

---

*Statement on the use of AI.* This report was written by humans and AI. Large language model (LLM) based assistants were used in the preparation of this manuscript for editing, some initial drafts, writing, especially of the Appendix, and rewriting of human text, table and plots creation, formatting, literature triage, and equations. The code of our submission has been entirely written by AI under our supervision. For this write up, our main submission is **#337631**. The submission code (see [here](#), around 3800 lines of code), has been distilled by an LLM (Table 5.1) to recap the main ideas (see [here](#), around 1000 lines of code). We mainly used LLMs such as GPT in the Sol (6, 6.1) and Astra families, and Claude Opus (5, 5.5) and Fable families (5, 5.1), and we also briefly used Gemini Argon 4. No text, derivation, figure, or result was accepted without verification by the authors, who also ran cross-verification with LLMs, who have reviewed and edited all content and take full responsibility for the correctness of the scientific claims presented here.

---

## The ARC White Box Challenge Phase 2 writeup

TEAM PUFFI\*

### ABSTRACT

We present our Phase 2 submissions (**#337479** and **#337631**), which modify the  $K = 3$  cumulant propagation algorithm of [Wu et al. \(2026\)](#). Our improvements are based on mechanistic observations, as well as on engineering work. In particular, we exploit the positive homogeneity of ReLU activation functions in two ways. First, decomposing the inputs of the networks in a spherical frame allows us to avoid capturing radial stochastic fluctuations that otherwise would need to be corrected by the cumulant propagation algorithm. Second, given that we approximate the true cumulants, we use a unit-variance coordinate system that prevents high-variance neurons from dominating propagated quantities. On the engineering side, we consider compressed third and fourth cumulants, as well as employ a bunch of techniques to save FLOPs. Most importantly, we do not perform *any* fitting to ground-truth Monte Carlo or any other kind of targets in our submissions. We avoid learned correctors and per-layer regression, relying purely on analytical closures to reach a raw final-layer MSE of  $1.174 \times 10^{-8}$  (16.0% of the  $2^{41}$  FLOP budget; adjusted score  $1.884 \times 10^{-9}$ ) in **#337479** and  $1.211 \times 10^{-8}$  (15.4% of the budget; adjusted score  $1.861 \times 10^{-9}$ ) in **#337631** on the 100-network public benchmark. Finally, we finish with presenting stronger estimators at the end of the text (Appendix C), developed after the submission deadline, which improve our estimators' MSE at the final layer by around 8–11% at a similar computational budget (15.7%).

### 1. INTRODUCTION

Modern Artificial Intelligence (AI) systems are mainly based on deep neural networks. In particular, the extremely popular and powerful Large Language Models (LLMs), based on the transformer architecture, have deep neural networks at their core and have reached unprecedented capabilities that might also pose risks to humans. However, we still do not understand how deep neural networks work internally, and hence we do not fully understand which consequences we should expect from such systems. Assuming future increases in capabilities of AI systems will still be based on deep neural networks at their heart, working towards mechanistic understanding is of extreme importance.<sup>1</sup>

More concretely, following an Alignment Research Center (ARC) blog post setup ([Neyman et al. 2025](#)), consider a trained deep neural network  $M_{\theta}$  with fixed parameters  $\theta$ . This network takes an input  $\vec{X} \in \mathcal{X}$  (e.g. a user query), distributed according to an input law  $p(\vec{X})$ , and outputs  $\vec{Y} = M_{\theta}(\vec{X}) \in \mathcal{Y}$ . Imagine we have white-box access to the internals  $\theta$  of this network: how can we know whether the system might induce rare, catastrophic consequences?

If the network  $M_{\theta}$  is fairly robust, it will likely have developed a behaviour where catastrophic failures  $\mathcal{R}$  are very rare events (as frequent failures would otherwise be washed out during training). To formalize this, imagine we have a *perfect* classifier  $\mathbf{C}$  that outputs 0 if the network has no problems, with respect to our choices that, for the sake of

\* Emails: [omardsubs@proton.me](mailto:omardsubs@proton.me), [eppie.gov@gmail.com](mailto:eppie.gov@gmail.com), [lawrence.dam55@gmail.com](mailto:lawrence.dam55@gmail.com)

<sup>1</sup> Even in the case where deep neural networks may be superseded, we can hope to transfer progress to other systems.

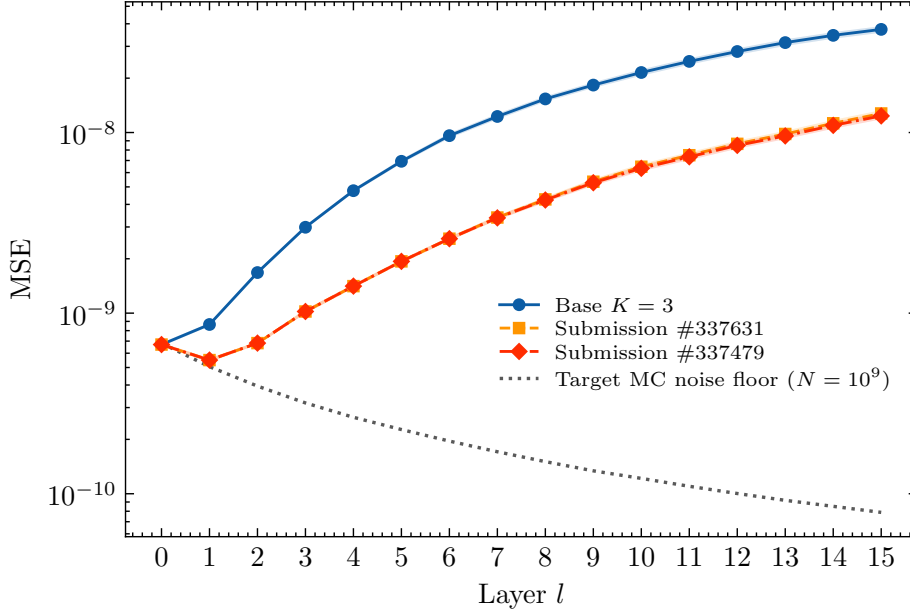


**Figure 1.** Mean squared error (logarithmic scale) as a function of layer depth  $l \in \{0, \dots, 15\}$ , averaged over 30 width-1024, depth-16 networks (IDs 600--629 from the full split of [aicrowd/arc-whetbench-public-2026](#), revision v2-phase2; shaded bands show  $\pm 1.96$  SEM, i.e. 95% confidence intervals). Shown are the baseline  $K = 3$  cumulant propagation algorithm (Wu et al. 2026), our two zero-fit Phase 2 submissions (#337631 at  $1.275 \times 10^{-8}$  and #337479 at  $1.236 \times 10^{-8}$  at  $l = 15$ , compared to  $3.724 \times 10^{-8}$  for Base  $K = 3$ ), and the analytical sampling-variance floor  $\text{Var}[h^{(l)}]/10^9$  of the  $N = 10^9$  Monte Carlo ground-truth targets on the same 30 networks ( $6.81 \times 10^{-10}$  at  $l = 0$  down to  $7.89 \times 10^{-11}$  at  $l = 15$ , matching the dataset’s empirical final-layer variance floor  $7.88 \times 10^{-11}$ ). For comparison, spending the entire  $2^{41}$  FLOP budget on naive Monte Carlo forward passes ( $N_{\text{MC}} = 2^{41}/(16 \times 2n^2) = 65,536$ ) yields an expected final-layer MSE of  $\text{Var}[h^{(15)}]/N_{\text{MC}} = 1.20 \times 10^{-6}$ .

the argument, we will describe as *good*, and 1 if the network has *problems*,

$$\mathbf{C}(M_{\theta}(\vec{X})) \in \{0, 1\}. \quad (1)$$

The quantity of interest is then the expected failure rate over the input distribution,

$$\mathbb{E}_{\vec{X} \sim p(\vec{X})} [\mathbf{C}(M_{\theta}(\vec{X}))] = ? \quad (2)$$

One obvious way to evaluate this expectation is with Monte Carlo sampling. We could draw, say,  $N = 10^9$  or even  $N = 10^{12}$  inputs. However, black-box sampling has three major issues: first, we need to run the system on every sample, so even  $N = 10^6$  forward passes can be very expensive; second, a problematic event might lie so far in the long tails that we would need  $N = 10^{16}$  samples to see it even once; and third, what if by running the system itself we end up triggering a rare catastrophe that we could have avoided with better analytical methods? For example, triggering could be caused by  $Q$  copies of the original system running in parallel. We do not want that.

To build intuition of “catastrophe”, imagine having a candy machine  $M_{\theta}$  whose internal gears  $\theta$  take a random scoop of ingredients  $\vec{X} \sim p(\vec{X})$  and dispense a piece of candy  $\vec{Y} = M_{\theta}(\vec{X})$ . Suppose the machine is slightly broken so that, very rarely, a particular combination of ingredients causes the gears to jam and output a hard stone instead of a candy, and biting into that stone would completely crush your teeth ( $\mathbf{C}(M_{\theta}(\vec{X})) = 1$ ). If you treat the machine as a black box and only test it by running it  $N$  times on a limited budget (say  $N = 100$  candies when each run costs money), seeing zero stones in those 100 runs does not rule out a stone appearing on run 1000 and crushing your teeth or those of your customers. On the other hand, if you can open the candy machine and inspect its internal gears  $\theta$  directly (white-box access), you can trace how the distribution of ingredients  $p(\vec{X})$  propagates through the mechanism to estimate how often a stone comes out, without having to manufacture billions of candies, making both your bank account and your customers happy.

In the ARC White Box Phase 2 benchmark, this problem is distilled into a clean mathematical testbed: given a bias-free  $L$ -layer ReLU network with known weight matrices  $\theta = (W_1, \dots, W_L)$  and standard Gaussian input  $\vec{h}^{(0)} =$

$\vec{X} \sim \mathcal{N}(\vec{0}, I_n)$ , our goal is to mechanistically estimate the expected activation vector  $\vec{\mu}_l = \mathbb{E}[\vec{h}^{(l)}(\vec{X})]$  at every layer  $l = 1, \dots, L$  under a strict compute budget of  $2^{41}$  FLOPs.<sup>2</sup>

Figure 1 shows the MSE behaviour with layer, spanning a few methods across 30 width-1024, depth-16 networks (IDs 600–629 from the full split of the official Hugging Face benchmark dataset [aicrowd/arc-whestbench-public-2026](https://huggingface.co/datasets/aicrowd/arc-whestbench-public-2026), revision v2-phase2). In particular, we highlight how our two Phase 2 submissions (#337631 and #337479) reduce the error accumulation with depth by around  $3\times$  ( $2.92\times$  in #337631 and  $3.01\times$  in #337479) at the final layer relative to the baseline  $K = 3$  cumulant propagation algorithm of Wu et al. (2026), alongside the target noise floor of the  $N = 10^9$  Monte Carlo ground truth.

The goal of the challenge was to improve upon the original ARC cumulant propagation algorithm, under the given compute budget. Using LLMs, we came up with fit-free, but structurally tuned, mean estimators that are based on ARC’s original  $K = 3$  algorithm. The goal was to minimize the *adjusted score*, a measure proportional to mean-squared-error (MSE) and compute used. While fitting can save FLOPs directly, through replacement of more expensive analytical states, and indirectly, through avoiding biases due to approximations one would need to make to carry out expensive states, we decided to opt for a strategy mainly based on fit-free prompting with the hope of LLMs achieving a new kind of estimator. However, given that our estimator has a similar error-growth-per-layer behaviour to the original  $K = 3$  algorithm, there is still work to do.

In Section 2 we will present a summary of our solution. Then, we will conclude in Section 3. The Appendix has an extensive set of details, as well as a modification of a previous version of our submission that has a 10% lower MSE and similar compute, but that we did not manage to make lower-cost before the end of the competition.

Throughout these notes we use  $\mathbb{E}[\cdot]$ ,  $\mathbb{E}[\cdot]$ , or  $\langle \cdot \rangle$  to indicate expectations, and when needed we use subscripts to indicate with respect to what we are averaging, e.g.  $\mathbb{E}_{\vec{X}}[\cdot]$  or  $\langle \cdot \rangle_{\theta}$ .

## 2. SUMMARY OF OUR SOLUTION

### 2.1. Core components of the estimator

We build an estimator by modifying the basic cumulant propagation estimator  $K = 3$  from Wu et al. (2026). As a design choice we decided to avoid any form of fitting to ground truth to improve our accuracy while reducing the effective number of FLOPs. However, we do have some structural tuning, such as choosing thresholds for pruning or iterating over low-dimensional ranks of matrices to improve our adjusted score. But except for this, we base our solution on analytical components, to be mechanistic as much as possible (Lee et al. 2018; Roberts et al. 2022).

As in the original  $K = 3$  algorithm, we carry layer by layer a state that allows us to determine at layer  $l$  its mean  $\vec{\mu}_l$ , covariance  $C_l$ , third and fourth cumulants  $\kappa_{3,l}$ ,  $\kappa_{4,l}$  estimates, respectively (McCullagh 1987). In particular, we similarly compress and factorize  $\kappa_3$  and  $\kappa_4$  into  $\mathcal{O}(n^2)$  objects that allow us to stay under the competition budget.

Our main findings specific to our algorithm are the following:

- The positive homogeneity of the ReLU activation function married to the zero-bias in the linear layers allows us to factor out the radial component of the inputs and propagate only the angular component on the sphere, removing norm fluctuations that are present in the standard input  $\vec{X}$  (Cho & Saul 2009). On our 5-network component-ablation slice (IDs 32–36, five width-1024, depth-16 networks from the full split of [aicrowd/arc-whestbench-public-2026](https://huggingface.co/datasets/aicrowd/arc-whestbench-public-2026); see Figure 3), this reduces the final MSE at the last layer by around 18–20% (18.1% when paired with the exact first-layer Gram covariance  $\Sigma_1 = W_1^T W_1$ , and 13.4% with an isotropic scalar, reaching up to 36% in early layers). See Appendix B.1 for details.
- By positive homogeneity of the ReLU again, and by its elementwise application on each neuron, we can rescale each neuron’s incoming weights by the reciprocal of its pre-activation standard deviation,  $s_{l,i} = 1/\sigma_{l,i}$ , and multiply the outgoing weights in the next layer by  $\sigma_{l,i}$ . In the limit of exact cumulant propagation and infinite precision, this is an exact coordinate reparameterization (a gauge transformation that we call the unit-prevariance gauge). Crucially, however, while rescaling leaves the exact network unchanged, it changes what a *compressed* approximation retains: choosing unit-variance coordinates before taking unweighted partial traces (for the fourth-cumulant core  $A_l$  and third-cumulant covariance star  $\vec{v}_l$ ) and Frobenius-norm SVD truncations (in the skewness history  $\mathcal{H}_l$ ) prevents high-variance neurons from dominating the retained state by powers of  $\sigma_{l,i}$ , making the compressed

<sup>2</sup> In mathematical descriptions,  $l = 0$  denotes the input  $\vec{X}$  and  $l = 1, \dots, L$  denote the  $L$  network layers; in the benchmark dataset, code, and figures, the  $L = 16$  network layers are 0-indexed as  $l \in \{0, \dots, 15\}$ .

closure invariant to arbitrary positive neuron rescalings. Even at the very first layer of a near-homogeneous He-initialized network, accounting for per-neuron variance differences in the spherical fourth-cumulant Gram closure reduces final-layer MSE on the  $K = 3$  recurrence by an extra 5.5% relative to an isotropic variance assumption ( $3.232 \times 10^{-8} \rightarrow 3.056 \times 10^{-8}$  on IDs 32--36); across all 16 layers, while the online gauge changes MSE by only 0.7% on unperturbed i.i.d. He-initialized networks whose raw pre-activation variances already cluster near  $O(1)$  ( $1.152 \times 10^{-8}$  vs.  $1.144 \times 10^{-8}$  on IDs 32--36), under an exact ReLU-preserving hidden-neuron rescaling in  $[0.5, 2.0]$  the ungauged estimator degrades by over  $5,000\times$  ( $6.105 \times 10^{-5}$ ) whereas the gauged estimator remains invariant up to float32 rounding ( $1.154 \times 10^{-8}$ ). See Appendix B.2 for details.

- *Replacing the scalar fourth cumulant with an  $n \times n$  matrix core  $A_l$  that remembers the past:* whereas the baseline  $K = 3$  algorithm compresses the entire  $n^4$  fourth cumulant  $\kappa_{4,l}$  into a single scalar  $\alpha_l$  at every layer — forgetting which specific neurons or directions have heavier tails — our estimator carries a symmetric  $n \times n$  trace-core matrix  $A_l \in \mathbb{R}^{n \times n}$  across layers (transported forward via  $G_l = \widetilde{W}_l^T A_{l-1} \widetilde{W}_l$ , where  $\widetilde{W}_l$  is the gauged weight matrix). Intuitively,  $A_l$  tracks the partial trace of  $\kappa_{4,l}$  in the unit-prevariance gauge via the reconstruction  $\kappa_{4,l} \approx \text{Sym}_6(A_l \otimes I_n)$  (McCullagh 1987),<sup>3</sup> and is updated at each ReLU layer so that its diagonal reproduces the single-neuron kurtosis  $\kappa_{4,l,iii}$  within the fourth-order closure. Replacing the scalar  $\alpha_l$  with this matrix core  $A_l$  (alongside the covariance-star vector summary  $\vec{v}_l$  of the third cumulant) gives our largest non-Gaussian accuracy gain, reducing the final-layer MSE by a further  $\sim 45\%$  relative to the baseline (see Figure 3 and Appendix B.3.2).
- *Correcting the matrix core  $A_l$  using the skewness history bank  $\mathcal{H}_l$ :* because  $A_l$  only captures an isotropic partial trace of  $\kappa_{4,l}$ , it misses directional fourth-cumulant terms that are generated by the third-cumulant (skewness) history bank  $\mathcal{H}_l$  (introduced in Section 2.2 below). To recover these missing pieces without forming an  $n^4$  tensor or double-counting contributions already represented by  $A_l$ , we add three targeted  $\mathcal{O}(n^2)$  analytical corrections on top of  $A_l$ :
  - (i) *tree-level quartic terms* read directly from the skewness bank  $\mathcal{H}_l$ , explicitly subtracting each term’s analytic trace projection onto  $A_l$  so nothing is double-counted with the core;
  - (ii) *multi-generation  $\kappa_3 \rightarrow \kappa_4$  trace contractions:* earlier skewness births continue to generate fourth-order kurtosis as they pass through subsequent ReLU kinks across depth. For every consumer layer from  $l = 4$  to the terminal layer, we retain a sliding window of the last 3 skewness birth generations in  $\mathcal{H}_l$  (omitting interactions older than 3 layers and elder-switched legs, and organizing the nested multi-step products via a backward Horner scheme that shares weight-matrix prefixes) to recover their diagonal kurtosis  $\kappa_{4,l,iii}$  and fold their  $(3, 1)$  gradient into the cubic slice, reducing final-layer MSE by 14.8% on IDs 32--36; and
  - (iii) *anisotropic directional response:* updating the fourth-order core requires contracting  $\kappa_{3,l}$  along the Gaussian variance-sensitivity direction  $u_k = \frac{\partial}{\partial \mu_k} \text{Var}_G[\text{ReLU}(Z_k)] = 2h_k(1 - d_k)$  (Edgeworth 1905; Cramér 1946). Because the core update only approximates this contraction through the isotropic covariance-star vector  $\vec{v}_l$ , our directional response contracts  $\vec{u}$  directly with the represented bank legs in  $\mathcal{H}_l$  and subtracts the covariance-star approximation’s existing contribution, reducing matched final-layer MSE by 4.1% (and improving the adjusted score by 1.0%) in a controlled 50-network public test.

Together, these three corrections reduce the final-layer MSE by an additional  $\sim 8\text{--}10\%$  of the baseline error (see Appendices B.3.1 and B.3.2 for details).

In Appendix B (see Figure 3) we give more details about which components drive the accuracy gains across depth.

## 2.2. Computational cost reductions

To understand how we save some computational cost, we will first give more details about how we represent higher order cumulants, than the second one, such as  $\kappa_3$  and  $\kappa_4$ . Details can be found in Appendix B but we will give some description here.

Following Wu et al. (2026), how is new skewness  $\Delta\kappa_3(\vec{h}^{(b)})_{ijk}$  created when pre-activations  $\vec{z}^{(b)}$  pass through the ReLU function? First of all, due to the large width of the network,  $n = 1024$ , we can employ the Central Limit Theorem

<sup>3</sup> For a symmetric  $A \in \mathbb{R}^{n \times n}$ ,  $\text{Sym}_6(A \otimes I_n)$  denotes the fully symmetric four-index tensor obtained by summing over the six distinct ways of assigning an index pair to  $A$  and the complementary pair to  $I_n$ :  $\text{Sym}_6(A \otimes I_n)_{ijkl} = A_{ij}\delta_{kl} + A_{ik}\delta_{jl} + A_{il}\delta_{jk} + A_{jk}\delta_{il} + A_{jl}\delta_{ik} + A_{kl}\delta_{ij}$ . In particular, its diagonal entries are  $6A_{ii}$ .

(CLT) that allows us to approximate pre-activations with a Gaussian distribution (with mean  $m_{b,m}$  and unit variance in our gauge). This helps us in writing an estimate for the third cumulant. Under this Gaussian reference, the expected first derivative (linear slope) of ReLU at neuron  $m$  is the probability that the neuron is active ( $z > 0$ ), given by the standard normal cumulative distribution function (CDF)  $d_{b,m} = \Phi(m_{b,m}) = \int_{-\infty}^{m_{b,m}} \phi(t) dt$ , while its expected second derivative (quadratic curvature right at the ReLU kink  $z = 0$ ) is given by the standard normal probability density function (PDF)  $\phi(m_{b,m}) = \frac{1}{\sqrt{2\pi}} e^{-m_{b,m}^2/2}$ . For three distinct neurons  $(i, j, k)$ , new skewness arises when one “hub” neuron  $m$  bends quadratically at the ReLU kink (with second-derivative curvature  $\phi(m_{b,m})$ ) while two off-diagonal covariance edges connect  $m$  to  $j$  and  $k$ . Thus, the third cumulant  $\kappa_3$  generated at birth layer  $b$  can be written as a sum over hub neurons  $m$  of factored two-dimensional leg matrices: one denoted by  $P^{(b)}$ , which we call the “Hub Identity” leg because at birth it is literally the identity matrix ( $P_{mi}^{(b)} = \delta_{mi}$ ), and two “Covariance-Curvature” branch legs  $H^{(b)}$  built from the off-diagonal covariance weighted by the ReLU gate  $\Phi$  and curvature  $\phi$ .<sup>4</sup> Figure 2 gives a schematic view of this leading birth structure, which reads

$$\Delta\kappa_3^{(b)} \approx \sum_{m=1}^n 3 \text{Sym} \left( P_{m\cdot}^{(b)} \otimes H_{m\cdot}^{(b)} \otimes H_{m\cdot}^{(b)} \right). \quad (3)$$

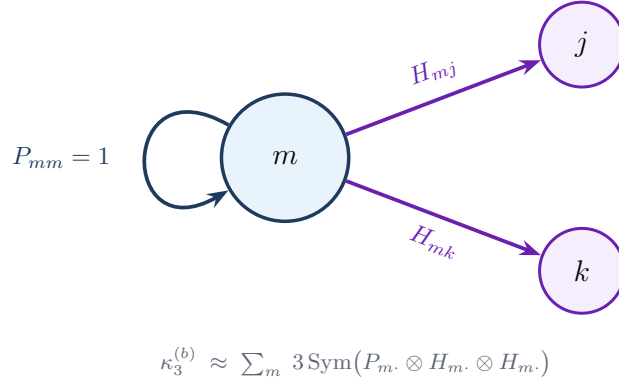


**Figure 2.** Schematic diagram of a third-cumulant source born at hub neuron  $m$  of layer  $b$ , formed by one hub-identity leg  $P_{mm} = 1$  and two covariance-curvature branch legs  $H_{mj}, H_{mk}$ . This picture has been generated with Fable 5.1.

Appendix B.3.1 gives more details about the full representation. How does a skewness source born at layer  $b$  propagate to a later layer  $l > b$ ? At each intervening layer  $l - 1$ , neuron  $k$  transmits small fluctuations linearly when it is active — with average slope  $d_{l-1,k} = \Phi(m_{l-1,k})$  at leading Gaussian order (and Edgeworth-corrected in the full closure) — and these fluctuations are then mixed across neurons by the weight matrix  $\widetilde{W}_l$ . Collecting the active-neuron probabilities into the diagonal gate matrix  $D_{l-1} = \text{diag}(\vec{d}_{l-1})$ , the effective linear step from layer  $l - 1$  to layer  $l$  is multiplication by the gated weight matrix  $D_{l-1} \widetilde{W}_l$ . Crucially, by multilinearity of cumulants, as we move to later layers  $l > b$  the  $P$  and  $H$  factors are transported forward simply by right-multiplying each leg by  $D_{l-1} \widetilde{W}_l$ . Most importantly, depending on the ReLU regime we can drop inactive rows and columns where a neuron is nearly dead ( $\Phi(m) \approx 0$ ) or sits deep in the positive linear regime where the ReLU has negligible curvature ( $\phi(m) \approx 0$ ). The axes involved in right-multiplying  $(P, H)$  by  $D_{l-1} \widetilde{W}_l$  are called the *transport axes*, whereas the row and column axes of the repeated-index matrix  $b_{ij} = \kappa_{3,l,ii}$  assembled from  $(P, H)$  to update the covariance and fourth cumulant are called the *slice axes*. Based on these, we have freedom to prune and compress the leg matrices while preserving the downstream skewness contractions.

Because  $\Delta\kappa_3^{(b)}$  is a sum of cubic products  $P^{(b)}(H^{(b)})^2$ , we cannot naively add  $P^{(1)} + P^{(2)}$  and  $H^{(1)} + H^{(2)}$  across different birth layers without creating fake cross-terms between layers. This means that as we go forward through layers, we must transport a growing list (“history bank”) of birth pairs  $\mathcal{H}_l = \{(P^{(b,l)}, H^{(b,l)})\}_{b=1}^{l-1}$ . However, transporting and

<sup>4</sup> To get explicit expressions, see the Appendix, but one can also simply expand the ReLU function, with Gaussian inputs, take expected value, and use Wick’s theorem.

reading every birth separately costs  $\mathcal{O}(L^2 n^3)$  computational work across depth, since each birth requires four  $\mathcal{O}(n^3)$  matrix multiplications per layer (two for forward transport and two for the quadratic slice readout). Hence, once early births reach age  $\geq 6$ ,<sup>5</sup> we project them onto a single shared right-subspace  $G_{\text{elder}}^{(l)} \in \mathbb{R}^{d \times n}$  ( $d \ll n$ ) that is transported once per layer.

With this picture in place, we can summarize the main techniques we use to reduce computational cost (see Appendix B.5 for explicit equations and implementation details):

- *Active-leg gate thinning and minimum-norm rank-2 refill of saturated slice rows:* while nearly dead neurons ( $d_i \approx 0$ ) can be pruned directly on transport and slice axes, saturated neurons deep in the positive linear regime ( $d_i > 0.985$ , where ReLU curvature  $\phi(m_i) \approx 0$  is negligible) still transmit existing skewness linearly with slope  $d_i \approx 1$ . As a result, their repeated-index slice rows  $b_{ij} = \kappa_{3,l,ii}$  enter the  $(2,2)$  fourth-order core update with  $\mathcal{O}(1)$  weight, and simply zeroing those rows roughly doubles ( $2.2\times$ ) the final-layer MSE. Instead of assembling the full slice on saturated rows, we evaluate the two required weighted row and column contractions directly on the factored bank legs and reconstruct the omitted rows via a minimum-norm rank-2 matrix that preserves both functionals while avoiding the full slice computation.
- *Shared elder basis:* cubic sources aged  $\geq 6$  are projected onto a shared orthogonal right-subspace  $G_{\text{elder}} \in \mathbb{R}^{d \times n}$  ( $d = 320$  in #337479,  $d = 256$  in #337631) transported once per layer, while their rank-3 birth-repair factors (which carry an analytic rank-1 trace-conservation column) remain on their cheap dense transport path.
- *Recursive  $2 \times 2$  block fast matrix multiplication and call fusion:* dense matrix products are routed through a Strassen–Winograd  $(2,2,2;7)$  block backend (Strassen 1969; Winograd 1971) with cached operand transforms, reducing matrix-multiplication FLOPs by  $(7/8)^{\text{levels}}$ , while scalar/vector recurrences are compiled into fused batched `einsum` tables to stay safely below the official 0.40s residual runtime cap (targeting an internal 0.35s safety margin; both official runs had maximum residuals of 0.374–0.375s).

### 3. CONCLUSIONS

In this report we presented our Phase 2 submissions (#337479 and #337631), which build upon the  $K = 3$  cumulant propagation algorithm of Wu et al. (2026). Our main contributions combine two exact consequences of the positive homogeneity of ReLU: radial-angular separation on the sphere and the neuron-by-neuron unit-prevariance gauge. Then, we implement a carried  $n \times n$  fourth-cumulant harmonic trace core  $A_l$ , targeted tree-level and Horner trace-contraction responses, and FLOP-saving reductions (including active-gate thinning with rank-2 slice refill, a shared elder subspace, and block fast matrix multiplication). We emphasize again that our submissions do not use *any* fitting to ground-truth Monte Carlo targets, relying solely on analytical closures and discrete structural tuning.

Looking ahead to future work, our main practical difficulty was the factored  $\kappa_3$  history bank: transporting and reading out the growing list of cubic births consumes the vast majority (75–80%) of our FLOP budget. While we found it difficult to compress this bank further during the competition without losing accuracy, we believe this  $\mathcal{O}(L^2 n^3)$  bottleneck is solvable, and it would already be a “breakthrough” solving it.

Finally, did we reach the “North Star”? If the North Star of mechanistic moment propagation is an estimator whose MSE is flat as a function of layer depth  $l$ , we are still far from it. As can be seen from Figure 1, although our estimators shift the error down by around  $3\times$  at the final layer, the slope (derivative) of the MSE curve with respect to layer depth remains very similar to that of the baseline  $K = 3$  case. In this sense, we think our estimator still belongs to the same asymptotic class of cumulant propagation as the original algorithm of Wu et al. (2026). Achieving a depth-flat MSE curve remains our North Star for the future; it is not obvious that this can be solved, though hopefully yes.

### ACKNOWLEDGEMENTS

We thank the Alignment Research Center (ARC) for organizing the White Box Estimation Challenge. We thank AICrowd for managing the competition, providing the FlopScope and WhetBench evaluation framework, and their support along the competition. We thank all the participants that made this competition fun.

### REFERENCES

<sup>5</sup> This number is again, a structural tuning, in the sense we iterate over numbers to find a good one that gives us a good adjusted score.

- Cho, Y., & Saul, L. K. 2009, in *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 22, 342–350
- Cramér, H. 1946, *Mathematical Methods of Statistics* (Princeton, NJ: Princeton University Press)
- Edgeworth, F. Y. 1905, *Transactions of the Cambridge Philosophical Society*, 20, 36
- Kibble, W. F. 1945, *Mathematical Proceedings of the Cambridge Philosophical Society*, 41, 12
- Lee, J., Bahri, Y., Novak, R., et al. 2018, in *International Conference on Learning Representations (ICLR)*
- McCullagh, P. 1987, *Tensor Methods in Statistics* (Chapman and Hall/CRC)
- Mehler, F. G. 1866, *Journal für die reine und angewandte Mathematik*, 66, 161
- Neyman, E., Lecomte, V., Wu, W., et al. 2025, *Competing with sampling*, Alignment Research Center Blog. <https://www.alignment.org/blog/competing-with-sampling/>
- Price, R. 1958, *IRE Transactions on Information Theory*, 4, 69
- Roberts, D. A., Yaida, S., & Hanin, B. 2022, *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks* (Cambridge University Press)
- Strassen, V. 1969, *Numerische Mathematik*, 13, 354
- Winograd, S. 1971, *Linear Algebra and its Applications*, 4, 381
- Wu, W., Lecomte, V., Winer, M., et al. 2026, *Estimating the expected output of wide random MLPs more efficiently than sampling*. <https://arxiv.org/abs/2605.05179>

## APPENDIX

## A. PROBLEM STATEMENT

We will focus on a neural network with  $L$  layers,  $n \times n$  weight matrices  $W_l$  (and zero bias, which in general could be absorbed into an extra coordinate with constant input 1, though the competition networks are strictly bias-free), and activation function ReLU.

In this write up we will use the following convention:

$$\vec{h}^{(0)} = \vec{X} \sim \mathcal{N}(\vec{0}, I_n), \quad \vec{z}^{(l)} = W_l^T \vec{h}^{(l-1)}, \quad \vec{h}^{(l)} = \text{ReLU}(\vec{z}^{(l)}) , \quad (\text{A1})$$

where  $\vec{X}$  is the input, and  $\vec{z}^{(l)}$ ,  $\vec{h}^{(l)}$ , are the pre-activation and activation at layer  $l$ , respectively, with layer 0 being the input. Our goal is to estimate *mechanistically*

$$\mathbb{E} [\vec{h}^{(L)}] \equiv \mathbb{E}_{\vec{X} \sim \mathcal{N}(\vec{0}, I_n)} [\vec{h}^{(L)}] \quad (\text{A2})$$

for fixed weights.

## A.1. Our harness

Our research harness orchestrated multiple LLM agents — mainly from the Fable, Astra, Opus, and Sol families — working in loops over the codebase and evaluated locally against the official FlopScope and WhestBench metering harness. We relied on single agents orchestrating same family agents, but we also paired agents of different families, so that one agent monitored, audited, or cross-checked the proposals and code changes of others against strict accuracy, FLOP, and runtime gates. Interestingly, we observed distinct behavioural tendencies across model families during optimization: for example, Opus agents tended to be much more optimistic and aggressive in proposing and attempting further optimizations, whereas Fable agents were noticeably more conservative and reticent to push speculative changes when progress became marginal.

In developing our solution, we generally asked the LLMs to avoid sampling-based techniques as much as possible, including quasi-Monte Carlo (QMC), as well as any form of fitting to targets, such as linear or non-linear regression. We were still open to fitting, but our hope was that a constrained LLM-based system would “evolve” towards a solution that is as mechanistic as possible, going into some new family of estimators beyond the ARC’s one.

Looking back, we think our solution, while it has some nice mechanistic observations, is mainly driven by careful closures and engineering, rather than a fundamentally new scientific methodology. Across the competition, we observed a tendency for LLM-guided searches to converge toward similar engineering optimizations (e.g. see Phase 1 write ups on the AICrowd forum). We feel a true scientific breakthrough would have achieved an order-of-magnitude ( $\sim 10\times$ ) reduction in MSE relative to the baseline  $K = 3$  algorithm under the  $2^{41}$  FLOP budget.

## B. OUR SOLUTION IN DETAIL

As mentioned in the main text, our estimator builds upon a modification of the baseline  $K = 3$  cumulant propagation algorithm of Wu et al. (2026). In this appendix, we distill the main analytical and engineering components of the LLM-developed solution we arrived at.

Starting from the original  $K = 3$  recurrence (Wu et al. 2026), we propagate at each layer the mean and covariance alongside factored third-cumulant ( $\kappa_3$ ) information and selected fourth-cumulant ( $\kappa_4$ ) responses, which together update the non-linear post-ReLU mean. The central challenge for us is that maintaining accuracy across depth requires carrying a growing history of cubic and quartic contributions without exceeding the  $2^{41}$  FLOP budget; our solution combines analytical closure improvements with computational reductions to make this tractable.

In the following subsections, we present our main improvements, some of which we think are mechanistic, over the baseline  $K = 3$  algorithm. To isolate the effect of the mathematical closures from pure FLOP-saving approximations, we evaluate progressive improvements stacked on top of the baseline  $K = 3$  estimator on 5 width-1024, depth-16 networks (IDs 32--36 from the full split of [aicrowd/arc-westbench-public-2026](#), revision v2-phase2).

## B.1. Radial homogeneity of the network

One of the cleanest mechanistic improvements found during our exploration exploits the positive homogeneity of bias-free ReLU networks. Figure 3 shows the relative reduction in MSE across layers as each progressive improvement is added to the base  $K = 3$  algorithm (Wu et al. 2026). Notably, in terms of compute, the spherical-radial separation itself requires virtually zero additional FLOPs.

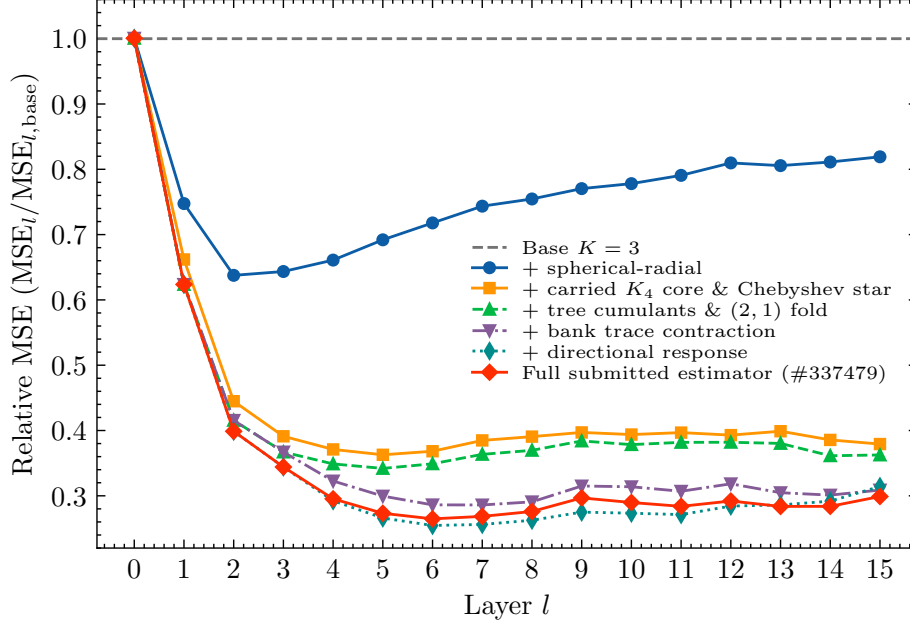


**Figure 3.** Relative mean squared error ( $\text{MSE}_l / \text{MSE}_{l,\text{base}}$ ) as a function of layer  $l \in \{0, \dots, 15\}$  across progressive improvements stacked on the base  $K = 3$  algorithm, up to our full submitted estimator (#337479), evaluated on 5 width-1024, depth-16 networks (IDs 32--36 from the full split of [aicrowd/arc-whetstbench-public-2026](#), revision v2-phase2). The spherical-radial curve uses the exact first-layer Gram covariance  $\Sigma_1 = W_1^T W_1$  (18.1% reduction at  $l = 15$ , vs. 13.4% for an isotropic scalar), and the intermediate rungs show the uncompressed analytical closures prior to elder-subspace projection and rank-2 slice refill.

To achieve this result, one has to start from the basics. Consider the property of the ReLU activation function:  $\text{ReLU}(\alpha z) = \max(0, \alpha z) = \alpha \max(0, z)$ ,  $\forall \alpha > 0$ . This is called positive homogeneity property of the ReLU.

Now, remember that a Gaussian input  $\vec{X}$  can be written as a radius vector times its direction on the unit sphere in  $n$ -dimensions:

$$\vec{X} = \|\vec{X}\|_2 \frac{\vec{X}}{\|\vec{X}\|_2},$$

where we measure length with an  $L_2$ -norm. To keep the coordinates of the direction vector with unit variance, the radius is scaled by  $\sqrt{n}$ . Hence, we end up separating the Gaussian input  $\vec{X}$  into a radial  $R$  and angular  $U$  components

$$\vec{X} = R\vec{U}, \quad R = \chi_n / \sqrt{n}, \quad \vec{U} \sim \text{Unif}(\sqrt{n} S^{n-1}),$$

where we use the fact that for a Gaussian distribution the scaled length  $R$  and direction  $\vec{U}$  are statistically independent.

Now, because the competition networks have bias-free linear layers and the positive homogeneity of the ReLU:

$$\vec{h}^{(0)}(\vec{X}) = R \vec{h}^{(0)}(\vec{U}) \implies \vec{h}^{(1)}(\vec{X}) = R \vec{h}^{(1)}(\vec{U}) \implies \dots \implies \vec{h}^{(l)}(\vec{X}) = R \vec{h}^{(l)}(\vec{U}).$$

This is regardless of the distribution in the middle layers! It implies that, exploiting independence of the variables,

$$\mathbb{E}[\vec{h}^{(l)}(\vec{X})] = \mathbb{E}[R] \mathbb{E}[\vec{h}^{(l)}(\vec{U})].$$

Our estimator takes advantage of this result. It propagates the angular input through the network, and considers separately the (scaled) radius. But why can this be better than propagating the cumulants of  $\vec{X}$  directly? An intuitive idea is that in the cumulant propagation algorithm, we do not need anymore to really consider fluctuations in the radius  $R$ ! Indeed, consider what happens after the first ReLU layer if  $R$  is left inside the input:

1. Before the first ReLU,  $\vec{z}^{(1)} = W_1^T \vec{X}$  has mean  $\vec{0}$ .
2. After the first ReLU,  $\vec{h}^{(1)} = \text{ReLU}(\vec{z}^{(1)}) = R \text{ReLU}(W_1^T \vec{U})$  has a positive mean vector  $\mathbb{E}[\vec{h}^{(1)}] > 0$  (componentwise).
3. The fluctuation of  $R$  acts on all coordinates of  $\vec{h}^{(1)}$  at once:
  - a draw with  $R > 1$  scales every neuron up together, along the direction of  $\mathbb{E}[\vec{h}^{(1)}]$ ;
  - a draw with  $R < 1$  shrinks every neuron together, along the same direction.
4. Formally, since  $R$  is independent of  $\vec{U}$  and  $\mathbb{E}[R^2] = 1$ , the law of total covariance gives, at every layer  $l = 1, \dots, L$ ,

$$\text{Cov}(\vec{h}^{(l)}(\vec{X})) = \text{Cov}(\vec{h}^{(l)}(\vec{U})) + \text{Var}(R) \vec{\mu}_l \vec{\mu}_l^T, \quad \vec{\mu}_l = \mathbb{E}[\vec{h}^{(l)}(\vec{U})], \quad \text{Var}(R) \approx \frac{1}{2n}.$$

The radius therefore injects a global rank-one variance “spike” into the covariance (along with corresponding terms in  $\kappa_{3,l}$  and  $\kappa_{4,l}$ ).

5. If we propagated  $\vec{X}$  directly, the covariance and cumulant approximations would spend capacity tracking the fluctuations of  $R$  through all  $L$  layers, even though the effect of  $R$  on the final mean  $\mathbb{E}[\vec{h}^{(l)}(\vec{X})]$  is known exactly in closed form!

By running the cumulant propagation on the fixed-radius input  $\vec{U}$  instead:

- the input norm  $\|\vec{U}\|_2 = \sqrt{n}$  does not fluctuate at all, so the cumulant propagation algorithm only has to track angular correlations on the sphere.
- only at the end we reintroduce the radial component  $R$  into every layer’s predicted mean  $\mathbb{E}[\vec{h}^{(l)}(\vec{U})]$ , by multiplying by an exact constant

$$\mathbb{E}[R] = \sqrt{\frac{2}{n}} \frac{\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n}{2})} = 1 - \frac{1}{4n} + \frac{1}{32n^2} + O(n^{-3}) \approx 0.99975589 \quad (n = 1024).$$

Hence, one only needs to make a small variation on the usual cumulant propagation algorithm for  $K = 3$ . Simply switching the input ingredients at layer 0 does the job. Explicitly, the angular input  $\vec{U}$  shares its first three cumulants with the Gaussian input:

- Mean:  $\mathbb{E}[\vec{U}] = \vec{0}$  (same as Gaussian).
- Covariance:  $\text{Cov}(\vec{U}) = I_n$  (same as Gaussian, so  $C_0 = I_n$ ).
- Third cumulant:  $\kappa_{3,0} = 0$  by the symmetry  $\vec{U} \leftrightarrow -\vec{U}$  (same as Gaussian).
- Fourth cumulant: because  $\|\vec{U}\|_2^2 = n$  cannot have large tails, a uniform vector on the sphere has a slightly negative, completely isotropic fourth cumulant (that is propagated to get the propagated trace used in the usual  $K = 3$  ARC algorithm),

$$\kappa_{4,0}[U_a, U_b, U_c, U_d] = -\frac{2}{n+2} (\delta_{ab}\delta_{cd} + \delta_{ac}\delta_{bd} + \delta_{ad}\delta_{bc}).$$

When pushed through the first weight matrix,  $\vec{z}^{(1)} = W_1^T \vec{U}$ , multilinearity gives the exact fourth cumulant of the first pre-activation in terms of the first-layer Gram covariance  $\Sigma_1 \equiv \text{Cov}(\vec{z}^{(1)}) = W_1^T W_1$ :

$$\kappa_4(z_i^{(1)}, z_j^{(1)}, z_k^{(1)}, z_m^{(1)}) = -\frac{2}{n+2} (\Sigma_{1,ij}\Sigma_{1,km} + \Sigma_{1,ik}\Sigma_{1,jm} + \Sigma_{1,im}\Sigma_{1,jk}) .$$

In particular, every neuron of the first pre-activation starts with a negative fourth cumulant proportional to the square of its column norm,

$$q_{4,i} = -\frac{6}{n+2} \Sigma_{1,ii}^2 = -\frac{6}{n+2} \|w_{1,i}\|_2^4 \quad \left( \text{or } -\frac{6}{n+2} \text{ in the unit-prevariance gauge } \tilde{\Sigma}_{1,ii} = 1 \right).$$

Evaluating this first-layer fourth cumulant with the exact per-neuron Gram covariance  $\Sigma_1 = W_1^T W_1$  yields an 18.1% final-layer MSE reduction over the baseline  $K = 3$  algorithm on IDs 32--36 ( $3.724 \times 10^{-8} \rightarrow 3.056 \times 10^{-8}$ ), compared to 13.4% ( $3.232 \times 10^{-8}$ ) when replacing  $\Sigma_{1,ii}$  with an isotropic scalar approximation at layer 1 (a 5.5% relative improvement from exact first-layer variance weighting alone). Because  $\Sigma_1 = W_1^T W_1$  is already computed for the first-layer covariance, this improvement is virtually free in FLOPs.

### B.2. Neuron-by-Neuron Scale Freedom

From the previous section, we discussed positive homogeneity of the ReLU function. Notice that ReLU acts elementwise on each neuron  $j \in \{1, \dots, n\}$  at each layer  $l$ . This means we can in principle take different positive scale factors  $s_{l,j} > 0$  for each individual neuron  $j$  at every hidden layer  $l$  without changing the network's output! How?

Consider the transition from layer  $l-1$  through hidden layer  $l$  into layer  $l+1$ , written neuron by neuron:

$$z_j^{(l)} = (W_l^T \vec{h}^{(l-1)})_j, \quad h_j^{(l)} = \text{ReLU}(z_j^{(l)}), \quad z_k^{(l+1)} = \sum_{j=1}^n h_j^{(l)} (W_{l+1})_{j,k} . \quad (\text{B3})$$

Suppose we multiply column  $j$  of  $W_l$  by a positive factor  $s_{l,j} > 0$  and, at the same time, divide row  $j$  of the next weight matrix  $W_{l+1}$  by the same  $s_{l,j}$ . By positive homogeneity of the ReLU, the hidden neuron is simply rescaled,

$$\tilde{z}_j^{(l)} = s_{l,j} z_j^{(l)}, \quad \tilde{h}_j^{(l)} = \text{ReLU}(\tilde{z}_j^{(l)}) = s_{l,j} \text{ReLU}(z_j^{(l)}) = s_{l,j} h_j^{(l)}, \quad (\text{B4})$$

and when  $\tilde{h}^{(l)}$  enters the next layer with row-rescaled weights  $\widetilde{W}_{l+1,j,k} = (W_{l+1})_{j,k}/s_{l,j}$ , the two factors cancel neuron by neuron:

$$\sum_{j=1}^n \tilde{h}_j^{(l)} \widetilde{W}_{l+1,j,k} = \sum_{j=1}^n (s_{l,j} h_j^{(l)}) \left( \frac{1}{s_{l,j}} (W_{l+1})_{j,k} \right) = \sum_{j=1}^n h_j^{(l)} (W_{l+1})_{j,k} = z_k^{(l+1)} . \quad (\text{B5})$$

Choosing positive per-neuron scales  $s_{l,j}$  is therefore an exact gauge transformation of the network: inside layer  $l$ , all moments and cumulants live in the gauged coordinates  $\tilde{z}_j^{(l)} = s_{l,j} z_j^{(l)}$ , and the physical mean at any layer  $l$  in the original coordinates is recovered simply by dividing the gauged post-ReLU mean by  $s_{l,j}$ :

$$\mathbb{E}[h_j^{(l)}] = \frac{\mathbb{E}[\tilde{h}_j^{(l)}]}{s_{l,j}} . \quad (\text{B6})$$

The second simplification we make is fixing this gauge to standardize the pre-activation  $\vec{z}^{(l)}$  entries to have unit variance. In a random network, different columns of  $W_l$  point in different directions relative to  $\text{Cov}[\vec{h}^{(l-1)}]$ , so without standardization some neurons would have raw variance  $\sigma_{l,j}^2 = 0.4$  while others have  $\sigma_{l,j}^2 = 2.5$ .

To do this recursively at every hidden layer  $l = 1, \dots, L-1$ , we first undo the previous layer's scale on the rows of  $W_l$  by setting  $\widehat{W}_l = \text{diag}(\vec{s}_{l-1})^{-1} W_l$  (with  $\vec{s}_0 = \vec{1}$ ), and compute the raw pre-activation covariance and standard deviation:

$$C_l^{\text{raw}} = \widehat{W}_l^T \text{Cov}[\vec{h}^{(l-1)}] \widehat{W}_l, \quad \sigma_{l,j} = \sqrt{(C_l^{\text{raw}})_{jj}} . \quad (\text{B7})$$

We then choose the scale  $s_{l,j} = 1/\sigma_{l,j}$ , which amounts to dividing column  $j$  of  $\widehat{W}_l$  by  $\sigma_{l,j}$  (and multiplying row  $j$  of  $W_{l+1}$  by  $\sigma_{l,j}$  at the next step):

$$\widetilde{W}_l = \widehat{W}_l \text{diag}(\vec{s}_l), \quad \vec{m}_l = \vec{m}_l^{\text{raw}} \odot \vec{s}_l, \quad C_l = \text{diag}(\vec{s}_l) C_l^{\text{raw}} \text{diag}(\vec{s}_l) , \quad (\text{B8})$$

where  $\vec{m}_l^{\text{raw}} = \widehat{W}_l^T \mathbb{E} \left[ \tilde{h}^{(l-1)} \right]$ . The gauged pre-activation  $\tilde{z}^{(l)} = \widehat{W}_l^T \tilde{h}^{(l-1)}$  then automatically has standardized mean  $m_{l,j} = m_{l,j}^{\text{raw}} / \sigma_{l,j}$ , unit variance on the diagonal ( $C_{l,jj} = 1$ ), and correlation coefficients off the diagonal ( $C_{l,ij} = C_{l,ij}^{\text{raw}} / (\sigma_{l,i} \sigma_{l,j}) \in [-1, 1]$ ). The entire ReLU closure of layer  $l$  is run on  $\tilde{z}^{(l)}$ . At the terminal layer  $l = L$  (which still applies ReLU), we multiply row  $j$  of  $W_L$  by  $\sigma_{L-1,j} = 1/s_{L-1,j}$  to undo the last hidden gauge, but we do not rescale the columns of  $W_L$ , so the final post-ReLU output mean  $\vec{\mu}_L$  comes out directly in physical units without needing a terminal covariance matrix.

Note that if we propagated the full, uncompressed cumulant tensors  $\kappa_3 \in \mathbb{R}^{n^3}$  and  $\kappa_4 \in \mathbb{R}^{n^4}$  in infinite precision, then the  $\sigma_{l,j}$  and  $1/\sigma_{l,j}$  factors would cancel out completely! However, in our practical algorithm, this unit-variance gauge yields two critical improvements:

1. A statistical reason (equalizing unweighted traces and SVDs): In our practical algorithm, we compress  $\kappa_3$  and  $\kappa_4$  in ways that are *not* invariant to neuron rescaling:

- *Fourth-cumulant harmonic trace core:* To compress the  $n^4$  post-ReLU fourth cumulant into an  $n \times n$  matrix core  $A$ , we take the partial trace over the third and fourth indices,  $t_{ij} = \sum_{k=1}^n \kappa_4(h_i^{(l)}, h_j^{(l)}, h_k^{(l)}, h_k^{(l)})$ . Since  $h_k^{(l)} = \sigma_{l,k} \tilde{h}_k^{(l)}$ , without the gauge this unweighted sum reads

$$t_{ij}^{\text{raw}} = \sigma_{l,i} \sigma_{l,j} \sum_{k=1}^n \sigma_{l,k}^2 \kappa_4(\tilde{h}_i^{(l)}, \tilde{h}_j^{(l)}, \tilde{h}_k^{(l)}, \tilde{h}_k^{(l)}) , \quad (\text{B9})$$

weighting every neuron  $k$  by  $\sigma_{l,k}^2$  (so a neuron with  $\sigma_{l,k} = 2$  dominates one with  $\sigma_{l,k} = 0.5$  by a factor of  $16\times$ ). Worse, the isotropic reconstruction ansatz  $\kappa_4 \approx \text{Sym}_6(A \otimes I_n)$  treats the traced-out indices isotropically with unit diagonal  $\text{diag}(C_l) = I_n$ , which requires all neurons to share unit variance.

- *Third-cumulant covariance star:* Similarly, fitting the vector  $\vec{v}$  in  $\kappa_3^{\text{star}}(z)_{ijk} = v_i C_{jk} + v_j C_{ik} + v_k C_{ij}$  requires taking the partial trace  $\text{rt}_i = \sum_{k=1}^n \kappa_3(z_k^{(l)}, z_k^{(l)}, z_i^{(l)})$  and solving  $(2C_l + \text{tr}(C_l)I_n)\vec{v} = \text{rt}$ . Standardizing  $\sigma_{l,k} = 1$  prevents high-variance directions from dominating this trace.
- *Low-rank SVD compression in the cubic history  $\mathcal{H}$ :* When truncating the factored cubic bank legs via SVD, singular vectors minimize the Frobenius norm  $\sum_{i,j} M_{ij}^2$ . In the unit-variance gauge, all entries are in dimensionless correlation units, so the SVD preserves directions with the strongest statistical skewness rather than merely the largest raw scale  $\sigma_{l,i}$ .

To quantify this statistical effect directly on IDs 32--36, we ran two complementary ablations of the unit-prevariance gauge: (a) on the unperturbed benchmark networks (whose i.i.d. He-initialized weights  $W_{l,ij} \sim \mathcal{N}(0, 2/n)$  already keep raw pre-activation variances clustered near  $O(1)$  with  $\sim 4\%$  relative spread, and where our unary/pair formulas carry explicit  $\sigma_{l,i}$  denominators), toggling the online hidden-layer rescaling  $s_{l,i} = 1/\sigma_{l,i}$  vs.  $s_{l,i} = 1$  across layers  $l = 1, \dots, 15$  changes final-layer MSE by 0.7% ( $1.152 \times 10^{-8}$  vs.  $1.144 \times 10^{-8}$ ), whereas at the first layer accounting for exact per-neuron Gram variances  $\Sigma_{1,ii} = \|w_{1,i}\|_2^2$  in the spherical fourth-cumulant closure reduces the  $K = 3$  final-layer MSE by 5.5% relative ( $3.232 \times 10^{-8} \rightarrow 3.056 \times 10^{-8}$ ); (b) when we apply an exact ReLU-preserving hidden-neuron rescaling  $W_l \mapsto D_{l-1}^{-1} W_l D_l$  with log-uniform scales  $d_{l,j} \in [0.5, 2.0]$  across hidden layers  $l = 1, \dots, 15$  (which leaves the true network output  $y(\vec{x})$  mathematically identical), disabling the online gauge causes the unweighted partial traces and SVDs to be dominated by high-scale neurons, degrading final-layer MSE by over  $5,000\times$  ( $1.144 \times 10^{-8} \rightarrow 6.105 \times 10^{-5}$ ), whereas the gauged estimator remains invariant up to `float32` rounding ( $1.152 \times 10^{-8} \rightarrow 1.154 \times 10^{-8}$ ).

2. A numerical and computational reason (simplifying Hermite/Mehler formulas and conditioning `float32`): For a Gaussian pre-activation  $Z_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ , the  $m$ -th derivative of  $\mathbb{E}[\text{ReLU}(Z_i)]$  carries a denominator  $\sigma_i^{m-1}$ :

$$\frac{\partial^m}{\partial \mu_i^m} \mathbb{E}[\text{ReLU}(Z_i)] = \frac{(-1)^m}{\sigma_i^{m-1}} \text{He}_{m-2}\left(\frac{\mu_i}{\sigma_i}\right) \phi\left(\frac{\mu_i}{\sigma_i}\right) \quad (m \geq 2) , \quad (\text{B10})$$

and bivariate Mehler terms require dividing by powers of  $\sigma_i$  and  $\sigma_j$  to form correlations  $\rho_{ij} = C_{ij}/(\sigma_i \sigma_j)$ . Enforcing  $\sigma_{l,i} = 1$  makes all  $\sigma_{l,i}^{m-1}$  denominators equal to 1, turns  $C_l$  directly into the correlation matrix, and

eliminates  $O(n^2)$  matrix divisions. Moreover, because the hidden-layer loop runs in `float32` (which costs half the FLOPs of `float64` under the challenge accounting), this exact coordinate reparameterization (distinct from an architectural LayerNorm) keeps pre-activation variances standardized to 1 across depth and avoids scale drift in  $\kappa_3 \propto \sigma^3$  and  $\kappa_4 \propto \sigma^4$ , while costing less than 0.02 GFLOPs of elementwise  $O(n^2)$  vector scaling.

### B.3. Standard linear and non-linear propagation

Based on the representation above, we propagate a non-Gaussian state at every layer  $l$ .<sup>6</sup>

To be under the competition computational budget, we propagate an evolving state  $\mathcal{S}_l$  of quantities at layer  $l$  given by:

1. mean  $\hat{\mu}_l$  (and gauged pre-activation mean  $\hat{m}_l$ )
2. gauged pre-activation covariance  $\hat{C}_l \equiv \widehat{\text{Cov}[\tilde{z}^{(l)}]}$ , with a unit diagonal (as explained in the previous section), alongside the post-activation covariance  $\widehat{\text{Cov}[\tilde{h}^{(l)}]}$
3. a third-cumulant  $\hat{\kappa}_3$  obtained from a combination of objects and a covariance star  $v$ . Following the LLMs terminology we call this a factored source bank  $\mathcal{H}_l$  with young legs and a shared elder basis  $G_{\text{elder}}$
4. a fourth-cumulant  $\hat{\kappa}_4$  given by a harmonic trace core, a bank trace contraction, tree-level native cumulants, and a directional response

Note, we use here the hat sign to describe that all of these are estimates for the true quantities. In what follows, we will drop the hats, for ease of notation. In summary, the carried state variables are listed in Table 1.

| State         | Meaning                                                                                               |
|---------------|-------------------------------------------------------------------------------------------------------|
| $m, C$        | Estimated gauged pre-activation mean and correlation matrix ( $\text{diag}(C) = I_n$ )                |
| $\mathcal{H}$ | Factored cubic history, including its repairs and shared elder basis $G_{\text{elder}}$               |
| $v$           | Vector representing the cubic covariance-star component                                               |
| $A$           | Matrix representing the fourth-order trace core ( $A = T + \zeta I_n$ )                               |
| $\mathcal{Q}$ | Additional quartic-response history: tree cumulants, bank trace contraction, and directional response |
| $s$           | Per-neuron coordinate scales ( $s_{l,j} = 1/\sigma_{l,j}$ ) used for gauge normalization              |

**Table 1.** State variables carried across layers.

In the next subsections we describe in a little more detail the main ideas behind the propagation of the third and fourth cumulants. We essentially care about two things: the cost in terms of FLOPs and memory, as well as total number of operations required, and preserve enough information to get the final mean with enough accuracy. Of particular care, given that in our representation each cumulant depends on the others, and for a large enough width we do not need a deep “tower” of cumulants, we want to make sure that common errors are minimized as much as possible.

#### B.3.1. Third cumulant $\kappa_3$ representation and propagation

Both  $\kappa_3$  and  $\kappa_4$  are objects that live in  $\mathbb{R}^m$ , with  $m = n^3, n^4$ . Not only does materializing them cost memory of order  $\mathcal{O}(m)$ , but updating them naively across a linear layer costs  $3 \times 2n^4 \approx 6.6$  TFLOPs per layer for  $\kappa_3$  alone ( $3 \times$  the entire 16-layer competition budget on a single layer in `float64`, or  $1.5 \times$  in `float32`), even though tensor factorization can bring the per-step contractions down to  $\mathcal{O}(n^3)$  and  $\mathcal{O}(n^2)$ .

Crucially, to update the mean, covariance, and fourth cumulant at layer  $l$ , our estimator never needs the full  $n \times n \times n$  cube  $\kappa_{3,l,ijk}$  for arbitrary distinct triples  $(i, j, k)$ . Instead, for  $\kappa_3$  our estimator needs only four contractions:

<sup>6</sup> In principle, one could have also decided to propagate completely different objects from cumulants, and materialize a mean only at the end, or taken different strategies. But for the challenge goals, given we were doing an LLMs based research, it was easier, assuming our resources and strategy, to just build upon the ARC’s paper.

- its unary diagonal  $z_{3,i} = \kappa_{3,l,iii} \in \mathbb{R}^n$ , which directly shifts the post-ReLU mean and unary moments via the Edgeworth expansion;
- its partial trace  $\text{rt}_j = \sum_{i=1}^n \kappa_{3,l,ijj} \in \mathbb{R}^n$  (the column sum of the repeated-index slice  $b_{ij}$ ), used to fit the covariance-star vector  $\vec{v}_l$  in  $\kappa_3^{\text{star}}(z)_{ijk} = v_{l,i}C_{l,jk} + v_{l,j}C_{l,ik} + v_{l,k}C_{l,ij}$  by solving  $(2C_l + \text{tr}(C_l)I_n)\vec{v}_l = \text{rt}$  via a knob-free Chebyshev polynomial recurrence;
- an isosceles-configuration, or repeated-index slice  $b_{ij} = \kappa_{3,l,ijj} \in \mathbb{R}^{n \times n}$  (two indices at neuron  $i$  and one at neuron  $j$ ), which feeds into the post-ReLU covariance and  $(2, 1)$  pair powers via the Mehler power series;
- a directional contraction  $K_{ij}(u) = \sum_{k=1}^n \kappa_{3,l,ijk}u_k$  along the Gaussian variance-response direction  $u_k = \frac{\partial}{\partial \mu_k} \text{Var}_G[\text{ReLU}(Z_k)] = 2h_k(1 - d_k)$ , which captures non-isotropic skewness-to-kurtosis coupling in the fourth-order trace core.

Why, then, can we not simply store these four contractions at layer  $l - 1$  and push them to layer  $l$ ? Because the linear layer  $\tilde{z}^{(l)} = \widetilde{W}_l^T \tilde{h}^{(l-1)}$  mixes distinct neurons  $(a, b, c)$  of layer  $l - 1$  into the diagonal and repeated-index slice of layer  $l$ :

$$\kappa_{3,l,iii} = \sum_{a,b,c=1}^n \widetilde{W}_{l,ai} \widetilde{W}_{l,bi} \widetilde{W}_{l,ci} \kappa_{3,l,abc}(\tilde{h}^{(l-1)})_{abc} . \quad (\text{B11})$$

To propagate this three-way structure in  $\mathcal{O}(n^2)$  memory and  $\mathcal{O}(n^3)$  FLOPs, we exploit how skewness is born at each ReLU layer  $b$  (Wu et al. 2026). After  $\tilde{h}^{(b)} = \text{ReLU}(\tilde{z}^{(b)})$ , newly created skewness consists of three pieces: (i) a 2-edge covariance star where a central ‘‘hub’’ neuron  $m$  bends with curvature  $e_{b,m} = \phi(m_{b,m})$  and connects to two other neurons via  $\tilde{C}_b \vec{d}_b$ ; (ii) a  $(2, 1)$  pair-collision term where two indices coincide; and (iii) a single-neuron unary skewness diagonal  $\beta_m^{(b)} \delta_{mi} \delta_{mj} \delta_{mk}$ . By completing the square between the 2-edge star and the  $(2, 1)$  collision term into a single branch matrix  $H^{(b)} = \sqrt{\tilde{e}_b} \tilde{C}_b \text{diag}(\vec{d}_b) + F^{(b)}$ , the entire third cumulant born at layer  $b$  takes the factored form

$$\Delta \kappa_3^{(b)}(\tilde{h}^{(b)})_{ijk} = \sum_{m=1}^{n_r} \left[ 3 \text{Sym}_{ijk}(P_{mi}^{(b)} H_{mj}^{(b)} H_{mk}^{(b)}) + 3 \text{Sym}_{ijk}(P_{mi}^{(b)} P_{mj}^{(b)} R_{mk}^{(b)}) + \beta_m^{(b)} P_{mi}^{(b)} P_{mj}^{(b)} P_{mk}^{(b)} \right] - 3 \text{Sym}_{ijk}(K_i^{(b)} V_j^{(b)} V_k^{(b)}) , \quad (\text{B12})$$

where  $3 \text{Sym}_{ijk}(u_i v_j v_k) = u_i v_j v_k + v_i u_j v_k + v_i v_j u_k$  sums the three cyclic placements,  $P^{(b)} = I_{n_r \times n}$  is the hub-identity leg,  $H^{(b)} \in \mathbb{R}^{n_r \times n}$  is the completed-square branch leg,  $K^{(b)}(V^{(b)})^2$  is a rank-1 negative-square correction removing the spurious  $(F^{(b)})^2$  cross-term, and  $R^{(b)}$  is a repeated-index repair matrix chosen so that Eq. (B12) reproduces the two-neuron  $(i, i, k)$  post-ReLU slice within the birth closure. To avoid paying a third dense matrix multiplication per layer to transport  $R^{(b)}$ , we immediately compress  $R^{(b)}$  at birth into a rank-3 factorization  $\text{LEFT}_b \cdot \text{RIGHT}^{(b)}$  ( $n_r \times 3$  times  $3 \times n$ , combining a rank-2 truncated SVD with 1 analytic conservation column that preserves the exact slice row sums, plus a diagonal shift folded into  $\tilde{\beta}^{(b)}$ ), so only the  $3 \times n$  factor  $\text{RIGHT}^{(b)}$  is multiplied by the weight matrices at later layers (costing  $\sim 0.3\%$  of a dense matrix product).

When the network advances from layer  $l - 1$  to layer  $l > b$ , multilinearity of cumulants implies that each 2D leg matrix is transported forward along the effective ReLU transport gate  $D_{l-1} = \text{diag}(\vec{d}_{l-1})$  and gauged weight matrix  $\widetilde{W}_l$ :

$$P^{(b,l)} = P^{(b,l-1)} D_{l-1} \widetilde{W}_l, \quad H^{(b,l)} = H^{(b,l-1)} D_{l-1} \widetilde{W}_l, \quad \text{RIGHT}^{(b,l)} = \text{RIGHT}^{(b,l-1)} D_{l-1} \widetilde{W}_l . \quad (\text{B13})$$

At layer  $l$ , reading out the unary diagonal  $z_{3,i}$  takes only  $\mathcal{O}(n^2)$  elementwise operations on the columns of  $P^{(b,l)}$  and  $H^{(b,l)}$ , whereas assembling the repeated-index slice  $b_{ij} = \kappa_{3,l,ijj}$  requires forming quadratic elementwise coefficient matrices  $C_P^{(b,l)}, C_H^{(b,l)}$  and computing two matrix products  $(C_P^{(b,l)})^T P^{(b,l)} + (C_H^{(b,l)})^T H^{(b,l)}$  per birth. Rather than forming the directional contraction  $K_{ij}(u) = \sum_k \kappa_{3,l,ijk}u_k$  as a separate  $\mathcal{O}(n^3)$  matrix product, we contract  $\vec{u}$  with the legs in  $\mathcal{O}(n^2)$  time ( $P^{(b,l)} \vec{u}$  and  $H^{(b,l)} \vec{u}$ ) and fold the result directly into  $C_P^{(b,l)}$  and  $C_H^{(b,l)}$  at zero extra  $\mathcal{O}(n^3)$  cost.

### B.3.2. Fourth cumulant $\kappa_4$ representation and propagation

Just as for  $\kappa_3$ , our closure only consumes  $\kappa_{4,l}$  through its unary diagonal  $q_{4,i} = \kappa_{4,l,iiii} \in \mathbb{R}^n$  (in the post-ReLU mean and moments) and its bivariate  $(2, 2)$  and  $(3, 1)$  slices  $q_{22,ij} = \kappa_{4,l,iiij}$  and  $q_{31,ij} = \kappa_{4,l,iiij}$  (in the 21-row Mehler

expansion). Rather than compressing  $\kappa_{4,l}$  to a single scalar  $\alpha_l$  as in the baseline  $K = 3$  recurrence (Wu et al. 2026) — which discards all neuron-specific tail heaviness — we follow the augmented odd- $K$  hierarchy (Wu et al. 2026) and decompose the pre-activation fourth cumulant into a broadly distributed harmonic trace core plus three targeted  $\mathcal{O}(n^2)$  directional residual mechanisms (McCullagh 1987; Roberts et al. 2022):

- *Harmonic trace core*  $A_l = T_l + \zeta_l I_n$ : represents the isotropic trace sector  $\kappa_{4,l}^{\text{core}} = \text{Sym}_6(A_l \otimes I_n)$  in the unit-prevariance gauge. Taking the partial trace over the last two indices on the unit-variance sphere gives  $\sum_{k=1}^n \kappa_{4,l,ijkk} = (n+4)T_{l,ij} + \text{isotropic terms}$ , which is why  $T_l$  is obtained by dividing the symmetrized mixed fourth trace by  $n+4$  while  $\zeta_l = -\text{tr}(\text{trace})/[2(n+2)(n+4)]$ . The core is propagated across layers via  $G_l = \widetilde{W}_l^T A_{l-1} \widetilde{W}_l$  with diagonal  $g_{a,i} = (G_l)_{ii}$  and updated from the 21-row Mehler pair-power table alongside the  $(2,2)$  row-sum identity so that  $\kappa_{4,l,iiii}^{\text{core}} = 6g_{a,i}g_{v,i}$ , where  $g_{v,i} = \|\widetilde{w}_{l,i}\|_2^2$ .
- *Tree-cumulant quartic corrections and  $(2,1)$  fold*: reads the traceless tree-level fourth cumulants directly from the cubic bank legs in  $\mathcal{O}(n^2)$  elementwise time — single-neuron quartic births  $\sum_a \widetilde{W}_{l,ai}^4 (\kappa_4(h_a) - 6A_{l-1,aa})$ , the 3-edge star, the  $u$ -term, and  $(3,1)$  pair reads — while subtracting each family’s analytic projection onto the harmonic trace core  $A_l$  so that no fourth-order contribution is double-counted against the core.
- *Multi-generation bank trace contraction*: earlier skewness births generate persistent fourth-order contributions as they pass through subsequent ReLU layers. For consumer layers  $l \geq 4$  up to the terminal readout, we contract a sliding horizon of the last three cubic birth generations via a backward Horner recurrence (sharing weight-matrix prefixes across generations, while omitting interactions older than three layers and elder-switched legs) to recover multi-layer  $\kappa_{4,l,iiii}$  diagonal terms and fold their  $(3,1)$  gradient panels into the repeated-index cubic slice.
- *Anisotropic directional response*: evaluates the directional contraction of the cubic bank  $\mathcal{H}_l$  along the Gaussian variance-sensitivity vector  $u_k = \frac{\partial}{\partial \mu_k} \text{Var}_G[\text{ReLU}(Z_k)] = 2h_k(1 - d_k)$  and subtracts the covariance-star approximation’s existing radial contribution, capturing non-isotropic skewness-to-kurtosis coupling without double-counting.

#### B.4. ReLU closure

At each hidden layer  $l$ , given the gauged pre-activation mean  $m_{l,i}$  (with unit variance  $\sigma_{l,i} = 1$ ), diagonal skewness  $z_{3,i} = \kappa_{3,l,iii}$ , and diagonal kurtosis  $q_{4,i} = \kappa_{4,l,iiii}$ , single-neuron post-ReLU moments  $m_k = \mathbb{E}[(h_i^{(l)})^k]$  ( $k = 1, 2, 3, 4$ ) and effective Hermite derivatives are evaluated via a fourth-order Edgeworth expansion around the exact Gaussian ReLU integrals (Edgeworth 1905; Cramér 1946; Cho & Saul 2009):

$$m_1 = h + \frac{f}{6} z_3 + \frac{g}{24} q_4, \quad (\text{B14})$$

where  $h = \phi(m) + m\Phi(m)$ ,  $f = -m\phi(m)$ , and  $g = (m^2 - 1)\phi(m)$  (with the terminal layer  $l = L$  evaluated in physical units and including the additional sixth-derivative  $\kappa_3^2$  Edgeworth term  $\frac{z_{3,L}^2}{72} \text{He}_4(m_L)\phi(m_L)/\sigma_L^5$ ). Bivariate post-ReLU moments — the next-layer post-activation covariance  $\text{Cov}[\widetilde{h}^{(l)}]$ , the  $(2,1)$  cubic slice, the  $(3,1)$  quartic slice, and the 21-row trace-core update matrix  $Z_l$  — are evaluated simultaneously from a third-order Mehler expansion (Mehler 1866; Kibble 1945; Price 1958) in powers of the off-diagonal correlation  $x_{ij} = C_{l,ij}$  ( $i \neq j$ ), combined with the  $(2,2)$  pair cumulant  $q_{22,ij} = g_{a,i}g_{v,j} + g_{v,i}g_{a,j} + 4G_{l,ij} \text{Gram}_{l,ij}$  (where  $\text{Gram}_l = \widetilde{W}_l^T \widetilde{W}_l$  at the first layer and  $\text{diag}(\widetilde{g}_v)$  at later layers) and the repeated-index cubic slice  $b_{ij} = \kappa_{3,l,ij}$ .

#### B.5. Engineering

We now go to the part that we mainly think is engineering, and where most of the LLMs based solution searches, not just ours, spent most of their time. We believe from this point of view there is room for improvement, not just on the engineering side, but also on building some particular harness that prevents LLMs from playing the “simple” engineering point of view.

Once the analytical cumulant recurrence above is fixed, reducing the computational cost from the dense baseline  $K = 3$  budget (69.5% of  $2^{41}$  FLOPs, or  $\sim 1.53$  TFLOPs) down to 15.4–16.0% (338–353 GFLOPs across all 100 official evaluation networks: 15.36% in #337631 and 16.04% in #337479; 341–348 GFLOPs on IDs 600–629), while respecting the grader’s strict 120s round-trip and 0.40s residual wall-time caps, relies on four concrete engineering mechanisms:

1. *Active-leg gate thinning and minimum-norm rank-2 refill of saturated slice rows:* Because ReLU zeroes out negative inputs, neurons in the deep negative tail ( $m_{l,i} \ll 0$ ) have negligible activation probability  $\Phi(m_{l,i})$  and curvature  $\phi(m_{l,i})$ . At each layer we build shared thresholded index plans (padded up to multiples of `PLAN_QUANTUM` = 64 so cropped matrices align with recursive block boundaries): we retain cubic birth hubs with  $\Phi(m_{b,m}) > 0.015$ , contract leg transport over input neurons with corrected gate  $d_{l-1,a} > 0.03$ , and evaluate the repeated-index cubic slice  $b_{ij} = \kappa_{3,l,ij}$  only on output columns with  $\Phi(m_{l,j}) > 0.01$ .

More subtly, on the repeated-index row side  $i$  of  $b_{ij}$ , strongly positive *saturated* neurons ( $1 - \Phi(m_{l,i}) \leq 0.015$ , i.e.  $m_{l,i} \gtrsim 2.17$ ) have almost no ReLU curvature ( $\phi(m_{l,i}) \approx 0$ ) and act nearly linearly, yet they cannot simply be zeroed out: the (2, 2) pair-power row sums (urow, which set the diagonal of the next layer’s fourth-cumulant trace core  $T_l$ ) weight  $b_{ij}$  and  $b_{ji}$  by  $g_i'' = 2(d_i - m_i e_i) \approx 2$  and  $g_j' = 2m_j(1 - d_j)$ . Zeroing saturated slice rows creates a coherent bias that compounds across layers and degrades final-layer MSE by  $2.2\times$ . Instead, we omit the dense matrix product on the saturated row set  $S = \{i : 1 - \Phi(m_{l,i}) \leq 0.015\}$  and evaluate the two required linear functionals directly from the factored bank in  $\mathcal{O}(\text{sources} \times \text{hubs} \times n)$  vector operations ( $\sim 0.1$  GF/layer):

$$\vec{a}^T = \sum_{i \in S} w_{r,i} b_{i,\cdot}, \quad c_i = \sum_j b_{ij} w_{c,j} \quad (i \in S), \quad (\text{B15})$$

where  $w_{r,i} = d_i - h_i \phi_i$  on  $S$  and  $w_{c,j} = h_j(1 - d_j)$  on the active columns. We then refill the omitted rows  $S$  with the unique minimum-Frobenius-norm rank-2 matrix  $F \in \mathbb{R}^{n \times n}$  (supported on  $S \times \text{OUT}$ ) satisfying  $\vec{w}_r^T F = \vec{a}^T$  and  $F \vec{w}_c = \vec{c}$ :

$$F = \frac{\vec{w}_r \vec{a}^T}{\|\vec{w}_r\|_2^2} + \frac{\left(\vec{c} - \vec{w}_r \frac{\vec{a} \cdot \vec{w}_c}{\|\vec{w}_r\|_2^2}\right) \vec{w}_c^T}{\|\vec{w}_c\|_2^2}. \quad (\text{B16})$$

Because  $\vec{w}_r \cdot \vec{c} = \vec{a} \cdot \vec{w}_c = \sum_{i \in S, j} w_{r,i} b_{ij} w_{c,j}$ , both constraints are exact and consistent. On a 3-network slice (IDs 32--34 of the same `full` split), this rank-2 refill and its companion row-plan refinements save 22.4 GFLOPs per network ( $369.0 \rightarrow 346.6$  GFLOPs,  $-6.1\%$ ) while changing final-layer MSE by only  $+0.85\%$  (a 5.3% improvement in adjusted score).

2. *Shared elder right-subspace ( $G_{\text{elder}}$ ) and deterministic Krylov small-rank SVDs:* Transporting two dense  $n_r \times n$  leg matrices ( $P^{(b,l)}, H^{(b,l)}$ ) for every historical birth layer  $b < l$  would make the per-layer bank transport grow linearly with depth. Once a cubic birth reaches age  $l - b = 6$ , its active leg directions have already been filtered through six layers of weight matrices and ReLU gates, so we project its  $P$  and  $H$  legs via least squares ( $P^{(b,l)} \approx Y_P^{(b)} G_{\text{elder}}^{(l)}$ ,  $H^{(b,l)} \approx Y_H^{(b)} G_{\text{elder}}^{(l)}$ ) onto a single shared orthonormal right-subspace  $G_{\text{elder}}^{(l)} \in \mathbb{R}^{d \times n}$ . The rank  $d$  is chosen by a  $2^{-8}$  singular-value tail-energy rule capped at  $d = 320$  (Submission #337479) or  $d = n/4 = 256$  (Submission #337631). Because  $G_{\text{elder}}^{(l)}$  is shared across all elder births, transporting the entire elder cohort across  $\widetilde{W}_l$  requires only a single  $d \times n$  matrix multiplication per layer, saving  $\sim 33$  GFLOPs per network with virtually zero MSE loss compared to maintaining separate per-source bases.

A related systems subtlety arises in small-rank  $k \leq 8$  SVD truncations (used for the rank-2 birth repair  $R^{(b)}$ , the rank-1 negative-square packet, and the rank- $n/256$  repeated-index trace-correction query): while FlopScope bills a rank- $k$  call `linalg.svd(M, k = k)` at the truncated price  $4mnk$ , the server backend executes a full economy SVD regardless of  $k$ , costing 0.2–0.4s of wall time per  $1024 \times 1024$  call. We therefore replace small-rank SVDs with a metered deterministic depth-2 block-Krylov Rayleigh–Ritz projection (`_svd_topk`) seeded by the  $k + 2$  largest-norm rows of  $M$ .

3. *Recursive Strassen–Winograd (2, 2, 2; 7) block fast matrix multiplication (`fmm.py`):* Because the challenge meters arithmetic FLOPs rather than hardware instructions, large matrix products in the bank and engine are evaluated via a recursive  $2 \times 2$  Strassen–Winograd (2, 2, 2; 7) schedule up to  $L_{\text{max}} = 5$  recursion levels (with  $16 \times 16$  leaf blocks), multiplying the 7 recursive sub-blocks at each level rather than 8 and reducing the asymptotic GEMM FLOP count by  $(7/8)^L$  (for example,  $(7/8)^4 \approx 0.586$ ). Crucially, when multiple left leg factors multiply the same right operand (such as the gauged weight matrix  $\widetilde{W}_l$  or the shared elder basis  $G_{\text{elder}}^{(l)}$ ), the recursive  $7^L$  linear-combination transform of the shared operand is computed once and cached by operand key (`a_key` / `b_key`), amortizing the Strassen addition overhead across the entire batch.

4. *Call-priced scheduling and residual wall-time engineering (0.40s cap)*: Under the official client-server evaluation harness, every logged FlopScope operation and every array slice view incurs  $\sim 6.9\text{--}10.7\mu\text{s}$  of client-side RPC/residual overhead, making the official 0.40s residual wall-time cap a hard ceiling of roughly 35,000–50,000 logged calls per network. To stay safely below this ceiling (with an internal safety target of 0.35s and official maximum residuals of 0.374–0.375s), we: (a) price each logged call at `CALL_PRICE` =  $5 \times 10^6$  FLOPs when selecting the recursion depth of every matrix product (so small products run as a single plain `matmul` rather than spawning hundreds of recursive leaf calls); (b) evaluate each Strassen recursion level along a 7-slot tensor axis using arithmetic-progression views and single-call `split` partitions (`SPLIT_FILL`); (c) group consecutive cubic sources that share the same padded row count into single batched 3D matrix products (`_equal_source_groups` and `_sum_families`); and (d) in Submission #337631, remove the final-step dense 7-to-4 `matmul` shortcut in `fmm.py` (used in #337479) to restore the compiled pointwise add/subtract combine, reducing final-combine arithmetic from 13 to 2 FLOPs per output element.

### C. EXTENSIONS

After the official Phase 2 submission deadline, we continued investigating two zero-fit mechanistic extensions of our cumulant-propagation framework on a 30-network canonical benchmark cohort (IDs 600--629 from the full split of [aicrowd/arc-whestbench-public-2026](#), revision v2-phase2, width  $n = 1024$ , depth  $L = 16$ ):

1. *Extended trace-contraction horizon and exact-zero  $H$ -leg transport*: Starting from our rank-2 slice-row refill architecture, we increased the Horner trace-contraction history horizon from 3 to 6 generations (5 at the terminal readout), lowered the late-layer core-read collar to (0.015, 0.05) with bilateral third-cumulant core contractions enabled, and eliminated redundant fast block matrix multiplications on provably zero rows of the completed-square  $H$  leg when the active  $H$ -row count  $n_v$  is strictly smaller than the active  $P$ -row count  $n_r$ . We also tested an exact-minus-radial off-diagonal cubic trace contraction feeding directly into the fourth-cumulant core update  $Z_l$ . Across the 30 networks (IDs 600--629), this extended trace-horizon estimator reduces the final-layer MSE from  $1.275 \times 10^{-8}$  (Submission #337631) and  $1.236 \times 10^{-8}$  (Submission #337479) down to  $1.141 \times 10^{-8}$  ( $3.26\times$  below Base  $K = 3$  at  $3.724 \times 10^{-8}$ ), while achieving a mean adjusted score of  $1.651 \times 10^{-9}$  on IDs 600--603 at 344.2 GFLOPs (15.7% of the  $2^{41}$  FLOP budget).
2. *Paired cubic history with deterministic block-Krylov windowing and multi-birth quartic replay*: To test how far purely analytical cumulant propagation can be pushed when a larger share of the  $2^{41}$  FLOP budget ( $\sim 29.3\%$ , 645.1 GFLOPs) is invested in higher-order history retention, this second extension replaces the single shared elder subspace with a deterministic block-Krylov projection window (seeded by 7 per-neuron response vectors to rank  $\lfloor \sqrt{n} \rfloor = 32$ ) for older cubic births while contracting the newest cubic births exactly via two-Gram products, and maintains a multi-layer quartic source window (Gaussian-pair, star, and third-cumulant families born at each layer and replayed through a Horner chain). Across all 30 networks (IDs 600--629), this Krylov and quartic-replay estimator breaks the  $10^{-8}$  barrier, reaching a pooled final-layer MSE of  $9.752 \times 10^{-9} \pm 0.224 \times 10^{-9}$  ( $\pm 1$  SEM;  $\pm 0.439 \times 10^{-9}$  at 95% CI,  $3.82\times$  lower than Base  $K = 3$ , and 21% lower than Submission #337479), demonstrating that substantial untapped accuracy remains purely within analytical, target-free cumulant propagation.

Figure 4 compares the layer-by-layer MSE (and  $\pm 1.96$  SEM 95% confidence bands across the 30 networks IDs 600--629) of the baseline  $K = 3$  algorithm, our two official Phase 2 submissions (#337631 and #337479), and these two post-submission extensions.

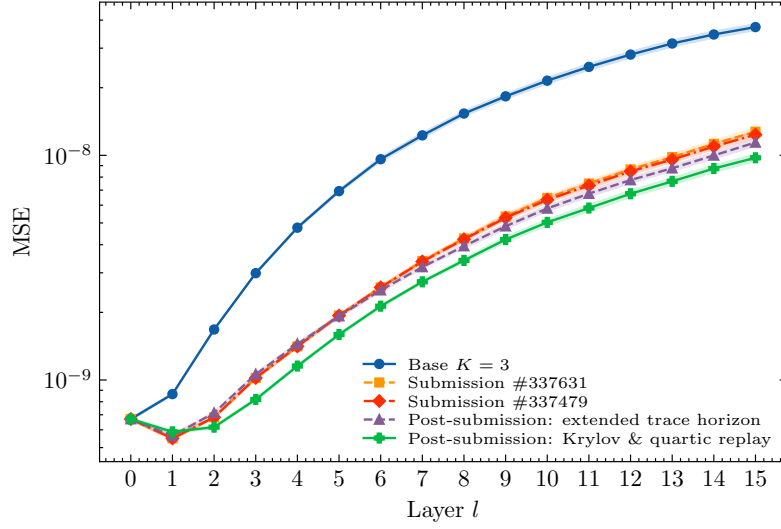


**Figure 4.** Mean squared error (logarithmic scale) as a function of layer depth  $l \in \{0, \dots, 15\}$  across 30 canonical width-1024, depth-16 networks (IDs 600--629 from the full split of [aicrowd/arc-whestbench-public-2026](#), revision v2-phase2; shaded bands show  $\pm 1.96$  SEM, i.e. 95% confidence intervals). Shown are the baseline  $K=3$  algorithm ( $3.724 \times 10^{-8}$  at  $l=15$ ), our two Phase 2 competition submissions #337631 ( $1.275 \times 10^{-8}$ ) and #337479 ( $1.236 \times 10^{-8}$ ), and our two post-submission zero-fit extensions: extended trace horizon ( $1.141 \times 10^{-8}$ ) and block-Krylov + quartic replay ( $9.752 \times 10^{-9}$ ).