

Higher-Order Cumulant Propagation with Learned In-Propagation Corrections

An account of our Phase 2 solution to the ARC White-Box Estimation Challenge (submission #337501)

Marius Binner

University of Bergen

October 2026

Abstract

We describe the estimator behind submission #337501 to Phase 2 of the ARC White-Box Estimation Challenge. It predicts the expected activation of every neuron of a random ReLU network of width 1024 and depth 16 without sampling. On the 100 graded public networks it scores 1.299×10^{-9} (final-layer MSE 1.087×10^{-8} at 0.119 of the FLOP budget), lower than any other team’s public score when submissions closed; plain sampling with the same FLOPs would have an MSE about 860 times higher. The estimator propagates cumulants layer by layer: the means, the full covariance, a factored third cumulant with low-rank memory of earlier layers, a scale-mixture model of the fourth cumulants, and an Edgeworth readout. Two ingredients go beyond ARC’s reference method. First, we model *births*, the skew that a ReLU creates when its input is already skewed, and fold the resulting (3, 1) slice of the fourth cumulant into an effective (2, 1) slice by an exact ReLU identity; an error map shows that this effective slice is the main local root of the error that remains. Second, three small learned heads correct the propagation inside the network. They are trained end to end, together with the analytic constants, through a differentiable twin of the whole estimator, and the graded steps that introduced them multiplied the score by about 0.82. We measure how much the submitted code depends on each component by switching it off, compare this with the gain each gave when it was introduced, and state which parts build on ARC’s reference method and on other participants’ public work.

1 Introduction

Mechanistic estimation asks for a quantity such as the expected output of a neural network to be derived from the weights rather than measured by running the network on samples. For wide random multilayer perceptrons, Wu et al. [23] showed that propagating the cumulants of the activations through the layers estimates the expected output more accurately than sampling at equal compute, with an error that shrinks quickly as higher cumulant orders are kept. The ARC White-Box Estimation Challenge [4] turns this into a contest. In Phase 2 an estimator receives the weights of a random ReLU network of width $n = 1024$ and depth $L = 16$ and must return the expected activation of every neuron. The final-layer mean squared error (MSE) is multiplied by the fraction of a 2^{41} -FLOP budget that the estimator used (counted by `flopscope`, an instrumented NumPy, with a floor of 0.1), so accuracy and compute trade one for one.

This paper describes the estimator behind our submission #337501. On the 100 graded public networks it scores 1.299×10^{-9} : a final-layer MSE of 1.087×10^{-8} at 0.119 of the budget. That is 312 times lower than the starter kit’s covariance propagation (4.05×10^{-7}) and lower than the best public score of any other team when submissions closed (1.334×10^{-9} , read from the public submissions after the close). Plain Monte Carlo with the same FLOPs, about 7,800 forward passes, would have an MSE of about 1×10^{-5} on our 64 test networks, some 860 times the estimator’s. The estimator is deterministic and uses no samples. It propagates the means, the full covariance, a factored representation of the third cumulants and a model of the fourth cumulants through all 16 layers, and reads the final means with an Edgeworth expansion. Three small learned heads correct this propagation inside the network.

Our contributions are:

- **Births and the effective slice** (Section 3.6). A ReLU whose input is already skewed creates new skew from it, a second-order effect that the first-order transport of third cumulants misses. We derive these terms, compute them within the budget by a backward recursion, and fold the (3, 1) slice of the fourth cumulant into an effective (2, 1) slice by an exact ReLU identity.
- **A map of where the remaining error lives** (Section 4). Given its true inputs the readout is accurate to under 1% of the estimator’s error. The error is mean error made in layers 6–15, it enters through the variance, and its local root at every layer is the effective slice: in our tests the true slice alone removes the covariance error created at a layer, and injected at layers 6–14 it multiplies the final MSE by 0.65.

- **Learned corrections inside the propagation** (Section 5). The heads read the propagated joint state at each layer and correct the post-activation means, the $(2, 1)$ slice and the cumulant and covariance diagonals. They are trained end to end through a differentiable twin of the whole estimator, together with the analytic constants. The graded score steps that introduced them multiplied to about 0.82. Other write-ups report that learning inside the propagation, or end to end, did not work for them [8, 13, 20, 21]; Section 5.3 discusses why it worked here.
- **A component-level account of the submitted code** (Section 7.2). Each component is tied to a switch in the code, and we measure what switching it off costs on the exact submitted file, next to the graded gain it gave when it was introduced. Section 6 briefly describes the engineering that fits third-order propagation into 12% of the budget; its fast multiplication schemes come from the literature.

What is ours. The third-cumulant propagation started as a port of ARC’s reference code [3, 23], which stores each third-cumulant source with its legs and models the fourth cumulants as an isotropic scale mixture; as ported it cost 149% of the budget. Other participants published related work during the competition: consumption forms and shared bases for the factored third cumulant [1, 5, 15], the spherical input [5, 9], scale-mixture closures and an exact $(3, 1)$ slice [5, 10], and birth-type terms [5, 15, 20]. We mark these overlaps where they occur. To our knowledge, the effective-slice fold, the learned in-propagation heads with their end-to-end training, the error map, and the kink-local selection with tails are not covered by other write-ups. Most of the code, and most of the specific ideas, came from AI agents working under the author’s direction; Section 8 and the statement on AI use describe how.

2 Problem and background

Task and score. A network has $L = 16$ layers of width $n = 1024$ with He-initialised weights $W_\ell \in \mathbb{R}^{n \times n}$, entries i.i.d. $\mathcal{N}(0, 2/n)$, no biases, and input $x \sim \mathcal{N}(0, I_n)$. Layer $\ell = 0, \dots, 15$ computes $z_\ell = W_\ell^\top y_{\ell-1}$ and $y_\ell = \text{ReLU}(z_\ell)$, with $y_{-1} = x$ (we number layers from 0, as the code does). The estimator returns $\mathbb{E}[y_\ell]$ for every neuron of every layer; only the last layer, y_{15} , is scored. For one network the score is $\text{MSE} \times \max(0.1, F/2^{41})$, where the MSE is taken over the 1024 final-layer neurons against labels from 10^9 samples and F is the number of FLOPs that the `flopscope` meter bills. Above the 10% floor, one percent fewer FLOPs is worth exactly as much as one percent lower MSE. `flopscope` bills a product of an $m \times k$ and a $k \times n$ matrix as $2mkn - mn$ FLOPs, element-wise operations per element, and decompositions such as the truncated SVD by rank-dependent formulas. The grader runs the estimator as a client that sends every array operation to a metering server. Per network it allows 120 s of wall time, 8 GB of memory and 0.4 s of *residual time*, the Python time the client spends outside metered operations. Every call to the server adds 17–27 μs of residual time, so this limit caps the number of array operations, and it shaped much of the engineering.

Competition setting. During the competition each submission was *graded*: run by the official grader on 100 public networks, whose average score is the public score. The organisers also released a dataset of networks with labels (splits *mini*, 100 networks, and *full*, 1,000 networks), which we used for training and validation; the graded networks are not among them. Each team then nominated two submissions for a private re-evaluation on fresh networks, which decides the ranking. The starter kit supplied simple example estimators, such as the covariance propagation below.

Cumulant propagation. Write $m_\ell = \mathbb{E}z_\ell$, $C_\ell = \text{Cov}(z_\ell)$ and $\kappa_{k,\ell}$ for the order- k joint cumulant tensor of z_ℓ . A linear layer maps cumulants exactly and multilinearly,

$$\kappa_k(z_\ell)_{i_1 \dots i_k} = \sum_{j_1, \dots, j_k} W_{j_1 i_1} \dots W_{j_k i_k} \kappa_k(y_{\ell-1})_{j_1 \dots j_k}, \quad (1)$$

so the means and covariances of z_ℓ are $W^\top \mathbb{E}y_{\ell-1}$ and $W^\top \text{Cov}(y_{\ell-1})W$. The ReLU is the hard part. At large width each pre-activation is a sum of many weakly dependent terms, its joint distribution is close to Gaussian, and the deviations are carried by small higher cumulants. The cumulants of y_ℓ are therefore expanded around the Gaussian with the same mean and covariance (an Edgeworth or Hermite expansion). For one neuron with mean m , standard deviation σ and $\alpha = m/\sigma$,

$$\mathbb{E} \text{ReLU}(z) = \mathbb{E}_G \text{ReLU} + \frac{\kappa_3}{6} \mathbb{E}_G \text{ReLU}''' + \frac{\kappa_4}{24} \mathbb{E}_G \text{ReLU}'''' + \frac{\kappa_5}{72} \mathbb{E}_G \text{ReLU}^{(6)} + \dots, \quad (2)$$

where $\mathbb{E}_G$ is the Gaussian expectation and the derivatives are distributional. All terms are closed forms in α : $\mathbb{E}_G \text{ReLU} = \sigma(\alpha \Phi(\alpha) + \varphi(\alpha))$, $\mathbb{E}_G \text{ReLU}' = \Phi(\alpha)$, $\mathbb{E}_G \text{ReLU}'' = \varphi(\alpha)/\sigma$, $\mathbb{E}_G \text{ReLU}''' = -\alpha \varphi(\alpha)/\sigma^2$, and so on. Pairs and triples of neurons use the multivariate version of the same expansion. Keeping cumulants up to order K gives an estimator whose error falls quickly with K at large width [23]. $K = 2$ (means and covariances) is the starter kit’s covariance propagation, which scores 4.05×10^{-7} at 2.4% of the budget.

The cost problem. The third-cumulant tensor of a layer has $n^3 \approx 10^9$ entries, and transporting it densely through one layer by Eq. (1) costs about $6n^4 \approx 6 \times 10^{12}$ FLOPs, three times the whole budget. ARC’s reference implementation [3] never forms it. Every ReLU creates new third cumulant (a *source*) with a simple low-rank structure, and later layers transport that source multilinearly, so each source is stored as a small core together with the matrices that map it to the current layer (its *legs*); only the slices that the next ReLU needs are formed. Ported to the metered NumPy, this reference $K = 3$ propagation cost 149% of the budget at width 1024 and depth 16. Most of the work in this paper is about getting more accuracy than that propagation at 12% of the budget.

Other Phase 2 work. Several public Phase 2 estimators build on the factored $K = 3$ chain of 504aldo [1]. Artyom_Mazur [5] analyses such a chain in depth, with a scale-mixture closure for the fourth cumulants and the argument that a corrector acting on each neuron separately can learn only a few percent of a chain’s error; Nurlanggs [15] measures where the per-source leg representation stops paying; kai_leal_gonzalez [10] computes the $(3, 1)$ slice of κ_4 exactly; il.miscusi [9] removes the global scale mode at every layer; hyojun_kwon [8] adds a learned terminal corrector; and abhimanyu_singh [2] measures how much slice error a chain tolerates. Susupro [20] runs ARC’s exact $K = 4$ recursion at the challenge size, with a final-layer MSE of 1.72×10^{-9} on five networks and no fitted constants, far more accurate than any in-budget estimator but far over the budget. Among the write-ups published by 22 October, the lowest in-budget final-layer MSE is 1.72×10^{-8} (at 30% of the budget) and the best score 2.97×10^{-9} ; our estimator reaches 1.09×10^{-8} at 12% and a score of 1.30×10^{-9} .

3 Method

Our estimator never samples. Following ARC’s cumulant propagation [23], it carries a truncated description of the joint law of the activations through the network, exactly through each linear map and approximately through each ReLU. Its third-cumulant part started as a port of ARC’s reference implementation [3], which we restructured to fit the budget and extended. This section describes what is computed; Section 6 describes how it is made cheap, and Section 5 the learned corrections.

At a glance. One layer of the estimator (Fig. 1) does the following.

1. *Linear transport* (Section 3.3): the mean and covariance are mapped exactly; every stored third-cumulant *source* is carried one layer further by the first-order response of the ReLUs it passes. [ARC’s reference.]
2. *Slices near the kink* (Sections 3.5, 3.6 and 3.8): from the sources, the κ_3 diagonal and the $(2, 1)$ slice that the ReLU map needs, computed only for neurons near their kink; *births*, the skew that a ReLU creates from skew, are added, and the $(3, 1)$ slice is folded into an effective $(2, 1)$ slice. [Closure; births and the fold are ours.]
3. *Fourth cumulants* (Section 3.7): a scale-mixture *host* gives their level and carried *stars* their variation across neurons. [Closure; the isotropic host is ARC’s, the covariance blend and the stars are ours.]
4. *Learned corrections* (Section 5): small heads rescale the cumulant diagonals and correct the $(2, 1)$ slice before the ReLU map. [Learned; ours.]
5. *ReLU map* (Section 3.4): Edgeworth expansions give the post-activation mean, covariance and new third cumulant, which is stored as a new source; a learned head then shifts the mean. [Closure: ARC’s moment machinery with our additions; the shift is learned.]
6. *Readout* (Section 3.9): after layer 15, the one-dimensional Edgeworth mean of each neuron, scaled by an exact radius factor (Section 3.1). [Closure; the input conditioning is an exact identity.]

The analytic steps contain 36 named constants (138 values) that were trained together with the heads; Section 3.10 compares them with their derived values. Table 1 collects the terms we use.

Notation. Layers are numbered as in the code, $z_\ell = W_\ell^\top y_{\ell-1}$, $y_\ell = \text{ReLU}(z_\ell)$ for $\ell = 0, \dots, 15$ with $y_{-1} = x$; only $\mathbb{E}y_{15}$ is scored. For the pre-activations z of the current layer, m_i , C_{ij} , $v_i = C_{ii} = \sigma_i^2$ and $\alpha_i = m_i/\sigma_i$ are the mean, covariance, variance and standardised mean, and κ_{ijk} , κ_{ijkl} the joint cumulants, with diagonals $\kappa_{3,i} = \kappa_{iii}$, $\kappa_{4,i} = \kappa_{iiii}$ and slices $(\kappa_{21})_{jk} = \kappa_{jjk}$, $(\kappa_{31})_{jk} = \kappa_{jjjk}$, $(\kappa_{22})_{jk} = \kappa_{jjkk}$ ($j \neq k$). Bars mark post-activation quantities ($\bar{m}$, $\bar{C}$, $\bar{v}$, $\bar{\kappa}$). $\mathbb{E}_G$ is the expectation under the Gaussian $\mathcal{N}(m_i, v_i)$, $\psi_{k,i} = \mathbb{E}_G[\text{ReLU}^{(k)}(z_i)]$ (distributional derivatives, $\psi_0 = \mathbb{E}_G \text{ReLU}$), φ and Φ are the standard normal density and distribution function, and $\circ$ is the elementwise product.

Table 1: Terms used in this paper.

Term	Meaning
source	The third cumulant created by the ReLUs of one layer, stored in factored form (Section 3.5).
vertex	A neuron μ at which a ReLU nonlinearity joins covariance paths; its coefficient $(\psi_{2,\mu}, \dots)$ is the vertex factor.
path	A covariance between a vertex and another neuron, $P_{j\mu} \approx \text{Cov}(y_j, z_\mu)$; carried forward as Q .
leg, age	The first-order map T from the layer where a source was born to the current layer; its age is the number of layers in between.
shared basis	One rank-256 basis B onto which the legs of all sources of age 7 or more are projected.
consumption	Contracting all sources with their legs to obtain the κ_3 diagonal and the $(2, 1)$ slice at the current layer.
slice	A two-index section of a cumulant tensor, e.g. the $(2, 1)$ slice κ_{jjk} or the $(3, 1)$ slice κ_{jjjk} .
birth	Skew created by a ReLU from skew that is already in its input, a second-order effect that linear transport misses (Section 3.6).
host, stars	The level of the fourth cumulants (a Gaussian scale mixture) and their variation across neurons, carried on the legs (Section 3.7).
kink-local selection, tail	Doing cumulant work only for fixed-size sets of neurons near their kink; extrapolated values (“tails”) for the rest.
heads, carry	The learned corrections of Section 5; the carry is a 16-dimensional learned state per neuron passed between layers.

3.1 Exact input conditioning

A bias-free ReLU network is positively homogeneous, $y_\ell(tx) = t y_\ell(x)$ for $t > 0$, and for $x \sim \mathcal{N}(0, I_n)$ the radius $\|x\|$ is independent of the direction $\hat{x} = x/\|x\|$. Hence, exactly,

$$\mathbb{E} y_\ell(x) = \nu \mathbb{E} y_\ell(\sqrt{n} \hat{x}), \quad \nu = \frac{\mathbb{E}\|x\|}{\sqrt{n}} = \sqrt{\frac{2}{n}} \frac{\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n}{2})} = 0.999756 \quad (n = 1024). \quad (3)$$

We propagate $\tilde{x} = \sqrt{n} \hat{x}$ and multiply the predicted means by ν . This removes the input’s radial fluctuation, a global scale mode, from what the truncated propagation has to carry. The law of $\tilde{x}$ is known exactly: mean 0, covariance I , no odd cumulants, and fourth cumulant $\kappa_{abcd}(\tilde{x}) = -\frac{2}{n+2}(\delta_{ab}\delta_{cd} + \delta_{ac}\delta_{bd} + \delta_{ad}\delta_{bc})$, a scale mixture of the form used in Section 3.7 with level $h_0 = -6/(n+2)$. Other teams used the same identity independently [5, 9].

3.2 State carried between layers

Between layers the estimator stores:

- the post-activation mean $\bar{m}$;
- the covariance $\bar{C}$; at layers 5–14 only neurons with $\alpha_i > -3$ keep their off-diagonal entries (at least 512 of them, rounded up to a multiple of 32), the others keep only their variance;
- the third cumulant as one source per earlier layer, each with two $n \times n$ matrices, its leg T and its transported path Q (Section 3.5), or, from age 7, coordinates in the shared basis;
- the fourth cumulant as the host level plus the stars carried by each source (Section 3.7);
- the legs of the last six layers, which the births need (Section 3.6);
- the carry of the learned heads (Section 5).

No n^3 object is ever formed.

3.3 Linear layer and source transport

A linear layer maps cumulants exactly, contracting every index with $W = W_\ell$: $m = W^\top \bar{m}$, $C = W^\top \bar{C} W$ and $\kappa_{abc} = \sum_{ijk} W_{ia} W_{jb} W_{kc} \bar{\kappa}_{ijk}$. Neurons that keep only their variance add $\sum_i \bar{v}_i W_{ia}^2$ to the diagonal of C . Sources are the one part of the state that is not mapped exactly. Each ReLU they pass is linearised, $\delta y_i \approx g_i \delta z_i$, so that crossing one more layer multiplies every leg and path by $W^\top D$ with $D = \text{diag}(g)$. The second-order terms this drops are the births of Section 3.6. For g we use the expected slope under the Edgeworth law of z_i , not the Gaussian slope $\Phi(\alpha_i)$ of the reference implementation:

$$g_i = \psi_{1,i} + \frac{\kappa_{3,i}}{6} \psi_{4,i} + \frac{\kappa_{4,i}}{24} \psi_{5,i} + \frac{\kappa_{3,i}^2}{72} \psi_{7,i}. \quad (4)$$

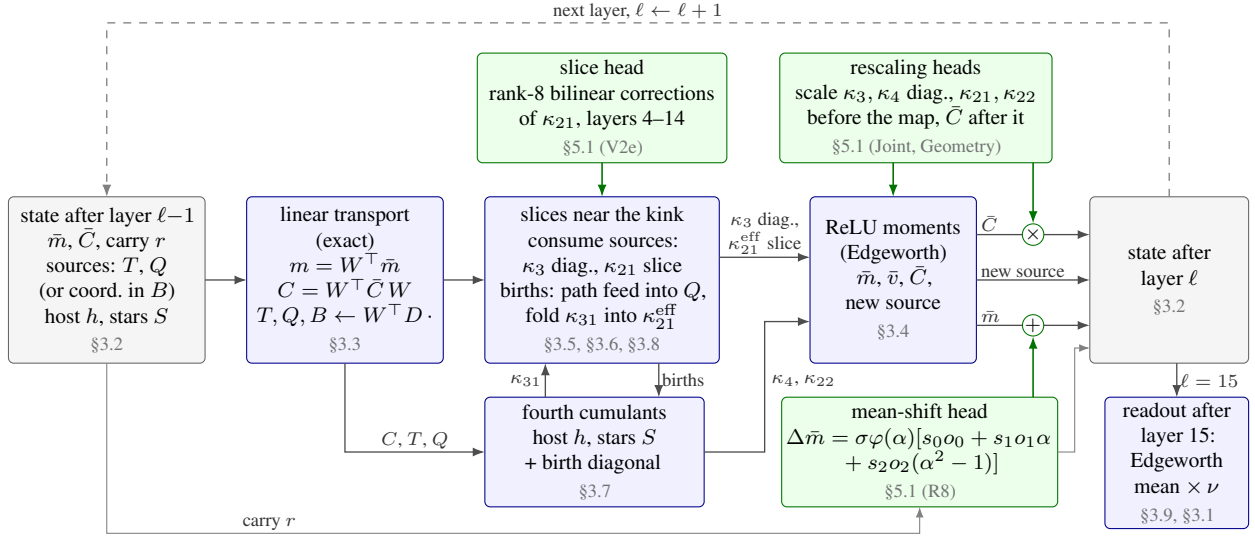


Figure 1: One layer ℓ of the estimator. Grey boxes hold the state carried between layers, blue boxes are analytic steps, and green boxes are learned heads, trained end to end through the whole estimator; $\times$ and $+$ mark where a head rescales or shifts a quantity. A *source* is the third cumulant that the ReLUs of one earlier layer created, carried by its leg T and path Q (Fig. 2); *births* are the skew that a ReLU creates from an already skewed input, which linear transport misses; the fourth cumulant is a scale-mixture *host* h plus *stars* S , the quartic vertices of the sources carried along their legs. The *carry* r is a learned 16-dimensional state per neuron that the mean-shift head updates and passes to the next layer; after layer 15 the readout turns each neuron’s m , v , κ_3 and κ_4 into its Edgeworth mean, times the radius factor ν .

3.4 ReLU layer: Edgeworth moments and the new source

Gaussian closed forms. Under the Gaussian reference, $\mathbb{E}_G[\text{ReLU}^k]$ ($k \leq 4$) have closed forms, and $\psi_0 = \sigma\varphi(\alpha) + m\Phi(\alpha)$, $\psi_1 = \Phi(\alpha)$ and $\psi_k = (-1)^k \text{He}_{k-2}(\alpha) \varphi(\alpha) \sigma^{1-k}$ for $2 \leq k \leq 7$, with the probabilists’ Hermite polynomials He . In particular $\psi_2 = \varphi(\alpha)/\sigma$ and $\psi_3 = -\alpha\varphi(\alpha)/\sigma^2$, so that

$$\psi_3 = -\frac{m}{v} \psi_2 \quad \text{exactly, for the ReLU.} \quad (5)$$

Marginals. A one-dimensional Edgeworth expansion of the law of z_i gives the moments $p_k \approx \mathbb{E}[y_i^k]$, $k = 1, \dots, 4$,

$$p_k = \mathbb{E}_G[\text{ReLU}^k] + \frac{\kappa_3}{6} \mathbb{E}_G[(\text{ReLU}^k)^{(3)}] + \frac{\kappa_4}{24} \mathbb{E}_G[(\text{ReLU}^k)^{(4)}] + s_{34} \frac{\kappa_3 \kappa_4}{144} \mathbb{E}_G[(\text{ReLU}^k)^{(7)}], \quad (6)$$

where each derivative reduces to a ψ (e.g. $\mathbb{E}_G[(\text{ReLU}^2)^{(3)}] = 2\psi_2$). The coefficient $s_{34} = 0.40$ and a factor on the κ_4 term per depth band (0.81–0.83) are trained. The moments give $\bar{m}_i = p_1$ (before the learned shift of Section 5) and the diagonals $\bar{v}$, $\bar{\kappa}_3$ and $\bar{\kappa}_4$.

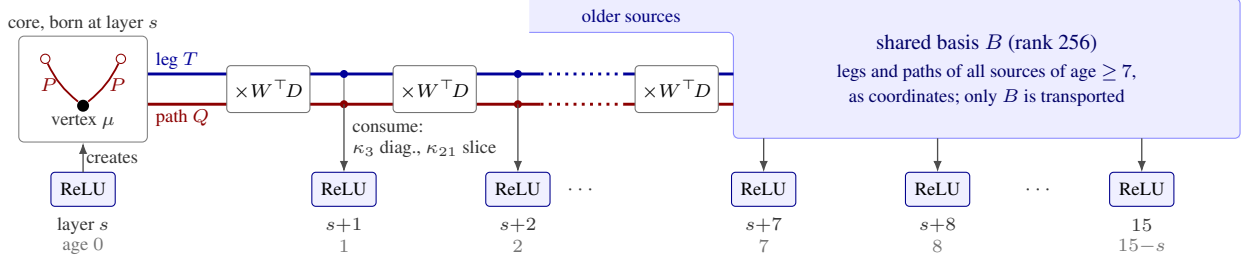
Pairs. For $i \neq j$ a bivariate Hermite–Edgeworth expansion in the joint cumulants gives

$$\bar{C}_{ij} \approx \tilde{\psi}_{1,i} \tilde{\psi}_{1,j} C_{ij} + \frac{1}{2} (\tilde{\psi}_{1,i} \tilde{\psi}_{2,j} \kappa_{ijj} + \tilde{\psi}_{2,i} \tilde{\psi}_{1,j} \kappa_{iij}) + \tilde{\psi}_{2,i} \tilde{\psi}_{2,j} (\frac{1}{2} C_{ij}^2 + \frac{1}{4} \kappa_{iiij}), \quad (7)$$

and the same expansion of $\text{Cov}(y_i^2, y_j)$ and $\text{Cov}(y_i^2, y_j^2)$ gives the output slice $\bar{\kappa}_{iij} = \text{Cov}(y_i^2, y_j) - 2p_{1,i} \bar{C}_{ij}$ and the total pair fourth cumulant $\sum_{i \neq j} \bar{\kappa}_{iijj}$, which sets the host level (Section 3.7). In Eq. (7) each neuron’s Gaussian factors are corrected for its own skewness and kurtosis, $\tilde{\psi}_k = \psi_k + a_3 \frac{\kappa_3}{6} \psi_{k+3} + a_4 \frac{\kappa_4}{24} \psi_{k+4}$ ($k \leq 3$); the Edgeworth expansion gives $a_3 = a_4 = 1$, and training gives $a_3 = 1.00$ and $a_4 = 0.27$. Next-order products ($C\kappa_{21}$, κ_{21}^2 , $\kappa_3\kappa_{21}$, the cubic Hermite term in C^3 , and quadratic κ_3 terms in the output slice) enter with trained coefficients. These additions, the self-cumulant correction and the trained factors are ours; the closed forms, Eqs. (6)–(7) and the conversion from moments to cumulants are the moment machinery of the reference implementation [3, 23].

The new source. To first order the output third cumulant contains the transported input cumulant $g_a g_b g_c \kappa_{abc}$ plus a part that the ReLU creates, with diagonal $\bar{\kappa}_{3,i} - g_i^3 \kappa_{3,i}$ and slice $\bar{\kappa}_{iij} - g_i^2 g_j \kappa_{iij}$. As in the reference implementation [3], the created part is stored as a new source.

(a) The life of a source



(b) A birth

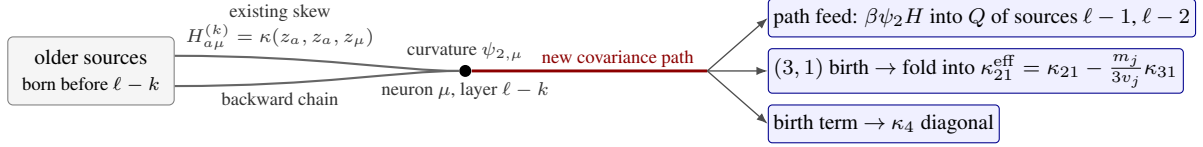


Figure 2: Third-cumulant sources and births. (a) The ReLUs of layer s create a *source*, a third cumulant whose core is a sum over *vertices* μ (neurons of layer s , weighted by their curvature $\hat{\psi}_{2,\mu}$), each joining two *paths* P , the covariances of the layer’s outputs with z_μ . The source reaches later layers through its *leg* T , the first-order map from y_s to the current pre-activations, and its transported path $Q = TP$; both are multiplied by $W^\top D$ at every layer, and at *age* $\ell - s = 7$ they are replaced by coordinates in one rank-256 *shared basis* B that holds all old sources, so that only B is transported. At every later layer the source is *consumed*: contracting its core with T and Q gives its share of the κ_3 diagonal and the κ_{21} slice that the ReLU map of that layer needs. (b) A *birth*: at an intermediate ReLU (layer $\ell - k$, $k = 1, 2$) the curvature $\psi_{2,\mu}$ of neuron μ joins an existing skew $\kappa(z_a, z_a, z_\mu)$, obtained from older sources by a backward chain $H^{(k)}$, to a new covariance path; this creates skew that the linear transport of sources misses, and it enters the estimator in the three places shown.

3.5 Third cumulants as sources $\times$ legs

Source form. The source born at layer s stores the third cumulant that the ReLUs of that layer create, with vectors and matrices over the neurons μ of layer s , the source’s vertices (Fig. 2a):

$$\kappa_{ijk}^{(s)} = \sum_{\mu} \left(\text{Sym}_{ijk} \left[\hat{\psi}_{2,\mu} \delta_{i\mu} P_{j\mu} P_{k\mu} + 2u_{\mu} \delta_{i\mu} \delta_{j\mu} P_{k\mu} + \frac{1}{3} \delta_{i\mu} \delta_{j\mu} R_{k\mu} \right] + f_{\mu} \delta_{i\mu} \delta_{j\mu} \delta_{k\mu} \right), \quad (8)$$

where Sym_{ijk} sums the three distinct placements of i, j, k in each monomial. The first term is the Gaussian tree: the curvature $\hat{\psi}_2 = \psi_2 + \frac{\kappa_3}{6} \psi_5$ of vertex μ joined to two paths $P_{j\mu} \approx \text{Cov}(y_j, z_\mu)$,

$$P_{i\mu} = \Phi_i \left(C_{i\mu} - \frac{\eta}{4} \frac{m_{\mu}}{v_{\mu}} \kappa_{\mu\mu i} \right) + \frac{\eta}{4} \psi_{2,i} \kappa_{ii\mu} - \tau \frac{m_{\mu}}{v_{\mu}} \psi_{2,i} C_{i\mu}^2 \quad (i \neq \mu), \quad P_{\mu\mu} = 0, \quad (9)$$

with trained closure gains $\eta/4 = 0.47$ (the expansion gives $\frac{1}{2}$ for the $\psi_{2,i} \kappa_{ii\mu}$ term) and $\tau = 0.48$. With $u = \tilde{\psi}_0 - p_1 \tilde{\psi}_1$, the second term is the Gaussian part $2u_{\mu} P_{k\mu}$ of $\bar{\kappa}_{\mu\mu k}$. A rank-2 matrix R and a diagonal f make the source reproduce the created diagonal exactly and the created slice up to the rank-2 truncation.

Legs and consumption. At a later layer $\ell > s$ the source is contracted on every index with its leg T , the first-order map from y_s to z_{ℓ} : $T = W_{\ell}^\top D_{\ell-1} \cdots D_{s+1} W_{s+1}^\top$ with $D_{\ell} = \text{diag}(g_{\ell})$. Each source carries T and the transported path $Q = TP$ ($Q_{a\mu} \approx \text{Cov}(z_a, z_{\mu}^{(s)})$), both multiplied by $W^\top D$ at every layer. Contracting Eq. (8) (*consumption*) gives

$$\kappa_{aaa} = \sum_s \sum_{\mu} T_{a\mu} (3\hat{\psi}_{2,\mu} Q_{a\mu}^2 + f_{\mu} T_{a\mu}^2 + \tilde{R}_{a\mu} T_{a\mu} + 6u_{\mu} T_{a\mu} Q_{a\mu}), \quad (10)$$

$$\kappa_{aab} = \frac{1}{3} \sum_s \sum_{\mu} (X_{a\mu} Q_{b\mu} + Y_{a\mu} T_{b\mu} + T_{a\mu}^2 \tilde{R}_{b\mu}), \quad (11)$$

with $\tilde{R} = TR$, $X = 6\hat{\psi}_2 T \circ Q + 6u T \circ T$ and $Y = 3\hat{\psi}_2 Q \circ Q + 3f T \circ T + 2\tilde{R} \circ T + 12u T \circ Q$ (vertex vectors scale columns). The all-distinct κ_{abc} is never formed, and a source costs four products of n^3 size per layer: two transports and the two slice contractions $[X, Y][Q, T]^\top$. An earlier form of ours needed six; when we switched, the FLOPs fell from 41% to 28% of the budget and the MSE rose by 1.6%, a change we did not decompose further. The factored source with legs is the reference implementation’s [3]; a similar consumption form, with a low-rank residual leg and a shared basis for old sources, was developed in parallel [1].

Ages and the shared basis. A leg is a product of $\ell - s$ random matrices, and its spectrum concentrates with age: the top 256 singular directions hold 67–70% of its energy at age 1 ($T = W^\top$) and 99–100% at age 6. From age 7 we therefore write $T \approx BU_T^\top$ and $Q \approx BU_Q^\top$ with one basis $B \in \mathbb{R}^{n \times 256}$ shared by all old sources. Only B is transported, and the slices of all joined sources are summed in basis coordinates before one product with $B^\top$. The basis comes from one truncated SVD of the leg of the first source that reaches age 7, weighted toward the columns that feed Q (Appendix B). Age-gated shared bases also appear in [1, 5].

3.6 Births and the effective slice

What linear transport misses. Source transport keeps only the first-order response of each ReLU. To second order, a neuron μ of an intermediate layer responds as

$$y_\mu - \mathbb{E}y_\mu \approx g_\mu \delta z_\mu + \frac{1}{2} \psi_{2,\mu} (\delta z_\mu^2 - v_\mu), \quad \delta z_\mu = z_\mu - m_\mu. \quad (12)$$

By the product formula for cumulants (Leonov–Shiryaev), the quadratic term adds to every cumulant that involves y_μ . For other variables X and A, B, C ,

$$\text{Cov}(y_\mu, X) \supset \frac{1}{2} \psi_{2,\mu} \kappa(z_\mu, z_\mu, X), \quad (13)$$

$$\kappa(y_\mu, A, B, C) \supset \psi_{2,\mu} [\kappa(z_\mu, A, B) \text{Cov}(z_\mu, C) + \kappa(z_\mu, A, C) \text{Cov}(z_\mu, B) + \kappa(z_\mu, B, C) \text{Cov}(z_\mu, A)]. \quad (14)$$

If the ReLU’s input is already skewed, Eq. (13) feeds the existing third cumulant into a covariance path, and Eq. (14) creates fourth cumulants that are products of an existing third cumulant and a covariance. We call these terms *births*: skew that a ReLU creates from skew (Fig. 2b). Linear transport drops them.

Computing them. The births need the cross-layer slice $H_{a\mu}^{(k)} = \kappa(z_a^{(\ell)}, z_a^{(\ell)}, z_\mu^{(\ell-k)})$ between the current layer ℓ and an intermediate layer $\ell - k$. We compute it recursively with the blocks of Eq. (11): (1) start k layers back and contract the blocks X, Y of every source born before layer $\ell - k$ with that source’s legs at layer $\ell - k$; (2) move up one layer at a time by multiplying the column index by that layer’s DW , adding the blocks of the source born there; (3) after k steps the result is the ordinary slice $H^{(0)} = \kappa_{21}$. Births use $k = 1, 2$ at layers 5–15. From layer 10 the recursion starts up to six layers back, and the terms from three to six layers back enter only the κ_4 diagonal. Births enter the state in three places:

- *Path feed.* By Eq. (13), $\text{Cov}(y_a, z_\mu) \supset \frac{1}{2} \psi_{2,a} H_{a\mu}^{(k)}$. We add $\beta(\psi_{2,a}/g_a) H_{a\mu}^{(k)}$ to the paths Q of the two youngest sources ($k = 1, 2$), so that the next transport carries $\beta \psi_2 H$; the derivation gives $\beta = \frac{1}{2}$, and training gives $\beta = 0.28\text{--}0.42$.
- *The (3, 1) slice.* By Eq. (14), $\kappa_{aaab} \supset 3 \sum_\mu \psi_{2,\mu} H_{a\mu} (T_{a\mu} \hat{Q}_{b\mu} + \hat{Q}_{a\mu} T_{b\mu})$, with the legs of the source born at the vertex layer and $\hat{Q} = Q + T \text{diag}(\Phi_\mu v_\mu)$, the path including the vertex’s own variance. The two terms carry trained weights of 2.3–3.1 instead of 3, and at age 2 only the half of the vertices with the largest ψ_2 enter. This slice is folded into the effective slice below.
- *The κ_4 diagonal* receives $12 \sum_\mu \psi_{2,\mu} H_{a\mu} T_{a\mu} \hat{Q}_{a\mu}$ with a trained weight per layer (0.27–0.54, against 1), plus the older births (weight 0.37) and a small per-layer rescaling of the host term.

The effective slice. In Eq. (7) the slice κ_{jjk} is multiplied by $\frac{1}{2} \psi_{2,j} \psi_{1,k}$, the curvature of the neuron that appears twice. The next term of the same expansion, not written in Eq. (7), multiplies the (3, 1) slice κ_{jjjk} by $\frac{1}{6} \psi_{3,j} \psi_{1,k}$. With Eq. (5), $\frac{1}{2} \psi_{2,j} \kappa_{jjk} + \frac{1}{6} \psi_{3,j} \kappa_{jjjk} = \frac{1}{2} \psi_{2,j} (\kappa_{jjk} - \frac{m_j}{3v_j} \kappa_{jjjk})$, so the two terms combine into one effective slice,

$$(\kappa_{21}^{\text{eff}})_{jk} = \kappa_{jjk} - \frac{m_j}{3v_j} \kappa_{jjjk}. \quad (15)$$

We fold the available (3, 1) slice into κ_{21} this way. It has two parts: the (3, 1) slice of the stars that every source carries (Section 3.7), with exact combinatorial weights and one trained factor per layer (0.81–1.05, against 1), and the (3, 1) birth above. Section 4 shows that this effective slice is the main local source of the remaining error.

Related work. Birth-type terms appear in parallel work: path births [5], an “edge history” of $\kappa_3 \times C$ terms [15], an exact (3, 1) slice in a factored chain [10], and, outside the budget, an exact $K = 4$ chain in which $C \cdot \kappa_3$ births are the largest single term [20]. To our knowledge the effective slice of Eq. (15) is not in other write-ups.

3.7 Fourth cumulants: host and stars

Host. A Gaussian scale mixture with base covariance Λ has $\bar{\kappa}_{abcd} = \frac{h}{3}(\Lambda_{ab}\Lambda_{cd} + \Lambda_{ac}\Lambda_{bd} + \Lambda_{ad}\Lambda_{bc})$ for one scalar level h . The reference implementation uses this form with $\Lambda = I$, i.e. the weight metric $W^\top W$ after the next layer [3]. We blend in the propagated covariance, $\Lambda = (1 - \vartheta)\frac{\text{tr}\bar{C}}{n}I + \vartheta\bar{C}$ with trained $\vartheta = 0.61$. After every ReLU we refit h to the total pair fourth cumulant of Section 3.4, and after the next layer the host gives the pre-activation diagonal and pair slice:

$$h = \frac{3 \sum_{i,j} \bar{\kappa}_{iiij}}{(\text{tr}\Lambda)^2 + 2\|\Lambda\|_F^2}, \quad \|\Lambda\|_F^2 = (1 - \vartheta^2)\frac{(\text{tr}\bar{C})^2}{n} + \vartheta^2\|\bar{C}\|_F^2, \quad (16)$$

$$\kappa_{4,i} = h\lambda_i^2, \quad \kappa_{iiij} = \frac{h}{3}(\lambda_i\lambda_j + 2c_{\text{off}}C_{ij}^2), \quad \lambda_i = (W^\top\Lambda W)_{ii} = (1 - \vartheta)\frac{\text{tr}\bar{C}}{n} \sum_a W_{ai}^2 + \vartheta v_i, \quad (17)$$

with a trained $c_{\text{off}} = -0.56$ (an exact mixture has $c_{\text{off}} = 1$ and $(W^\top\Lambda W)_{ij}$ in place of C_{ij}). The sphere input of Section 3.1 is exactly of this form, with $h_0 = -6/(n+2)$ and $\Lambda = I$. A scale-mixture closure of κ_4 was also developed in parallel [5].

Stars. Each source also carries the quartic vertices that its ReLU creates at Gaussian order: the three-path vertex ψ_3 , the $(2, 2)$ vertex $c_{22} = 2(\Phi(1 - \Phi) - \psi_0\psi_2)$, the $(3, 1)$ vertex $c_{31} = \kappa(y, y, y, z)/v = [\bar{\kappa}_4 + (p_1 - m)(\bar{\kappa}_3 - 3p_1\bar{v} + p_1^3)]/v$ (an identity for the ReLU), and the $(4, 0)$ vertex $\bar{\kappa}_4$. Transported on the legs they give the stars

$$S_a = \sum_s \sum_\mu \left[s_\star \psi_{3,\mu} T_{a\mu} Q_{a\mu}^3 + (s_{22}c_{22,\mu} + \psi_{3,\mu}) T_{a\mu}^2 Q_{a\mu}^2 + s_{31}c_{31,\mu} T_{a\mu}^3 Q_{a\mu} \right], \quad (18)$$

with trained $s_\star = 4.8$, $s_{22} = 4.7$ and $s_{31} = 3.4$ (the combinatorial weights are 4, 6 and 4), and $\kappa_{4,a} = h\lambda_a^2 + (S_a - \text{mean}_b S_b) + \text{births}$. The host, refit from the total fourth cumulant, sets the level, and the centred stars add only the variation across neurons. The $(3, 1)$ slice of the vertices, with exact weights, is the first part of the fold in Section 3.6.

3.8 Kink-local selection

Every κ_3 - or κ_4 -dependent term of the ReLU map carries a ReLU derivative of order two or more, $\psi_k \propto \varphi(\alpha)$, so it matters only near the kink, and nearly dead neurons ($g \approx 0$) add little to the transport. We therefore do the third- and fourth-cumulant work only for sets of neurons near the kink. Each set has a fixed size, so that array shapes, FLOPs and the number of calls are nearly the same on every network, as the time limits require (Section 6); which neurons belong to it is decided per network. From layer 5 the neurons are ordered by $\Phi(\alpha)$ or $|\alpha|$, and the sets shrink with depth, for example from 960 to 576 rows for the $(2, 1)$ slice (Appendix B lists all sizes). Skipped rows get *tails* in units of their own scale, $\kappa_{3,i} \approx \langle \kappa_3/\sigma^3 \rangle \sigma_i^3$ and $S_i \approx \langle S/\sigma^4 \rangle \sigma_i^4$, averaged over the last 64 computed rows.

3.9 Readout

At the last layer each neuron’s prediction is the one-dimensional Edgeworth mean

$$\widehat{\mathbb{E}y_i} = \nu \max \left\{ 0, \psi_0 + s_3 \frac{\kappa_3}{6} \psi_3 + s_4 \frac{\kappa_4}{24} \psi_4 + s_{34} \frac{\kappa_3 \kappa_4}{144} \psi_7 + \Delta_i \right\}, \quad (19)$$

with trained scales $(s_3, s_4, s_{34}) = (1.01, 0.81, 0.40)$, the learned shift Δ_i of Section 5, and ν from Eq. (3). The readout keeps the κ_3 and κ_4 terms of Eq. (2) and the next-order $\kappa_3\kappa_4$ term, but not the κ_3^2 term: with true inputs, adding it would lower the readout’s own error only from 9.6×10^{-11} to 6.0×10^{-11} , against 1.1×10^{-8} for the estimator (Section 4.2).

3.10 Trained constants

The closures above contain 36 named constants with 138 values in all: gains, per-layer factors and blend weights. They were trained end to end together with the learned heads (Section 5.2). Table 2 compares the main ones with the values that the derivations give. Several sit far from them, for example a_4 , c_{off} and the κ_4 birth weights. We read these constants as closure gains that absorb what the truncated expansions miss and what the heads correct, not as estimates of the true coefficients.

Table 2: Analytic constants of the closures, trained end to end together with the learned heads. Theory is the value that the derivation gives (none for terms without one), ratio is trained/theory, and ranges run over layers or depth bands.

Term	Symbol	Theory	Trained	Ratio
<i>ReLU map</i>				
Skewness correction of the Gaussian factors	a_3	1	1.00	1.00
Kurtosis correction of the Gaussian factors	a_4	1	0.27	0.27
Factor on κ_4 in the moments p_k (3 depth bands)	–	1	0.81–0.83	0.81–0.83
Quadratic κ_3 terms of the output slice (3 terms)	–	1	1.3, 13.0, 12.7	1.3–13
κ_{21}^2 and $\kappa_3\kappa_{21}$ terms of $\bar{C}$	–	none	6.4, 3.7 ^b	–
<i>Sources and paths</i>				
κ_3 terms of the path P	$\eta/4$	$\frac{1}{2}$	0.47	0.94
C^2 term of the path P	τ	$\frac{1}{2}$	0.48	0.96
<i>Births</i>				
Path feed into Q ($k = 1, 2$)	β	$\frac{1}{2}$	0.28–0.42	0.56–0.84
(3, 1) birth, two terms (3 depth bands)	–	3	2.3–3.1	0.76–1.05
(3, 1) slice of the stars folded into κ_{21}	–	1	0.81–1.05	0.81–1.05
κ_4 birth diagonal, $k = 1, 2$	–	1	0.27–0.54	0.27–0.54
κ_4 birth diagonal, $k = 3-6$	–	1	0.36–0.37	0.36–0.37
<i>Fourth cumulants</i>				
Covariance blend of the host	ϑ	1 ^a	0.61	0.61
Pair coefficient of the host	c_{off}	1	−0.56	−0.56
Three-path star	$s_\star$	4	4.76	1.19
(2, 2) and (3, 1) vertices	s_{22}, s_{31}	6, 4	4.66, 3.42	0.78, 0.85
<i>Readout</i>				
κ_3 and κ_4 terms	s_3, s_4	1, 1	1.01, 0.81	1.01, 0.81
$\kappa_3\kappa_4$ term (also in p_k)	s_{34}	1	0.40	0.40

^a Exact scale mixture ($\Lambda \propto \bar{C}$); the reference implementation uses $\vartheta = 0$. ^b Multiples of the code’s starting coefficient $\frac{1}{12}$; these product closures have no derived value.

4 Where the error comes from

This section locates the error that remains and connects it to the mechanisms of Section 3. The analyses were made on the estimator of 9–10 October (graded 1.54×10^{-9}), which has the architecture of the submission but older weights and fewer cost cuts.

4.1 Versions and measurements

Table 3 lists the versions of the estimator that our numbers refer to. All MSEs are final-layer MSEs against labels from 10^9 samples, *noise-corrected* by subtracting the label-noise term $\overline{\text{Var}}(y_{15})/10^9$ that the dataset supplies. Comparisons are paired: we compute the per-network log ratio of the MSE (and of the FLOPs) and report its geometric mean, with a 95% confidence interval or a z score, and the number of networks that improved. Selection used a fixed validation panel (VAL32, public networks 832–863); the switch-off ledger of Section 7.2 uses 64 other public networks (896–959). Three kinds of experiment appear below.

- *Switch-off*: a component’s coefficients are set to zero and nothing is retrained. *Heads off* switches off all learned parts of Section 5; *heads frozen* keeps them at their trained values while something else is changed.
- *Oracle injection*: one propagated quantity is replaced by its Monte Carlo truth at the given layers, and the estimator runs on from there.
- *Linearised counterfactual*: the error of the post-activation mean at layer ℓ is split into the part inherited through the linear response, $\Phi_\ell \circ \delta m_\ell$, and a *local innovation* $\varepsilon_\ell = \delta \bar{m}_\ell - \Phi_\ell \circ \delta m_\ell$ created at that layer. Propagating every ε_ℓ to the output by the true linear response reproduces the final mean error (correlation 1.000), so the final error splits by layer. Splitting each ε_ℓ further by the ReLU map’s first-order response to errors in its inputs (variance, κ_3 , κ_4 , learned parts) splits it by channel, and removing one channel’s propagated contribution estimates what making that channel exact would give. For the variance at layers 6–14 this predicts $\times 0.650$, against $\times 0.645$ measured by injection.

Table 3: Versions of the estimator that the numbers in this paper refer to.

Version	Graded score	Used for
8 October build	–	heads switched off (Table 5; VAL32 MSE 1.29×10^{-8})
10 October build	1.54×10^{-9}	error map and oracles (this section)
cost-map build (October)	–	FLOP shares (Section 6)
#337501 (submitted)	1.299×10^{-9}	graded results, switch-off ledger (Section 7)
#337556 (second nominated)	1.295×10^{-9}	graded results only

4.2 The readout is accurate

A neuron’s mean depends only on its own marginal law. Fed with the true sphere-conditioned m , v , κ_3 and κ_4 of every final-layer neuron (32 validation networks) and unit coefficients, the readout series has an MSE of 1.8×10^{-7} with the Gaussian term alone, 1.7×10^{-8} with the κ_3 term and 9.6×10^{-11} with the κ_4 term as well (6.0×10^{-11} with the κ_3^2 term added), against 1.3×10^{-8} for the estimator of that time and 1.09×10^{-8} for the submission. The readout’s own error is under 1% of the total, so almost all of the error is in its inputs.

4.3 The error is mean error made upstream

With only the pre-activation mean of the final layer wrong, the MSE is 1.0×10^{-8} (77% of it); a wrong variance, κ_3 or κ_4 alone gives 1.8, 1.1 and 1.1×10^{-9} (Fig. 3a). The final pre-activation mean is exactly $W_{15}^\top \bar{m}_{14}$, so this part of the error was made at earlier layers. By the layer whose ReLU created it, 3% of the final error comes from layers 0–5, 14% from 6–9, 29% from 10–12 and 31% from 13–14; the remaining 23% is created at the last ReLU itself, by the errors of its variance and higher cumulants (Fig. 3b).

4.4 The local cause: the variance, and behind it the effective slice

A neuron’s post-ReLU mean depends on its pre-activation variance (Eq. (2)), so variance errors become mean errors. Making the variance exact at every layer would multiply the MSE by 0.51 without heads and by 0.53 with them (Fig. 3c): the heads barely touch this channel. The pre-activation variance of the next layer is $w_i^\top \bar{C} w_i$, and its error is the image of the post-activation covariance error, $\delta v_{\ell+1} = \text{diag}(W^\top \delta \bar{C}_\ell W)$ (correlation 0.9997), 87% of it from off-diagonal entries. About 60% of the energy of that covariance error is inherited from earlier layers. The part created at the layer itself is removed by one exact input: on four networks at layers 10 and 14, the exact bivariate Edgeworth pair map with all-true inputs leaves $1.0\text{--}2.4 \times 10^{-5}$ RMS per covariance entry, the estimator’s local error is $4.4\text{--}6.5 \times 10^{-5}$, and giving it the true effective slice κ_{21}^{eff} of Eq. (15) alone brings it down to that floor. The true κ_{21} or the true κ_{31} alone barely help. The estimator’s slice has a 13–21% relative error against κ_{21}^{eff} and carries about half of the needed fold. End to end, with the heads off, the true κ_{21}^{eff} in the covariance update at layers 6–14 multiplies the final MSE by 0.651, the plain true κ_{21} only by 0.819 (Fig. 3d). Most of the value lies in layers 6–11 ($\times 0.729$ for those alone), whose local errors propagate furthest.

4.5 The heads absorb the third- and fourth-cumulant errors

Table 4 collects the oracle results. Without heads, exact κ_3 or κ_4 diagonals at every layer would give $\times 0.75$ and $\times 0.86$; as deployed they give $\times 0.97$ and $\times 1.17$. The heads have learned to compensate these errors, so fixing one of them alone breaks the compensation. With the heads off, making the variance, κ_3 and κ_4 exact together would leave only 4% of the MSE; these channels carry almost all of the error. Moving κ_{21}^{eff} , κ_3 and κ_4 halfway to the truth at layers 6–14 gives $\times 0.44$ without heads and $\times 0.52$ with the heads frozen, so the gains stack. With frozen heads a half move toward the true κ_{21}^{eff} is better than the exact value ($\times 0.744$ against $\times 0.789$): the slice head’s correction still compensates other errors until it is retrained.

4.6 What does not fix it

The covariance error is dominated by its component along the mean vector, the global scale mode, but removing that component with exact per-network oracles changes nothing (last row of Table 4): the rest of the estimator is consistent with this deficit. This contrasts with the large gain that il.miscusi [9] report from removing the scale mode in a chain with a carried fourth-order part. Three attempts to fix the effective slice directly also failed:

- *Learning it per neuron.* Only 3–5% of the energy of the needed correction lies in the span of 21 per-neuron estimator features on each side, and a rich nonlinear basis explains 2% on held-out networks. Least-squares rescaling of the existing slice toward κ_{21}^{eff} makes the local error worse (+3 to +11%).

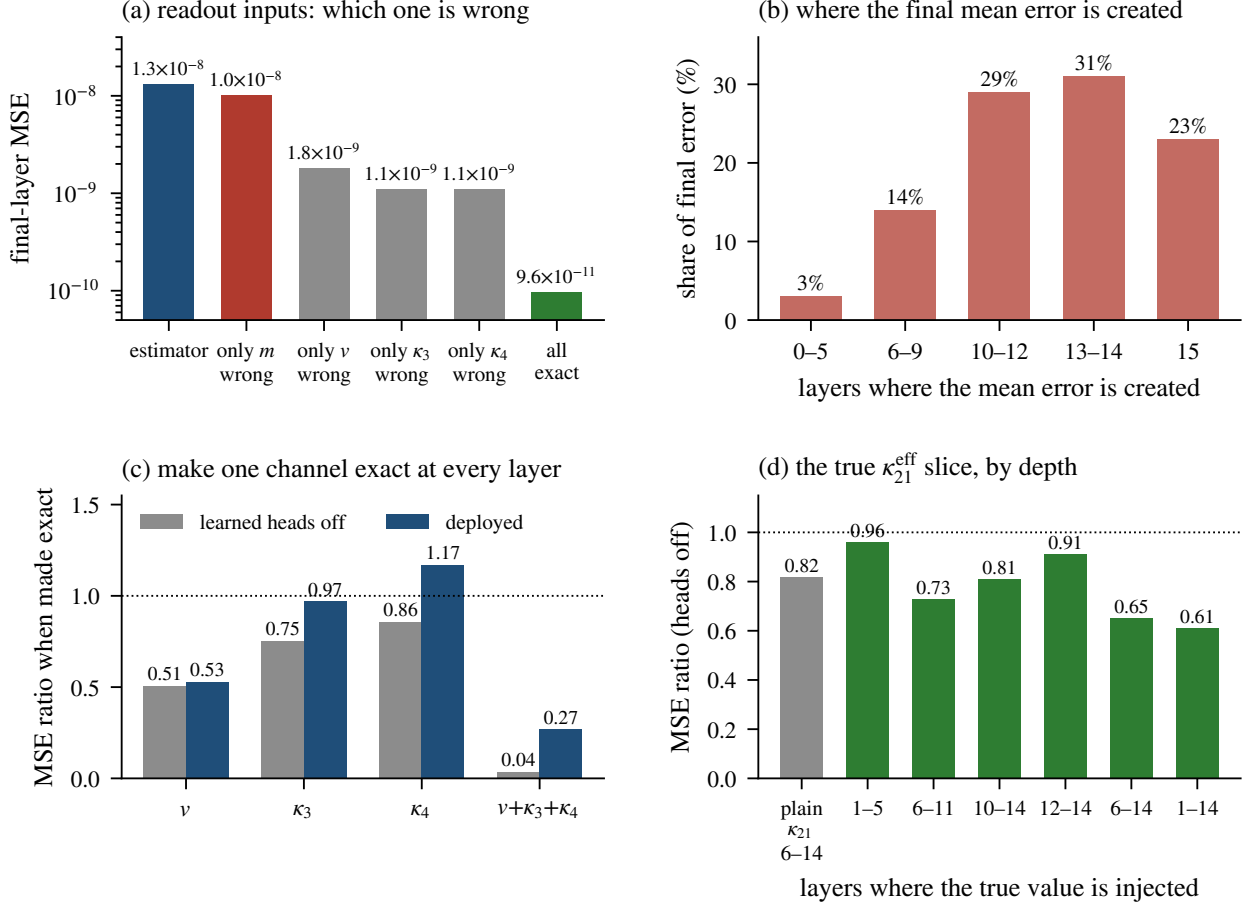


Figure 3: Error map of the 10 October estimator on validation networks with 10^9 -sample truth for every layer. (a) Final-layer MSE when only one input of the readout is replaced by the estimate and the rest are true (22 networks). (b) Share of the final mean error created at each group of layers, from the local innovations of Section 4.1. (c) MSE ratio when one channel is made exact at every layer (linearised counterfactual, Section 4.1), with the learned heads switched off (grey, 22 networks) and as deployed (blue). (d) MSE ratio when the true effective slice κ_{21}^{eff} is injected into the covariance update at the given layers, heads off (8 networks); grey: the plain true κ_{21} instead.

- *Adding the missing one-step $\kappa_4 \rightarrow \kappa_3$ feed*, the part of the $(2, 1)$ slice after a ReLU that comes from the fourth cumulant of its input (5–9% of $|\kappa_{21}|$). The estimator already carries 75–85% of it through the stars and the fold, and adding the rest double-counts (local error 1.3–1.9 times worse).
- *Any single missing term*. The needed correction is nearly orthogonal to every newborn term we computed (correlation 0.03–0.04): it accumulates from many small per-layer errors, not from one missing term.

5 Learned corrections inside the propagation

Three small learned heads correct the cumulant propagation of Section 3 inside the network: a *mean-shift head* (code name R8) at layers 1–15, a *slice head* (V2e) that corrects the $(2, 1)$ slice at layers 4–14, and *rescaling heads* (Joint and Geometry) that rescale the cumulant diagonals and slices from layer 4 on. They modify the propagated state rather than the output, so their corrections travel with the state. They are trained together with the 36 named analytic constants of Section 3.10 by back-propagating the final-layer error through a differentiable twin of the whole estimator. They cost about 2% of the FLOPs. When they were introduced, the graded score steps that added them multiplied to about 0.82; switched off without retraining in the estimator of 8 October they raised its MSE by a quarter (Section 5.4).

5.1 Where the heads act and what they see

Order within a layer. We use the notation of Section 3, with ρ_{ij} the correlation of z_i and z_j . Within a layer the rescaling heads first rescale the cumulants that enter the ReLU map, the slice head corrects κ_{21} , and the mean-shift head computes a shift of $\bar{m}$. The Edgeworth ReLU map then uses the corrected cumulants, the shift is added to its mean, and the rescaling heads rescale $\bar{C}$ (layers 4–14).

Table 4: Oracle results on the 9–10 October estimator: MSE ratio when one quantity is made exact. Injection: the true value is inserted at the given layers and the estimator runs on. Linearised: counterfactual of Section 4.1. Heads on means frozen for injections and deployed for counterfactuals. Networks: VAL32 or subsets of it.

Exact quantity	Layers	Method	Heads off	Heads on	Networks
variance	all	linearised	$\times 0.51$	$\times 0.53$	22
variance	6–14	injection	$\times 0.645$	$\times 0.718$	32
κ_3 diagonal	all	linearised	$\times 0.75$	$\times 0.97$	22
κ_3 diagonal	6–14	injection	$\times 0.803$	$\times 0.913$	32
κ_4 diagonal	all	linearised	$\times 0.86$	$\times 1.17$	22
κ_4 diagonal	6–14	injection	$\times 0.892$	$\times 1.276$	32
variance, κ_3 and κ_4	all	linearised	$\times 0.036$	$\times 0.27$	22
full post-activation covariance	6–14	injection	$\times 0.554$	$\times 0.567$	32
κ_{22}^{eff}	6–14	injection	$\times 1.132$	$\times 1.82$	32
κ_{21}^{eff} (covariance update)	6–14	injection	$\times 0.651$	$\times 0.789$	8 / 4
plain κ_{21} (covariance update)	6–14	injection	$\times 0.819$	–	8
κ_{21}^{eff} , κ_3 , κ_4 halfway	6–14	injection	$\times 0.439$	$\times 0.522$	8 / 4
covariance along the mean direction	8–14	injection	$\times 1.002$	$\times 1.007$	16 / 8

Mean-shift head (R8). A permutation-equivariant network over the neurons of a layer. Each neuron supplies 49 inputs plus the depth $\ell/15$: 9 marginal statistics (e.g. α_i , $\kappa_{3,i}/\sigma_i^3$, $\kappa_{4,i}/\sigma_i^4$, $\sum_k W_{ki}^2$), 10 row and column reductions of ρ_{ij} , $\kappa_{21,ij}/(v_i\sigma_j)$ and $\kappa_{22,ij}/(v_iv_j)$, 14 statistics of the births, and its 16-dimensional carry r_i . A 64-wide embedding passes through two residual blocks, each with a context term (the mean over neurons of the layer-normalised embeddings, the only learned interaction across neurons) and a GELU layer of width 128. The output $o_i = \tanh(\cdot) \in (-1, 1)^{19}$ sets the shift

$$\Delta \bar{m}_i = \sigma_i \varphi(\alpha_i) [s_0 o_{i,0} + s_1 o_{i,1} \alpha_i + s_2 o_{i,2} (\alpha_i^2 - 1)], \quad (20)$$

added to the Edgeworth mean before transport, with trained scales $s_0 = 1.35 \times 10^{-3}$, $s_1 = -2.77 \times 10^{-3}$ and $s_2 = 3.93 \times 10^{-3}$. This basis coincides with the first-order response of the Edgeworth mean to errors in the variance and higher cumulants, $\sigma \varphi(\alpha) [\delta v / (2\sigma^2) - \alpha \delta \kappa_3 / (6\sigma^3) + (\alpha^2 - 1) \delta \kappa_4 / (24\sigma^4)]$, so the head can be read as a bounded correction of the standardised variance, skewness and kurtosis as the mean sees them (at most 2.7×10^{-3} , 0.017 and 0.094 per layer). This reading is ours; the basis was chosen empirically.

The other 16 outputs, o_i^c , update the carry, which crosses the next layer’s weights W :

$$r_j^+ = (1 + \zeta_j) \tanh a_j + (1 - \zeta_j) \tanh \left(\sum_i \frac{\Phi_i}{2} W_{ij} r_i \right), \quad a_j = \sum_i \left[\frac{\Phi_i}{2} W_{ij} M_1 + \left(W_{ij}^2 - \frac{2}{n} \right) M_2 + W_{ij}^3 M_3 \right]^\top o_i^c, \quad (21)$$

with learned $M_k \in \mathbb{R}^{16 \times 16}$ and a gate $\zeta_j = \tanh(G^\top a_j + b)$. The carry is thus a small recurrent network that crosses each layer by the operators that move means, variances and third cumulants. Each parameter tensor of the head is a quadratic polynomial in the depth fraction $\ell/15$, so the 16 per-layer networks (47,075 weights each) share 141,225 trainable parameters.

Slice head (V2e). One network with a one-hot layer code serves layers 4–14. Two GELU layers of width 128 map 12 per-neuron statistics, similar to the mean-shift head’s, to an embedding ξ_i , and three rank-8 bilinear forms correct the slice:

$$\kappa_{21,ij} \leftarrow \kappa_{21,ij} + \frac{\gamma_\ell}{\sqrt{8}} [v_i \sigma_j \langle A_1^\top \xi_i, B_1^\top \xi_j \rangle + \kappa_{21,ij} \langle A_2^\top \xi_i, B_2^\top \xi_j \rangle + \sigma_i C_{ij} \langle A_3^\top \xi_i, B_3^\top \xi_j \rangle] \quad (i \neq j), \quad (22)$$

with learned A_t, B_t of rank 8 and per-layer gains γ_ℓ . The three terms are additive, relative, and proportional to the covariance (the form of the $\kappa_3 \times C$ births). The head has 26,416 trainable weights.

Rescaling heads (Joint and Geometry). From layer 4 on, a one-layer tanh network reads 12 marginal features and their layer means and emits per-neuron factors of the form $1 + c \tanh(\cdot)$, or $c \tanh(\cdot)$ for additive terms. They multiply the κ_3 and κ_4 diagonals, κ_{21} (as the sum of a row factor and a column factor), κ_{22} , and, after the ReLU map, $\bar{C}$. A per-layer geometry read-out, which also sees 12 row reductions of the normalised slices and correlations, adds at most $0.125 \sigma_i^3$ and $0.004 \sigma_i^4$ to the diagonals. The submitted amplitudes $|c|$ (per-channel scales times one global amplitude) are 0.24 (κ_3), 0.32 (κ_4), 0.13 and 0.20 (κ_{21}) and 0.36 (κ_{22}), but under 1% on $\bar{C}$: these heads reshape the pre-activation cumulants and barely touch the covariance. They have 11,536 trainable weights; with the 138 values of the analytic constants, 179,315 scalars are trained in all.

5.2 The differentiable twin and end-to-end training

Twin and gradients. The estimator is written once against a NumPy-like array interface. From the same source the build emits the `flopscope` submission and a Torch twin, in which the interface is an adapter onto Torch, so the twin runs the same statements in the same order, fast multiplication circuits included. In our parity tests of the CPU twin, twin and native predictions agree to a relative ℓ_2 error below 2×10^{-6} and paired MSE ratios to about 0.1%; the estimator’s own relative error is about 10^{-4} . Each forward pass runs the twin on one network, and the loss is back-propagated through all 16 layers: the transport of $\bar{m}$, $\bar{C}$ and the sources, the births, the fourth cumulants, the Edgeworth maps, the readout, and every head and carry. A head at layer ℓ thus receives the derivative of the final error through the Jacobian of layers $\ell + 1, \dots, 15$, later heads included. Discrete choices (the selected neuron sets) keep their forward values, and the truncated-SVD bases that compress old legs are held constant.

Objective and data. Only the scored layer enters the loss. For a minibatch of 16 networks N ,

$$\mathcal{L}(\theta) = \frac{1}{16} \sum_N w_N \frac{\text{MSE}_\theta(N)}{\text{MSE}_{\text{ref}}(N)}, \quad \text{MSE}_\theta(N) = \frac{1}{n} \sum_i (\widehat{\mathbb{E}} y_{15,i}(\theta; N) - y_{15,i}^{\text{MC}}(N))^2, \quad (23)$$

where y_{15}^{MC} is the Monte Carlo label and $\text{MSE}_{\text{ref}}(N)$ is the error of the fixed starting weights, so each network contributes its relative error. The 15,384 training networks are networks 0–799 of the public `full` split, with all-layer means from 10^9 samples; 10,488 of the 14,048 networks of a public corpus [11], regenerated from their seeds, with 10^8 -sample final-layer labels; and 4,096 networks we generated, with 10^7 -sample labels. Label noise is about 0.6%, 6% and 70% of the estimator’s MSE in these sources, so the last has $w_N = 1/8$ and the others $w_N = 1$. Public networks 800–999 and the other 3,560 corpus networks are never trained on. The 100 graded networks are not in the released dataset at all: none of their names occurs in its `mini` split, and the one name that also occurs in its `full` split belongs to a different network.

Optimisation. We use Adam with per-group learning rates, constant after a short warm-up, and export exponential moving averages of the weights every 200 updates. The submitted weights are the average of the 0.998-EMAs of two training runs at update 30,600: one continuous run warm-started on 12 October from the previous best weights (about 32 passes over the data), and a copy of it from update 20,232 on with a different data order. Weight sets are selected on 128 held-out corpus networks and confirmed once on fresh ones: the average lowered the MSE by 0.14% against the previous weights (each run alone by 0.10%), confirmed on 512 fresh networks ($z = 6.2$). Swaps of trained weights of this kind account for most of the drop in graded MSE from 1.160×10^{-8} to 1.087×10^{-8} between 7 and 20 October. Appendix B gives the details of training, export and the transfer to the submitted code.

5.3 Why it works here when it failed elsewhere

Other write-ups report difficulties with learning inside the propagation. A per-layer calibration fed back into the propagation made the error 69 times larger [21]. A learned readout re-propagated through the chain was $1.09\text{--}1.60\times$ worse, while a corrector that reads the chain without changing it and is added at the output gave $\times 0.887$ [20]. Fits through a differentiable float32 replica did not generalise [8]. End-to-end tuning of a chain’s constants gave nothing in [13], whose authors predict that a layer-by-layer corrector transported by W would work. A proposition in [5] shows that per-neuron variance errors are random projections $w_i^\top \Delta \bar{C} w_i$ of the previous layer’s covariance error, so a model of a neuron’s moments from its own state can recover only a few percent of a third-order chain’s error. We cannot prove why our heads avoid these failures. Three observations, based on the oracles of Table 4, suggest reasons.

(1) The heads correct the joint state before it is projected. The slice and rescaling heads change $n \times n$ objects before $W_{\ell+1}$ mixes them, and the mean-shift head shifts $\bar{m}$ before $W_{\ell+1}^\top$ projects it; this projected mean error is 77% of the final error. The carry is built from the weight columns, and the mean-shift head’s pair inputs are reductions of the joint state. A correction made upstream of a projection is projected together with the error it cancels, so no head has to predict a random projection. This is consistent with the proposition: the heads barely touch the variance channel (exact variances are worth about as much with the heads as without them), and the main local defect, the effective slice, is beyond per-neuron features (Section 4). The heads earn their value in the κ_3 and κ_4 channels, which they absorb. We think these are closure errors at the births that a learned map can see, but we have not tested this directly.

(2) They are trained on their own trajectory, through the rest of the propagation. A layer-wise fit learns from the uncorrected trajectory and a local target; fed back, it shifts the inputs of every later layer, and its errors compound. Trained end to end, each head sees the state that the corrected upstream estimator produces, and its gradient carries the final error back through all later layers and heads, so it learns the correction with the right downstream effect, which

Table 5: Heads switched off without retraining in the 8 October build (native, VAL32), against the same build with all heads on. Δ MSE is noise-corrected; “off better” counts networks on which the switched-off variant had the lower error.

switched off	Δ MSE	paired z	off better	Δ FLOPs	score ratio
mean-shift head (with its carry)	+20.5%	23.1	0/32	−1.3%	$\times 1.188$
slice head	+10.2%	14.1	0/32	−0.6%	$\times 1.095$
rescaling heads	+8.1%	10.1	0/32	−0.1%	$\times 1.079$
all three	+25.5%	18.4	0/32	−2.2%	$\times 1.227$

Table 6: Main cost levers, in the order they were introduced, with their approximate effect on the graded score when they were introduced. Steps that bundled several changes are split approximately (about ± 2 –3 points).

Lever	Effect when introduced
Port of the reference $K = 3$ propagation, with products shared between consumers and across source ages	149% $\rightarrow$ 78% of the budget
Fast matrix multiplication, first generation (Strassen–Winograd)	score $\approx -40\%$
Four-product consumption (Section 3.5)	FLOPs 41% $\rightarrow$ 28% of the budget; score -30%
Exact restructuring outside the multiplication kernel (reductions, Q carried with T , joint basis solves)	score $\approx -22\%$
Kink-local selection with tails (Section 3.8)	score $\approx -13\%$
Shared rank-256 basis for old legs (Section 3.5)	score $\approx -13\%$
Fast multiplication, second generation (in place, alternative basis, larger top-level schemes)	score $\approx -10\%$

need not be locally correct. In one test we trained an MLP offline to predict the error of the propagated covariance along its leading eigen-directions (held-out $R^2 = 0.54$). Subtracting the predicted error during propagation doubled the final MSE ($\times 2.05$, worse on all 32 networks). Correcting only the part of the true error that the MLP could not predict, an oracle, helped ($\times 0.79$). The predictable part was already compensated elsewhere in the estimator, and correcting it a second time did harm.

(3) Heads and mechanism co-adapt. With the heads frozen, the true κ_4 diagonal and the true κ_{22} make the estimator worse (Table 4), half moves toward the truth beat exact replacement, and removing the learned parts’ propagated contribution in the linearised map costs $\times 1.27$. The two parts form one fitted system, so we compare mechanistic changes refit against refit. The price is that a better mechanism pays only after long retraining (a 200-update refit moves the mean-shift head’s weights by 0.03%), and a correction learned against one version of the propagation can become a liability for another, which may explain some of the failures above. Other conditions probably help: the twin is generated from the source of the native estimator, not re-implemented; the corpus is larger (15,384 networks, against 3,048 in [20] and 200 in [8]); and the heads are bounded and start near the identity. We have not ablated these.

5.4 Measured value

Table 5 switches the heads off, without retraining, in the native estimator of 8 October, on VAL32 (MSE 1.288×10^{-8} at 0.139 of the FLOP budget), against the same build with all heads on. Every head is worth more than 8%. Removing all of them raises the MSE by 25.5% on all 32 networks while saving only 2.2% of the FLOPs. The single costs are sub-additive (sum 38.8%): with the mean-shift head off, the other two together are worth only 4.2%, so the heads are complements, as expected from joint training. A switch-off without retraining measures a head within the co-adapted system; an estimator tuned without heads would recover part of the loss through its constants. In the submitted estimator, after a further week of joint training, single heads cost far more to switch off; Section 7.2 discusses why.

6 Fitting the compute budget

The submitted estimator bills 262 GFLOPs per network on average (0.116–0.123 of 2^{41} over the 100 graded networks). In a cost map of an October version, 82% of the FLOPs went to the third-cumulant sources; their three largest parts were building the slices on the selected rows (31%), the backward chain of the births (27%) and transport (15%). About 10% went to the covariance transport, and 93% of all FLOPs are large matrix products. Three levers bring the cost down. Fast matrix multiplication is exact up to rounding; the restructuring is exact except for one step, and the kink-local selection trades a little accuracy for cost. Table 6 lists their approximate graded effects.

Table 7: Official results on the 100 graded public Phase 2 networks. Wall and residual (unmetered) time are per network: mean / maximum (limits 120 s and 0.4 s). #337505 and #337567 are repeat runs of the same folders. The baselines are the starter kit’s examples.

Submission		Score	MSE	$F/2^{41}$	Wall (s)	Residual (s)
#337501	this paper	1.29915×10^{-9}	1.0867×10^{-8}	0.119	85.8 / 95.3	0.316 / 0.340
#337505	repeat	1.29915×10^{-9}	1.0867×10^{-8}	0.119	86.6 / 98.5	0.318 / 0.340
#337556	second nominated	1.29525×10^{-9}	1.0871×10^{-8}	0.119	89.0 / 102.9	0.324 / 0.345
#337567	repeat	1.29525×10^{-9}	1.0871×10^{-8}	0.119	88.0 / 95.1	0.319 / 0.332
Starter kit: covariance propagation		4.05×10^{-7}	4.05×10^{-6}	0.024		
Starter kit: mean propagation		2.22×10^{-5}	2.22×10^{-4}	0.00004		

Structure. Within the factored form, algebra decides the cost. The four-product consumption needs four products per source instead of six; products that two consumers share are computed once; the path Q is carried together with the leg T ; and the joins of young sources to the shared basis are solved together. These changes are exact up to rounding, except the four-product form, which raised the MSE by 1.6% when it was introduced.

Kink-local selection. Fixed-size sets of neurons near the kink (Section 3.8) keep the cost and the number of calls nearly the same on every network. Cutting the sets further costs 2–3% MSE per 1% of FLOPs saved, so the sizes sit near the score optimum.

Fast matrix multiplication. The large products run through a recursive kernel built from published schemes: Winograd’s variant [22] of Strassen’s algorithm [19], computed in place; the stable alternative-basis form of Schwartz et al. [17] at the deepest levels; larger top-level schemes from PLinOpt [6], from Hopcroft and Kerr [7] with straight-line programs from Perminov [16], and from AlphaEvolve [14] via Sedoglavic’s catalogue [18]; and the structured circuits of Khoruzhii et al. [12] for the covariance step (Appendix B lists them). Every circuit was checked exactly against its tensor. A large product costs 0.45–0.55 of its plain price, and the predicted means of every layer agree with plain multiplication to a relative ℓ_2 difference of 2×10^{-6} , the tolerance we set.

Time limits. Each array call through the metering client costs 17–27 μ s of the 0.4 s residual allowance, and each extra recursion level multiplies the number of tiny leaf products. Kernel routes were therefore priced as FLOPs plus a cost per call and per leaf, and the depth of recursion was chosen per product. On the official run the submission used 85.8 s of wall time on average (95.3 s at most, limit 120 s) and 0.316 s of residual time (0.340 s at most, limit 0.4 s). This engineering earns no accuracy, and other teams used similar schemes [1, 5, 8]; we include it because the estimator of Section 3 would not fit the budget without it.

7 Results

7.1 Graded scores

Table 7 gives the official results. #337501 and #337556 are our two nominated final submissions. They run the same estimator with the same trained weights; #337556 adds further exact FLOP cuts and wall-time engineering and scored 0.3% better. We describe #337501, the safer of the two: over two clean official runs it had the larger timing margins. Everything in this paper applies to both. The grader is deterministic for a given submission: the repeat run of #337501 (#337505) reproduced its score exactly, and only the timings differ between runs. Before nominating, we also ran the exact submitted folder on 1,174 further networks, including extreme ones, to check that it stays finite and within the time limits (Appendix B). Figure 4 shows how the score developed over the eight weeks. Since the first third-cumulant submission on 29 August (2.77×10^{-8}) the score fell 21-fold: 6.5-fold from FLOPs (78% $\rightarrow$ 12% of the budget) and 3.3-fold from MSE ($3.56 \times 10^{-8} \rightarrow 1.09 \times 10^{-8}$).

7.2 Switch-off ledger on the submitted code

Table 8 switches off one component at a time in the exact submitted file, without retraining anything else, and measures the noise-corrected final-layer MSE on 64 public networks (896–959). The switch-off ratio measures how much the submitted estimator *depends* on a component. The graded change when the component was introduced, with a refit of everything else, measures what it was *worth* at the time. The two can differ widely. The covariance blend of the host

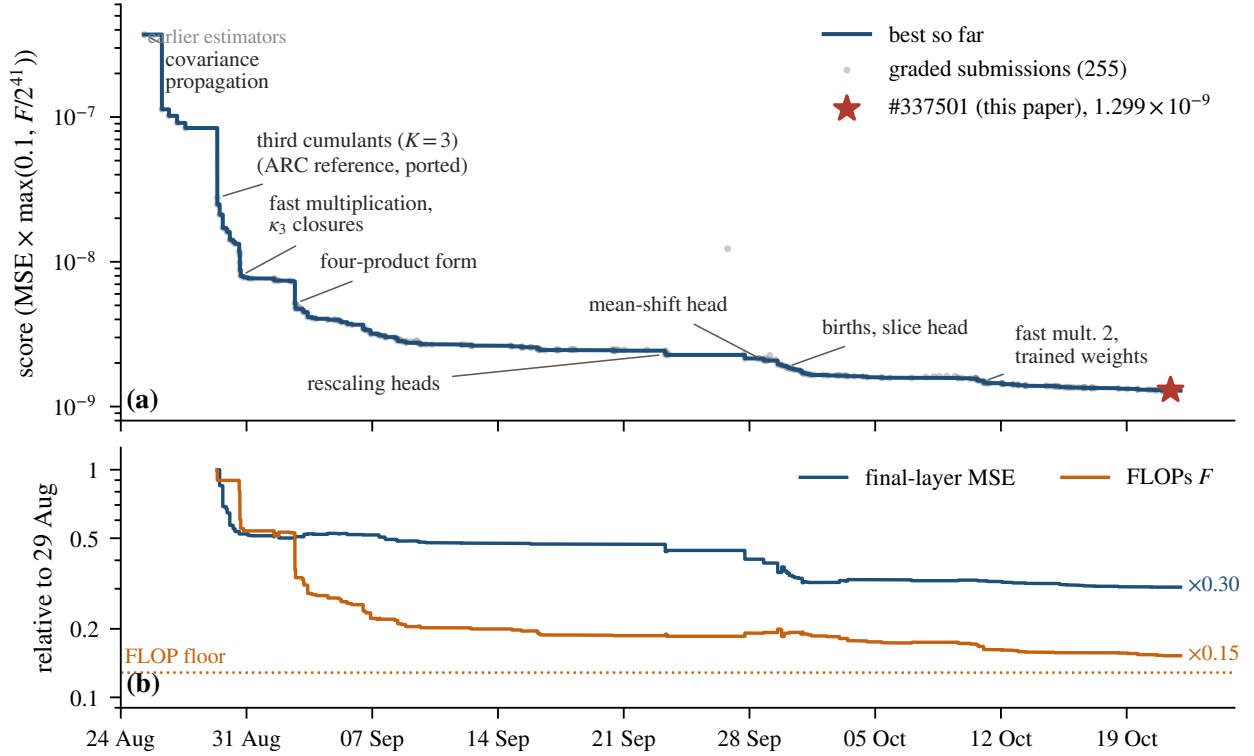


Figure 4: Development of the graded score. (a) All graded submissions of our team (grey) and the best score so far (blue), with the main steps labelled; the submissions before 29 August used earlier estimators that are not described here. (b) Final-layer MSE and billed FLOPs of the best submission so far, relative to the first third-cumulant submission on 29 August (FLOPs then 78% of the budget, so the 10% floor lies at 0.13). The score is their product until the FLOPs reach the floor.

improved the graded score by 19% when it was introduced, before the stars and the heads existed; switching it off now costs only $\times 1.036$, because later parts took over much of its role.

Every switch-off but one raises the MSE on at least 57 of the 64 networks. The exception, the $\kappa_3\kappa_4$ term ($\times 1.001$, worse on 36 networks), could be dropped. The largest analytic term is the readout’s κ_3 correction ($\times 13.3$): even at width 1024 the final layer is skewed enough that a Gaussian readout would be far off. Upstream, switching off the self-cumulant correction multiplies the MSE by 1.448, the stars by 1.237, the births by 1.212 and the exact sphere input by 1.150. The four birth terms cost between 3.8% and 6.2% each, and their costs add up.

For the learned heads the two kinds of number differ most. Switching off all three multiplies the MSE by 1.560, against +25.5% in the 8 October estimator (Table 5). A single head costs far more: the mean-shift head $\times 2.60$, the slice head $\times 11.2$ and the rescaling heads $\times 11.8$. The heads were trained together and partly correct for one another, so switching off one leaves the others unbalanced. Between the two builds, training also moved several analytic constants well away from their earlier values, mostly in the fourth-cumulant channel: the readout’s κ_4 scale from 1.02 to 0.81, the κ_4 factors of the ReLU map from about 1.0 to 0.82, the host’s covariance blend from 0.83 to 0.61 and its pair coefficient from -0.18 to -0.56 ; the global amplitude of the rescaling heads grew from 4.5 to 7.6. The estimator improved, but its parts now expect each other. The all-heads ratio is the closest analogue of the 8 October +25.5%; it is larger, but it also contains the drift of the constants, which a refit without heads would partly undo. Switch-offs of learned parts therefore measure entanglement rather than value. The graded steps that introduced the heads, which multiply to about 0.82, and the 8 October numbers are the better measure of what the heads add.

7.3 What did not work

Besides the attempts on the effective slice (Section 4), these changes failed, for the reasons given:

- **Cheaper legs.** Compressing late-born sources earlier costs $\times 1.23$ MSE for -1.0% FLOPs; pruning newborn leg columns by their vertex costs $\times 3$ – $\times 11$, because the newest leg is the transport operator itself.
- **Exact third cumulants on more rows.** Exact κ_3 diagonals at layers 12–14 would give $\times 0.95$ (top 128 rows) to $\times 0.87$ (all rows) MSE, but computing them costs about 15% more FLOPs.
- **Earlier births** (from layer 3 or 4) help only with a refit: $\times 0.97$ with the heads off but $\times 1.27$ with the heads frozen, because the heads expect no birth inputs there.
- **Deeper fast-multiplication recursion** breaks our accuracy tolerance (1.9 – 2.9×10^{-6} per layer), and extra rank-20

Table 8: Switch-off ledger on the exact submitted code of #337501. Each row switches one component off, by setting its registry parameters to zero before `setup` or by an exact source patch (both listed in Table 9), and changes nothing else; no retraining. MSE ratio: geometric mean over the paired networks of the noise-corrected final-layer MSE against the unmodified estimator (64 networks 896–959 of the public dataset, mean MSE 1.14×10^{-8} ; label noise 0.77×10^{-10} subtracted), with a 95% confidence interval; *Worse*: networks whose MSE rose. ΔF : change of billed FLOPs; zeroing a coefficient usually leaves the computation in place, so this column does not measure what a component costs. Last column: change of the graded score when the component was introduced, for comparison; -18% is the product of the steps that introduced the three heads.

Group	Switched off	MSE ratio	95% CI	Worse	ΔF	Graded, introduced
Learned heads	mean-shift head (R8)	$\times 2.60$	[2.54, 2.67]	64/64	0	-6.3%
	slice head (V2e)	$\times 11.2$	[10.8, 11.6]	64/64	0	-3.6% , refits -2.4%
	slice head skipped entirely	$\times 11.2$	[10.8, 11.6]	64/64	-0.4%	
	rescaling heads (Joint, Geometry)	$\times 11.8$	[11.4, 12.3]	64/64	0	-6.9%
	Joint part only	$\times 5.21$	[4.98, 5.44]	64/64	0	
	Geometry part only	$\times 4.06$	[3.93, 4.19]	64/64	0	
	all three heads	$\times 1.560$	[1.524, 1.597]	64/64	0	-18% (product)
Births	all birth terms	$\times 1.212$	[1.194, 1.231]	64/64	-0.5%	-9.0% (refit)
	(3, 1) births folded into κ_{21}	$\times 1.051$	[1.044, 1.058]	64/64	0	
	(3, 1) star slice folded into κ_{21}	$\times 1.062$	[1.053, 1.070]	63/64	-0.5%	
	path feed into Q	$\times 1.038$	[1.030, 1.045]	59/64	0	-1.2%
	birth diagonal into κ_4	$\times 1.052$	[1.044, 1.061]	61/64	0	
Fourth cumulants	stars (all quartic vertices)	$\times 1.237$	[1.220, 1.254]	64/64	-0.8%	-2.0%
	(2,2), (3,1) vertex weights	$\times 1.108$	[1.096, 1.120]	64/64	-0.6%	-5.0% , -2.1%
	covariance blend of the host	$\times 1.036$	[1.030, 1.043]	57/64	0	-19%
ReLU moments	self-cumulant correction (a_3, a_4)	$\times 1.448$	[1.426, 1.470]	64/64	0	-12%
	next-order pair terms	$\times 1.037$	[1.031, 1.044]	60/64	0	
	$\kappa_3 \kappa_4$ term (s_{34})	$\times 1.001$	[1.000, 1.002]	36/64	0	
Input, readout	sphere input (Gaussian instead)	$\times 1.150$	[1.128, 1.173]	63/64	0	-2.8%
	readout κ_3 term	$\times 13.3$	[12.9, 13.8]	64/64	0	
	readout κ_4 term	$\times 1.163$	[1.151, 1.174]	64/64	0	

levels saved 0.67% of FLOPs for 7.8% more wall time.

- **Trusting the validation panel.** More training data alone did not improve the heads, and the 32-network validation panel overstated some gains: weight sets gained less on 488 fresh networks, and a late-layer tail setting that gained 0.25% on the panel gained nothing on 223 fresh networks.

8 How the solution was developed

An AI-agent research team. The work was done by one person directing a team of AI agents over eight weeks (25 August to 21 October 2026). A Claude Code session acted as the “driver”: it planned experiments, wrote task packets, verified and merged the agents’ deliverables, ran the pre-submission checks and submitted. Further Claude Code sessions audited the driver’s claims against the raw records. Up to fifteen headless OpenAI Codex agents worked in parallel on separate research lanes, each from a written packet that stated the motivation, a numeric target and a time box. Each agent worked in its own snapshot of the code and delivered a patch that the driver checked before merging. Most of the specific ideas, including the births, the effective-slice fold and the designs of the learned heads, were proposed by the agents. The author directed the research, including which kinds of learning methods to try, set the rules for evidence and risk, and decided which submissions to make and nominate.

Evidence rules. With hundreds of small changes, most of them worth well under one percent, the main risk was fooling ourselves. We used these rules, in addition to the paired statistics of Section 4.1:

- One change per experiment, each with a short record of its question, change, result and verdict.
- Selection on the fixed validation panel; a separate test panel only for tracking progress; no confirmed result quoted from fewer than 32 held-out networks.
- One-time panels of fresh networks, read once for a single decision. They caught changes that looked good on the validation panel but did not hold up (Section 7.3).
- Decision rules written into the log before the result was read, for example the rules for moving the final submission and for choosing the two nominated submissions.

- Checks before every submission: exact determinism, FLOP, call and memory limits, a 37-case fault-robustness suite, and a timing census on 240 networks.

Over the competition we ran more than 500 recorded experiments and made about 290 submissions, 255 of which were graded.

9 Limitations

- **The remaining error is diffuse.** We know where the error enters (the effective slice at layers 6–14) and that the needed correction is nearly orthogonal to every single term we could compute: it accumulates from many small per-layer errors. We found no affordable way to compute it.
- **Fitted parts.** Besides the learned heads, the analytic closures contain 138 trained values, several far from their derived values (Table 2). The heads and constants were trained on networks from the challenge’s generator at width 1024 and depth 16; other widths, depths or weight distributions would need retraining, and we have not tested the method at other sizes. Because the parts were trained together, they depend on each other (Section 7.2), which makes the attribution of accuracy to components approximate.
- **Older analyses.** The error map and the oracles (Section 4) were measured on the 9–10 October version of the estimator, which has the same architecture but older weights and fewer cost cuts; the switch-off ledger was measured on the submitted code.
- **Out-of-distribution accuracy.** We tested fresh and extreme networks only for finiteness, cost and time (Appendix B), not for accuracy.
- **Engineering specific to the metric.** Neuron-set sizes, kernel routes and call budgets were tuned for 1024×16 networks, for `flopscope` 0.12.3’s prices and for the grader’s timing limits. The residual-time margin (0.34 s of 0.4 s) depends on the grader’s hardware.
- **Statistics.** The ledger uses 64 networks and the validation panel 32. Effects under about 0.3% are below what these panels resolve; smaller effects were confirmed only on panels of 500 or more networks.

10 Conclusion

Cumulant propagation, as proposed by ARC, works at the scale of this challenge once its third-order part is factored, compressed and restricted to the neurons near their kink. Two further ingredients distinguish our estimator. The first is to model how ReLUs turn existing skew into new skew (births) and to fold the resulting $(3, 1)$ slice into the effective $(2, 1)$ slice that drives the next covariance. The second is to let small learned heads correct the propagation inside the network, trained end to end through a differentiable twin of the whole estimator together with the analytic constants. With these, the estimator reaches a final-layer MSE of 1.09×10^{-8} at 12% of the budget, and the error that remains is in the readout’s inputs, not in the readout.

That error is mean error accumulated in the middle and late layers, and its local root is the effective slice. Moving that slice, together with the κ_3 and κ_4 diagonals, halfway to the truth at layers 6–14 would roughly halve the MSE. Our analyses suggest that the missing part is spread over many small per-layer errors rather than one missing term. We think the most promising directions are learned corrections that act on pair-level rather than per-neuron objects, a cheaper carried representation of the fourth cumulants’ contribution to the slices, and, further out, a fourth-order propagation that fits the budget.

Acknowledgements

We thank the Alignment Research Center and AICrowd for organising the challenge, the authors of ARC’s reference implementation, and the participants whose public write-ups we cite.

Use of AI tools

AI tools were used throughout this project, and most of the code was written by them.

- **OpenAI Codex agents** (GPT models, at maximum reasoning effort) worked on research lanes: new estimator components, cost reductions, the learned heads and their training code, analyses such as the error map, and test tooling.
- **ChatGPT** prepared the Python implementation of some fast-multiplication circuits (the PLinOpt rank-48/49 and the alternative-basis rank-7 circuits) from the published schemes.

- **Anthropic Claude Code** (Claude models) ran as the orchestrating “driver” session, as auditing sessions, and as cloud agents that proposed cost-cutting patches. It also wrote parts of the estimator and most of the evaluation and submission tooling.
- **Google Gemini** was used for reviews and independent analyses.

Most of the specific ideas, including the births, the effective-slice fold and the designs of the learned heads, were proposed by these agents. The author directed the research, including which kinds of learning methods to try, set the acceptance tests and the rules for evidence and risk, reviewed the results and made every submission and nomination decision. This paper was drafted by Claude Code (Claude Opus) from the project’s code and records, including the switch-off experiments run for it, and was reviewed and edited by the author. All numbers are taken from grader receipts, experiment records or the runs described here, and interpretations that we have not tested are marked as such.

References

- [1] 504aldo. Everything we tried: a factorized $K=3$ cumulant-propagation estimator at $0.25 \times B$, where its FLOPs go, and 25 measured dead ends (team 504aldo, rank ~ 10). AICrowd forum, topic 18218, <https://discourse.aicrowd.com/t/18218>, 2026.
- [2] abhimanyu_singh. [Phase 2 write-up] A bias/variance error budget for factored $K=3$ cumulant propagation: what the slices can tolerate, measured (submission #337475). AICrowd forum, topic 18261, <https://discourse.aicrowd.com/t/18261>, 2026.
- [3] Alignment Research Center. mlp_cumulant_propagation: reference implementation of cumulant propagation. https://github.com/alignment-research-center/mlp_cumulant_propagation, 2026.
- [4] Alignment Research Center and AICrowd. ARC white-box estimation challenge 2026. <https://www.aicrowd.com/challenges/arc-white-box-estimation-challenge-2026>, 2026.
- [5] Artyom_Mazur. [Phase 2 write-up] A third-cumulant chain at its boundary: random projections, a skew law, and where the next order’s accuracy lives (submission #333534). AICrowd forum, topic 18249, <https://discourse.aicrowd.com/t/18249>, 2026.
- [6] Jean-Guillaume Dumas, Bruno Grenet, Clément Pernet, and Alexandre Sedoglavic. PLinOpt (software). <https://github.com/jgdumas/plinopt>.
- [7] John E. Hopcroft and Leslie R. Kerr. On minimizing the number of multiplications necessary for matrix multiplication. *SIAM Journal on Applied Mathematics*, 20(1):30–36, 1971.
- [8] hyojun_kwon. [Phase 2 write-up] Cumulant propagation under a compute budget: state-matched corrections, verified fast arithmetic, and a learned terminal corrector (submission #335413). AICrowd forum, topic 18255, <https://discourse.aicrowd.com/t/18255>, 2026.
- [9] il.miscusi. [Phase 2 write-up] Removing the global scale mode makes a carried κ_4 core pay: extraction on/off $\times 0.380$ on the grader inside e30_a’s engine (submission #336317). AICrowd forum, topic 18248, <https://discourse.aicrowd.com/t/18248>, 2026.
- [10] kai_leal_gonzalez. [Phase 2 write-up] The $(3,1)$ slice of κ_4 : where V29 loses accuracy, and a two-term closure confirmed in a pre-registered test on 77 held-out networks (submission #334821). AICrowd forum, topic 18246, <https://discourse.aicrowd.com/t/18246>, 2026.
- [11] keenanpepper. arc-whestbench-p2-d8b-corpus-14k. Hugging Face dataset, <https://huggingface.co/datasets/keenanpepper/arc-whestbench-p2-d8b-corpus-14k>, 2026.
- [12] Kirill Khoruzhii, Patrick Gelß, and Sebastian Pokutta. Faster algorithms for structured matrix multiplication via flip graph search. *arXiv preprint arXiv:2511.10786*, 2025.
- [13] LordProcciuto. Everything we measured and did not get to work: a census of public chains, 111 closed avenues with their conditions, the levers we could not use, and open questions. Write-up of submission e30_a, <https://github.com/siming767-creator/whestbench-lordprocciuto>, 2026.
- [14] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

- [15] Nurlanggs. [Phase 2 write-up] Where the per-source leg representation of factorized K=3 ends (adj 3.8e-9), and the measurements behind it (submission #336225). AICrowd forum, topic 18252, <https://discourse.aicrowd.com/t/18252>, 2026.
- [16] Andrew Perminov. FastMatrixMultiplication (software). <https://github.com/dronperminov/FastMatrixMultiplication>, 2025.
- [17] Oded Schwartz, Sivan Toledo, Noa Vaknin, and Gal Wiernik. Alternative basis matrix multiplication is fast and stable. *Numerische Mathematik*, 2026. doi: 10.1007/s00211-026-01531-9.
- [18] Alexandre Sedoglavic. Yet another catalogue of fast matrix multiplication algorithms. <https://fmm.univ-lille.fr/>.
- [19] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.
- [20] Susupro. [Phase 2 write-up] Zero-Sample K=4 via Factored n^3 -Memory Transport at 1024 x 16: 11.8x below augmented K=3, why memoryless closures stall, #1 write-up (#336537). AICrowd forum, topic 18254, <https://discourse.aicrowd.com/t/18254>, 2026.
- [21] venture_agent_ai. Phase 2 write-up: exact bivariate ReLU covariance plus per-layer calibration, 3x over the kit baseline, with negative results (submission 329951). AICrowd forum, topic 18214, <https://discourse.aicrowd.com/t/18214>, 2026.
- [22] Shmuel Winograd. On multiplication of 2×2 matrices. *Linear Algebra and its Applications*, 4(4):381–388, 1971.
- [23] Wilson Wu, Victor Lecomte, Michael Winer, George Robinson, Jacob Hilton, and Paul Christiano. Estimating the expected output of wide random MLPs more efficiently than sampling. *arXiv preprint arXiv:2605.05179*, 2026.

A Reproducibility

Code. The submitted folder of #337501 (one generated file, `estimator.py`, with about 30,600 lines, plus 3.5 MB of learned weights and lookup tables in `.npz` files) will be released, together with the training code, the Torch twin and the switch-off scripts, with its licences and third-party notices at <https://github.com/marbiner/arc-whitebox-challenge-2026-submission-marius-binner>. Its SHA-256 over the folder is `00bc59df...8cc15eda`. The second nominated submission, #337556 (score 1.29525×10^{-9} , MSE 1.0871×10^{-8}), adds further exact FLOP cuts and wall-time engineering to the same estimator.

Environment. Python 3.10, numpy 2.2.6, flopscope 0.12.3 and whetbench 0.16.1, as on the grader. One network takes 1.5–3 minutes on one CPU core. A public network can be scored with

```
whest run --estimator submission-337501/estimator.py \
  --no-residual-wall-time-limit \
  --dataset hf://aicrowd/arc-whetbench-public-2026 --revision v2-phase2 \
  --split mini --streaming --n-mlps 4
```

(the residual-time limit is disabled because a local in-process run on a slower or busy machine exceeds the grader’s 0.4 s).

Switch-off experiments. Table 8 was produced with `ablation/ablate.py` from the code repository. It loads the submitted file in-process, switches one component off by overriding registry defaults before `setup` or by an exact source patch (Table 9 lists them), and predicts networks 896–959 of the public `full` split, whose labels come from 10^9 samples; `ablation/summarize.py` computes the paired ratios. MSEs are noise-corrected by subtracting the label-noise term supplied with the dataset; FLOPs are `flopscope`’s billed count.

B Implementation details

Kink-local selection. Table 10 lists the sizes of the neuron sets of Section 3.8. From layer 5 the neurons are ordered by $\Phi(\alpha)$ for the cumulant work and the leg transport, and by $|\alpha|$ for the $(2, 1)$ slice. At layers 5–14, rows with $\alpha < -2.5$, and trailing blocks of 32 rows that hold under 1% of $\sum_i \varphi(\alpha_i)/\sigma_i$, are skipped; vertices with $\alpha > 2.5$ drop their path columns. Skipped slice columns are replaced by their mean. The tails of skipped rows are averaged over the last 64 computed rows, at the last layer separately for $m > 0$ and $m \leq 0$.

Table 9: Switches used for Table 8: registry parameters of the submitted file set to zero before `setup`, or exact source patches. Variant: the name used by `ablation/ablate.py`.

Switched off	Variant	Parameters set to zero
mean-shift head (R8)	<code>r8_off</code>	<code>r8_output_scale</code>
slice head (V2e)	<code>v2e_off</code>	<code>k21_slot_v2e_lambda</code>
slice head skipped entirely	<code>v2e_slot_off</code>	<code>k21_slot</code>
rescaling heads (Joint, Geometry)	<code>joint_geometry_off</code>	joint_amplitude and the Geometry output scale (patch)
Joint part only	<code>joint_off</code>	joint_amplitude
Geometry part only	<code>geometry_off</code>	output scale <code>dscale</code> of the Geometry heads (patch)
all three heads	<code>all_learned_off</code>	<code>r8_output_scale</code> , <code>k21_slot_v2e_lambda</code> , joint_amplitude, Geometry output scale (patch)
all birth terms	<code>births_all_off</code>	<code>h12_fold_gain</code> , <code>feed_fold</code> , <code>h12_beta/gamma_layers</code> , <code>fe30</code> , <code>exact_k31_betas</code>
(3, 1) births folded into κ_{21}	<code>h12_fold_off</code>	<code>h12_fold_gain</code>
(3, 1) star slice folded into κ_{21}	<code>k31_fold_off</code>	<code>exact_k31_betas</code>
path feed into Q	<code>feed_off</code>	<code>feed_fold</code>
birth diagonal into κ_4	<code>h12_off</code>	<code>h12_beta/gamma</code>
stars (all quartic vertices)	<code>k4_star_off</code>	<code>k4_star_scale</code>
(2,2), (3,1) vertex weights	<code>k4_vertices_off</code>	<code>k4_d22c/d31c_scale</code>
covariance blend of the host	<code>k4_host_cov_off</code>	<code>k4_cov_weight</code>
self-cumulant correction (a_3, a_4)	<code>dress_off</code>	<code>k3/k4_vertex_dress</code>
next-order pair terms	<code>pair_terms_off</code>	<code>cov_k3cross</code> , <code>cov_k3sq</code> , <code>split_pair_c3_cov</code>
$\kappa_3\kappa_4$ term (s_{34})	<code>edgeworth34_off</code>	<code>edgeworth_34</code>
sphere input (Gaussian instead)	<code>sphere_off</code>	radius factor $\rightarrow 1$ and input κ_4 core $\rightarrow 0$ (patch)
readout κ_3 term	<code>term_k3_off</code>	<code>final_k3_scale</code>
readout κ_4 term	<code>term_k4_off</code>	<code>final_k4_scale</code>

Table 10: Sizes of the fixed neuron sets of the kink-local selection.

Set	Layers	Size
rows for the cumulant work (by $\Phi(\alpha)$)	7–9 / 10–12 / 13–14	960 / 928 / 896
slice columns	7–9 / 10–12 / 13–14	832 / 768 / 704
rows whose legs are transported	7–9 / 10–12 / 13–14	896 / 864 / 832
rows of the (2, 1) slice (smallest $ \alpha $)	4–5 / 6–7 / 8–9 / 10–11 / 12 / 13–14	960 / 896 / 768 / 704 / 640 / 576
κ_3 and stars at the last layer (smallest $ \alpha $)	15	640
births at the last layer (largest $ \psi_4 $)	15	128
covariance rows with correlations	5–14	$\alpha > -3$, at least 512

Shared basis. The basis B of Section 3.5 is computed once, when the first source reaches age 7: the 255 leading left singular vectors of $T \text{diag}(\omega)$, with $\omega_j^2 = 1 + \|P_j\|^2 / \text{mean}_k \|P_k\|^2$ favouring the columns that feed Q , plus the normalised part of $T\mathbf{1}$ that they miss, so that $T\mathbf{1}$ is kept exactly. In the submitted build this is a single truncated SVD on the selected rows. The coordinates are $U_T = T^\top B$ and $U_Q = P^\top U_T$. Sources that reach age 7 at layers 8–12 join by a ridge solve for 255 coordinates (ridge 10^{-6} times the mean Gram diagonal); the remaining column is replaced by the new leg’s unexplained mean column, and the older sources are re-expressed in the updated basis. Later sources stay dense. The rank-2 remainder R of a new source (Section 3.5) comes from a randomised range finder with four power iterations.

Training. Adam with per-group learning rates (4.1×10^{-5} for the mean-shift head, 1.4×10^{-4} for the slice head, 1.4×10^{-5} for the rescaling heads, 1.4×10^{-3} relative for the analytic constants), constant after a 50-update warm-up, with 16 networks per update. Exponential moving averages of the weights (decays 0.995–0.9998) are exported every 200 updates. The submitted weights average the 0.998-averages of two runs at update 30,600; the second run was forked from the first at update 20,232 with a different data order. In selection on 128 corpus networks the average lowered the MSE by 0.142% (each run alone by 0.096%); in a one-time confirmation on 512 fresh networks by 0.140% ($z = 6.2$, better on 327).

Export and transfer. The exporter writes the chosen weights, with the native float64-to-float32 set-up rounding, to one file per group plus the scalar constants, all checked against the parameter registry. The native set-up folds them linearly into per-layer matrices, so averaging exported arrays averages the weights. The submitted code is a later, FLOP-reduced version of the training source; trained weights move onto it by a weights-only swap that leaves the

FLOPs unchanged, are re-measured natively against the current best and pass the usual pre-submission checks. After training, the carry’s transports and memory path were compressed to factors of rank 8, 2, 2 and 8, fitted to the full carry’s outputs and refitted for each weight set (0.5% fewer FLOPs, under 0.1% more MSE).

Fast multiplication schemes. Each binary level of the recursive kernel uses Winograd’s variant [22] of Strassen’s algorithm [19], computed in place in a blocked buffer; the deepest three levels use the stable alternative-basis form of Schwartz et al. [17], which needs 3+3 encoding and 6 decoding additions instead of 4+4+7. Wide and long-inner products start with larger top-level schemes: rank-40 $\langle 3, 3, 6 \rangle$ and rank-48/49 circuits from PLinOpt [6], the rank-20 $\langle 2, 3, 4 \rangle$ scheme of Hopcroft and Kerr [7] with straight-line programs from Perminov [16], further schemes from the same repository (for example $\langle 4, 4, 10 \rangle$ of rank 115), and the rank-54 $\langle 3, 4, 6 \rangle$ scheme found by AlphaEvolve [14], taken from Sedoglavic’s catalogue [18]. The triangular and Gram products of the covariance step use the structured circuits of Khoruzhii et al. [12].

Robustness of the final submission. A network that runs over a time limit scores as if every prediction were zero, which would dominate the average. Before nominating #337501 we ran the exact submitted folder in the grader’s software environment on 400 training networks (all finite, no fallbacks, client memory under 0.8 GB), on 756 freshly generated networks (all finite, no fallbacks), and on 18 networks picked from 2,800 generator seeds, 16 of them extreme in their output mean, fraction of dead neurons, spectral gain or largest activation (up to 5.7 standard deviations from typical). An extreme-value model fitted to the per-network timing tails of the two official runs puts the probability that any of 1,000 private networks exceeds a time limit at about 10^{-4} , and at a few percent under the most pessimistic tail model we tried. The run time would have to grow by 17.5% before the residual-time limit is reached. These tests check finiteness, cost and time, not accuracy.