Solving cumulant propagation's depth problem with scale separation

A fit-free estimator that beats sampling at every width tested when depth grows with width

Pranav Viswanath

Code: https://github.com/PranavViswanath/scale-separation

The result

ARC's mechanistic estimation paper [1] leaves one regime open: matching random sampling [3] when depth grows in proportion to width. This note answers it on ARC's own criterion, with ARC's own code.

1. **A fit-free deterministic estimator beats naive sampling at every width I tested on the depth = width/4 diagonal, from 64x16 to 1024x256:** 2.3x, 1.6x, 1.5x, 1.4x and 1.2x at equal FLOPs, 10 random nets per cell. It costs about what ARC's K2 costs, and it is 1.3x to 4.1x more accurate than ARC's best estimate at my compute, in cells where ARC's ladder inverts or diverges.
2. Why ARC's ladder breaks there: one object, the collective scale mode, which is the random overall size of each layer's activations. Removing it from the state ARC's K4 propagates cuts K4's error 42x and makes the ladder improve with every order again.
3. The fix has two parts. ReLU's positive homogeneity factors the scale out of every vertex exactly. The scale's variance then has to be carried with its mean-coupled terms: the leading-order rule shrinks toward zero at depth, and with it my margin over sampling fell below 1x at 128x32 and 256x64. With the exact recursion the estimator stays ahead of sampling at every width, through 1024x256.
4. ARC's own ladder converges again once the scale is carried this way: its K4 is 37x better than ARC's K4 at 64x16, and its K3 is about 5x better than the best ARC's ladder reaches at 128x32 and 256x64.
5. At the challenge's own shape, 1024x16, the separation lowers ARC's factored K3 error 1.25x (SIMPLE) and 1.44x (AUGMENT) on every net tested, and it improves my graded entry's leaderboard score 1.73x (Section 7).

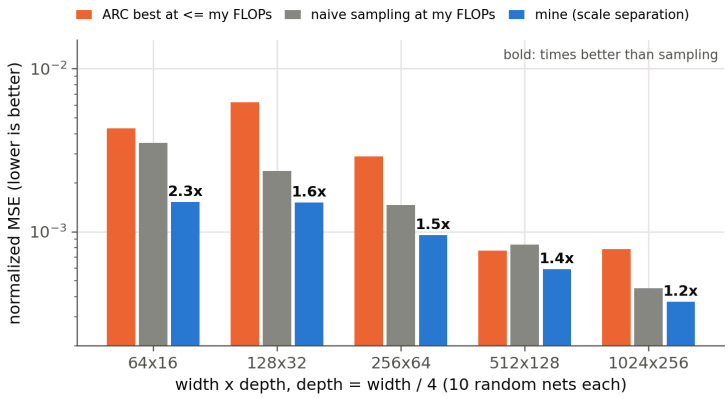


Figure 1. Depth = width/4, 10 random nets per cell. For each cell: ARC's best estimate at or below my FLOPs, naive sampling at my FLOPs, and mine. Bold: how many times lower my error is than sampling's.

Everything below uses ARC's cumulant propagation [4] as implemented in its repository (commit 93d091a, unmodified) [2], its FLOP counter, many random nets per cell, and predictions written down with timestamps before the numbers were computed.

1. ARC's ladder when depth grows with width

I ran ARC's SIMPLE and AUGMENT ladders, K1 to K4, with depth = width/4:

- 64x16: K4 is worse than K3 on 10 of 10 nets (1.0e-2 vs 3.5e-3). AUGMENT K4 diverges.
- 128x32: K4 is worse than K3 on 9 of 10 nets (0.15 vs 4.9e-3). AUGMENT inverts from K2 to K3 on 7 of 10 nets and K3 diverges.
- 256x64: K3 is about the same as K2, K4 diverges on 5 of 5 nets, and AUGMENT K2 and K3 diverge. The usable level, K2, is worse than naive sampling at its own cost (0.86x).

I also reproduce the paper's 256x8 observation: K4 there is slightly worse than sampling (0.94x).

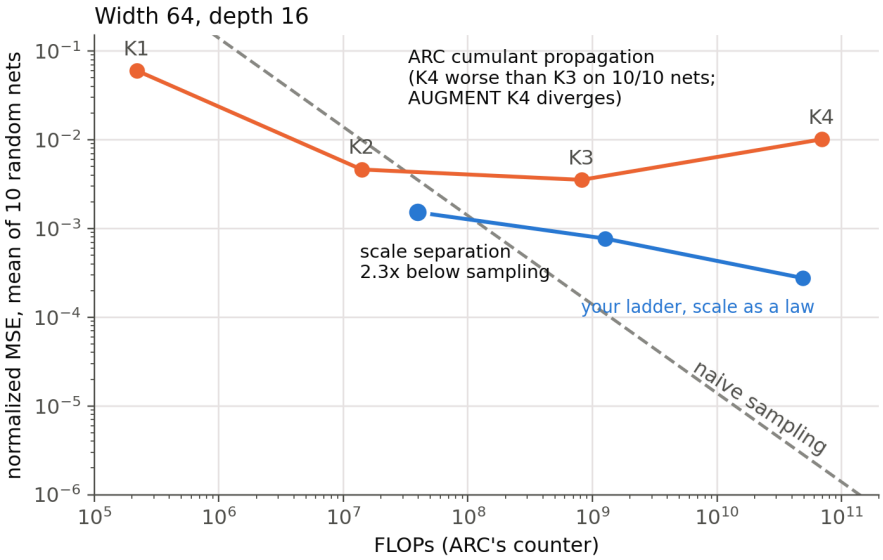


Figure 2. Width 64, depth 16, mean over 10 random nets. Orange: ARC's ladder. Grey: naive sampling at the same FLOPs. Blue: my estimator (left point) and ARC's ladder run with the scale carried as a law.

2. Why: the collective scale mode

At finite width the squared norm of a layer, $\|h\|^2/n$, fluctuates from input to input, and the fluctuation grows with depth over width [5, 6, 7]. It is one collective coordinate shared by every neuron. A truncated cumulant list carries it as large, correlated fourth and higher cumulants across every pair of neurons, and truncating those is what makes higher orders hurt.

The test is direct. At 64x16 I de-scale the state ARC's dense chain propagates at every layer (factor out the scale, propagate the shape, nothing fitted):

	ARC's ladder	with the scale removed
K3	3.5e-3	7.9e-4
K4	1.0e-2	2.4e-4 (**42x**)

The inversion disappears and the ladder becomes monotone.

3. The fix: scale separation

Model each pre-activation layer as

$$z = \sqrt{S}\,(m + g), \qquad g \sim \mathcal{N}(0, C), \qquad S \sim \text{Gamma}(\text{mean } 1,\ \text{variance } V).$$

ReLU is positively homogeneous, so $\text{ReLU}(\sqrt{S}\,y) = \sqrt{S}\,\text{ReLU}(y)$, and every moment factorizes exactly:

$$\mathbb{E}\big[h_j^a\, h_l^b\big] \;=\; \mathbb{E}\big[S^{(a+b)/2}\big] \;\cdot\; \mathbb{E}_{y\sim\mathcal{N}(m,C)}\big[\text{ReLU}(y_j)^a\,\text{ReLU}(y_l)^b\big].$$

The vertex is one Gaussian vertex times closed-form Gamma moments, with no quadrature over the scale. The state per layer is the mean, the covariance and the scalar V, and m and C are exact functions of the carried mean and covariance.

The part that decides the deep regime is how V moves from layer to layer. Averaging the norm statistic of the next layer over the fresh weights gives an exact equation in quantities the law already knows. With $a = \mathbb{E}\,h$, $K = \text{Cov}\,h$ and $s_1(V) = \mathbb{E}\sqrt{S}$:

$$(1 + V')\left[N_0 - \|a\|^4\,(4b + 2b^2)\right] = \mathbb{E}\,\|h\|^4, \qquad b = \frac{1}{s_1(V')^2} - 1,$$

$$N_0 = \left(\|a\|^2 + \text{tr}\,K\right)^2 + 2\,\text{tr}\,K^2 + 4\,a^\top K a.$$

$\mathbb{E}\,\|h\|^4$ comes from the law's own pair moments. At leading order this reads

$$V' \approx \frac{\sum_{j,l}\kappa_4(h)_{jjll} + 4\,a\cdot\beta}{(\text{tr}\,K)^2 + 2\,\text{tr}\,K^2 + 2\|a\|^2\,\text{tr}\,K + 4\,a^\top K a},$$

where β is the trace of $\kappa_3(h)$. The simpler rule without the mean-coupled terms underestimates the scale variance at depth by 0.10 to 0.19 and drifts toward zero as width grows. The exact recursion tracks the true value to within 0.01 on 7 of 9 development nets. Nothing is fitted to truth, and the cost is the covariance propagation ARC's K2 already does plus a pair vertex.

4. Results on the diagonal

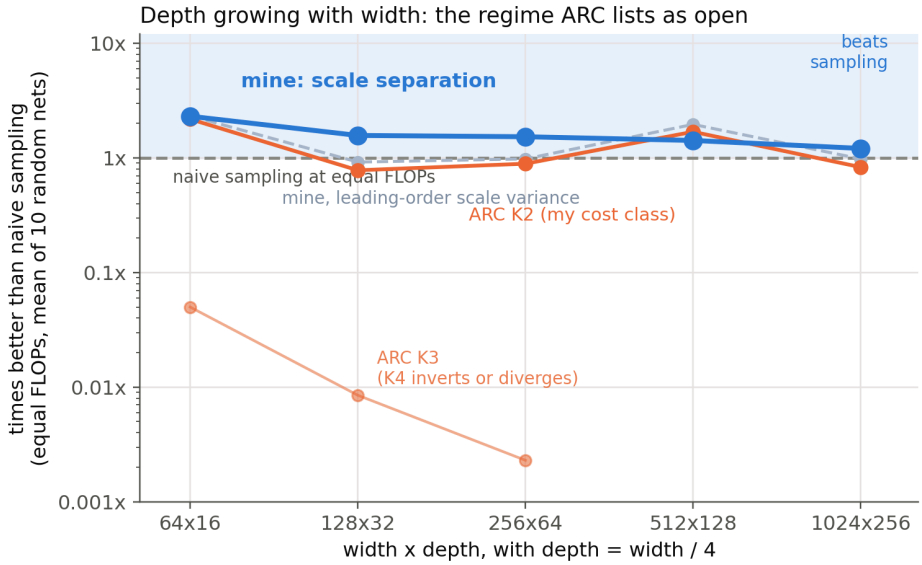


Figure 3. Depth = width/4, 10 random nets per cell, every method at its own FLOPs. Above the dashed line a method beats naive sampling. Blue: my estimator. Dashed grey: the same estimator with the leading-order scale variance. Orange: ARC's K2 (the level at my cost) and ARC's K3 (not run at 512 and 1024); ARC's K4 inverts or diverges in every cell where it runs.

width x depth	vs naive sampling	vs ARC's best at equal or lower FLOPs
64x16	**2.31x**	2.8x
128x32	1.57x	4.1x
256x64	1.53x	3.1x
512x128	1.42x	1.3x
1024x256	1.21x	2.1x

At 64x16 the estimator also beats antithetic sampling and radial-exact sampling. At 64x16, 128x32 and 256x64 my estimator is more accurate than every level of ARC's ladder, at any compute. At 512x128 ARC's K2 also beats sampling (1.69x at its own, smaller, FLOPs), and my estimator is 1.3x more accurate than it. At 1024x256 ARC's K2 falls below sampling (0.83x) and AUGMENT K2 diverges, while my estimator stays ahead of sampling (1.21x) and is 2.1x more accurate than ARC's best.

One numerical detail matters at depth. Deep layers carry strongly correlated neuron pairs (correlations up to 0.98) and large normalized means, and the pair vertex's quadrature has to resolve them. I set it by a convergence rule on development nets against a fine reference, with no truth involved: 24 points, at 1.05x the cost of the coarser setting, converged to within 4e-7 at both 512x128 and 1024x256. A setting tuned at 64x16 alone under-resolves the deepest layers of the larger networks.

5. ARC's ladder converges again

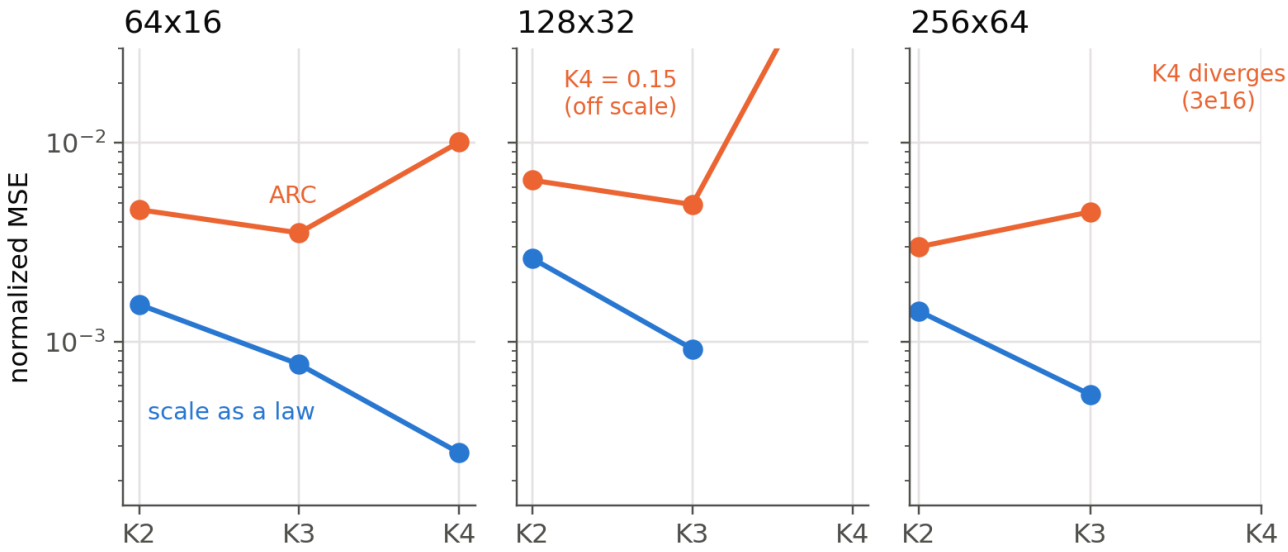


Figure 4. Orange: ARC's SIMPLE ladder. Blue: the same ladder with the scale carried as an exact law at every layer (ARC's code on the de-scaled shape; level 2 is my vertex). Setting V = 0 reproduces ARC's K3 exactly.

cell	best of ARC's ladder	ladder with the scale law
64x16	3.5e-3 (K3)	2.8e-4 (K4), 12.8x better; **37x** better than ARC's K4
128x32	4.9e-3 (K3)	9.2e-4 (K3), 5.3x better
256x64	3.0e-3 (K2)	5.4e-4 (K3), 5.5x better

The higher levels run on ARC's dense or factored chain, which is expensive at these widths, so they sit above the sampling line at their FLOPs. They show the mechanism: with the scale handled exactly, ARC's method converges where it currently diverges.

6. Where this leaves the deep regime

Across five widths, each double the last, the margin over sampling goes 2.3x, 1.6x, 1.5x, 1.4x, 1.2x: the estimator stays ahead of sampling all the way to 1024x256, at a cost of order ARC's K2, while ARC's ladder inverts or diverges above K2. Reaching every tolerance at a fixed width needs a level above this one that keeps the scale exact, and Section 5 shows ARC's own ladder converging once it does.

7. The challenge cell, 1024x16

The challenge itself is fixed at width 1024 and depth 16. I ran the same separation there, inside ARC's unmodified factored K3, on the challenge's own networks (Phase 2 public dataset, mini split) against their 1e9-sample truth.

ARC factored K3 at 1024x16	without scale separation	with scale separation	gain	nets improved
SIMPLE, final-layer MSE	3.89e-8	3.11e-8	1.25x	12/12
AUGMENT, final-layer MSE	1.93e-8	1.34e-8	1.44x	6/6

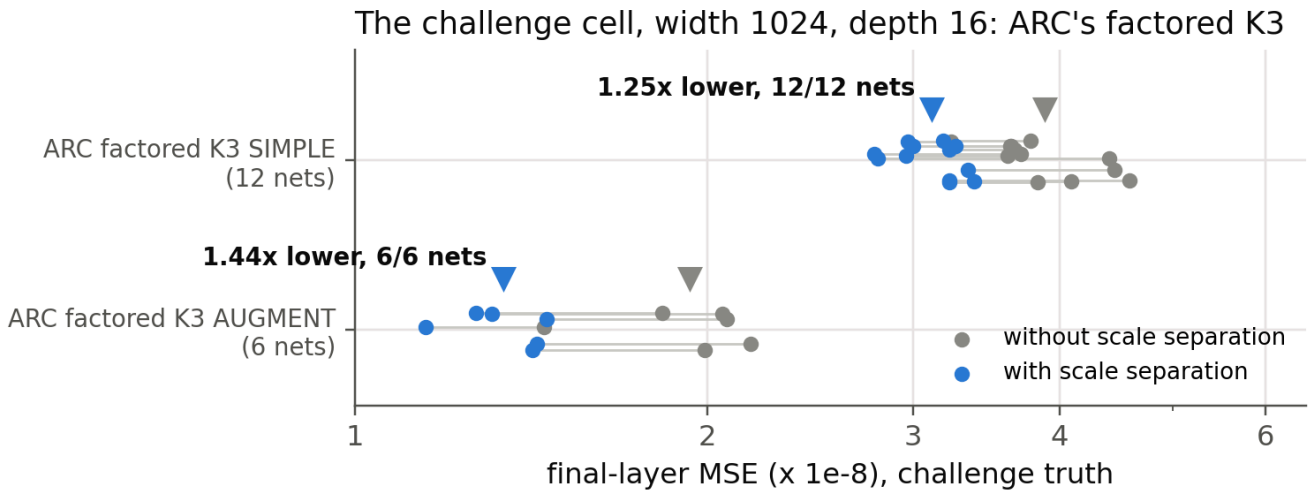


Figure 5. The challenge cell. Each pair of dots is one network, without (gray) and with (blue) scale separation; triangles are means.

In my graded entry (336823), the scale separation improves the leaderboard score 1.73x (1.17e-8 to 6.79e-9, 12/12 nets, same compute).

These measurements were made on 29 October 2026, after this write-up was first posted; scripts are in `challenge_1024x16/`.

Method

- Networks: ARC's MLP class, He initialization, no bias, fixed seeds. Development nets (separate seeds) were used only to freeze numerics and check the variance recursion, and are excluded from every score.
- Cells: the diagonal 64x16, 128x32, 256x64, 512x128 and 1024x256 (10 nets each); also 128x16, 256x8 (10 each) and a wide control at 256x4 (5). The challenge cell 1024x16 uses the challenge's own nets and truth (Section 7).
- Truth: Monte Carlo in two independent halves. Error is the cross-half product, so truth noise cancels, normalized by the output variance and averaged over nets with bootstrap intervals. The label noise floor is at least 180x below every error reported. Per-net results and intervals are in the repository.
- Baselines: ARC's repository unmodified, in float64: SIMPLE and AUGMENT, K1 to K4, factored for K >= 3. "ARC's best" at a FLOP level is the best estimate from any of those variants at that FLOP level or below.

- Compute: ARC's NamedFlopCounter for every method, plus a hand count of special-function evaluations. Naive sampling costs $2n^2(L+1)$ FLOPs per sample.
- Pre-registration: every test's prediction and pass line was written down with a timestamp before its numbers were computed. Results that failed their lines are in the repository alongside the passes.

Scope

- In wide, shallow networks (256x8, 256x4) ARC's AUGMENT K2 was about 2x better than my estimator with the leading-order variance, at similar cost. The scale mode is small there, and ARC's higher orders stay ahead wherever they converge.
- With the exact variance recursion, first-order and exact Gamma moments give the same accuracy on the diagonal. The gain comes from separating the scale and carrying its variance correctly.
- The pre-registered exact variance rule is reported in every cell. At 512x128 the leading-order rule happens to score higher (1.96x); at 128x32, 256x64 and 1024x256 it falls below sampling, and the exact rule stays ahead at all five widths.
- At the challenge shape (1024x16) my standalone K2-cost estimator is 1.3x worse than ARC's AUGMENT K2; the scale separation helps there as a layer on a K3 chain (Section 7).
- One prediction failed: a one-neuron series argument predicted that ARC's ladder would keep improving through K4, and at 64x16 it inverts at K4. The multi-neuron test in Section 2 is what identifies the cause.

Epistemic status and AI use

The mechanism in Sections 2 and 3 is established by direct intervention on ARC's own pipeline, and the variance recursion is checked against exact samples of the law. The results in Sections 4 and 5 are measured, with intervals over random nets. I used AI coding assistants (Claude) heavily for implementation, experiment management and analysis, under my direction. Every number here comes from code in the repository, with its pre-registered prediction and its truth labels.

Submission

Algorithmic Contribution Prize submission. Graded leaderboard entry: 336823. Code, data scripts, figures and the pre-registration log: https://github.com/PranavViswanath/scale-separation.

Reproducing

The repository contains the estimator, the variance recursion and its check, the de-scaling hook for ARC's code, the truth scripts, the pre-registration log, and one script per figure. Each cell runs in minutes on one GPU, apart from Monte Carlo truth.

References

1. W. Wu, V. Lecomte, M. Winer, J. Robinson, J. Hilton, P. Christiano. Mechanistic estimation for wide random MLPs. arXiv:2605.05179 (2026). Blog: https://www.alignment.org/blog/mechanistic-estimation-for-wide-random-mlps/
2. ARC. mlp_cumulant_propagation, reference implementation of cumulant propagation (commit 93d091a).
3. ARC. Competing with sampling: the matching sampling principle. Alignment Research Center blog.
4. P. Christiano, E. Neyman, M. Xu. Formalizing the presumption of independence. arXiv:2211.06738 (2022).
5. B. Hanin, M. Nica. Products of many large random matrices and gradients in deep neural networks. arXiv:1812.05994 (2018).
6. B. Hanin. Correlation functions in random fully connected neural networks at finite width. arXiv:2204.01058 (2022).
7. D. A. Roberts, S. Yaida, B. Hanin. The principles of deep learning theory. arXiv:2106.10165 (2021).