Structure-Aware Estimation for Random ReLU MLPs

Natasha Stewart Yangxinyu Xie

Team SOX

ARC WhiteBox Estimation Challenge 2026, Phase 1
Submission ID: 319341

Abstract

The ARC WhiteBox Estimation Challenge asks for estimates of post-ReLU mean activations in random MLPs under standard Gaussian inputs, with a score that rewards both low final-layer error and low effective compute. We contribute a structure-aware hybrid estimator and a study of how far its premises extend. The estimator is built on data-driven observations about the structure of the ReLU network. Measuring how these networks behave shows that neurons separate into three regimes: almost certainly inactive, almost certainly active, and uncertain. We thus apply an analytical pass to classify every neuron into one of these categories from the weights alone and spend Monte Carlo samples only where they are informative. Due to the structure of a ReLU MLP, sparsity persists at the second level, even after the almost certainly inactive neurons have been removed, so the sampled matrix products can skip the structurally zero regions exactly. Compute reductions follow from these observations rather than from a separate optimization pass, and we hope the result serves as a strong sampling baseline for subsequent white-box estimators. Because these premises arise from the ReLU nonlinearity rather than from the estimator itself, we next ask how well they generalize. Varying the activation function and the architecture independently on matched random weights, we study two complementary properties: the effective rank of the final-layer representation and the transferability of the three-region classification. Both depend strongly on architectural choices, but they are distinct properties and need not vary together.

Note on AI usage. This work was developed with heavy use of large language models (Claude, ChatGPT). The investigation began with human-led exploration—visualizing the networks' behavior and posing questions about the patterns that emerged—and the resulting ideas were implemented in code by LLMs under our direction. An initial draft of this write-up was likewise produced using LLMs and then substantially revised by the authors.

Contents

1 Introduction

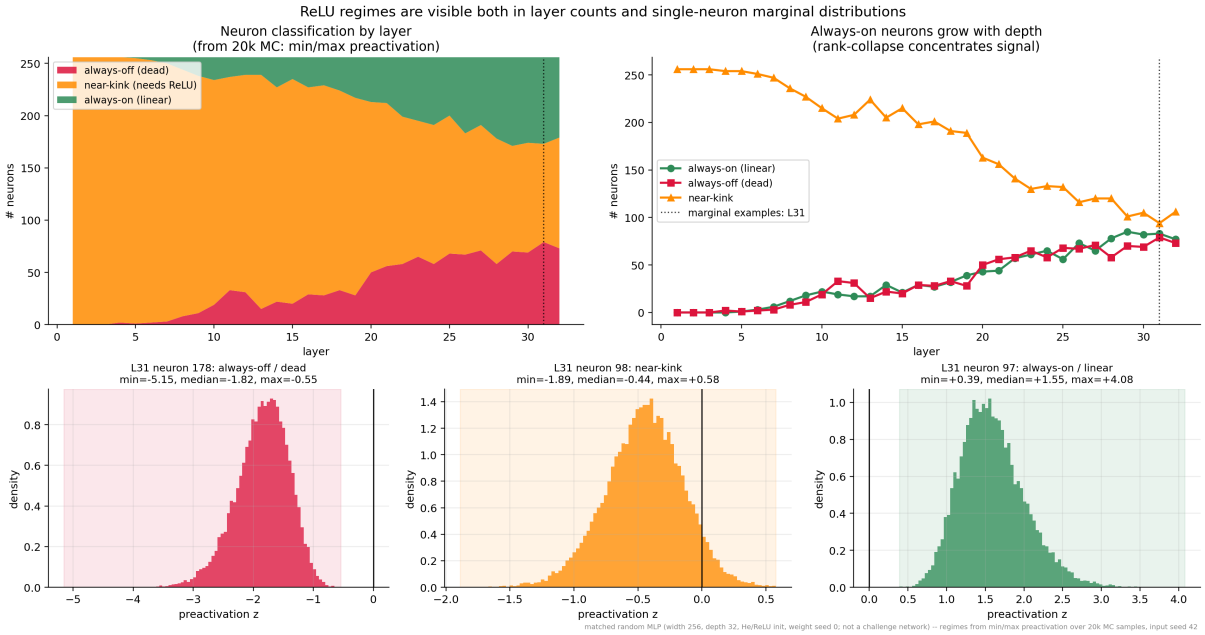

Figure 1: Monte Carlo diagnostic from the layer-collapse notebook, computed on a matched random MLP—width 256, depth 32, He-initialized ReLU weights (weight seed 0), the same architecture and initialization scale as the challenge networks, but not a challenge network itself. Neurons are classified by their preactivation range over 20k samples: always-off neurons never fire, always-on neurons pass linearly through ReLU, and near-kink neurons still require sampled nonlinear propagation. The upper panels show the layerwise regime counts; the lower panels show representative single-neuron preactivation marginals from layer 31, making the dead, kink, and on regimes visible at the distribution level.

The WHEST challenge aims to estimate the mean post-ReLU activation of every neuron using only the network weights and a fixed compute budget [2, 15]. With a large number of samples, direct Monte Carlo sampling can be highly accurate; however, it does not provide much insight into the structure of the network, and it is inefficient since it treats every neuron and every sampled path in the same way. A

white-box estimator, by contrast, can use the weights to decide which parts of the network are structurally predictable.

In a ReLU network, the most useful structural distinction is how likely a neuron's pre-activation is to fall on either side of the ReLU threshold at zero. Neurons that are negative for nearly all inputs ("dead") contribute little, neurons that are positive for nearly all inputs ("on") behave nearly linearly, and neurons whose pre-activations frequently fall on both sides of the threshold ("kink") are, in our submission, where Monte Carlo sampling is most informative. We show Figure 1 as an example, computed on a matched random MLP drawn with the challenge networks' width, depth, and He initialization scale: in the first layer, all 256 neurons are in the kink regime; by the 16th layer, 29 neurons are dead and 29 are always-on; by the 32nd layer, 73 neurons are dead (28% of the layer), 77 are always-on, and 106 remain in the kink regime. The always-on neurons are also strongly correlated with one another: as depth grows, most of the covariance mass concentrates into a small, tightly coupled core of neurons, so the late layers are far more structurally predictable than their width suggests.

The growing population of dead neurons is consistent with the well-known dying-ReLU phenomenon, where inactive ReLU neurons output zero across inputs and produce a vanishing-gradient failure mode [7, 17]. Lu et al. also prove a stronger depth-asymptotic statement: under their initialization model, dying ReLUs are inevitable once the network is deep enough, with deep ReLU networks dying in probability as depth tends to infinity.

These observations motivate a hybrid estimator that combines analytical methods with strategic sampling. Its core is not to replace sampling everywhere, but to use a mechanistic approximation to decide where sampling should be trusted, where it should be avoided, and how sampled activations should be propagated. The estimator first propagates a diagonal Gaussian approximation through the network, using the results to classify neurons as dead, kink, or on. It then refines uncertain classifications with a small set of samples. Finally, it turns the resulting ReLU behavior into implementation decisions: dead neurons are omitted from the sampled propagation, and their reported means are supplied by the analytical pass. The remaining non-dead neurons are evaluated using Monte Carlo samples, with the sampled matrix products reorganized so that zero blocks of the activation matrix (rarely-firing columns, crossed with the rows that never fire on them) are skipped exactly. In the final two layers, the estimator applies an additional optimization: on neurons are treated as linear, allowing their contributions to be propagated analytically through the weights of the final two layers. This implementation thus captures the idea of "competing with sampling" [2]: the code is faster because it follows the network's structure rather than treating every sampled coordinate uniformly as a black box.

1.1 How Far Do These Premises Transfer?

A structure-aware estimator buys its speed by assuming something about the network, so it is worth asking early how much of the method is tied to the particular architecture it was tuned on. The challenge networks are plain, bias-free ReLU stacks, but the premises the estimator rests on are properties of the ReLU activation. In Section 5 we therefore stress-test them directly, varying the nonlinearity and other architectural choices independently on matched random weights and re-running the same structural diagnostics. We study the effective rank of the final-layer representation and transferability of the kink classification and show that both axes are highly sensitive to architectural choices.

1.2 Contributions

This work makes two contributions.

1. **A sampling method that follows the structure of a ReLU network.** Section 3 presents the estimator as a sequence of data-driven decisions: we measure how these networks actually behave, and each measurement then determines a design choice. Visualizing pre-activation distributions on example MLPs shows that neurons separate into dead, kink, and on regimes, which motivates classifying every neuron from the weights before any sampling: dropping the dead, sampling the kinks, and directly calculating the expectation of on neurons through the last two layers analytically. Sorting the surviving columns by firing rate then shows that sparsity persists at a second level. Even after the dead neurons are removed, the rarely-firing "cold" columns are mostly zero, and a typical sample row fires on only two or three of them. We leverage this observation by splitting the sampled matrix product into a dense block and a packed cold prefix, with the prefix width and the row buckets read off the measured firing rates rather than fixed in advance. Compute reductions are thus consequences of these structural observations rather than a separate optimization pass, and we hope the result serves as a strong sampling baseline for subsequent white-box estimators built on the same mechanistic structure.

2. **A deep dive into how that structure changes with the architecture.** Two ablations, over the nonlinearity and over six architectural changes on shared weights, identify sign-preservation at the ReLU kink as the criterion that determines whether the structure survives, and separate it from low-rank collapse of the representation.

In essence, much of the final-layer mean can be explained by a small set of structural decisions: which neurons are almost certainly dead, which are almost certainly active and thus effectively linear, and which regions of the sampled activation matrix can be handled with sparse operations.

2 Problem Setup and Notation

We consider a fully connected feed-forward neural network (a multilayer perceptron, MLP) with ReLU activations and no bias terms. The network has *depth* L and *width* d. The input does not count toward the depth. Each of the L layers contains d neurons, and the transformation from layer $\ell - 1$ to layer ℓ is parameterized by a weight matrix $W_\ell \in \mathbb{R}^{d\times d}$. Throughout we use row-vector notation: activations are row vectors and multiply the weight matrices from the left, matching the layout of the implementation. The input $x_0 \sim N(0, I_d)$ is a random row vector with d independent standard Gaussian coordinates. The layers compute

$$z_\ell = x_{\ell-1} W_\ell \in \mathbb{R}^d, \tag{1}$$

$$x_\ell = \mathrm{ReLU}(z_\ell) \in \mathbb{R}^d, \qquad \ell = 1, \ldots, L, \tag{2}$$

where $\mathrm{ReLU}(z) = \max(z, 0)$ is applied elementwise. We call z_ℓ the *pre-activation* and x_ℓ the post-ReLU *activation* of layer ℓ. Entry $W_{\ell,ij}$ is the weight from neuron i of layer $\ell - 1$ to neuron j of layer ℓ. Thus

$$z_{\ell,j} = \sum_i x_{\ell-1,i}\, W_{\ell,ij}. \tag{3}$$

Given only the weights $W_1, \ldots, W_L$, the estimator must output the matrix of mean activations

$$M = \begin{pmatrix} m_1 \\ \vdots \\ m_L \end{pmatrix} \in \mathbb{R}^{L\times d}, \qquad m_\ell = \mathbb{E}[x_\ell] \in \mathbb{R}^d, \qquad \ell = 1, \ldots, L, \tag{4}$$

where the expectation is taken over the random input x_0.

The challenge networks have depth $L = 32$ and width $d = 256$. Accuracy is scored only on the last-layer means. Writing $\widehat{m}_L$ for the predicted final-layer mean, the score is the mean squared error

$$\mathrm{MSE}(\widehat{m}_L, m_L) = \frac{1}{d}\sum_{j=1}^{d}(\widehat{m}_{L,j} - m_{L,j})^2, \tag{5}$$

multiplied by the compute penalty $\max(0.1,\, C/B)$; lower is better. Here C is the compute the estimator consumed, measured in floating-point operations (FLOPs), and B is the per-network FLOP budget. The earlier rows are reported but serve only as diagnostics. In effect, the estimation problem is to approximate m_L as accurately as possible within the compute budget.

3 Method: A Structure-Aware Hybrid Estimator

3.1 Classifying Every Neuron as Dead, Kink, or On Before Any Sampling

Before choosing an algorithm, we selected a few example MLPs and drew a large number of Monte Carlo samples to visualize the pre-activation distribution for each neuron under Gaussian inputs. We noticed that, particularly for the middle and later layers of the MLP, there were many pre-activation distributions with almost all probability mass below zero, some with substantial probability mass near zero, and others with almost all probability mass above zero. These are the empirical dead, kink, and on regimes in Figure 1.

By using the weights of the MLP to predict each neuron's regime prior to any sampling, we can create a white-box estimator that leverages the classical observation that rectifiers produce sparse active sets [11, 18]. Suppose the previous post-ReLU activations have approximate mean $\mu_{\ell-1}$ and diagonal variance $v_{\ell-1}$. For each output neuron j (column j of W_ℓ), the preactivation moments are approximated by

$$\mu^{\mathrm{pre}}_{\ell,j} = \sum_i \mu_{\ell-1,i}\, W_{\ell,ij}, \tag{6}$$

$$(\sigma^{\mathrm{pre}}_{\ell,j})^2 = \sum_i v_{\ell-1,i}\, W^2_{\ell,ij}. \tag{7}$$

Writing $\alpha_{\ell,j} = \mu^{\mathrm{pre}}_{\ell,j}/\sigma^{\mathrm{pre}}_{\ell,j}$ and denoting the standard normal density and CDF by ϕ and Φ, the Gaussian ReLU moment update is

$$\mu_{\ell,j} = \mu^{\mathrm{pre}}_{\ell,j}\Phi(\alpha_{\ell,j}) + \sigma^{\mathrm{pre}}_{\ell,j}\phi(\alpha_{\ell,j}), \tag{8}$$

$$v_{\ell,j} = \left((\sigma^{\mathrm{pre}}_{\ell,j})^2 + (\mu^{\mathrm{pre}}_{\ell,j})^2\right)\Phi(\alpha_{\ell,j}) + \mu^{\mathrm{pre}}_{\ell,j}\sigma^{\mathrm{pre}}_{\ell,j}\phi(\alpha_{\ell,j}) - \mu^2_{\ell,j}. \tag{9}$$

These are the same closed-form ReLU Gaussian moments used in assumed-density filtering for Bayesian neural networks [10, 14]. The Gaussian assumption here is only an approximation, not an accurate representation of ReLU activations, especially in the later layers. Under Gaussian weight priors, unit-level pre- and post-activation marginals become increasingly heavy-tailed with depth and are characterized by sub-Weibull tails [25].

In this estimator, we use the normalized margin $\alpha_{\ell,j}$ to classify a neuron as dead if $\alpha < -3$, on if $\alpha > 3$, and kink otherwise.

3.1.1 What the Classification Buys: Skip the Dead, Sample the Kinks, Fold the On

The regime model determines which sampled coordinates are materialized. Dead neurons do not need a sampled matmul column: their post-ReLU mean is either filled by the analytic pass or set to zero when the residual contribution is negligible. Kink neurons remain sampled, because their ReLU masks change across samples. On neurons receive a linear treatment only in the last two layers, 31 and 32; in layers 1–30 they remain part of the sampled propagation alongside the kink neurons. If neuron i is on at layer 31, then sample by sample

$$x_{31,i} = z_{31,i} = (x_{30}W_{31})_i. \tag{10}$$

Its contribution to a layer-32 preactivation can therefore be composed through W_{31} and W_{32} without materializing the sampled layer-31 column, and the means of the on neurons at layer 32 are likewise computed linearly from the layer-31 mean activations. Kink neurons at layer 31 do not satisfy this identity, so their contribution remains sampled.

One disadvantage of using a classifier is that the boundary between regimes is not always clear. To overcome this shortcoming, a small set of samples is used to reclassify neurons near the decision boundary, which we call the "pilot pass," and will come back to in the next subsection. This verification is particularly important in the last two layers, where classification errors directly affect the scored layer.

3.2 Matrix Multiplication for Sampled Activations

The classification of neurons as dead, kink, or on improves efficiency primarily by removing the computation for dead neurons, but the sampled activations of the surviving neurons still contain many zeros. To see where they are, we sort the columns of the sampled post-ReLU activation matrix by firing rate, the fraction of Monte Carlo samples on which each neuron is active. Figure 2 shows the results across three networks and three depths. On the left side of the matrix, the usually inactive ("cold") columns are mostly zero, with only scattered firings, while usually active ("hot") columns to the right are dense. After the coldest-first sort, a typical row carries only two or three nonzeros across the leading columns (the "prefix"). The estimator therefore splits the multiplication into two regions. The dense portion is evaluated with Strassen matrix multiplication [24], while the prefix of cold columns is processed row by row, gathering only the few active entries in each row.

3.2.1 Evaluating the Cold-Region Contribution

To choose the prefix width, columns are sorted coldest-first, and the cutoff is chosen where the running sum of the fire rates (the *expected* number of nonzeros a row carries over the first s columns) reaches a target of 3 nonzeros per row. Finally, the width of the cold-column prefix is clamped so the remaining hot block keeps at least 96 columns. Next, rows are grouped into buckets by their nonzero count over the prefix, using the ladder $(0, 1, 2, 4, 8)$. The first bucket holds rows with no nonzero columns in the prefix, and the next bucket holds rows with exactly one nonzero. This pattern continues, with one bucket for each level of the ladder. There is a final remainder bucket to collect the rare rows with more than 8 active prefix entries.

After the dense block is evaluated for all rows using Strassen matrix multiplication, the prefix contribution is added separately for each bucket. The first bucket has no contribution and can be ignored. For the four buckets of rows with 1–8 active prefix entries, a gathered correction accounts for the nonzero elements. Finally, the last bucket computes the prefix block densely for its rows. See Figure 2 for an illustration.

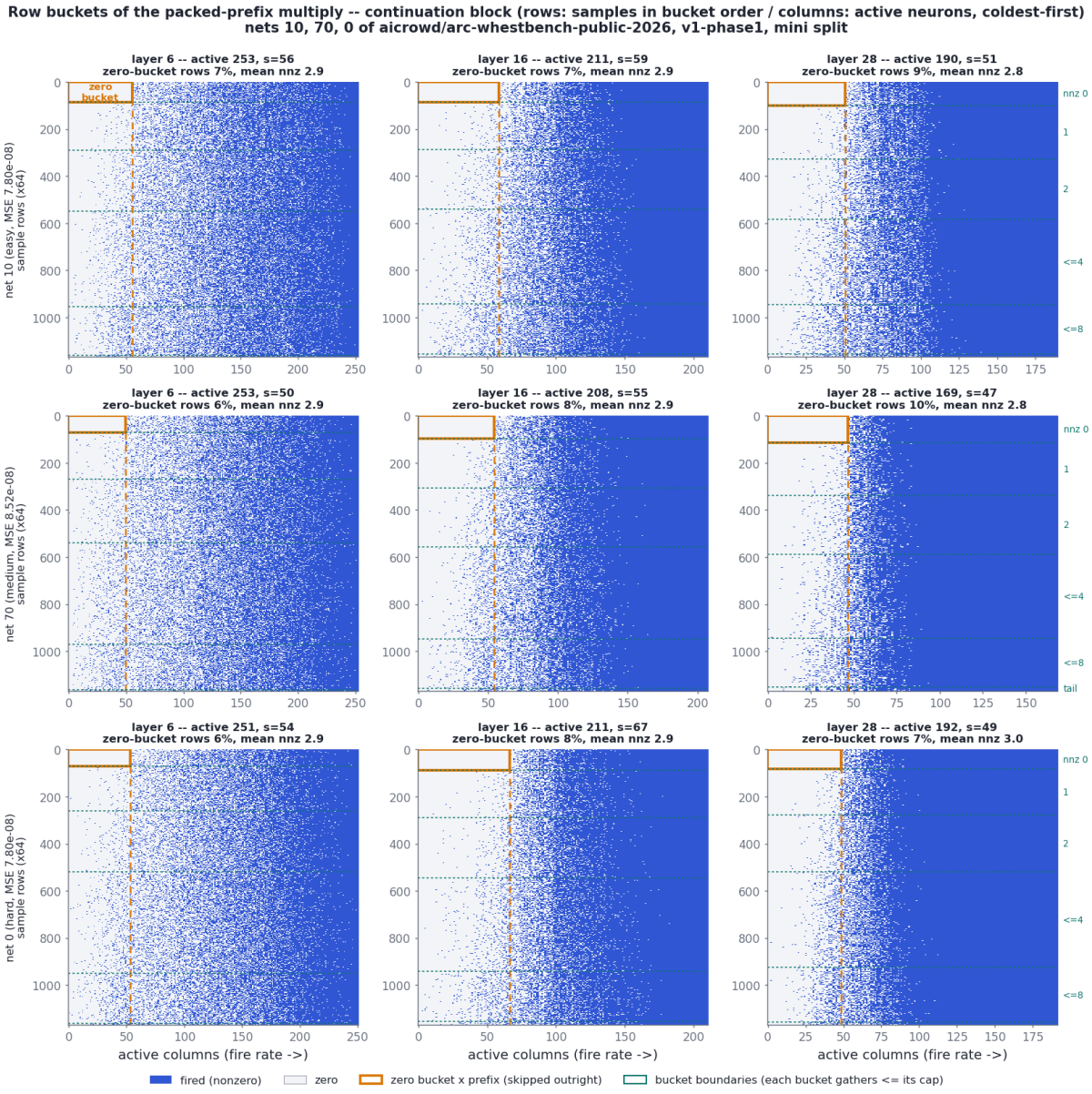

Figure 2: The zero structure the packed prefix exploits, captured from the submitted estimator's own continuation blocks (74,752 sample rows $\times$ active columns). Rows of the grid are three public mini networks of increasing difficulty (nets 10, 70, 0); grid columns are layers 6, 16, and 28. Within each panel, matrix columns are sorted coldest-first, the dashed vertical line marks the prefix width s, and sample rows are shown in the estimator's actual bucket order: rows with no nonzero on the prefix first, then buckets of at most 1, 2, 4, and 8 nonzeros, then the dense tail. The outlined top-left rectangle—the zero bucket crossed with the prefix—is all zero, and the propagating matmul skips it outright; each bucket below it gathers at most its cap in columns instead of multiplying the whole prefix.

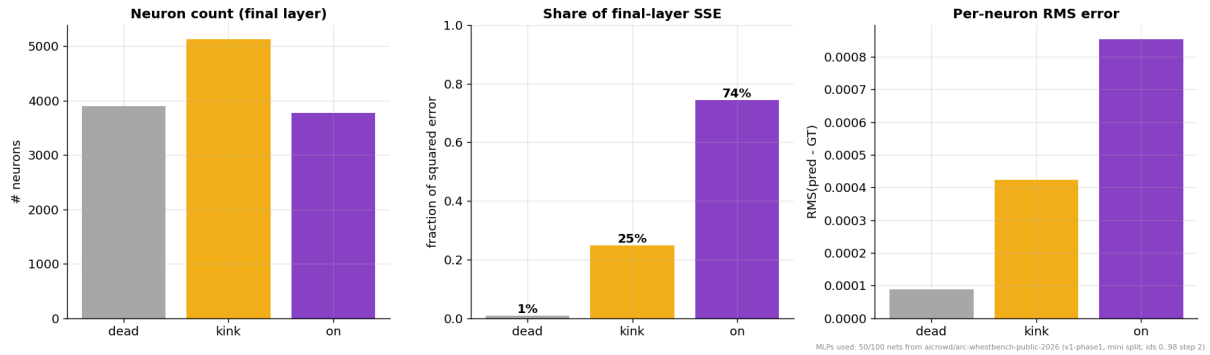

Figure 3: Final-layer error by neuron class (50 public mini networks). Always-on neurons carry 74.4% of the total squared error, kink neurons 24.8%, dead neurons 0.8%.

3.2.2 Sparse Matrix Multiplication for the Pilot Pass

To improve efficiency, we apply the same general techniques for matrix multiplication to the pilot pass as well. However, the pilot stage cannot use the empirical firing rates to set the prefix cutoff because they have not yet been determined. Instead, it determines the cutoff from $\Phi(\alpha_j)$, each column's firing probability under the diagonal Gaussian approximation. The same running-sum rule is then applied to the α-sorted columns. Two details differ from the continuation block described above. The pilot's row order must be preserved—the probe rows are a fixed Sobol prefix and the antithetic pairing is positional—so after the bucketed multiply an inverse-permutation scatter puts the rows back. And when the analytic prefix comes out narrower than 8 columns, the pilot falls back to a single-level carve, treating a column as cold if its margin satisfies $\alpha < -1.88$ ($\Phi(-1.88) \approx 0.03$) and splitting rows only by support/no-support.

3.3 Sampling: Antithetic Scrambled Sobol Points at a Fixed Budget

For sampled propagation, the estimator uses a fixed Sobol prefix and antithetic pairing. The Sobol sequence is a standard low-discrepancy variance-reduction ingredient [23]. The antithetic pair uses the Gaussian symmetry $z \mapsto -z$ to encourage cancellation of the "odd" part of the integration [20, Section 8.2]. Related randomized-QMC approaches were also discussed by other challenge participants, including randomized lattice/Sobol comparisons and Rao-Blackwellization for post-ReLU activation means [9], and variance-bias budgeting for deep white-box mean estimation [1].

Only half of the Gaussianized points are stored; the first-layer symmetry reconstructs the paired activations exactly after the first matmul. The submitted estimator uses a fixed $84{,}992$ effective samples for every MLP. The pilot stages read the leading 2.5% and 20% of the same Sobol prefix that the main pass consumes, so every probe sample is reused and classification costs no extra draws.

4 Results, Limitations, and Future Work

4.1 The Scored Error Concentrates in a Few Large Always-On Neurons

All measurements in this section use the submitted estimator (submission 319341, $N = 84{,}992$ Sobol samples) on 50 of the 100 public mini networks (even ids 0–98). For distribution-level measurements

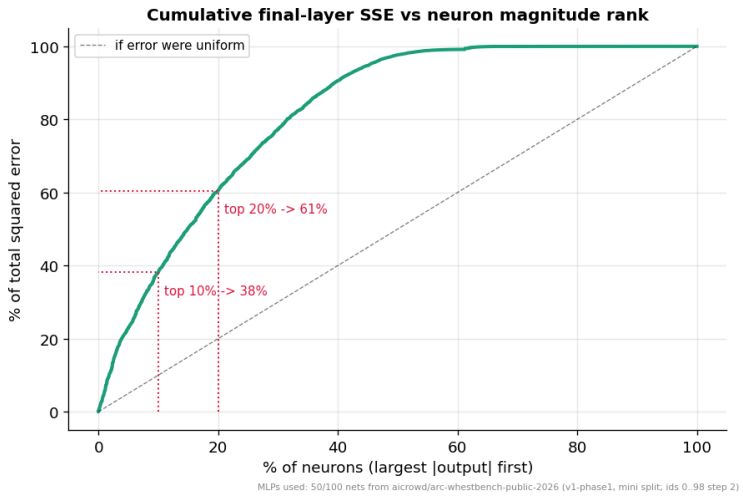

Figure 4: Cumulative final-layer squared error by output-magnitude rank (same 50 networks): the top 10% of neurons carry 38% of the squared error, the top 20% carry 61%.

we take four of these networks (ids 0, 32, 66, 98) and draw 16,384 antithetic samples each.[1] Since only final-layer error is scored, we restrict attention to the final layer, where the median MSE is 2.3×10^{-7} (mean 2.9×10^{-7}).

When we analyze the final-layer squared error by neuron class (Figure 3), always-on neurons turn out to carry most of it—74.4% of the total, against 24.8% for kink neurons and just 0.8% for dead ones. This is not because on neurons are more numerous but because they are bigger: their outputs average ≈ 1.5 in magnitude versus ≈ 0.17 for kink neurons, so even modest relative errors on the on block dominate everything else. The concentration continues within the on block itself (Figure 4), where the top 10% of neurons by output magnitude account for 38% of the squared error and the top 20% for 61%.

4.2 The On-Neuron Error Is Sampling Variance, Not Bias

Is the on-block error bias or just noise? Pooled over the 50 networks, the mean signed error is -2.0×10^{-5}—roughly forty times smaller than the error spread of 8.6×10^{-4}, so its squared contribution is under 0.1% of the MSE—and it has no meaningful direction. There is a slight lean toward under-estimation (51.4% of on neurons land below the truth, as do 28 of the 50 networks), but this is a median effect rather than a bias: frequent small undershoots are balanced by rarer, larger overshoots, so more than half of realizations land low. Because all 256 outputs share the same sample points, each network additionally picks up a common shift, which is why whole networks (not just neurons) lean one way. Further analysis also indicates that the size of the error matches what sampling variance predicts.

4.3 Limitations and Future Work: Heavy Tails and Jointly-Crossing Kinks

Prior work shows that on-neuron activations become increasingly heavy-tailed with depth [19, 30], following sub-Weibull distributions [25]. We see this directly, if modestly at this width: both skewness and excess kurtosis grow with depth (Figure 5). These heavy tails expose a limitation of the diagonal

[1] All statistics are produced by `estimator_error_attribution_319341.ipynb` in the repository; the figures are rendered by `writeup/figures.ipynb` from the same cached predictions.

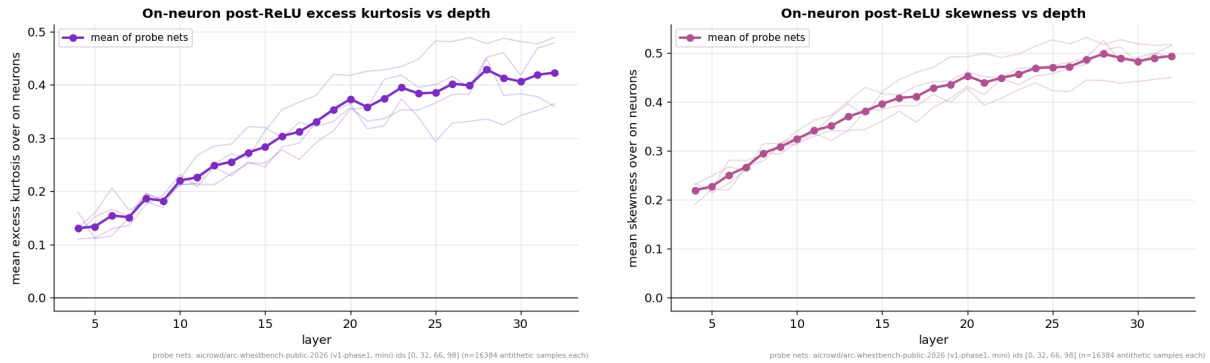

Figure 5: Excess kurtosis (left) and skewness (right) of on-neuron post-ReLU activations versus depth (four probe networks; thin lines are individual networks, bold their mean). Both grow with depth, consistent with sub-Weibull tails.

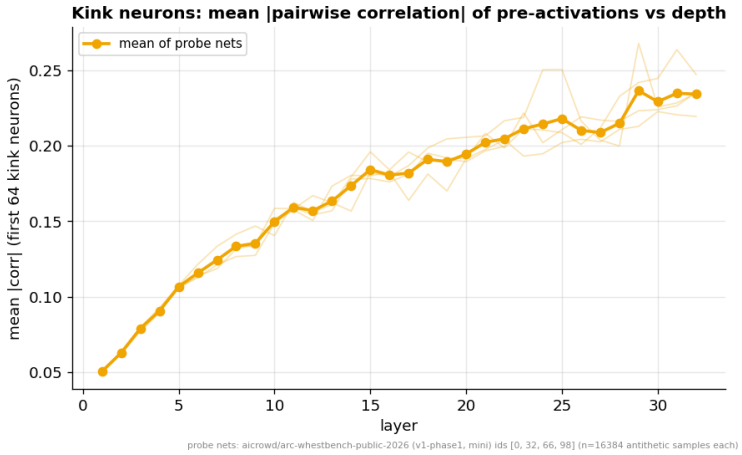

Figure 6: Mean absolute pairwise correlation of kink-neuron pre-activations versus depth (same four probe networks), growing from ≈ 0.13 in early layers to ≈ 0.21 in late layers.

Gaussian propagation pass. In deeper layers, it overestimates on-neuron means ($+1.5\%$ in late layers, $+1.8\%$ at the final layer), while its Gaussian tail underestimates the clipping probability by roughly a factor of two. The submitted estimator largely eliminates this error: kink neurons are sampled, on neurons are eventually treated linearly, and the analytical pass survives only in classification and the dead-neuron correction. How to propagate heavy-tailed distributions well remains an open question.

For kink neurons, a natural next step is to parametrize their joint distribution more carefully, for instance with multivariate probit models [4, 5]. There is a clear signal to exploit here: kink pre-activations are substantially cross-correlated, and the coupling grows with depth (Figure 6). Near-threshold neurons tend to cross the kink together—a joint event that independent-marginal propagation cannot represent.

5 Discussion

The estimator is not a pure cumulant-propagation or covariance-propagation method [26, 27]. Instead, it combines analytical approximations with antithetic Sobol sampling. Its primary mechanistic contribution is using structural information about the MLP to sample more strategically: the diagonal Gaussian pass

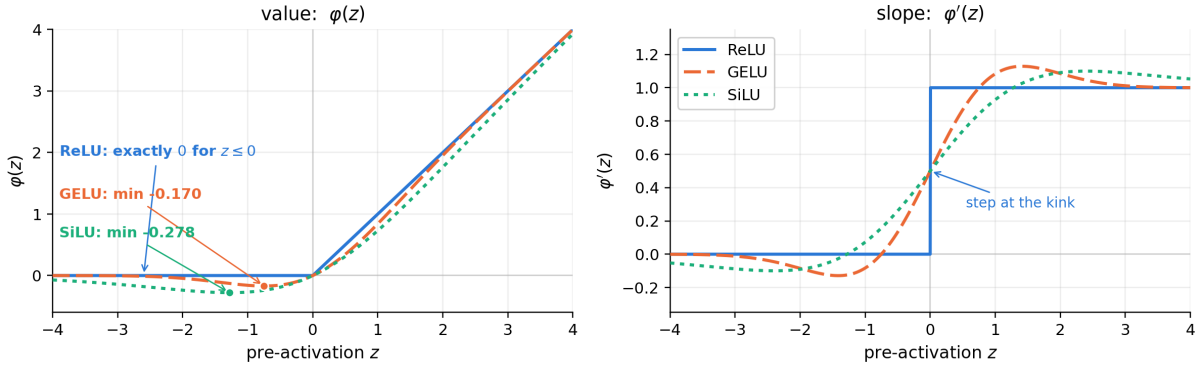

Figure 7: The three activation functions used in the ablation study.

first predicts which neurons are dead, kink, and on; the pilot samples refine uncertain classifications; and the resulting activation structure determines both which neurons are sampled and how the sampled matrix products are evaluated. We hope our methods can serve as a sampling baseline for more sophisticated white-box estimators that exploit the same mechanistic structure.

A useful stress test is to ask what the estimator's premises depend on. We run two ablation studies, varying the nonlinearity and the architecture independently, to see whether the estimator's assumptions survive. Each ablation is run on a matched random MLP with width 256, depth 32, and He-initialized Gaussian weights.

Both ablations report the *effective rank* of the activation covariance, which we define here. Let $\Sigma_\ell \in \mathbb{R}^{n\times n}$ be the empirical covariance of the layer-ℓ activations over the sampled inputs, with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$. We use the participation-ratio form

$$\operatorname{erank}(\Sigma_\ell) \;=\; \frac{\left(\sum_{i=1}^{n} \lambda_i\right)^2}{\sum_{i=1}^{n} \lambda_i^2} \;=\; \frac{(\operatorname{tr}\,\Sigma_\ell)^2}{\|\Sigma_\ell\|_F^2}, \tag{11}$$

which is scale-free, equals k when k eigenvalues are equal and the rest vanish, and drops toward 1 as the spectrum concentrates on a single direction. It is a smooth stand-in for $\operatorname{rank}(\Sigma_\ell)$ that does not require a cutoff on near-zero eigenvalues — which matters here, because the SiLU stack decays to an activation scale of 4.6×10^{-5} and any absolute threshold would report its rank as a property of the threshold. Note that this is the participation ratio rather than the entropy-based effective rank of Roy and Vetterli [22].

5.1 Activation Functions

The first ablation varies the *nonlinearity*. Figures 8 and 9 repeat the diagnostic with ReLU, GELU [13], and SiLU [8]. The latter two activations are smooth, see figure 7, and therefore do not produce the hard dead and always-on populations that the estimator exploits.

Thus, although ReLU creates clean always-off and always-on regions, GELU and SiLU produce neither. Instead, under the initialization studied here, their activations gradually shrink with depth, so that by layer 32, the typical activation magnitude is close to zero even though neurons remain responsive to their inputs. This distinction is important when designing diagnostics. A method that declares neurons "dead" whenever their activation falls below a fixed absolute threshold could incorrectly label an entire

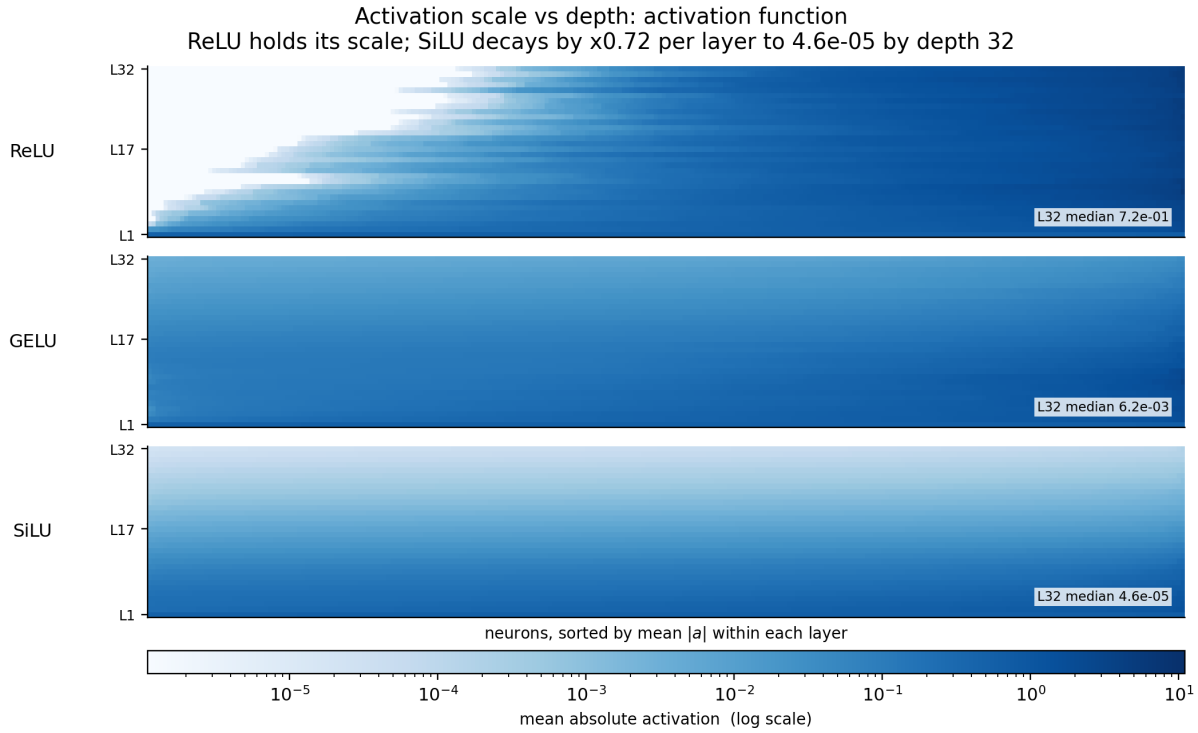

Figure 8: The activation magnitude $\overline{|a|}$ on a shared log scale for three networks with different activation functions.

deep SiLU layer as dead simply because the overall activation scale has decayed, which is a property of the threshold and not of the network; see figure 8.

Similar to ReLU networks, the correlation matrices of the activations in GELU and SiLU networks collapse onto a very correlated core, but the ReLU network is the only one that does so with the exact dead/on identities that the estimator exploits. The smooth activations do not produce the same hard regimes, and therefore the estimator's assumptions do not hold for them. We think a mechanistic method exploiting the low rank of the activations could be an interesting direction, but we haven't found a method that achieves higher accuracy than sampling.

5.2 Architectural Tweaks

The second ablation varies the *architecture*. Figures 10 and 11, summarized in table 2, compare six wirings on the same weight tensors and ReLU activation function: plain $\mathrm{ReLU}(xW)$, a residual block $x + \mathrm{ReLU}(xW)$, batch, layer, and root mean square (RMS) normalization applied to the pre-activation, and a pre-norm residual block $x + \mathrm{ReLU}(\mathrm{LN}(x)\, W)$. BatchNorm is used in its inference-time form, a fixed per-neuron affine calibrated from population statistics. Table 1 writes out all six.

As shown in figure 10, the estimator's assumptions survive some of these architectural tweaks but not others. Residual reinforces the classification, because the skip connection preserves the sign of the pre-activation and therefore the dead and always-on populations. RMS preserves the classification, because it rescales the pre-activation by a positive scalar and therefore cannot move a neuron from one side of the kink to the other. LayerNorm preserves the classification in expectation, but shifts the pre-activation by a per-input amount that can move a neuron from one side of the kink to the other. On the other hand,

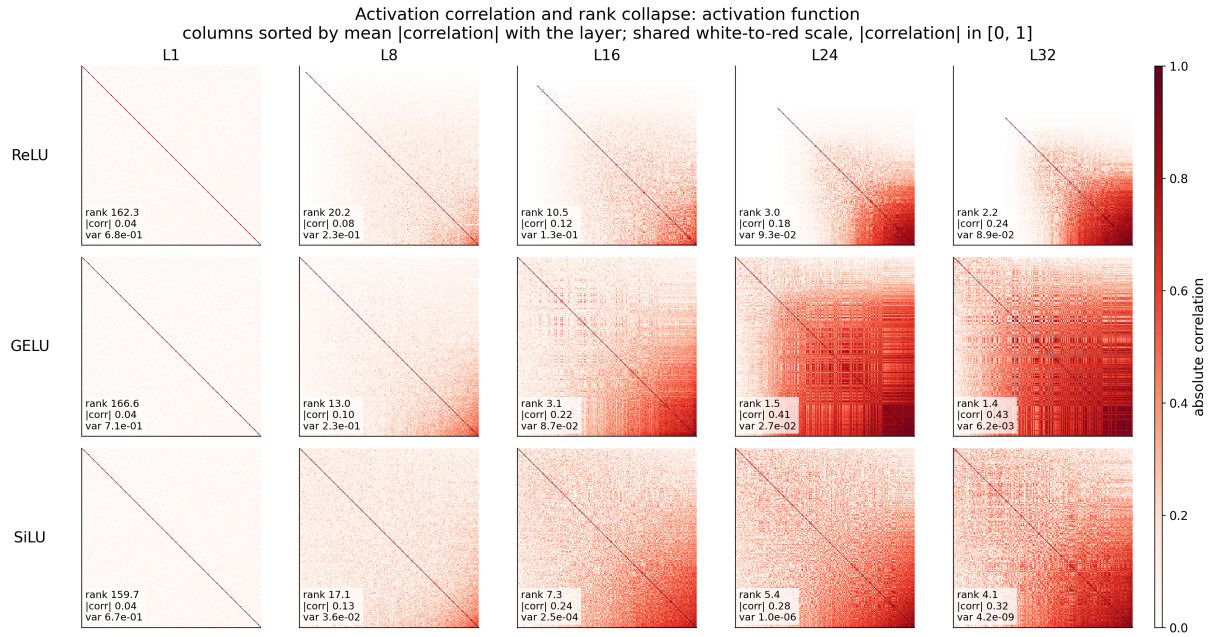

Figure 9: Activation covariance comparison for the matched activation functions. Columns are ordered by each neuron's mean absolute correlation with the rest of its layer, so weakly coupled neurons collect at the top left and the correlated core at the bottom right; the white-to-red scale and its colorbar are shared across all panels. All three collapse onto a late correlated core, but only ReLU does so with the exact dead/on identities the submitted estimator exploits.

BatchNorm and pre-norm residual destroy the classification, thus collapsing the estimator's assumptions.

In our error analysis in section 4, we saw that the estimator's error is dominated by the on neurons, with RMSNorm, we expect the exact magnitude of the errors to shrink according to the rescaling. For residual connections, we expect the classification step in the estimator to be more accurate, and therefore the error to shrink as well. For layer normalization, it might take a more sophisticated estimator to identify the per-input shift and correct for it. For BatchNorm and pre-norm residual, the estimator's assumptions are violated, and the estimator reduces to plain sampling. These intuitions are confirmed in table 2 and figure 12, which show that the estimator's classification survive residual and RMSNorm, are partially violated by LayerNorm, and are completely destroyed by BatchNorm and pre-norm residual.

These different architectures also produce varying effective ranks of the activation covariance, which is a more quantitative measure of the collapse of the representation, as shown in different views in table 2 and figures 11 and 12. As seen in figure 11, the residual collapses the most to a near rank-one representation, which might be attributed to the sub-optimality in He-initialization for residual blocks [29]. BatchNorm and LayerNorm are in between, with BatchNorm collapsing more than LayerNorm. RMSNorm is an exception: it raises effective rank without changing a single regime label, since it rescales magnitudes but cannot move signs. Pre-norm residual block maintains a near-full-rank representation, which is consistent with its popularity in modern architectures [21, 28].

Across layers, as shown in figure 12, we see the effective rank decreases the fastest for the residual stack, followed by the plain ReLU network. Interestingly, RMSNorm's effective rank decreases at the same rate as layer normalization, instead of trailing the plain ReLU network. Although batch normalization initially show a slower decay, it eventually matches the RMSNorm and layer normalization. Daneshmand

Table 1: The six wirings, written out. For one input, $x \in \mathbb{R}^d$ is the layer input (a row vector, as in the rest of the paper), W its weight matrix, $z = xW \in \mathbb{R}^d$ the pre-activation, $d = 256$ the width, and $\varepsilon = 10^{-5}$. No normalizer carries a learned affine ($\gamma = 1$, $\beta = 0$), so each row differs from *plain* only by its normalizer. μ_i and σ_i are the mean and standard deviation of neuron i's pre-activation *over the input population*, estimated once and then held fixed, which is what makes inference-time BatchNorm a per-neuron affine; $\bar{z} = \frac{1}{d}\sum_j z_j$ and s_z are the mean and standard deviation of the d coordinates of z *within a single input*. The last column is what the estimator actually depends on: whether the normalizer can change which side of the kink a pre-activation falls on.

Variant	Layer update	Normalizer $N(z)$	Effect on $\operatorname{sign}(z)$
plain	$x \mapsto \mathrm{ReLU}(xW)$	—	—
residual [12]	$x \mapsto x + \mathrm{ReLU}(xW)$	—	—
BatchNorm [16]	$x \mapsto \mathrm{ReLU}(N(xW))$	$\frac{z_i - \mu_i}{\sigma_i + \varepsilon}$	per-neuron shift
LayerNorm [3]	$x \mapsto \mathrm{ReLU}(N(xW))$	$\frac{z - \bar{z}\mathbf{1}}{s_z + \varepsilon}$	per-input shift
RMSNorm [31]	$x \mapsto \mathrm{ReLU}(N(xW))$	$\frac{z}{\sqrt{\frac{1}{d}\sum_j z_j^2} + \varepsilon}$	none (positive scale)
pre-norm residual	$x \mapsto x + \mathrm{ReLU}(\mathrm{LN}(x)\,W)$	LN, applied to the branch input	per-input shift, upstream

Table 2: Architecture ablation on matched He-Gaussian weights (width 256, depth 32). Regime counts and effective rank are all measured at the scored final layer; the cold-prefix column is the mean over depth of the prefix width as a fraction of the layer width. BatchNorm and the pre-norm residual block leave nothing to classify.

	Final-layer regimes				
Architecture	dead	kink	on	Eff. rank	Cold prefix
plain $\mathrm{ReLU}(xW)$	73	106	77	2.2	35.3%
residual	117	5	134	1.0	42.4%
BatchNorm	0	256	0	9.4	2.1%
LayerNorm	60	130	66	11.4	34.6%
RMSNorm	73	106	77	10.9	35.3%
pre-norm residual	0	256	0	94.8	2.0%

et al. [6] prove that batch normalization provably avoids rank collapse in randomly initialized deep networks, so we don't expect the effective rank to continue to decay in deeper networks.

The pre-norm residual block maintains a near-full-rank representation across all layers, which is consistent with the visualizations in figure 11, and might be the most difficult to develop a mechanistic estimator that attempts to exploit the low-rank structure of the representation.

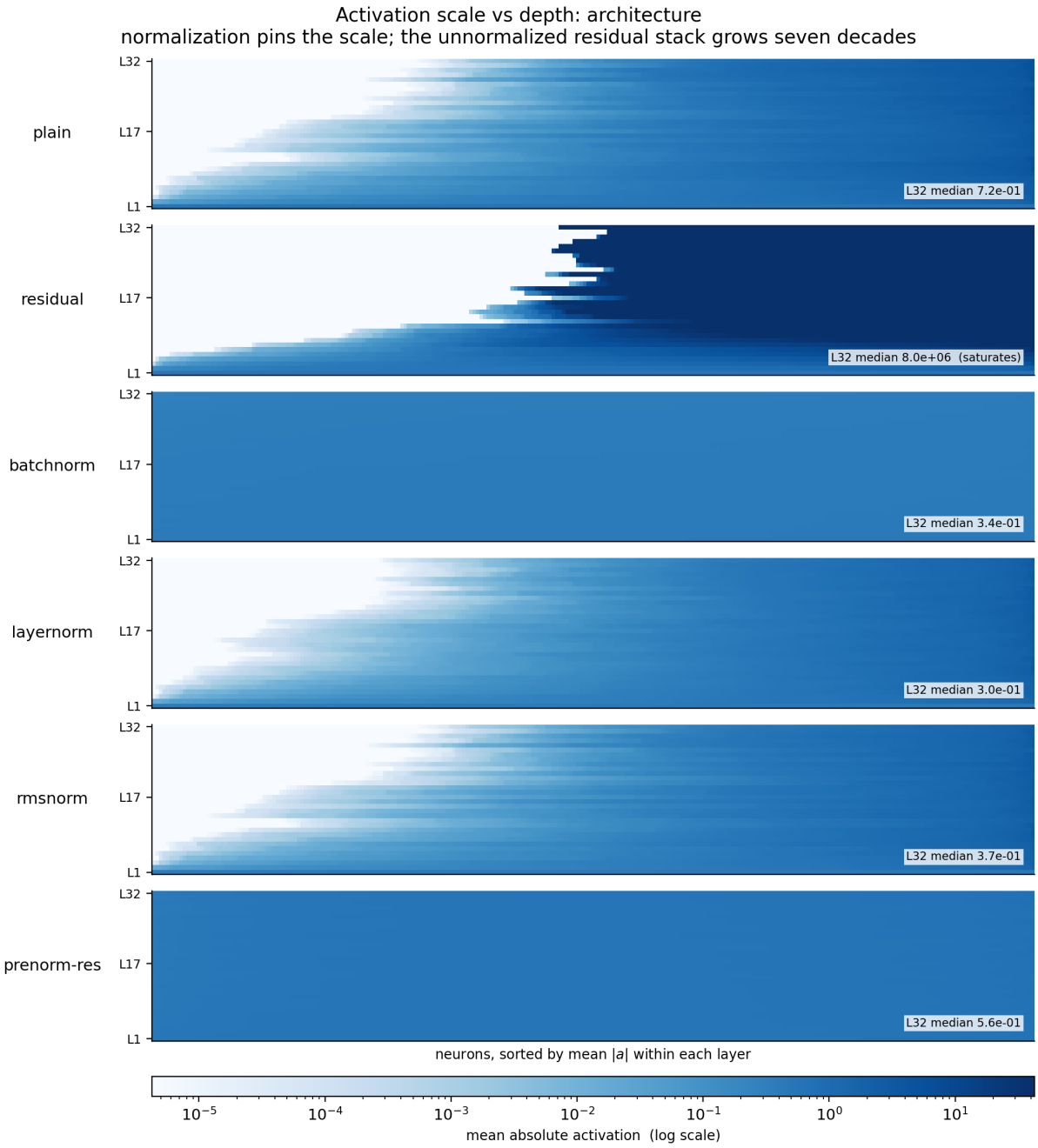

Figure 10: The same two threshold-free views for the six architectures, directly comparable to figure 8. Grey in the right panel marks neurons that must be sampled; red and green mark the dead and always-on populations the estimator prunes and folds. BatchNorm and the pre-norm residual block are flat in *both* panels — uniform scale and uniform grey at every depth — which is the visual form of an architecture that leaves nothing to classify. The unnormalized residual stack saturates the magnitude ramp, its median activation reaching 8.0×10^6 by depth 32.

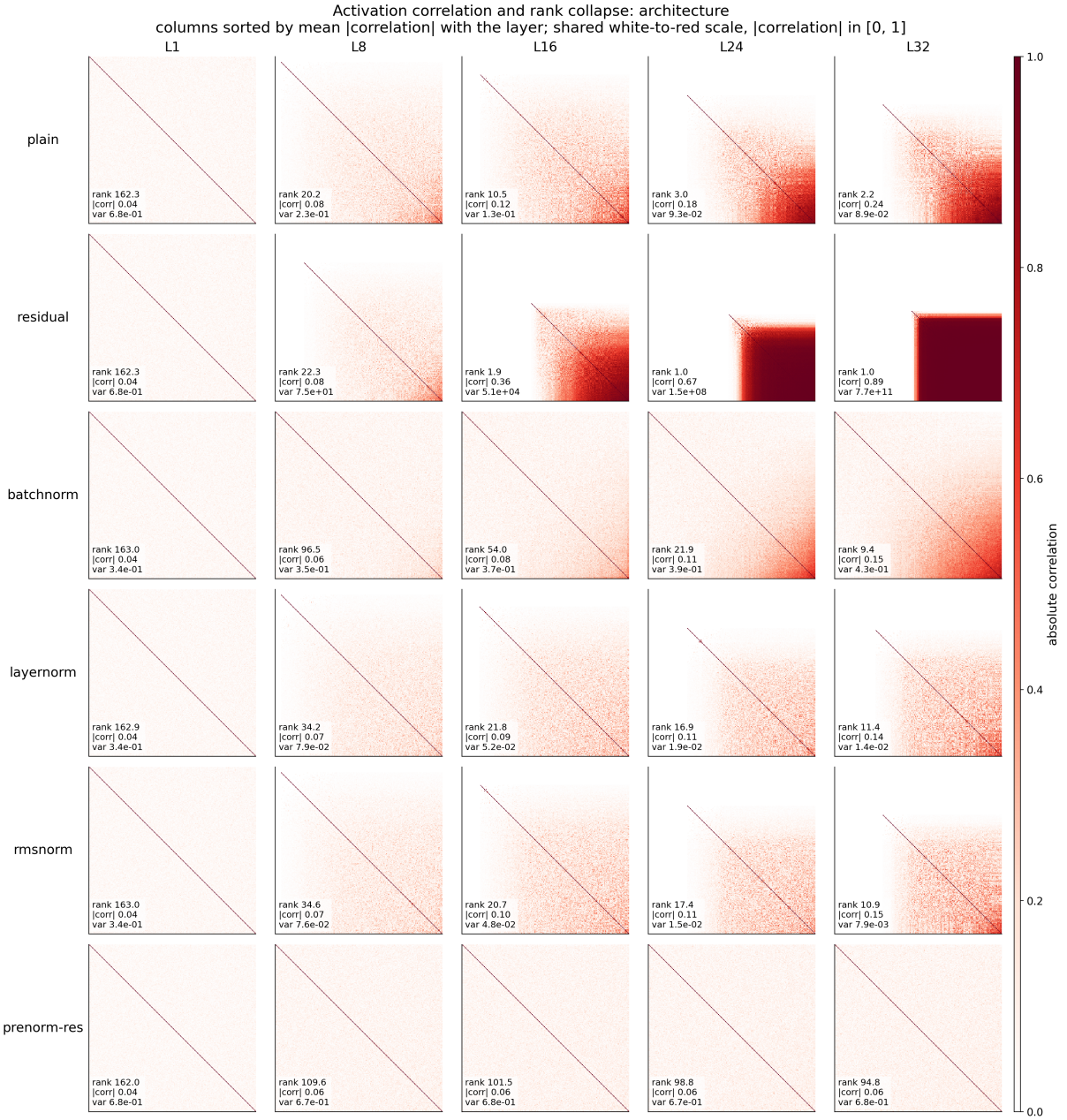

Figure 11: Activation correlation matrices for the six architectures, the same view as figure 9, columns ordered by mean absolute correlation with the rest of the layer and annotated with effective rank. The residual stack concentrates into a single saturated correlated core by L16; BatchNorm develops a weaker late core; the pre-norm residual block stays near-diagonal to depth 32, which is the visual form of a representation that never collapses.

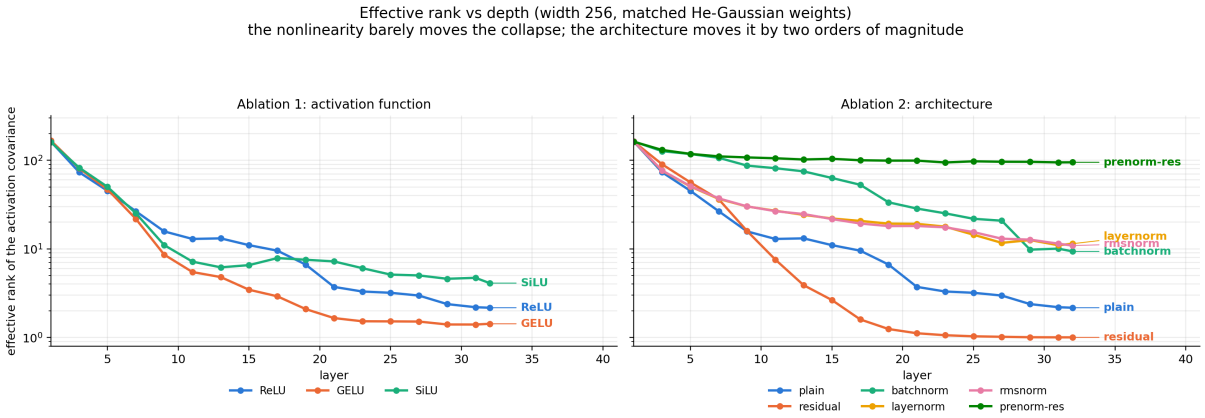

Figure 12: Effective rank of the activation covariance versus depth for both ablations on shared axes. Changing the nonlinearity moves the scored-layer rank only between 1.4 and 4.1; changing the architecture moves it from 1.0 to 94.8. The plain and residual stacks collapse toward rank one, which is what creates the hard regimes; normalization slows the collapse and the pre-norm residual block halts it. RMSNorm is the exception to the correspondence: it raises effective rank without changing a single regime label, since it rescales magnitudes but cannot move signs.

References

[1] AIcrowd participants. Phase 1 write-up: a variance-bias budget for deep white-box mean estimation (submission 317660). AIcrowd Forum, 2026. Forum topic. Accessed 2026-07-22.

[2] Alignment Research Center. Competing with sampling. https://www.alignment.org/blog/competing-with-sampling/, 2025. Accessed 2026-07-12.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Siddhartha Chib and Edward Greenberg. Metropolis-Hastings algorithms for bayesian multivariate probit models. *Journal of Business & Economic Statistics*, 19(3):307–320, 2001.

[5] John G Cragg. Some statistical models for limited dependent variables with application to the demand for durable goods. *Econometrica: journal of the Econometric Society*, pages 829–844, 1971.

[6] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. *Advances in Neural Information Processing Systems*, 33:18387–18398, 2020.

[7] Simon Dufort-Labbe, Pierluca D'Oro, Evgenii Nikishin, Irina Rish, Pierre-Luc Bacon, Razvan Pascanu, and Aristide Baratin. Maxwell's demon at work: Efficient pruning by leveraging saturation of neurons. *Transactions on Machine Learning Research*.

[8] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[9] evaaaz. Unbiased randomized-qmc + rao-blackwell for post-relu activation means: method, unbiasedness proof, and where the frontier is. AIcrowd Forum, 2026. Forum topic. Accessed 2026-07-12.

[10] Soumya Ghosh, Francesco Maria Delle Fave, and Jonathan Yedidia. Assumed density filtering methods for learning bayesian neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1589–1595, 2016.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[13] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[14] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pages 1861–1869. PMLR, 2015.

[15] Jacob Hilton. Mechanistic estimation for wide random mlps. https://www.alignment.org/blog/mechanistic-estimation-for-wide-random-mlps/, 2026. Accessed 2026-07-12.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[17] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying ReLU and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28(5):1671–1706, 2020.

[18] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010.

[19] Lorenzo Noci, Gregor Bachmann, Kevin Roth, Sebastian Nowozin, and Thomas Hofmann. Precise characterization of the prior predictive distribution of deep relu networks. *Advances in Neural Information Processing Systems*, 34:20851–20862, 2021.

[20] Art B. Owen. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/, 2013.

[21] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[22] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference (EUSIPCO)*, pages 606–610, 2007.

[23] I. M. Sobol'. Distribution of points in a cube and approximate evaluation of integrals. *Zh. Vych. Mat. Mat. Fiz.*, 7:784–802, 1967. In Russian. English translation: U.S.S.R. Comput. Maths. Math. Phys. 7:86–112.

[24] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[25] Mariia Vladimirova, Jakob Verbeek, Pablo Mesejo, and Julyan Arbel. Understanding priors in bayesian neural networks at the unit level. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6458–6467, 2019.

[26] Oren Wright, Yorie Nakahira, and José M. F. Moura. An analytic solution to covariance propagation in neural networks. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, 2024.

[27] 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.

[28] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pages 10524–10533. PMLR, 2020.

[29] Ge Yang and Samuel Schoenholz. Mean field residual networks: On the edge of chaos. *Advances in neural information processing systems*, 30, 2017.

[30] Jacob Zavatone-Veth and Cengiz Pehlevan. Exact marginal prior distributions of finite bayesian neural networks. *Advances in Neural Information Processing Systems*, 34:3364–3375, 2021.

[31] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.