

A learned corrector that reads the engine's cross-neuron terms, saturation-aware compute and a free fourth-cumulant fold on factored cumulant propagation at width 1024

Phase 2 technical write-up for submission 336955 (participant yeongjunyoo; author Yeongjun Yoo, 유영준).

Summary

Submission 336955 estimates the expected final-layer activations of the 1024×16 He-initialised ReLU networks without sampling. The grader scored it at **2.2984e-09** under flopscope 0.12.3: final-layer MSE $1.6616e-08$ at 0.1383 of the FLOP budget. Our better designated submission, 337488 (the other is 335519, an earlier file kept for its wall-time margin), is this file plus later changes and scored **2.1878e-09** (final-layer MSE $1.4869e-08$ at 0.1471): compression rank 256 with one more restriction (section 7.3), and il.miscusi's scale-mode extraction in place of the memoryless law, with the corrector retrained (section 7.6). The repository's `submissions/shots.csv` holds the grader's values for every submission. Every ablation in this write-up measures 336955 unless a table says otherwise.

The engine is ARC's factored $K = 3$ augmented cumulant propagation (Wu et al., arXiv:2605.05179; reference code `mlp_cumulant_propagation`, commit 93d091a, MIT), made affordable at this shape with structure team 504aldo published in forum post 18218: two dense legs per third-cumulant source, a memoryless one-scalar-per-layer law for the off-diagonal of the fourth-cumulant core, compressed old sources, and Strassen-Winograd written as flopscope operations. We re-implemented these from the paper, the reference code and the post's prose. Section 2 says, part by part, whose idea each piece is.

We add three things:

- 1. A learned corrector that reads the engine's cross-neuron terms** (section 4). After the engine has run, a small recurrent network proposes a correction to the mean at every layer from quantities the engine already computes. Among its inputs are eleven structure taps, per-neuron summaries of the engine's cross-neuron terms (the off-diagonal third cumulant and pre-activation correlations on each neuron, the fourth-order folds of section 6, the leg energy lost to compression). Effect: $\times 0.790$ [0.776, 0.805] of the engine's final-layer MSE on 48 corpus networks nothing of ours had touched, lower on all 48 ($\times 0.798$ on the 90 validation networks that selected it). Retrained without the structure taps it keeps $\times 1.085$ [1.075, 1.095] more error, on every seed, and with the neuron's own-state inputs held at their mean $\times 1.055$ (section 4.5). The corrections are also carried forward through the engine's mean-to-mean partial derivative, $dp_t = D_t o(W_t^T dp_{t-1}) + u_t$, but retraining finds no reliable effect of that carry ($\times 1.005$, with seeds on both sides of 1), and replacing its full readout derivative by the ReLU slope changes nothing ($\times 1.0001$). Lineage: it adapts mliston's learned center correction (a GRU over the closure's moment state with messages through W ; forum 18170, Phase 1) to a sampling-free estimator. Susupro's write-up (forum 18254, posted 2026-10-18) describes a corrector with the same linear carry; our graded submission 334200 (shot 11, 2026-10-05) already had it, earlier and independently, but since it is worth little here we do not count it as a contribution. What we claim is the inputs, the structure taps above all, the training and the measurements.
- 2. Saturation-aware compute** (section 5). Every large product runs only over the rows and columns of neurons above a layer-dependent threshold in $a = m / \sigma$, with shapes fixed in setup and chosen per network, and dead birth neurons are gathered out of new sources. Restricting third-cumulant products to active neurons is also in koushik_rudra's V34 (rows) and in Susupro's extension to columns (forum 18254); we built ours independently (graded 2026-10-07). Our parts are the thresholds on rows, columns and the $3T^T$ band, the per-network choice among fixed shapes, and the birth rule. Effect, with the corrector off: adjusted $\times 0.687$ [0.679, 0.695] against the same engine with the restriction switched off (FLOPs $\times 0.747$, final MSE $\times 0.919$), lower on all 48 untouched networks. That it also lowers the MSE was not the aim. Section 5 measures what it is: the entries it drops are correct, so dropping them must offset error elsewhere in the engine. The ablations tie that error to the input-norm scale mode carried in the fourth-cumulant core rows: the scale-mode extraction of 337488 takes away most of the band's gain (we did not trace the path).
- 3. The star (3,1) fold** (section 6). Several fourth-cumulant sources a layer's nonlinearity creates live on the two legs of that layer's third-cumulant source. For the young, uncompressed sources we fold the (3,1) slice of the "star" source into the product that already forms the third-cumulant matrix: no extra dense n^3 product, 0.44% more FLOPs.

Kai_leal_gonzalez (forum 18246) measured that the (3,1) slice carries much of 504aldo's estimator's error (the exact slice lowers its final MSE by 18.1%); our fold carries one source's exact share of that slice. Effect, with the corrector off: final MSE x0.982 [0.974, 0.990], adjusted x0.989 [0.981, 0.998], lower MSE on 35 of the 48 untouched networks.

Section 3 maps where the engine's error is at width 1024: 78% of the last layer's error comes in through the layer-14 mean; replacing the engine's per-neuron variance, third and fourth cumulants by Monte Carlo truth at every layer leaves 8-10% of the final MSE, with nearly all of that value in layers 8-15. These diagnostics (on earlier engines, two to forty networks) led us to correct the mean at every layer and carry the corrections along the chain; section 4.5 later found that the gain comes from what the corrector reads, not from the carry.

How to read the numbers. Three kinds appear, and each table says which: (a) clean ablations of the submitted file on networks nothing was trained, tuned or selected on (section 7.1, and for 337488 the end of section 7.6); (b) acceptance reads during development, each on 48 public networks no candidate trained on but all candidates were judged on (section 7.5); (c) exploratory diagnostics on earlier engines (section 3) and on public training networks (section 5's mechanism, section 7.6's readable engine). Only (a) is confirmation; section 7.2 repeats (a) on networks that were used once for selection.

1. Setting and notation

Problem. Layers $t = 0, \dots, 15$ of width $n = 1024$: $z_t = W_t^T h_{t-1}$ with $h_{-1} = x \sim N(0, I)$, and $h_t = \text{ReLU}(z_t)$. Weights are i.i.d. $N(0, 2/n)$, no biases. The estimator returns $E[h_{15}]$. The score is the final-layer MSE against a 10^9 -sample Monte Carlo reference, times $\max(0.1, C/B)$, where C is the FLOP count flopscope assigns analytically to the declared operations and $B = 2^{41}$. Above 0.1 B a 10% cut in cost is worth exactly a 10% cut in MSE.

Carried state (activation side, as in the reference): the mean $\mu_t = E[h_t]$, the covariance C_t , the third cumulant held as sources (below) and a fourth-cumulant core.

Pre-activation quantities of layer t . $m = W_t^T \mu_{t-1}$, $S = W_t^T C_{t-1} W_t$ (exact, since z_t is linear in h_{t-1}), $\sigma^2 = \text{diag}(S)$, $a = m / \sigma$, $S_{\text{off}} = S$ with its diagonal set to 0, $t3_i = \kappaappa3(z_i, z_i, z_i)$, $d4_i = \kappaappa4(z_i, z_i, z_i, z_i)$.

Wick coefficients. For $Z \sim N(m_i, \sigma_i^2)$, $w_k = E[\text{ReLU}^k(Z)]$: $w_0 = \sigma \phi(a) + m \Phi(a)$, $w_1 = \Phi(a)$, $w_2 = \phi(a)/\sigma$, $w_3 = -a \phi(a)/\sigma^2$, $w_4 = (a^2 - 1) \phi(a)/\sigma^3$, $w_5 = (3a - a^3) \phi(a)/\sigma^4$ (code: `w01 ... w51`). Note $d w_k / d m = w_{k+1}$.

Readout. $\mu_t = w_0 + t3 w_3 / 6 + d4 w_4 / 24$ per neuron.

Third-cumulant sources and legs. Each layer's nonlinearity creates one source. A source is a sum over its birth neurons ("centers") j of symmetrized products of three columns ("legs") H_j, E_j, G_j . At birth $G = I$, $H = \text{diag}(w_1) S_{\text{off}}$ and $E = \text{diag}(x) + 3 Y^T$, where x and Y are the new third cumulant's repeated-index slices with the parts already carried subtracted (the reference's construction). Between layers every leg is multiplied by $\text{diag}(w_1)$ of its layer and then by W^T into the next layer; we call the W^T step the **push**.

T and $3T^T$. $T_{ab} = \kappaappa3(z_a, z_a, z_b)$ for $a \neq b$. The engine forms $3T^T$ from all sources at once in one stacked product of "A rows" (Hadamard products of legs with coefficients) against the legs; the code calls it `TT`. Its diagonal is $3 t3$. The next covariance reads T through the Wick (2,1) term $(1/2)(w_{2,a} w_{1,b} T_{ab} + w_{1,a} w_{2,b} T_{ba})$; the new source's legs and the fourth-order rows read it too.

Fourth-cumulant core. The augmented term set carries the activation fourth cumulant through an $n \times n$ core $\Gamma = \text{diag}(\gamma) + \text{off-diagonal}$ (the reference's representation; section 2.2 says how we fill it).

2. The engine, and whose idea each part is

2.1 One layer, in the order the code runs it

The loop is `_AugEngine.propagate` in the submitted `estimator.py`. For layer t :

1. **Linear step.** $m = W^T \mu$, and $S = W^T (C W)$ as a symmetric product by blocks (`_covsym`), on some layers riding on the push's Strassen-Winograd product. Then σ , a , and the restricted sizes of section 5.

2. **Push.** Every stored source's dense legs go through W^T in one stacked Strassen-Winograd product ($_{sw}$), restricted to the rows and columns of section 5; thin legs by plain products; compressed sources are rebuilt from their bases.
3. **$3T^T$.** One stacked product forms $3T^T$ (diagonal $3t_3$) from all live sources, with the star-fold rows of section 6.
4. **Per-neuron algebra.** Wick coefficients from exact polynomial tables in a , σ , $\phi(a)$, $\Phi(a)$; the readout with t_3 and d_4 ; the diagonal of the fourth-cumulant core.
5. **Matrix stage.** One einsum over a small table of diagrams gives the next covariance C' , the new source's legs and the rows R_4 that feed the fourth-cumulant core's diagonal; the $\text{diag}(w_1)$ factor of the legs is applied here (the G fold of section 6 just before it).
6. **Birth.** The new source in two-leg form (section 2.2).
7. **Compression** of the source that has just turned six layers old (section 7.3).

After the last layer the corrector of section 4 adds its correction to the final mean.

2.2 Parts taken from published work

Part	Source	What we did with it
Factored $K = 3$ augmented propagation: term set, Wick/Hermite coefficients, source-and-leg factorisation	ARC: arXiv:2605.05179, <code>mlp_cumulant_propagation</code> (commit 93d091a, MIT)	Ported to flopscope operations; checked against the reference numerically (per-layer means within $2.6e-9$ relative at $n = 16, 48$ and 96 , float64).
Two dense legs per source: the third leg written through the other two times diagonals, plus a low-rank residual from a randomized range finder	504aldo, forum 18218, section 2.1 ($M = P d(s) + 3 A d(e) + Z L^T$, rank 16)	Our version: $E = G \text{diag}(xv) + H \text{diag}(e) + Z L_z^T$, where xv is E 's birth diagonal, $e_j = \langle E_{\text{off}}[j], H[j] \rangle / \langle H[j], H[j] \rangle$ per birth neuron j (least squares; the post does not say how its $d(e)$ is computed), and $Z L_z^T$ is the best rank-4 approximation of the residual $E_{\text{off}} - H \text{diag}(e)$ inside the range of (residual) Ω . Ω is the next layer's first 32 weight columns: an i.i.d. Gaussian test matrix that costs no random numbers at prediction time. It does not select directions.
Memoryless fourth-cumulant core: $\text{Gamma}_{\text{off}} = \text{lambda}'_t C_{\text{off}}$, one scalar per layer, diagonal exact	504aldo, forum 18218, section 2.2	Our table $\text{LAM_REFIT} = (0.01799, 0.01539, 0.01345, 0.01320, 0.01194, 0.01089, 0.00880, 0.00787, 0.00747, 0.00755, 0.00731, 0.00691, 0.00821, 0.00669, 0.00687, 0.00436)$ starts, like theirs, from a regression: the per-layer least-squares coefficient of the exact core's off-diagonal on C_{off} , taken along the exact reference chain at $n = 256$ (one seed). We scaled it by 0.25 for $n = 1024$ as the $1/n$ power counting predicts (a scale scan of the final MSE on mini 50-52 put the optimum at 0.236 - 0.248), then refitted five layer-group scales (layers 0-2, 3-5, 6-8, 9-11, 12-14) by linearized least squares on finite differences of the final MSE on mini 50-52 (changes $+26\%$, $+28\%$, $+7\%$, $+17\%$, $+46\%$), worth $\times 0.939$ [$0.929, 0.948$] of the final MSE on mini 0-39 at the time. Mini 51 and 52 are also in the corrector's pool. The diagonal gamma is computed exactly each layer from row sums of the (4) and (2,2) slices.
Compressing old sources by age	504aldo, forum 18218, section 2.4 (one shared, age-gated basis)	Each source is compressed separately (section 7.3).
Strassen-Winograd as flopscope operations, batched across sources, depth limited by the wall-time caps	504aldo, forum 18218, section 2.7; the Phase 1 winners	Our recursive layout and in-place deepest level (section 7.4).

When they went in, the augmented terms with the memoryless law cut the final MSE to $\times 0.545$ [$0.528, 0.563$] of our port of the reference SIMPLE engine on 40 mini networks (0-39), 40 of 40 better, at the same cost (the two are measured together; section 7.1 separates the law).

3. Where the engine's error is (exploratory)

These are diagnostics on earlier versions of the engine; the version is named each time. None of them is a score.

Depth. The engine's per-layer MSE grows almost linearly with depth: from $6.9e-10$ at layer 0 (the Monte Carlo floor of the reference) to $2.33e-8$ at layer 15 on 40 public networks with the engine of 10-04, and from $6.7e-10$ at layer 1 to $2.07e-8$ at layer 15 on the 90 validation networks with the engine of shot 34. Replacing only the layer-14 mean by the truth and

recomputing layer 15 removes 78% of the layer-15 MSE of $2.33\text{e-}8$ ($1.81\text{e-}8$): that much arrives through the layer-14 mean. The other 20% ($4.62\text{e-}9$) comes through the layer-14 covariance and cumulants and layer 15's own closure; the two parts correlate at 0.028 (same 40 networks). Linear regressions on 32 layer-15 features explain 3.1% of the layer-15 error in sample. A correction that sees only the last layer cannot reach most of the error. 504aldo report the same from their side (18218, dead-end row 10, "Learned / online output corrections": final-layer features correlate below 0.09 with the error).

The same oracle on the submitted engine. On 30 of the 90 validation networks we set only layer t 's output mean to the truth (its central moments kept) and ran the rest of the submitted file unchanged. The engine's final-layer MSE falls to $\times 0.991$, $\times 0.873$, $\times 0.504$ and $\times 0.192$ of its value for $t = 4, 8, 12$ and 14 (30 of 30 networks better at $t = 12$ and 14). With the shipped corrector also left unchanged the final MSE is $\times 0.993$, $\times 0.922$, $\times 0.679$ and $\times 0.431$: trained on the engine's usual errors, the corrector then corrects errors that are no longer there. The corrector itself leaves $\times 0.75$ of the mean MSE at layers 4 and 8, $\times 0.77$ at layer 12 and $\times 0.80$ at layers 14 and 15, so the information it has, not where it is applied, bounds what a mean correction can do. Putting the corrector's own layer- t correction into the engine in place of the truth, and restarting its chain after t , leaves the final MSE where it was: $\times 1.0003$ [0.9999, 1.0008] at $t = 14$ and $\times 0.9997$ [0.9982, 1.0011] at $t = 12$, against the submitted file on the same 30 networks. The linear chain already carries that correction to the last layer, and nothing measurable comes through the variance path; without the restart the correction is counted twice ($\times 1.203$).

Channels. On two training networks (full 96 and 97) we measured every neuron's first four cumulants at every layer from 2^{25} Monte Carlo samples, replaced the engine's own values by these at every layer, and carried the mean to the end (engine of 10-05, final-layer MSE $2.05\text{e-}8$):

True values injected at every layer	full 96	full 97
variance only	$\times 0.708$	$\times 0.577$
third cumulant only	$\times 0.728$	$\times 0.683$
fourth cumulant only	$\times 0.717$	$\times 0.731$
variance and third	$\times 0.367$	$\times 0.346$
variance, third and fourth	$\times 0.099$	$\times 0.082$

Injecting at layers 1-7 only does nothing ($\times 0.96$ to $\times 1.06$); injecting at layers 8-15 only gives $\times 0.11$ - 0.12 for all three channels. Two networks are few, and the per-neuron Monte Carlo error of the third and fourth cumulants at 2^{25} samples is of the same order as the engine's own error on those neurons, so single-neuron comparisons are noise-limited; the injections average that noise over many neurons, but the noise they inject also adds error, so the 8-10% left is an upper bound on what the three channels leave (the MSE itself is against the dataset's 10^9 -sample means). The two diagnostics fit together: what reaches layer 15 through the layer-14 mean starts as closure error of the variance and cumulants in layers 8-14, carried forward through each layer's readout. Nurlanggs (forum 18252) report the same picture from oracle injections into 504aldo's chain (variance at layers 8-15: $\times 0.687$ and $\times 0.647$).

Neuron states. By the last layer's a (engine and corrector of shot 34, the 90 validation networks):

Last-layer state	Share of neurons	Share of the score
dead, $a < -2$	28%	0.4%
band, $\text{abs}(a) < 2$	44%	49%
on, $a > 2$	28%	50%

Width. Small-width runs of the reference code, depth 16, three seeds (so intervals are wide and several include 1), Monte Carlo truth with antithetic pairs ($6.7\text{e}7$ samples at $n = 128$, $1.34\text{e}8$ at $n = 256$, $3.4\text{e}7$ at $n = 512$; final-layer floors of at most $1.7\text{e-}9$, $7.8\text{e-}10$ and $1.9\text{e-}9$, subtracted). Final-layer MSE as a ratio to factored $K = 3$ augmented, geometric mean and 95% t interval:

Variant	$n = 128$	$n = 256$	$n = 512$
factored $K = 4$ SIMPLE (reference)	0.515 [0.264, 1.005]	0.244 [0.157, 0.381]	not run (memory)
same, without the non-hypertree $K = 3$ terms that factored $K = 3$ drops	2.10	0.838 [0.593, 1.183]	

Variant	n = 128	n = 256	n = 512
those non-hypertree K = 3 terms with the fourth cumulant reduced to its trace part		0.800	
fourth-cumulant sources up to age 1 read from shared legs (the factored K = 3 augmented terms plus the path, star, d4, d31, d22 and 3+2 kinds, without the d211 kind and the cross parts; not the shipped fold, which was not run at these widths)		0.846 [0.651, 1.101]	0.853 [0.795, 0.915]

A separate closure that projects the full fourth cumulant onto the top-8 eigenspace of the activation covariance kept $x0.715$ [0.609, 0.839] of the error of its own trace-only control at $n = 256$ and $x0.960$ at $n = 512$. What these runs support, with their wide intervals: in the reference code the $K = 4$ gain over augmented $K = 3$ was larger at $n = 256$ than at $n = 128$; at $n = 256$ it appeared only when the fourth cumulant came together with the non-hypertree third-order terms; the pieces our representation can afford did not grow from 256 to 512; and the subspace closure lost most of its value from 256 to 512. Separately, Susupro (forum 18254) ran ARC's exact unaugmented $K = 4$ at $n = 1024$ and report a raw final-layer MSE 11.8 times below augmented $K = 3$, a different algorithm from anything in our submission. Artyom_Mazur (forum 18249) report a closure that pays only together with a diagram the reference truncates.

4. Contribution 1: a learned corrector that reads the engine's cross-neuron terms

4.1 Definition

After the engine has run, the corrector computes a correction dp_t for layers $t = 1, \dots, 15$ ($dp_0 = 0$) and returns $mu_15 + dp_15$:

```
dp_t = D_t o (W_t^T dp_{t-1}) + u_t,
D_t = w_1 + w_4 t3 / 6 + w_5 d4 / 24 = d mu_t / d m_t at fixed sigma, t3, d4,
u_t = 1e-4 sigma_t o (w_head . H_t + b_head), H_t = GRUCell(x_t, W_t^T H_{t-1} / sqrt 2), H_0 = 0,
```

with o the elementwise product. $D_t o W_t^T$ is the partial Jacobian of layer t 's mean with respect to layer $t-1$'s mean, holding layer t 's variance and cumulants fixed. So $dp_15 = \sum_t (D_{15} W_{15}^T \dots D_{t+1} W_{t+1}^T) u_t$ carries each local correction u_t to the last layer along the mean-to-mean path of the engine. The paths through the later covariances and cumulants are not included: a mean correction also moves w_1 and so the next covariance and sigma, and the chain leaves that path to the network, which sees it only through the $(W o W)^T \text{abs}(dp)$ message. The chain changes only the mean. Section 4.5 finds no reliable effect of this carry once the network also passes its own state through W .

The input x_t of each neuron (code `_wmsg`, extra taps switch `_QX`):

Group	Contents
own state (14)	a , σ , $\phi(a)$, $\Phi(a)$, the $t3$ and $d4$ readout terms and their higher Edgeworth companions, higher ReLU cumulants of the neuron
structure taps (11)	the star slice's diagonal and two shapes of the G fold (section 6); the off-diagonal energy and signed column sum of T on the neuron, also as the covariance update reads it; the leg energy lost to compression, carried forward; the neuron's pre-activation correlation energy and signed sum
neighbourhood (16)	the previous layer's ReLU cumulants and readout terms summed through the elementwise powers $W^{\{02\}}, \dots, W^{\{06\}}$; column norms of W
messages	$W_t^T dp_{t-1} / \sigma_t$ and $(W_t o W_t)^T \text{abs}(dp_{t-1}) / \sigma_t$, and the depth

The hidden size is 32 and the head is linear. The weights (`wmsg.npz`, 176 kB, with the input standardisation folded in) are loaded in `setup()`. At prediction time the corrector costs about 2.2 GFLOPs, 0.7% of the bill.

4.2 Lineage

	mliston (Phase 1, forum 18170 and its PDF)	Susupro (forum 18254, 2026-10-18)	This submission (shot 11, graded 2026-10-05)
role	corrects the analytic center that a sample-based control variate then uses	adds a learned correction to a budget-pruned $K = 3$ chain	the whole correction; no samples
output per layer	the normalised mean correction of layer m , trained on $mu_ref - a_base$ at layers 2-31	the error the layer adds, carried forward linearly through the ReLU slopes and W	the local part u_t ; the carried part $D_t o (W_t^T dp_{t-1})$ with the full readout derivative

	mliston (Phase 1, forum 18170 and its PDF)	Susupro (forum 18254, 2026-10-18)	This submission (shot 11, graded 2026-10-05)
earlier corrections	input messages $\Phi(\alpha) \circ (W^T u_{\{m-1\}})$ and an auxiliary output through $W \circ W$	a 32-dimensional state and the incoming correction, through W and $W \circ W$	input messages through W and $W \circ W$, and the carried term in the output
inputs	nine analytic channels, five sample-comparison channels	35 chain statistics, 8 of the previous layer	engine state, structure taps, sums through $W^{\circ K}$
network	GRU, 96 hidden, 96-192-2 GELU readout	MLP, three layers of 128 GELU units	GRU, 32 hidden, linear head

The structure (local correction plus a linear carry through the engine's own mean derivative) was in our graded submission 334200 (shot 11) on 10-05; Susupro posted a corrector of the same structure on 10-18. We state that as chronology, not as a contribution: section 4.5 finds no reliable effect of the carry here, and its full readout derivative does no better than the ReLU slope Susupro use. What we claim as ours is the inputs, the structure taps above all (8.5% in section 4.5, against 5.5% for the neuron's own state and 0.8% for the neighbourhood sums), the training and distillation, and the measurements of sections 4.4 and 4.5. The idea of learned networks that consume cumulant estimates as features is also in Jacob Hilton's public LessWrong comment on the challenge announcement.

Before the corrector, shots 8 and 9 used two simpler learned parts, `1c` and `1aw` in the code; their flags are still on in the file, but the code skips both whenever the corrector runs. `1c` carried a corrected mean beside the engine, $a'_t = a_t + W_t^T (p_{t-1} - p_t) / \sigma_t$ and $p'_t = \text{readout}(a'_t) + \text{net}_t(\text{features})$, with one small ReLU network per layer; `1aw` scaled the fourth-order law per neuron by $1 + c_1 a + c_2 a^2$ with six fitted constants. The corrector replaced both in shot 11.

4.3 Training

Targets are the per-layer residuals $r_t = \text{truth}_t - \mu_t$, from the dataset's Monte Carlo means. The loss is

$$L = \sum_t w_t \text{MSE}(r_t - \text{dp}_t) / \text{MSE}(r_t) / \sum_t w_t, \quad w_t = 1 \text{ for } t < 15, w_{15} = 5,$$

so every layer's correction is trained, each layer's error is normalised by the engine's own, and the last layer counts five times. Adam (lr $2e-3$, weight decay $1e-4$), one network per step, up to 60 epochs with early stopping on mini 44-49 (patience 12). The shipped weights are a student of the same architecture trained on $(1 - \beta) r_t + \beta \text{dp}_t^{\text{teacher}}$ with $\beta = 0.5$, where $\text{dp}_t^{\text{teacher}}$ is the mean of twelve correctors that differ only in their seed. Nine students (β 0.5, 0.8 and 1.0, three seeds each) all passed the pre-set rule on the 90 validation networks, and the rule took the best. Appendix A lists every set of networks and its use.

4.4 Measured effect

Comparison	Networks	Metric	Ratio	Kind
corrector on vs off (engine alone), submitted file	48 untouched corpus networks	final MSE (floor-corrected)	$\times 0.790$ [0.776, 0.805], lower on 48 of 48 (adjusted $\times 0.796$)	clean
same	public one-shot selection set, 88 networks	final MSE	$\times 0.802$ [0.792, 0.812], lower on 88 of 88 (adjusted $\times 0.808$)	selection evidence (section 7.2)
same	90 validation networks	final MSE	$\times 0.798$ (band $\times 0.819$, on $\times 0.779$; weakest cell a in $[-1, 0]$ at $\times 0.918$)	picked the student
<code>1c</code> , then <code>1aw</code> , when introduced (shot 8 against shot 7, shot 9 against shot 8)	grader, 50 networks	final MSE	$\times 0.879$, then $\times 0.974$	development
distilled student vs the single corrector of shot 33	90 validation networks	final MSE	$\times 0.9817$ [0.9762, 0.9874], 70 of 90 (the twelve-seed teacher $\times 0.9785$; its single members $\times 0.988$ to $\times 1.004$)	picked the student
corrector replacing <code>1c</code> and <code>1aw</code> , when introduced (shot 11)	full 0-47 acceptance read	final MSE	$\times 0.9787$ [0.9707, 0.9862], 35 of 48	development

`1c` and `1aw` were fitted once, on the engine of shots 8 and 9, and never refitted, so we do not compare them with the corrector on the submitted engine. The corrector was retrained seven times as the engine and its inputs changed (shots 23, 28 and 30-34). The shipped weights were trained on dumps of the engine of shots 30-32 ($3T^T$ rows $a > -1.5$) and kept through the engine steps after it (push threshold 1.75, kept-subspace SVD, exact trims), so they never saw the submitted engine's own features. Section 4.5 takes the corrector apart by retraining.

4.5 What the chain and the inputs are worth

Retrains with the same settings and pool except 30 epochs instead of 60 (every variant and its reference alike, to fit the time left), three seeds per variant (2, 3, 4; single correctors differ by about 2% from seed to seed), read on the 84 full networks of the validation set (the six mini networks there were used for early stopping), each against a retrained reference with everything on and the same seed. The plan and the variants (`ladder.py`) were fixed on 10-20 before any of these numbers existed. We report the mean over seeds of the paired ratio with a bootstrap over networks (seeds averaged inside it), and each seed's own ratio, since the seed-to-seed spread is the larger uncertainty here:

Change	Final MSE ratio against the reference retrain	Seeds 2, 3, 4
no carried term in the output: $dp_t = u_t$ (the dp messages stay inputs)	x1.0046 [1.0012, 1.0081]	0.9994, 1.0124, 1.0021
no carried term and no dp messages: only the hidden-state carry $W^T H_{t-1}$ links the layers	x1.0052 [1.0017, 1.0086]	0.9998, 1.0128, 1.0029
carried term with $D_t = w_1$ only (ReLU slope, as in Susupro's)	x1.0001 [1.0000, 1.0001]	1.0001, 1.0000, 1.0001
without the structure taps	x1.0847 [1.0749, 1.0947]	1.0791, 1.0884, 1.0865
the neuron's own-state inputs (the 14 of section 4.1) held at their training mean	x1.0551 [1.0481, 1.0626]	1.0488, 1.0587, 1.0580
the neighbourhood inputs (the 16 sums through the powers of W) held at their training mean	x1.0082 [1.0046, 1.0120]	0.9989, 1.0168, 1.0089
without the messages	x0.9995 [0.9983, 1.0006]	0.9974, 1.0019, 0.9991
without the hidden-state carry ($H_{t-1} := 0$)	x1.0040 [0.9988, 1.0093]	0.9987, 1.0062, 1.0072

The references keep x0.810 of the engine's final MSE on these 84 networks (the shipped distilled student x0.798 on all 90, section 4.4).

Of the parts tested, two input groups matter. Retrained without the structure taps the corrector keeps 8.5% more of the error on every seed, and is better than its reference on only 2 to 7 of the 84 networks; with the neuron's own state held at its mean it keeps 5.5% more, also on every seed (5 to 8 networks better). The neighbourhood sums add little (x1.008, one seed of three below 1). Each group was held out alone, so the others may cover for it, and the three effects need not add up to the corrector's x0.81. We find no reliable effect of the linear carry: without it the mean is 0.5% worse, but one seed of three is better without it, with or without the dp messages. Removing the other link between layers, the network's own state passed through W , costs as little (x1.004); we did not remove both. The full readout derivative D_t makes no measurable difference against $D_t = w_1$, and the messages none. So the claim of section 4.2 shrinks to the inputs: what lowers the error is what the network reads, not how its corrections travel between layers.

4.6 What we do not know

We cannot say exactly what the corrector has learned. The combination results (it replaces `law` and does not stack with it, x1.0051 when both are on) suggest it absorbs at least the systematic part the fourth-order law was fitting. That reading is a hypothesis. Section 4.5 shows which of its inputs it needs, the structure taps, not what it computes from them. We did not test feeding the corrections back into the engine's covariance and cumulants.

5. Contribution 2: saturation-aware compute

Rule (code `_SAT`, `_R25B`, `_DC`, sizes in `_SAT_FRAC` and `_SAT_BINS`; all thresholds are empirical, chosen by paired reads):

- push, layers 6-10: input rows of neurons with $a > -1.75$ at layer $t-1$ (covariance rows always in full);
- push, layers 11-14: input rows with $a > -1.75$ at $t-1$ and output columns with $a > -3$ at t (the switch `_R25B['s8']` [`'out'`] still reads 3.5, but the count uses the $3T^T$ band's column threshold, which moved from 3.5 to 3 at shot 29);
- $3T^T$, layers 2-14: rows of neurons with $a > -1.5$ and columns in the band $\text{abs}(a) < 3$;
- last layer: covariance rows pushed in full (every neuron's variance is needed, as 504aldo also note in 18218 section 2.3), legs over inputs with $a > -1.75$ at layer 14 and outputs in the band $\text{abs}(a) < 3$ at layer 15; t_3 is 0 outside the band,

on both tails (the t3 readout term carries $\phi(a)$, below 0.0044 there; the band edge was set by paired reads, shot 29);

- births: a new source keeps its birth neurons with a > -2 ("dead centers" are gathered out rather than multiplied as zeros).

Lineage. As Susupro describe it (forum 18254; we have not read the code), koushik_rudra's V34 trims the third-cumulant leg products by the rows of neurons that are almost never active, and Susupro extend the cut to columns ($w_1 < 0.014$ skips their hub contraction and shared-basis lift) and prune young atoms by a norm bound. Our restriction was built independently and graded on 2026-10-07 (shot 13). What is specific to it is listed above: thresholds in a for push rows, push columns and the $3T^{\wedge}T$ band, the per-network choice among fixed shapes below, and the birth rule.

Shapes. flopscope bills shapes, and the wall-time caps punish per-network re-planning, so every restricted product has a short list of allowed shapes fixed in `setup()`: the 99th percentiles of the kept counts over 392 public networks (the dumped ones among the 416 of full 112-303 and 352-575, which include the 84 full validation networks and none from the acceptance, report or held-out sets), rounded up to the multiple the Strassen recursion needs. Each prediction sorts the neurons by a and uses the smallest allowed shape that holds every neuron above the threshold; if none does, that product runs unrestricted. The birth rule is the one fixed count: each source keeps its top- R_t birth neurons by a, where R_t is the same 99th percentile of the count with a > -2 . On most networks that keeps every neuron above -2 and a few just below; on about 1% it drops the lowest-a neurons above -2. The 99th-percentile fractions R_t / n for layers 1-15 are 1.0, 0.989, 0.965, 0.940, 0.917, 0.888, 0.874, 0.852, 0.838, 0.822, 0.810, 0.795, 0.784, 0.775, 0.759 (births, a > -2) and 0.999, 0.977, 0.944, 0.910, 0.885, 0.861, 0.841, 0.820, 0.809, 0.793, 0.783, 0.766, 0.756, 0.748, 0.733 (push inputs, a > -1.75); the other tables are in `_SAT_BINS` and `_R25B`.

Why it is not free. A small marginal activation probability does not bound a neuron's contribution to cross-neuron terms, and the reads show where the trade stops paying: one more output-column cut on top of the shipped rules gave adjusted $\times 0.9970$ [0.9927, 1.0015]; keeping only 70% of late sources' birth rows by an importance score cut cost 7.7% but raised the engine's MSE 19% (adjusted $\times 1.644$ on 48 networks).

Effect. The first four rows are the restriction, or one part of it, on against off in the submitted file's engine with the corrector off (section 7.1 gives them the other way round, with the lines each one changes); off means every restricted product at full width, dead birth neurons kept, and the compression SVD over all rows and columns.

Measured	Networks	Result	Kind
the whole restriction	48 untouched corpus networks	adjusted $\times 0.687$ [0.679, 0.695], FLOPs $\times 0.747$, MSE $\times 0.919$ [0.908, 0.930], lower MSE on 48 of 48	clean
the restricted shapes alone (push, $3T^{\wedge}T$ and last-layer products; birth rule and compression on in both arms)	same	adjusted $\times 0.725$ [0.717, 0.733], FLOPs $\times 0.790$, MSE $\times 0.918$ [0.907, 0.928], lower MSE on 48 of 48	clean
of which the $3T^{\wedge}T$ band alone (rows a > -1.5 , columns $\text{abs}(a) < 3$)	same	adjusted $\times 0.783$ [0.775, 0.792], FLOPs $\times 0.863$, MSE $\times 0.908$ [0.898, 0.917], lower MSE on 48 of 48	clean
of which the push rows and columns alone	same	adjusted $\times 0.930$ [0.925, 0.935], FLOPs $\times 0.923$, MSE $\times 1.008$ [1.002, 1.014], lower MSE on 17 of 48	clean
of which the last layer's legs alone	same	adjusted $\times 0.978$ [0.977, 0.980], FLOPs $\times 0.977$, MSE $\times 1.001$ [0.999, 1.002], lower MSE on 23 of 48	clean
the birth rule alone	same	adjusted $\times 0.882$ [0.876, 0.888], FLOPs $\times 0.873$, MSE $\times 1.010$ [1.003, 1.017]	clean
the compression SVD on kept rows and read columns alone	same	adjusted $\times 0.989$ [0.986, 0.991], FLOPs $\times 0.988$, MSE $\times 1.000$ [0.998, 1.003]	clean
first restriction, where it went in (shot 13 file against shot 11's exact twin, corrector on)	same	adjusted $\times 0.948$ [0.945, 0.951], FLOPs $\times 0.946$, MSE $\times 1.002$ [0.999, 1.006], better on 48 of 48	clean
same	24 validation networks, at the time	C/B -5.9%, MSE +0.6% [0.25%, 0.95%]	development
dead birth neurons gathered out (shot 17)	full 0-23	FLOPs $\times 0.951$, MSE $\times 1.0065$	development
the last four restrictions	mini 0-19 (inside the corrector's pool)	adjusted $\times 0.9546$ [0.9499, 0.9591], FLOPs $\times 0.9561$, MSE $\times 0.9984$, 20 of 20	development, in sample

The birth rule trades about 1% of MSE for 13% of the bill, as each step was meant to. The restricted shapes do more than save: with them off (the birth rule and the compression left as they are) the engine's final MSE is 9.0% higher, on every one of the 48 networks. Every restriction went in as a cost cut read at about equal MSE, with the corrector on and on earlier engines (the development rows), so this is an unexpected result of the clean ablation, not something we designed for. Switching the three families off one at a time says where it comes from: all of it is the $3T^T$ band. With the band on, the final MSE is $\times 0.908$, lower on 48 of 48; the push restriction costs a little MSE ($\times 1.008$, lower on only 17 of 48) for its 7.7% of the bill, and the last layer's restriction changes nothing ($\times 1.001$ [0.999, 1.002]). The two effects multiply to about the package's ($0.908 \times 1.008 = 0.915$ against 0.918). So the terms that matter are the entries of $3T^T$ the band leaves out. Entry (i, j) is $3 \text{ kappa3}(z_i, z_j, z_j)$, with i the single index and j the doubled one, and it is dropped when $a_i \leq -1.5$ or $\text{abs}(a_j) \geq 3$. On mini 0 (counted in shot 39's file, which applies the same rule) the band drops 6% of the off-diagonal at layer 3 and 53% at layers 13-14. Section 7.2 repeats the table on a second set of networks.

What the band's MSE gain is. We expected the dropped entries to be wrong, and measured them against Monte Carlo truth (10^7 samples per network on public mini 0-3, the organizers' input distribution; band on and off without truth on mini 0-15; [research/wu/mech/](#)).

- **The dropped entries are right.** At layers 2-10 they match the truth as well as the kept entries: sign agreement 1.00, correlation 1.00, slope of engine on truth 0.97-1.00, signed bias within 1% of their size. At layers 11-14 they are shrunk to 0.52-0.63 of the truth, because the push restriction writes no legs for neurons with $a < -3$.
- **Removing them still helps.** Putting their exact values back changes the band-off engine's final MSE by $\times 1.002$ -1.015. Zeroing them, as the band does, gives $\times 0.871$ -0.938. The gain is in the entries dropped for their doubled index: zeroing only those gives $\times 0.840$ -0.937, and zeroing only the row drops costs $\times 1.10$ -1.19. The gain remains when every other entry is exact.

So the band does not remove errors. It removes correct terms, and since that lowers the final MSE, their removal must offset error elsewhere in the engine.

Where that error is. One later change took most of the gain away, the scale-mode extraction of shot 39 (section 7.6). Band on against off on mini 0-15, as the geometric mean of the final-MSE ratio:

Engine	Band on / off	Band on better
this file (336955)	$\times 0.918$	16 of 16
shot 38	$\times 0.919$	16 of 16
shot 39 without the extraction	$\times 0.894$	16 of 16
shot 39 (337488)	$\times 0.983$	11 of 16
shot 39 with the memoryless law back	$\times 1.062$	0 of 16

The extraction has four parts, and switching one off at a time (mini 0-7) points mainly at the part that takes the scale mode out of the next layer's fourth-cumulant core rows. With only that part off, the band gains again ($\times 0.915$, 8 of 8). With only the feed pair off it gains a little ($\times 0.964$), and with the covariance or the new-leg part off not at all ($\times 0.992$, $\times 1.030$). With the core-row, covariance or new-leg part off the engine itself is far worse (band-on final MSE 5 to 28 times shot 38's), as are the partial versions of section 7.6; without the feed pair it stays near shot 39. The band changes the final means in nearly the same way in all six engines we measured (cosine 0.83-0.94). In this file that change points toward the error. In the engine with only the core-row part off, 87-92% of the band's gain lies along that part's own change.

Our reading, consistent with these ablations but not traced through the cumulants: while the fourth-cumulant core rows carry the input-norm scale mode's share, the engine makes an error that the band partly cancels, and the extraction's core-row part removes that error itself, which leaves the band little to do. The truth-based results rest on four networks.

6. Contribution 3: the star (3,1) fold

6.1 Which fourth-cumulant sources live on third-cumulant legs

In ARC's $K = 4$ SIMPLE step, a layer's nonlinearity creates fourth-cumulant sources in two bundles: one with $A_r = \text{sym}(v_r e_r^T)$, $v_r = w_1 \circ S_{\text{off}}[:, r]$ (kinds "path", "star" and "3+2"), and one with $A_r = e_r e_r^T$ (the repeated-index corrections d4, d31, d22, d211). We went through the reference's term list ([factor_k4.py](#), 91 of 95 terms used) and

checked our contraction formulas against a dense computation (relative error below $2e-16$ at $n = 12$). All these sources share two legs, e_r (the identity at birth) and v_r (the column r of H at birth): exactly the legs G and H of the third-cumulant source born at the same layer, transported the same way. Hence:

Kind	Extra legs to push	Note
path, star, d4, d22	none	fixed coefficients on the third-cumulant source's two legs
d31	one	one more $n \times n$ leg per layer
3+2	none, if K3 is factored	coefficients are another source's third leg
d211	several	does not close in leg form

Pushing them costs nothing beyond the third-cumulant push; reading them is the cost: about $16 n^3$ products per (bundle, layer) for the (4), (3,1) and (2,2) slices.

6.2 The star's (3,1) slice, folded into $3T^{\wedge}T$

The star source is the reference's term in which one birth neuron r connects to three others through the column v_r (its weight is $4 w_3(l, r)$ in the reference's normalisation, with $w_3(l, r)$ the third Wick coefficient of r at the birth layer l). After transport, with G and H the current legs of source l , its (3,1) slice is

$$K31_{ab} = \text{kappa4_star}(z_a, z_a, z_a, z_b) = \sum_j w_3(l, j) (3 G_{aj} H_{aj}^2 H_{bj} + H_{aj}^3 G_{bj}).$$

(This is the reference's slice contraction for sources on shared legs, checked against a dense computation as in section 6.1, evaluated with the star's coefficients.) Its Wick (3,1) term in the next covariance is

$$C'_{ab} += (1/6) (w_{3,a} w_{1,b} K31_{ab} + w_{1,a} w_{3,b} K31_{ba}),$$

with w at the current layer. The covariance reads T through $(1/2)(w_{2,a} w_{1,b} T_{ab} + w_{1,a} w_{2,b} T_{ba})$, and $w_{3,a} = \text{lambda}_a w_{2,a}$ with $\text{lambda}_a = -a_a / \text{sigma}_a$. So adding $\text{lambda}_a K31_{ab} / 3$ to T_{ab} yields exactly this term. $3T^{\wedge}T = \sum_s (A^{\wedge}H_s H_s^{\wedge}T + A^{\wedge}G_s G_s^{\wedge}T)$, so the code adds, per live dense source and elementwise,

$$A^{\wedge}H += 3 \text{lambda} \circ w_3(1) \circ G \circ H \circ H, \quad A^{\wedge}G += \text{lambda} \circ w_3(1) \circ H \circ H \circ H,$$

and then resets the diagonal of $3T^{\wedge}T$ to $3 t_3$ (the slice's diagonal belongs to $d4$). Four facts about it:

- The coefficient is the analytic one (`_k4f['scale'] = 1.0`).
- It is applied to the dense, uncompressed sources only (the six youngest). Compressed sources enter $3T^{\wedge}T$ through their projected blocks (section 7.3) and do not carry the fold.
- $3T^{\wedge}T$ also feeds the new source's legs and the fourth-order rows in the matrix stage. There the added rows enter with those contractions' own Wick factors, which we did not derive; the measured effect is that of the fold as built, side effects included.
- It adds no dense product, only elementwise work (block `k4f` in `propagate`): 1.3 GFLOPs per network, 0.44% of the engine's bill on the corpus networks (0.37% of the larger bill of shot 21, where it went in). That share is the fold row of section 7.1 less the corrector's input taps, which the code skips with the fold: switching the fold off saves 0.72%, of which the taps are 0.28%, measured on their own.

A companion fold (switch `_GF`) adds the part of the star's supply to the next third cumulant that has the shape of a G leg: $G += s \text{lambda}_l (w_2 / w_1) G \circ H$ before the w_1 step, with $\text{lambda}_l = -a / \text{sigma}$ at the source's birth layer. Our derivation of that supply term, from the G -slot part $w_3 \text{Sym3}(U, h, h)$ with $U = w_2 \circ G \circ H$, gives $s = 1$; we have not written it out here, and the fold is better read as a heuristic. Measured on mini 0-19 against the build before it, $s = 1.0, 0.5$ and 0.25 gave adjusted $x1.0065, x0.9880$ and $x0.9914$; we ship $s = 0.5$, an empirical damping of a derived term. That choice missed our screening threshold (0.985) and was settled by the acceptance read instead ($x0.9838 [0.9784, 0.9890]$, 41 of 48, the same corrector on both arms), a departure from the pre-set rule that we recorded at the time.

6.3 Effect

Measured	Networks	Result	Kind
star fold off, engine alone	48 untouched corpus networks	MSE x1.018 [1.010, 1.027], adjusted x1.011 [1.002, 1.020], higher MSE on 35 of 48	clean
G fold off, engine alone	same	MSE x1.004 [0.999, 1.009], adjusted x1.000 [0.996, 1.005], higher MSE on 30 of 48	clean
both off, engine alone	same	MSE x1.030 [1.019, 1.040], adjusted x1.022 [1.011, 1.032], higher MSE on 40 of 48	clean
both off, with the corrector (the code then also drops all eleven structure taps)	same	MSE x2.584 [2.465, 2.708], adjusted x2.56 [2.44, 2.68]	clean
the same split: folds left out, taps still built, corrector on	same	MSE x1.084 [1.073, 1.096], higher on 47 of 48	clean, confounded: the corrector was trained on the folded engine
the same split: taps off, folds kept, corrector on	same	MSE x2.510 [2.395, 2.628], higher on 48 of 48	clean: the shipped corrector without inputs it was trained on
star fold, when introduced (corrector on)	mini 0-19	adjusted x0.9774 [0.9640, 0.9917], 16 of 20 (scale 0.7: x0.9762)	development, in sample

In the clean ablation the G fold has no measurable effect on the engine alone. Its only support is the development read that adopted it with the corrector on (section 6.2), which is not a confirmation, and the corrector-on rows cannot isolate it (below). We leave it in the file as an unconfirmed heuristic and do not count it as a contribution. The two folds partly stand in for each other: switching the star fold off costs x1.018 of the MSE with the G fold on and x1.026 with it off, and switching the G fold off costs x1.004 and x1.011.

With the corrector on, switching the folds off also drops the corrector's taps, so the last three rows split that change in two (variants `v_tapsnofold` and `v_notaps`, added after review). The x2.56 is almost all the taps: the shipped corrector without them, folds kept, has x2.51 the MSE. It is not the taps' worth. A corrector retrained without the taps loses 8.5% (section 4.5); x2.5 is the price of withholding inputs a trained network relies on. The folds left out with the taps kept cost x1.084, more than the x1.030 the engine alone loses. The corrector was trained on the folded engine, so part of that may be its mismatch with an unfolded one; we did not retrain it without the folds.

Why the (3,1) slice. Before building the fold we put the exact fourth-cumulant slices into C' on two networks (an oracle on the engine of 10-14): all of the fourth-order gain was in the (3,1) slice (x0.816 and x0.791), none in the (2,2) slice (x1.048 and x0.972), and most of it in the part of the slice orthogonal to `S_off`. Kai_leal_gonzalez (forum 18246, 10-15) reached the same place independently on 504aldo's estimator: the exact slice lowers its final MSE by 18.1%, and their fitted two-term law recovers 39-59% of the slice's value in one covariance update and about a sixth of the exact slice's final-MSE gain. Our fold adds one source's exact contribution instead of a fitted law. We did not test whether the two combine.

7. Results

7.1 Clean ablation of the submitted file on untouched networks

Networks: 48 networks of Keenan Pepper's public corpus (`keenanpepper/arc-whestbench-p2-d8b-corpus-14k`), idx 3408 + 8k and 3409 + 8k for $k = 0, \dots, 23$. Their weights are regenerated from the corpus's seed rule, and their truth is the corpus's 10^8 -sample means. No lane of ours dumped, trained on or read them before this run. The truth's own expected squared error at the last layer, $\text{mean}_j (m_{2,j} - m_{1,j}^2) / 10^8$ from the corpus's raw moments, averages $7.6e-10$ per network (about 4.6% of our MSE); we report ratios with it subtracted from both arms ("floor-corrected") and without. Each row changes the listed lines of the submitted file (`make_variants.py` writes each variant and its diff). It runs in process under flopscope 0.12.3, one thread, the way the grader meters a prediction, and is paired with the submitted file on the same networks. Ratios are of means; the intervals are paired bootstrap 95% over networks.

Each ratio is the variant over its base (the Base column), so above 1 means the change makes it worse; the last column counts the networks on which the variant's adjusted score is lower than its base's.

Variant	Change	Base	Adjusted	MSE (floor-corrected)	FLOPs	Lower adjusted score
engine alone	gru=False, lc=False, law=False	submitted (336955)	1.257 [1.234, 1.280]	1.266 [1.243, 1.289]	0.993	0/48
engine alone, star fold off (the taps go too)	+ _K4F['scale'] = None	engine alone	1.011 [1.002, 1.020]	1.018 [1.010, 1.027]	0.993	16/48
engine alone, G fold off (the taps go too)	+ _GF['scale'] = None	engine alone	1.000 [0.996, 1.005]	1.004 [0.999, 1.009]	0.996	20/48
engine alone, both folds off (the taps go too)	+ both	engine alone	1.022 [1.011, 1.032]	1.030 [1.019, 1.040]	0.992	10/48
engine alone, memoryless law at 0	+ lam = (0.0,) * 16	engine alone	2.123 [2.072, 2.175]	2.123 [2.072, 2.175]	1.000	0/48
engine alone, restriction off	+ the three rows below together	engine alone	1.456 [1.438, 1.474]	1.088 [1.076, 1.101]	1.338	0/48
engine alone, restricted shapes off (birth rule and compression unchanged)	+ _SAT_BINS emptied, _R25B keys s1, s8, s2, s7 removed	engine alone	1.380 [1.365, 1.396]	1.090 [1.078, 1.102]	1.266	0/48
engine alone, 3T^T band off	+ _SAT_BINS 3T^T table emptied, _R25B key s7 removed	engine alone	1.276 [1.263, 1.291]	1.101 [1.090, 1.113]	1.159	0/48
engine alone, push restriction off	+ _SAT_BINS push table emptied, _R25B keys s1, s8 removed	engine alone	1.075 [1.069, 1.082]	0.992 [0.987, 0.998]	1.084	0/48
engine alone, last-layer restriction off	+ _R25B key s2 removed	engine alone	1.022 [1.021, 1.024]	0.999 [0.998, 1.001]	1.023	0/48
engine alone, dead birth neurons kept	+ _DC['theta'] = None	engine alone	1.134 [1.126, 1.142]	0.990 [0.983, 0.997]	1.146	0/48
engine alone, compression SVD on all rows and columns	+ _R25B['kc'] = None	engine alone	1.011 [1.009, 1.014]	1.000 [0.997, 1.002]	1.012	3/48
engine alone, taps off	+ QXON = False	engine alone	0.997	1 (identical predictions)	0.997	48/48
both folds off, corrector on (the taps go too)	_K4F, _GF	submitted (336955)	2.561 [2.443, 2.683]	2.584 [2.465, 2.708]	0.992	0/48
taps off, folds kept, corrector on	QXON = False	submitted (336955)	2.501 [2.386, 2.619]	2.510 [2.395, 2.628]	0.997	0/48
folds left out of 3T^T and the G legs, taps still built, corrector on	four additions removed (v_tapsnofold)	submitted (336955)	1.082 [1.071, 1.094]	1.084 [1.073, 1.096]	0.998	1/48
approximate level off	_APA_ORI = {}	submitted (336955)	1.009 [1.006, 1.013]	0.995 [0.991, 0.999]	1.014	16/48
our port of the reference SIMPLE engine (shot 1)	k3_fast.py	submitted (336955)	11.17 [10.82, 11.54]	2.300 [2.229, 2.376]	4.856	0/48
ARC augmented with 504aldo's structure, our implementation (shot 6)	k3_fast_aug_sc_21.py	submitted (336955)	2.457 [2.399, 2.515]	1.360 [1.328, 1.392]	1.807	0/48

Every prediction of every row was finite. Without the floor correction each MSE ratio moves toward 1 by at most 2.8% of its value (the engine alone reads 1.254 instead of 1.266); both tables and the per-network logs are in the repository.

The "(the taps go too)" rows need the code in view. The code builds the corrector's eleven structure taps (section 4.1) only when both folds are on, and it builds them even when the corrector is off. So every fold row also drops the taps. In the engine-alone rows that is work nothing reads: switching the taps off alone costs nothing in MSE (the predictions are bit-identical on all 48 networks) and saves 0.28% of the FLOPs, as the three fold-plus-tap rows had implied (0.72% + 0.38% - 0.82%). The corrector-on rows are split into folds and taps in section 6.3.

One part cannot be switched off inside the submitted file, because later code is built on it: two legs per source. It is compared where it went in, on the same networks: `k3_fast_aug_sc_21.py` (shot 6) against `k3_fast_aug_sc.py` gives adjusted $\times 0.745$ [0.742, 0.748], FLOPs $\times 0.739$, MSE $\times 1.008$ [1.004, 1.013], better on 48 of 48. The first saturation restriction is compared the same way in section 5.

7.2 The same table on the public one-shot selection set

Public mini 60-99 and full 304-351 (88 networks) were held out of all training and acceptance reads, then read once on 10-16 to choose the two designated submissions among five candidates. After the nomination window closed (10-23) the rows of section 7.1 ran there with the same harness. These networks are selection evidence, not a clean holdout: they chose between candidates, although nothing was trained or tuned on them. Their truth is the dataset's 10^9 -sample mean, whose own error is about a tenth of the corpus's (10^9 samples against 10^8), so these MSE ratios are not floor-corrected. The last column repeats the corpus's floor-corrected MSE ratio from section 7.1.

Variant	Base	Adjusted	MSE	FLOPs	Lower adjusted score	MSE on the corpus (7.1)
engine alone	submitted (336955)	1.238 [1.223, 1.253]	1.247 [1.232, 1.262]	0.993	0/88	1.266
engine alone, star fold off (the taps go too)	engine alone	1.014 [1.007, 1.020]	1.021 [1.014, 1.027]	0.993	32/88	1.018
engine alone, G fold off (the taps go too)	engine alone	1.003 [0.999, 1.006]	1.007 [1.003, 1.010]	0.996	34/88	1.004
engine alone, both folds off (the taps go too)	engine alone	1.025 [1.017, 1.034]	1.033 [1.025, 1.042]	0.992	25/88	1.030
engine alone, memoryless law at 0	engine alone	2.122 [2.077, 2.167]	2.122 [2.077, 2.167]	1.000	0/88	2.123
engine alone, restriction off	engine alone	1.441 [1.424, 1.457]	1.080 [1.068, 1.092]	1.334	0/88	1.088
engine alone, restricted shapes off	engine alone	1.370 [1.355, 1.385]	1.085 [1.073, 1.097]	1.263	0/88	1.090
engine alone, 3T^T band off	engine alone	1.273 [1.261, 1.285]	1.100 [1.090, 1.110]	1.157	0/88	1.101
engine alone, push restriction off	engine alone	1.074 [1.069, 1.079]	0.991 [0.986, 0.996]	1.083	0/88	0.992
engine alone, last-layer restriction off	engine alone	1.021 [1.021, 1.022]	0.998 [0.998, 0.999]	1.023	0/88	0.999
engine alone, dead birth neurons kept	engine alone	1.129 [1.123, 1.135]	0.985 [0.980, 0.991]	1.146	0/88	0.990
engine alone, compression SVD on all rows and columns	engine alone	1.013 [1.011, 1.014]	1.001 [1.000, 1.003]	1.012	4/88	1.000
engine alone, taps off	engine alone	0.997	1 (identical predictions)	0.997	88/88	1
both folds off, corrector on (the taps go too)	submitted (336955)	2.329 [2.268, 2.390]	2.349 [2.288, 2.411]	0.992	0/88	2.584
taps off, folds kept, corrector on	submitted (336955)	2.270 [2.209, 2.330]	2.277 [2.217, 2.337]	0.997	0/88	2.510
folds left out of 3T^T and the G legs, taps still built, corrector on	submitted (336955)	1.080 [1.072, 1.089]	1.083 [1.074, 1.092]	0.998	2/88	1.084
approximate level off	submitted (336955)	1.010 [1.008, 1.012]	0.996 [0.994, 0.997]	1.014	14/88	0.995
our port of the reference SIMPLE engine (shot 1)	submitted (336955)	11.31 [11.01, 11.61]	2.335 [2.274, 2.398]	4.843	0/88	2.300

Variant	Base	Adjusted	MSE	FLOPs	Lower adjusted score	MSE on the corpus (7.1)
ARC augmented with 504aldo's structure, our implementation (shot 6)	submitted (336955)	2.414 [2.369, 2.461]	1.340 [1.315, 1.365]	1.802	0/88	1.360
two legs, where they went in (shot 6 against its three-leg twin)	three legs	0.747 [0.744, 0.749]	1.010 [1.007, 1.013]	0.739	88/88	1.008
first restriction, where it went in (shot 13's file against shot 11's exact twin, corrector on)	shot 11's twin	0.950 [0.948, 0.953]	1.005 [1.002, 1.007]	0.946	88/88	1.002

Every row points the same way as on the corpus. The MSE ratios are within 1.5% of the corpus's, except the two corrector-on rows without the taps (x2.349 and x2.277 here against x2.584 and x2.510), where losing the taps costs the corrector less on these networks. The corrector itself is worth x0.802 [0.792, 0.812] of the engine's final MSE here, lower on all 88 (x0.800 on mini 60-99, x0.804 on full 304-351). The three restriction-family rows were added to section 7.1 after this set's plan was fixed and ran after it; they reproduce section 5's split almost exactly: all of the MSE gain is the $3T^{\wedge}T$ band (x1.100 with it off, corpus x1.101), the push restriction costs a little (x0.991, corpus x0.992), and the last layer's restriction is about neutral (x0.998, corpus x0.999).

7.3 Old-source compression

Each layer adds a source, so pushing all of them costs $O(L^2 n^3)$ over the network (the paper's bound for the factored method). A source becomes "old" after six pushes. Its stacked legs $[H; G]$ ($2n$ birth rows by n current-layer columns) are then factored once as $C_s Q_s$, with Q_s ($192 \times n$) the top right singular vectors of a truncated SVD and C_s ($2n \times 192$) kept aside; every later push moves only Q_s , and when $3T^{\wedge}T$ needs the legs they enter through $Q_s^{\wedge}T$ ($C_s^{\wedge}T [A^{\wedge}H; A^{\wedge}G]_s$). The SVD rows are weighted by each birth neuron's share of the T mass (floor 0.02), and the SVD runs only on the rows the birth rule keeps and the columns the next push reads (`_r25_kc`). Measured trades on four public networks: compressing one layer earlier (age 5) cut FLOPs 5.5% but raised MSE 22% (adjusted x1.152); rank 128 cut FLOPs 5.9% but raised MSE 29% (adjusted x1.218). On the submitted file the trade is nearly flat around 192: rank 128 raised the final MSE x1.358 for x0.949 the FLOPs (mini 0-19), and rank 256 lowered it to x0.942 for x1.051 the FLOPs on the acceptance read (adjusted x0.990 [0.980, 1.002], short of the adoption rule; both under flopscope 0.12.1). With one more restriction on top (the fold-layer pushes restricted from layer 4), rank 256 passed and was graded as shot 38, 2.2974e-09: the grader's 0.12.3 prices the larger truncated SVDs higher (FLOPs x1.065 there against x1.048 under 0.12.1), so it scored within 0.05% of the submitted file. The truncated SVDs cost 15.98 GFLOPs per network (5.3% of the bill).

7.4 Exact fast arithmetic and one approximate level

Bill of the submitted file (mean over public mini networks 0-2, flopscope 0.12.3, 303.4 GFLOPs per network):

Part	GFLOPs per network	Share
Strassen-Winograd leaf products	233.05	76.8%
Strassen-Winograd sums and copies	29.54	9.7%
propagation outside the products (einsums, small products, gathers)	20.23	6.7%
old-source compression (8 truncated SVDs)	15.98	5.3%
corrector	2.19	0.7%
approximate-level encodes	1.32	0.4%
layout reshapes	1.04	0.3%

Strassen-Winograd (credited in 2.2). At our shapes the products routed through it would cost about 480 GFLOPs per network as plain products; at recursion depth 4-6 their leaves cost about 240 and their sums about 30. That saving is the textbook one. Our addition is a layout: the seven children of every recursion level are one axis of a preallocated array, so all leaf products of a call are one batched product, each level's sums are a few elementwise operations, and the deepest level's sums overwrite their inputs. It does not change the arithmetic. Against our previous implementation it cut C/B from 0.196 to 0.194 at identical predictions (shot 13 to shot 15) and cut the client calls and wall time that bound the recursion depth: one level deeper on some products cut C/B 2.7% at equal MSE but multiplied wall time by 1.45.

Symmetric covariance by blocks (`_covsym`). $S = W^{\wedge}T (C W)$ is symmetric. We compute the upper off-diagonal half block and the diagonal blocks at finer scales, batched, then mirror. Exact up to float32 rounding.

One approximate level (switch `_APA_ORI`). At the top level of the 3T[^]T products and of the pushes that carry no covariance rows, one level of the Bini-Capovani-Lotti-Romani <3,2,2;10> approximate bilinear scheme replaces 12 block products by 10, with $\epsilon = 2^{-8}$ in float32; the 10 products then run through Strassen-Winograd. Which operand axes take the 3 x 2 and 2 x 2 block split is fixed per product kind in `_APA_ORI` and `_BINI`. This is not exact. Covariance rows do not tolerate it: on four networks MSE rose 17%, 4.9% and 1.1% at $\epsilon = 2^{-8}$, 2^{-9} and 2^{-10} . Its clean effect is in section 7.1.

7.5 Trajectory (development)

Grader scores of milestone submissions (all of them, with their grader values, are in the repository's `submissions/shots.csv`). Up to shot 33 the grader ran flopscope 0.12.1; from shot 34, 0.12.3, which prices truncated SVDs higher and raised our bill about 2% for the same code. Rows often bundle several changes. The last column is the acceptance read on full 0-47 (adjusted ratio against the build before it) where one was made; earlier shots were adopted on screens.

Shot	Date (KST)	Main change	Score	Final MSE	C/B	Acceptance read
anchor	10-04	the kit's covariance-propagation example	3.93e-07	3.93e-06	0.024	
1	10-04	port of the reference SIMPLE K = 3 engine	2.529e-08	3.780e-08	0.669	
4	10-04	+ old-source compression, Strassen-Winograd	1.368e-08	3.890e-08	0.352	
6	10-04	augmented terms with the memoryless law; two legs	5.288e-09	2.238e-08	0.236	
8	10-05	+ 1c	4.298e-09	2.007e-08	0.214	
9	10-05	+ 1aw	4.186e-09	1.955e-08	0.214	
11	10-05	the learned corrector replaces both	4.003e-09	1.920e-08	0.208	MSE x0.9787 [0.9707, 0.9862], 35/48
13	10-07	saturation-aware restriction	3.782e-09	1.927e-08	0.196	
15	10-07	recursive in-place Strassen-Winograd layout (same predictions)	3.730e-09	1.927e-08	0.194	
17	10-08	dead birth neurons gathered out	3.488e-09	1.936e-08	0.180	
19	10-12	symmetric covariance by blocks, the approximate level, deeper recursion	3.146e-09	1.958e-08	0.161	
21	10-14	weighted compression; star fold	3.002e-09	1.861e-08	0.161	
22	10-14	restriction package; split einsum	2.849e-09	1.856e-08	0.154	
23	10-14	corrector retrained on the new engine	2.837e-09	1.834e-08	0.155	corrector alone x0.988
27	10-15	G fold (s = 0.5)	2.749e-09	1.814e-08	0.152	x0.9838 [0.9784, 0.9890], 41/48
28	10-15	corrector reads the fold taps	2.654e-09	1.745e-08	0.152	x0.9793 [0.9720, 0.9863], 38/48
29	10-15	3T [^] T band 3.0 and rows a > -2.0	2.548e-09	1.754e-08	0.145	x0.960, 48/48
30	10-15	3T [^] T rows a > -1.5; more corrector taps	2.433e-09	1.714e-08	0.142	x0.948, 46/48
33	10-16	retrained corrector; push threshold 1.75	2.325e-09	1.681e-08	0.138	x0.9743 [0.9656, 0.9829], 38/48
34	10-18	distilled corrector; kept-subspace SVD (first shot under 0.12.3)	2.325e-09	1.664e-08	0.140	x0.9752 [0.9673, 0.9832], 37/48
36 (336955)	10-18	exact trims	2.298e-09	1.662e-08	0.138	x0.9935 [0.9916, 0.9955], 36/48 (against shot 35)

Shot	Date (KST)	Main change	Score	Final MSE	C/B	Acceptance read
38 (337371)	10-21	compression rank 256; fold-layer pushes restricted from layer 4	2.297e-09	1.559e-08	0.147	x0.9854 [0.9754, 0.9961], 32/48 (against shot 36, FLOPs under 0.12.1)
39 (337488)	10-21	memoryless law off in the engine, scale mode taken out of all three cumulants (section 7.6); corrector retrained	2.188e-09	1.487e-08	0.147	x0.9575 [0.9470, 0.9681], 42/48 (against shot 38)

Shots 38 and 39 came after the file this write-up describes. Shot 39 (337488) is one of our two designated submissions, with 335519; section 7.6 describes its change. The write-up stays with 336955, the file every ablation here measures. From shot 11 on, adoption needed the upper end of the 95% interval of the acceptance read below 1 and at least 30 of 48 networks better. The one departure we know shaped the shipped file is the G-fold scale (section 6.2).

7.6 The scale mode replaces the memoryless law (shot 39)

il.miscusi's Phase 2 write-up (forum 18248 and its PDF; prose only) takes a "scale mode" out of the carried cumulants: one scalar epsilon per layer, fitted from the off-diagonal (i,i,j) slice of the activation third cumulant and removed from the covariance, the third and the fourth cumulant together. They report that it makes a carried fourth-cumulant core pay and that it hurts a chain that keeps the fourth cumulant memoryless. Our engine is such a chain, so we measured it before porting it.

Readable engine (`research/r38/k3_acc_tc.py`, a copy of our augmented research engine), mini 0-9, engine alone. Here $\epsilon = (f/2) \langle D_{21}, S \rangle / \langle S, S \rangle$, with D_{21} the off-diagonal (i,i,j) slice and $S_{ij} = v_i m_j + 2 m_i C_{ij}$ (activation variance v , mean m , covariance C). The covariance becomes $(C - \epsilon m m^T) / (1 + \epsilon)$, the (4) and (2,2) slices of the fourth cumulant lose $12 \epsilon v_i^2$ and $4 \epsilon (v_i v_j + 2 C_{ij}^2)$, and the newborn third-cumulant source loses $2 \epsilon \text{Sym}(C(x) m)$. Ratios are of mean final-layer MSE against the same engine with the shipped law.

Law (multiplier on LAM_REFIT)	f	Final MSE ratio	Lower MSE
1	0.02	0.991	7/10
1	0.03	1.032	3/10
0.5	0.1	1.092	1/10
0.25	0.1	0.950	8/10
0.1	0.1	0.912	10/10
0	0.1	0.906	10/10
0	0	1.987	0/10

With the full law the extraction buys nothing, as il.miscusi found. With the law at zero it is worth x0.906, and the law at zero without it costs x1.99. On this chain the extraction recovers all of the law's measured benefit and about 9% more. Our reading is that much of what the one-scalar off-diagonal core was absorbing is the scale mode; that is not tested beyond these ratios. The partial versions we tried are much worse in this implementation: against the $f = 0.1$, law x0.1 row, taking the mode out of the covariance alone gives x35, out of the covariance and the fourth cumulant x25, out of the covariance and the third x6.5. So we take it out of all three together and do not read these ratios as the parts' separate contributions. Epsilon's spread across networks is small (standard deviation 1.4-6.4% of its mean per layer on mini 0-9, `research/r38/out/g3.jsonl`), so a fixed per-layer table costs little (x0.911 with the law at zero). Carrying the fourth-cumulant core on top of the extraction adds about 2% at best (x0.889: transport 0.5, $f = 0.1$, law x0.1) for extra n^3 products, so we did not port the carry.

Production (`estimators/r38/r38_sub_x100_s2`, generated from shot 38's file by `research/r38/make_r38.py`): the law's terms are zero in the engine's propagation (its two rows are masked in the engine's per-neuron vector), while the corrector still reads those rows as inputs. After the birth window the covariance, the fourth-cumulant row sums that feed the core's diagonal, the new source's E leg and one feed pair take the extraction with the fixed table. Engine alone against shot 38's engine on mini 0-4 (`research/r38/prod/`): final MSE x0.931 (5/5), FLOPs x0.997; scaling the table by 0.7, 0.85, 1.08 and 1.4 gave 1.056, 0.970, 0.931 and 1.072. Shot 38's corrector did not transfer to the new engine (final MSE about 2.9x shot 38's on mini 0 and 1), so we retrained it on the new engine's dumps of a 406-network pool (single model, 60 epochs, three seeds; the rule picked seed 2 on mini 44-49, x0.788 of the engine there, `research/r38/logs/pick.txt`).

Acceptance read against shot 38 on full 0-47 (`research/r25a/gru/s_read_r38.json`): x0.9575 [0.9470, 0.9681], 42/48 better, FLOPs x0.998. A student distilled from twelve seeds read x0.9973 [0.9928, 1.0020] against it on the same networks (`s_read_r38c_vs_r38.json`) and was not submitted. Graded: 2.1878e-09, x0.952 of shot 38.

Clean read of 337488 on the untouched corpus (the 48 networks of section 7.1, same harness, floor-corrected; `research/wu/kp/r38sub.json1`, `r38eng.json1`). The engine alone is 337488's file with the same lines changed as in section 7.1's "engine alone" row (`estimators/r38/r38d_eng_base`).

Comparison	Adjusted	MSE	FLOPs	Lower adjusted score
337488 against 336955	0.957 [0.943, 0.971]	0.899 [0.886, 0.913]	1.064	41/48
337488's engine alone against 336955's engine alone	0.948 [0.935, 0.962]	0.891 [0.878, 0.904]	1.064	43/48
337488's corrector, on against off	0.803 [0.787, 0.819]	0.797 [0.782, 0.813]	1.007	48/48

The same on the public one-shot selection set of section 7.2 (88 networks, read once on 10-16 to pick between earlier candidates, before 337488 existed; not floor-corrected; `research/wu/pub/r38sub_*.json1`, `r38eng_*.json1`):

Comparison	Adjusted	MSE	FLOPs	Lower adjusted score
337488 against 336955	0.954 [0.944, 0.965]	0.898 [0.888, 0.907]	1.063	74/88
337488's engine alone against 336955's engine alone	0.956 [0.945, 0.966]	0.898 [0.889, 0.908]	1.064	70/88
337488's corrector, on against off	0.807 [0.795, 0.819]	0.801 [0.790, 0.813]	1.007	88/88

The gain holds on networks no part of 337488's development touched, on both sets, and it matches the grader's ratio of the two scores (2.1878e-09 / 2.2984e-09 = 0.952). It comes from the engine: the retrained corrector is worth about what the shipped one is worth on 336955 (x0.790 on the corpus, x0.802 on the public set, section 4.4).

8. What did not work

Each line was measured before we closed it. Ratios are final-layer MSE against the engine of the time unless noted.

Idea	Result	Why, as far as we can tell
Sampling, alone or as a control variate on the engine	best pure design 66x worse than the engine of the time; hybrid 9.7x worse	at this budget, sample noise is far above the engine's error
Readout calibration (affine, shift, coefficient refits, a κ^3 Edgeworth term)	x0.996 to x1.003	the error is closure error growing with depth, not a readout bias
Last-layer-only learned corrections	best x0.943 (mini 40-49)	most of the last layer's error arrives through earlier layers (section 3)
More corrector capacity or other recipes (hidden 48 or 64, averaged epochs, per-network loss weights)	none improved by 1%; halving the pool to 200 cost 1.15%	the learning curve is flat near our pool size
Subspace closure of the full fourth cumulant	see section 3	the useful fourth-order part spreads beyond any small subspace as width grows
Proxies for the non-hypertree third-order terms	x0.791 at n = 256, x0.827 at n = 512 (three seeds)	not affordable at n = 1024 in any form we found
Carrying the fourth-cumulant core with transport and regeneration terms	explains 92-94% of the exact core's off-diagonal (R^2) at n = 256, but x1.042 [0.997, 1.090] out of sample at n = 1024. With il.miscusi's scale-mode extraction added (forum 18248: the mode fitted each layer from the chain's own (2,1) third-cumulant slice and taken out of the covariance, the third and the fourth cumulant together), the best carried configuration of our readable engine was x0.889 of the shipped law on mini 0-9 (engine alone), short of the x0.85 we had set before porting it and about 2% better than the extraction without the carry (section 7.6); with the reference's own K211 production of the core in place of the law, the chain was 7x worse	closer to the exact core was not closer to the right mean here. The reference's core is about 4x our refitted law (LAM256 against LAM_REFIT), and a chain that carries it piles up a fourth cumulant the K = 3 truncation cannot use

Idea	Result	Why, as far as we can tell
Retuning shot 39's engine: the extraction table by layer group (x1.4 on layers 0-2 down to x0.54-0.6 on layers 9-14) and the G-fold scale (0.5 to 0.25)	engine alone x0.974 [0.959, 0.990] on mini 0-9 (9 of 10 lower); with shot 39's corrector x1.014 [1.008, 1.021] on the acceptance read; with the corrector retrained on the retuned engine x0.9998 [0.993, 1.007], 23 of 48	the corrector had already absorbed what the retune fixes: on mini 44-49 the extraction-table part alone was x1.004 through the old corrector, and the retrained corrector's ratio to its own engine there rose from x0.788 to x0.800
A "gain mode": the mean-direction kappa4 measured 9-73x larger than the engine implies, injected as an oracle	worse on all 5 networks (x1.047 to x1.103)	the injected shift moved the mean toward the error, not away
Shanks or Aitken acceleration across cumulant orders	correlation of the last-order step with the error at most 0.17; Aitken x387 or worse	the next order's error is not along the last step
Deeper Strassen-Winograd	C/B -2.7% at equal MSE, wall time x1.45	wall-bound
Randomized SVD instead of truncated SVD for compression	no saving at equal quality	at flopscope's prices a range finder of the needed quality costs as much
Approximate level on pushes that carry covariance rows	MSE +17%, +4.9%, +1.1% at epsilon 2^{-8} , 2^{-9} , 2^{-10}	covariance rows are too sensitive

Two exclusions were made for rule reasons, not measurements. Swapping a gather for a cheaper-priced scatter that moves the same data was left out under the fair-accounting rule. The Karstadt-Schwartz alternative-basis variant of Strassen-Winograd was left out because it is covered by US patent US10387534B2.

9. Reproducing the numbers

The graded artifact. Submission 336955 is the folder

k3_fast_aug_sc_2l_b_1c2k_rd_wm_ex35_satrole_fbv8_rsw_cf_sp_d6p_sb2_dc20x_r143bcxa8tv_r24wc10_r25ak4f10L_gA_r25ces_r25bc3e7x_r25agf05_r25aqx_r25av2a_opb30a15_r25av4kd05s3_opi175_opkcA_ops1cs_opr30 (each suffix names one step of the trajectory) with two files: estimator.py (sha256 f908afd327e6f22e67b254989d9240f98acf58510168f9ff8d83051fff0cb942) and wmsg.npz (sha256 46bb22a1171552d81c9e9266b403882321ad9039291163c650087a48d0eaf30b). Submission 337488 is the folder r38_sub_x100_s2, generated from shot 38's file by research/r38/make_r38.py and research/r38/build_sub.py: estimator.py (sha256 0d69dc7d0f85065c1c657696c157457b3e208e63f9decf4e7e6d83039c8a5dee) and wmsg.npz (sha256 7438a1b7543a63a03a5abd51799cb1b0bf389723d6f9637acc7295904aa17fa2).

Where things are in the code.

Component	Code	Switch
per-layer engine	_AugEngine.propagate	
push and every large product	_sw, _SwPlan	sw_depth, sw_hi_push, sw_hi_tt
symmetric covariance	_covsym	_COVSYM
approximate level	_apa, _apa_enc, _BINI	_APA_ORI
memoryless law	LAM_REFIT	Estimator(lam=...)
star fold, G fold	blocks k4f, gf in propagate	_K4F, _GF
restriction	_sat_size, _sat_bins, _dc_sizes, _r25_up	_SAT, _DC, _R25B
compression	_r25_kc	age, rank, _WC, _R25B['kc']
corrector	_wmsg, wmsg.npz	Estimator(gru=False), _QX

Repository. <https://github.com/yeongjunyoo/arc-whitebox-2026-phase2> (MIT) holds the graded folders of 336955, shot 38 and 337488 with their hashes; make_variants.py, ablate.py, paired.py, kp_floor.py, report.py and the drivers that produced sections 5, 6.3, 7.1, 7.2 and 7.6, with the raw per-network results; the corrector's dump, training, distillation and evaluation scripts, with ladder.py and the per-seed results of section 4.5; the band study of section 5

(`research/wu/mech/`); the readable engine, generator and grids of section 7.6 (`research/r38/`); the scripts and outputs that fitted `LAM_REFIT` and sized the restricted shapes; and the trajectory of section 7.5 (`submissions/shots.csv`). Its `README.md` lists every path.

10. Credits and use of AI tools

Credits. ARC's factored cumulant propagation, `alignment-research-center/mlp_cumulant_propagation` at commit 93d091a (MIT License), and arXiv:2605.05179. Team 504aldo's forum post 18218 for the two-leg identity with its randomized residual, the memoryless fourth-cumulant law, age-based compression of old sources and Strassen-Winograd as flopscope operations. mliston's Phase 1 write-up (forum post 18170) for a GRU over the closure's moment state with messages through W. Jacob Hilton's LessWrong comment on learned networks that consume cumulant estimates. The Bini-Capovani-Lotti-Romani $\langle 3,2,2;10 \rangle$ coefficients from `github.com/arbenson/fast-matmul` (BSD-2-Clause), verified exactly. Keenan Pepper's public corpus `keenannepper/arc-whestbench-p2-d8b-corpus-14k` for the untouched evaluation networks of section 7.1. Related and concurrent Phase 2 work cited above: `kai_leal_gonzalez` (18246, the (3,1) slice), `Artyom_Mazur` (18249), `Nurlanggs` (18252, oracle injections), `Susupro` (18254: exact $K = 4$ at $n = 1024$, a corrector with the same linear carry, active-set trimming), `koushik_rudra` (V34, dead-unit pruning, as cited in 18254), `il.miscusi` (18248, the scale-mode extraction that submission 337488 ships, section 7.6). Ideas from other participants were taken from their posts' prose only. No other participant's code was read or used.

Use of AI tools. This submission was built with heavy use of AI agents. Anthropic's Claude models, running as coding agents in Claude Code, wrote nearly all of the estimator code and the research scripts, ran the experiments, kept the research notes and drafted this write-up. OpenAI GPT models (through the Codex CLI) and Google's Gemini models (through the Antigravity CLI) read drafts of this write-up as independent critical reviewers, and a GPT model checked flopscope's charges against its published cost model (it found no divergence). The author set the goals and the constraints, decided what to submit, reviewed the results and this text, and takes responsibility for its accuracy. Where we are unsure why something works, mainly what the corrector has learned, the text says so.

Appendix A. Networks and what they were used for

Public data: `aicrowd/arc-whestbench-public-2026`, revision v2-phase2 (mini: 100 networks; full: 16 networks per shard, shard k holds ids $16k$ to $16k + 15$).

Set	Networks	Used for
corrector training pool	mini 0-43 and 51-58; 348 full networks from shards 6-18 and 22-35	training the corrector, its teachers and students
early stopping	mini 44-49	every corrector training; also part of V90
validation (V90)	mini 44-49 and 84 other full networks of the same shards 6-18 and 22-35 (no network in common with the pool; both id lists are in the repository)	choosing the student; section 4.5 uses the 84 full ones
shape sizing	392 full networks, ids 112-303 and 352-575 (pool and V90)	the allowed shapes of section 5
law fitting	mini 50-52 (51 and 52 are also in the pool)	the scale and five group refits of <code>LAM_REFIT</code>
screen	mini 0-19 (inside the pool)	deciding what to read next
acceptance read	full 0-47	every adoption after shot 11
one-off reports	full 48-95	read once for single questions
one-shot selection set	mini 60-99, full 304-351	read once on 10-16 to choose the designated submissions; section 7.2
untouched corpus networks	Keenan Pepper's corpus, idx 3408-3593 (48)	the clean ablation of section 7.1 only (sections 4.4, 5 and 6.3 quote it)

No network is in two of the pool, V90, the acceptance read and the one-shot selection set.