0.900 against 0.899), so the whole $1.215\times$ gap is in the chain:

- about three quarters is compute (utilization ratio 1.175);
- about one quarter is raw error (1.043).

They report four steps we lack:

- Puffi's SSC closure for the κ_4 regeneration law (-5.4% raw);
- young-atom pruning and active-set trimming ($0.875\times$);
- hub level 6 with other rank choices ($0.963\times$);
- rank-4 feedback thin legs ($0.991\times$).

Their level-6 hub passes the residual cap where ours failed. We suspect it issues far fewer client operations, but have not verified this.

Dead ends (each closes only the configuration tested):

- Demoting legs earlier saved 2–25% of FLOPs, but raw MSE rose 9% to $10\times$. Demoting them later cost more than it gained.
- With all sources dense, raw fell 4.8% for +26.8% FLOPs. Without the second tier, raw fell 2.6% for +5.1% FLOPs.
- Retiring legs older than 13 layers cost 4.3% raw: legs shrink only about $1/\sqrt{2}$ per layer.
- A one-factor κ_{31} model captured 7.4% of the oracle gain.
- Structurally different recurrences failed: an order-8 positive copula was $160\times$ worse, and a rank-32 first-order recurrence lost positivity.

Grader pitfalls:

- Residual time is spent per client operation, not per FLOP. Measure it through the public client and server with warm calls and a matched baseline; in-process timings say nothing about the cap.
- Recursive fast matrix multiplication trades FLOPs for operations, so cap its depth by work.
- Repeated-operand einsum validates symmetry, and float32 roundoff can fail that check remotely.
- Compare `fnp.dtype` objects, not strings: a failed guard silently falls back.
- Mask degenerate standardized columns in learned parts.
- A local pass does not certify a remote pass.

9 Code, compliance, attribution, and use of AI

Code. The repository github.com/pranay172/ARC-WHEST-Challenge is the organizers' starter kit with one commit on top. That commit adds the submitted estimator files, with only internal experiment labels removed from comments, and removes tutorial

material. Run it with `uv sync`, then `whest validate`.

Compliance:

- The submission uses only the grader's Python, the `flopscope` client API and `whestbench` types: no NumPy, compiled code, threads or subprocesses.
- All arithmetic is metered, including the corrector and the λ scalars.
- Packaged data is the constant Wick/Hermite tables and the corrector's float32 weights, with no pickle.
- The estimator guards on geometry (16×1024), budget and float32 dtype, and otherwise falls back to a Gaussian estimate.
- Training, fitting and diagnostics used only public data: the public `full` and `mini` splits, Keenan Pepper's corpus, and our own 10^9 -sample moments on public development networks.

Piece	Origin
Factorized K=3 cumulant recursion; Wick/Hermite tables	Wu, Lecomte, Winer, Robinson, Hilton, Christiano 2026 (arXiv:2605.05179 , code , MIT)
Phase 2 chain design: legs, memoryless κ_4 regeneration, λ law, conveyor	team 504aldo (code , MIT)
Independent FlopScope port; cost engineering (§4); adaptive λ ; κ_4 coupling; residual-channel extraction	ours
Dead-row hub pruning	idea from koushik_rudra (V34) and Susupro; implementation ours
Residual diagnosis and repair (§5)	ours
Corrector design: inputs, carried state, linear carry, local target, schedule	Susupro (Yan Su), logxio/fourth-cumulant (MIT)
Corrector implementations, training on our chain, frozen evaluation, ablation, float16 fix; oracle map (§7)	ours
Training corpus	Keenan Pepper, arc-whestbench-p2-d8b-corpus-14k

Use of AI. LLM coding agents, mainly Anthropic's Claude in Claude Code, wrote nearly all of the research and submission code. They also ran the experiments on a local machine and a cloud VM, and drafted this report. The author set the research directions, approved every experiment, data download, cloud spend and upload, made the submissions, and checked the reported numbers against the grader. Guesses are marked in italics.

Thanks to the ARC authors, team 504aldo, Susupro, koushik_rudra, and Keenan Pepper (for the corpus and the Phase 1 variance/bias budgeting view), and to the organizers for a carefully metered challenge.