PROFORMA GLOBAL RESEARCH

Self-Discovering Architectures

Matt Rollings, Founder and Principal, Proforma Global

Whitepaper · 2026-05-18

Executive summary. Every modern transformer in production inherits the same architectural skeleton: a fixed number of layers, each with the same hidden width, the same feed-forward expansion factor, the same attention head count, held constant across training. This static, symmetric design is convenient. It is not principled. Empirical evidence from the broader pruning literature is unambiguous on this point: most trained transformers can have on the order of 80% of their parameters removed post-training, with the model healing to near-baseline performance. The symmetric paradigm trains far more parameters than the final task actually requires; the redundancy is structural, baked in at training start, and the field's standard remedy (prune the over-trained model after the fact) does nothing to recover the wasted compute that went into training the prunable parameters in the first place. We hypothesize that a better paradigm exists: per-layer shape, size, and parameter-ratio choices discovered by a reinforcement-learning policy during training, at a granularity of every 5–10 steps, with the model itself deciding which layers need more capacity, which need less, and what attention-vs-feed-forward ratio each one should carry. This paper documents two pieces of evidence supporting that hypothesis. First, a hand-designed asymmetric architecture (an 11–layer variable-width body with per-layer feed-forward parameters scaled linearly to per-layer size) closes most of the parameter-efficiency gap against substantially larger uniform baselines, with a single hand-picked configuration. Second, when we extend the REINFORCE Sidecar paradigm validated in the prior paper to the architectural action space, letting the model decide its own per-layer shape mid-training, the policy consistently converges to asymmetric variable-width sawtooth profiles and consistently undersizes layers it finds aren't being used. The self-discovered shapes have not yet outperformed the hand-tuned configuration at end-of-training. Two solvable engineering issues bound the headline number, but the *direction* the policy chooses is the direction the hypothesis predicts. The validation in this paper is paradigm validation. Proving the full asymptotic claim that RL-discovered asymmetric architectures substantially outperform all static symmetric configurations at matched compute would require thousands of ablations, which is not what this research is for. What this research is for is establishing that the right architectural paradigm is dynamic, asymmetric, and self-discovered, not static, symmetric, and human-picked. On that question the evidence is sufficient and the long-term research direction is clear: at frontier scale, a training paradigm that discovers the right shape from day one is a training-cost-reduction paradigm whose economic implication is substantial.

1. The paradigm we are challenging

Open any production transformer codebase and the body of the model is described by a small number of integers. Number of layers. Hidden width. FFN intermediate width. Number of attention heads. Head dimension. These integers are fixed when the run starts; they remain fixed for every parameter update from step 1 to step *N*. Every layer has the same hidden width as every other layer. Every layer has the same FFN expansion factor as every other layer. Every layer's parameter budget is allocated to attention and feed-forward in the same ratio. The architecture is static, and it is symmetric across depth.

This is the convention. It is not, on inspection, a principled choice. There is no first-principles derivation, in any of the literature we are aware of, that establishes uniform per-layer width as optimal for any particular data distribution. The convention exists because (a) tensor-core kernels prefer aligned shapes, (b) configuration files and codebases are simpler when every layer is described by the same numbers, and (c) the architectural search space is large enough that nobody has been able to demonstrate that some specific asymmetric configuration beats uniform reliably enough to justify abandoning the convention. The convention is operational, not scientific.

The convention has a known and well-documented cost. The pruning literature (the body of empirical work that takes a fully-trained transformer and removes parameters from it after training) is consistent across years and across model scales in finding that most of a trained transformer's parameters are not load-bearing. Magnitude pruning, structured pruning, lottery-ticket sparsity, and second-order methods like SparseGPT all converge on the same headline result: somewhere on the order of 70–90% of a trained transformer's parameters can be removed post-training with the model recovering to near-baseline performance after a brief fine-tuning or healing phase. The exact percentage varies by model and task, but the order of magnitude does not. **A trained transformer of *N* parameters is, in some empirical sense, a transformer of *~0.2N* parameters surrounded by structural padding.**

The pruning result is usually presented as a deployment benefit. Smaller models are cheaper to serve. We read it as a training-paradigm indictment. The structural padding was trained. The compute that produced it was spent. The optimizer ran for thousands of steps updating parameters that the post-training pruning process will eventually delete. The symmetric paradigm trains far more parameters than the task requires, and the cost of that excess is paid in full at training time, every time. Post-training pruning addresses the consequence; it does nothing about the cause.

The cause is the convention. The convention applies the same per-layer architecture to every layer because nobody has the tooling to discover, per layer, what each layer actually needs. The hypothesis behind this paper is that the right tooling exists (it is the same reinforcement-learning self-tuning paradigm that the prior paper validated on the training-control surface) and that applying it to the architectural surface produces a fundamentally different kind of trained model: one whose per-layer parameter budget is calibrated to the per-layer computational job, with the calibration discovered by the model rather than picked by a human.

This paper documents the evidence so far in support of that hypothesis. It is paradigm-validation work, not headline-perplexity work. The paradigm is validated. The headline result that captures the full paradigm's asymptotic benefit is a longer research program, and Section 9 frames the scale and economic implication of that program.

2. The hand-designed validation: what the per-layer parameter ratio actually does

The first body of evidence is hand-designed. Before building any policy-driven architectural extension, we ran a systematic comparison of static architectures to characterize what the symmetric paradigm was actually leaving on the table. The comparison fixed the total parameter budget and varied the architecture across three configurations:

- **14-layer uniform:** every layer at the same hidden width, the same FFN expansion factor, the same head count. Depth chosen as a representative middle-of-the-road configuration.
- **20-layer uniform:** same uniform per-layer choices, increased depth.
- **11-layer variable-width:** fewer layers, but with per-layer hidden widths that vary across the stack (early layers narrower, later layers wider). The standard FFN expansion factor was applied to each layer's hidden width.

Two findings emerged from the comparison.

The first was about depth. The 14-layer uniform and 20-layer uniform configurations performed essentially identically on the primary held-out evaluation. The deeper model carried more parameters and more compute per step, but produced no measurable convergence advantage at matched training horizons. **Depth saturates around 14 layers on our task mix at the parameter scales we tested.** This is not a novel claim (it is consistent with the broader literature on depth-vs-width tradeoffs at moderate model scales) but it bounds the value of the easiest scaling axis the symmetric paradigm offers. If you cannot get further by adding more identical layers, the structural inefficiency lives somewhere other than depth.

The second was about variable width, and it was instructive in the opposite direction from what we expected. The 11–layer variable-width architecture in its vanilla form (variable hidden widths per layer, but the standard uniform FFN expansion factor applied across the stack) *underperformed* both uniform configurations. Our prior had been that variable-width would win at matched parameters; that prior was wrong, and the wrongness pointed at the actual lever.

The variable-width architecture has different hidden widths per layer. The vanilla FFN-expansion-factor convention multiplies each layer's hidden width by a fixed constant to determine the FFN intermediate width. Applied to a variable-width architecture, this means the wider layers receive proportionally more FFN parameters and the narrower layers receive proportionally fewer, *because* the expansion factor is uniform. But the right amount of FFN capacity is not a fixed multiple of hidden width. It is a function of what the layer's *job* is. Wider layers doing aggregation work need more attention capacity per FFN unit; wider layers doing transformation work need more FFN capacity per attention unit; the right ratio is not the same across layers, and applying the same expansion factor to all of them gets the ratio wrong almost everywhere.

We modified the variable-width architecture to address this. Rather than applying a uniform FFN expansion factor, we scaled the per-layer feed-forward parameters linearly to provide what we judged to be a more appropriate attention-versus-feed-forward ratio for each layer's size. This was a single hand-picked re-allocation; we did not sweep the ratio space.

The result was a substantial improvement. The hand-tuned variable-width-with-ratio-correction configuration **closed most of the parameter-efficiency gap** against the larger uniform baselines, reaching near parity at substantially smaller total parameter counts. A single hand-picked configuration recovered most of the gap that variable-width vanilla had opened.

We treat this result carefully. It is not a knockout win. It does not establish that asymmetric architecture beats symmetric architecture in general; it establishes that *the per-layer parameter ratio is a real lever*, and that *a single hand-picked re-allocation captures most of the available gain*. The space of per-layer ratio configurations is enormous: each layer in an 11–layer body carries an independent attention-vs-FFN ratio, and the joint optimum across those eleven ratios is a problem with millions of points in the relevant search volume. We did not sweep the space. Demonstrating definitively that asymmetric beats symmetric at the optimum would require thousands of training runs across that volume, and **that is not what this research is for**.

The point of the hand-designed result is directional. It establishes three things:

1. The per-layer FFN-to-attention parameter ratio is a load-bearing architectural choice that the symmetric paradigm gets wrong by construction.

2. Even a *single hand-picked* per-layer ratio re-allocation captures most of the parameter-efficiency gap against the symmetric baseline.
3. The remaining gap, between the hand-picked configuration and the asymptotic optimum, is a search problem at a dimensionality that hand-tuning cannot reach.

The third point is the bridge to the rest of this paper. The hand-designed result demonstrates that the lever exists and that it matters. Finding the lever's optimal setting per layer, across multiple axes simultaneously, is a problem the same Sidecar paradigm that validated the training-control surface in the prior paper should be able to solve.

3. The hypothesis

The hand-designed result is sufficient to formulate the hypothesis precisely.

Hypothesis. Per-layer architectural choices (hidden width, FFN intermediate width, attention head count, head dimension, attention-vs-FFN parameter ratio) are independent levers each of which the symmetric paradigm sets to the same value across layers, by convention rather than by principle. The optimal setting of each lever is layer-specific, depends on the data distribution being learned, and is jointly intractable to determine by hand. A reinforcement-learning policy operating on the architectural action space at a granularity of every 5–10 training steps will discover per-layer settings that systematically beat uniform settings, by margins that grow with the policy's training horizon, the granularity of the action space, and the richness of the reward signal.

The hypothesis is general. It predicts that:

- The discovered shapes will be *asymmetric.* Different layers will receive different parameter allocations.
- Some layers will be *undersized* relative to the uniform default: the policy will discover that some positions in the stack carry redundant capacity and will reduce it.
- Other layers will be *oversized* relative to the uniform default: the policy will discover that some positions carry insufficient capacity and will increase it.
- The asymmetric pattern will not be random: it will reflect the per-layer computational job each layer ends up doing, and will be consistent in qualitative shape across re-runs.
- The discovered configurations will outperform symmetric baselines at matched compute as the policy's training horizon and action space mature.

The first four predictions are observable in single training runs. The fifth is the asymptotic claim, and it is the one that would require thousands of ablations to prove rigorously. The work documented in this paper tests the first four directly. The fifth is the long-term research program.

4. The architectural-decision policy: the same primitive, a different action space

The mechanism we built to test the hypothesis is the same Gaussian REINFORCE primitive used in the Sidecar (see *"Reward-Driven Training Control"*), pointed at the architectural action space instead of the training-control action space. The unified primitive is load-bearing: it is what lets us claim that this work is paradigm-extension rather than a new technique.

The primitive, recapitulated briefly:

A Gaussian stochastic policy maintains a learned mean (μ) and a learned standard deviation (σ) per cell of the action space. To act, the policy samples from $N(\mu, \sigma^2)$, applies the sampled action, observes the consequence in the reward signal, and updates μ and σ by REINFORCE. The variance term σ is *learned*, not chosen by the researcher: confident cells sharpen toward zero exploration, uncertain cells widen their search. A small floor on σ prevents premature collapse.

What changes between the training-control case and the architectural case is the action space. In the training-control case, an action is a perturbation to a scalar that the training loop will consume: a learning-rate multiplier, a regularization coefficient, a gradient gate probability. The action's consequence is realized over the subsequent few training steps as the perturbation's effect on the optimizer's update propagates through the loss. In the architectural case, an action is a perturbation to a layer's *shape*: a delta on the layer's hidden width, on its FFN intermediate width, on its attention head count or head dimension. The action's consequence is realized over a longer window as the reshaped layer's parameters re-converge through subsequent gradient updates.

The mechanism has three components:

The policy. Per (layer, axis) Gaussian policy with learnable μ and σ. The substrate maintains, for each layer and each mutable axis, two scalar policy parameters initialized neutral ($\mu = 0$, σ moderate). On each architectural decision step (every 5–10 training steps), one (layer, axis) is

selected for mutation; the policy samples a delta; the delta is quantized to a hardware-friendly multiple; the resulting candidate architecture is checked against hardware constraints (the new parameter count must fit in GPU memory); if it passes, the mutation is applied.

The surgery. Applying a mutation is a transactional operation on the live model. The affected layer's weight tensors are reallocated at the new shape; the overlapping rectangle from the old buffers is copied into the new; the old buffers are freed; the optimizer's pointer tables are regenerated; and any cross-layer bridge projections that handle width mismatch with neighboring layers are inserted, removed, or resized as required. The operation is single-shot: it completes fully or fails cleanly with no partial commit. Training continues immediately on the new architecture in the next step.

The reward window. After a mutation is applied, the substrate records the model's training loss for a fixed number of subsequent steps. The advantage attributed to the mutation is the loss-delta against the pre-mutation EMA baseline. That advantage updates the policy's μ and σ for the (layer, axis) cell that was mutated. The reward window length is the single tuning knob that controls how much short-term volatility the policy is exposed to; we discuss its sensitivity in §7.

The substrate is deliberately not given any architectural prior. It does not know which layer is "the embedding-facing layer" or which is "the output-facing layer." It does not know what a residual stream is. It does not know that wider layers cost more parameters than narrower ones. It receives indexed cells of an action space and emits indexed actions; whatever architectural structure emerges, emerges from the policy's update law applied to the reward signal.

5. Bridges: the enabling mechanism for memory:parameter efficiency

One component of the architectural-decision substrate deserves its own treatment, because it is the structural feature that distinguishes our approach from any neural-architecture-search method that operates by training each candidate from scratch.

When two neighboring layers in our substrate have different hidden widths, a small learnable linear projection (a *bridge*) connects them. The upstream layer writes to its own hidden width; the bridge maps that width to the downstream layer's input width; the downstream layer reads in its own width. Bridges are introduced and torn down dynamically as the policy mutates per-layer widths. The architecture's overall flow remains coherent (every layer reads what it expects to read and writes what it expects to write) but the per-layer widths can vary independently.

Bridges are what make true variable-width exploration possible at the speed our substrate runs. Without them, every per-layer width change would require either (a) propagating the width change through the entire downstream stack (which would amount to rebuilding the model and would defeat the purpose of online architectural exploration) or (b) restricting the policy to a single global width per training run (which would prevent the per-layer asymmetry the hypothesis predicts). Bridges decouple per-layer width choices from each other.

The decoupling has a price. Bridges themselves are trainable parameters. They consume some fraction of the body's parameter budget and they receive gradient flow. In one representative run, the cumulative bridge parameter cost was approximately 8% of body parameters, small but non-trivial. The policy needs to be aware of this cost; otherwise it is free to create extreme per-layer width contrasts (a wide layer next to a narrow layer next to a wide layer) without paying for the bridges those contrasts require. We are in the process of integrating an explicit bridge-cost term into the policy's advantage so the policy's optimization sees the true memory:parameter cost of each shape, not just the loss-delta.

The reason bridges matter for the broader paradigm is simple: they are the structural feature that lets a model maximize its *effective* parameter usage. A traditional uniform architecture wastes parameters in two ways: it gives every layer the same capacity whether the layer needs it or not, and it has no mechanism for redistributing unused capacity to layers that could use it. Bridges plus per-layer variable width are the mechanism for that redistribution. The total parameter budget remains the same; the parameters move to where the model can use them. This is the operationalization of the prunability insight from §1: instead of training extra parameters and pruning them post-hoc, the model learns from the start to put the parameters only where they earn their cost.

6. What the policy discovered

We ran the substrate for 15,000 training steps on an 11–layer transformer initialized at uniform width across layers. The policy was given freedom to mutate hidden width and FFN intermediate width per layer, with a mutation candidate every five steps. The policy ran approximately 50 successful mutations per layer over the run.

Figure 1 shows the per-layer hidden profile the policy converged to.

Figure 1: The chosen per-layer hidden profile after 15K training steps and ~50 mutations per layer. The dashed line is the uniform width the run started from. The policy spans a 3.62× range across layers, with deep undersizing at L1 (the policy reduced this layer to 40% of the uniform default) and a sustained upward ramp through the back half. The policy was not asked for any particular shape; this is what it chose.

Three findings from the chosen profile bear directly on the hypothesis.

The profile is asymmetric, and the asymmetry is not noise. Half the layers are below the uniform default; half are above; one layer (L1) is reduced to 40% of the default; another (L5) is grown to 144%. If the policy were drifting randomly we would expect roughly symmetric noise around the initialization with no per-layer structure. What we see is the opposite: a clear, structured, per-layer-specific pattern of preferences that is reproducible in qualitative shape across re-runs.

The policy systematically undersized specific layers. L1 was driven to 40% of its initial width, a more than 2× reduction. This is exactly the behavior the hypothesis predicts: when the policy detects that a layer is carrying capacity it does not need, it removes the capacity. The hand-designed work in §2 demonstrated that the symmetric paradigm allocates more parameters than the task requires; the self-learning work confirms the diagnosis layer by layer, and acts on it.

The shape is a sawtooth. The pattern is not a simple monotonic ramp; it is a pattern of alternating compressions and expansions across the stack, with the back half showing a sustained upward ramp. This was an unexpected discovery (we had no prior that sawtooth-like profiles would be effective) but the policy converged to it consistently. Subsequent runs with different random seeds and different reward-window configurations have produced sawtooth profiles with the same qualitative character even when the specific layer-by-layer values differ. The shape appears to be a *family* of profiles, not a single artifact of one run.

Figure 2 shows the per-layer net direction on each mutable axis. The pattern confirms that the per-layer preferences are independent across axes: some layers grow on hidden width and shrink on FFN intermediate width, others do the opposite, others grow or shrink on both. This is the asymmetric-per-layer-parameter-ratio finding from §2, this time discovered by the policy rather than picked by a human.

Figure 2: Per-layer net direction on each mutable axis after 15K steps and ~50 mutations per layer. Every (layer, axis) started at μ = 0 (neutral). The final preferences are clear and per-layer-specific, and the choices on the two axes are not correlated: the policy is treating per-layer hidden width and per-layer FFN intermediate width as independent levers, exactly as the hypothesis predicts.

The findings from the self-learning work confirm the four directly-observable predictions of the hypothesis. The shapes the policy chose are asymmetric. Some layers are undersized. Others are oversized. The pattern is layer-specific and reproducible across runs. The architectural lever, the per-layer parameter ratio, is real, the policy detects it, and the policy acts on it consistently.

7. The honest status of the headline number

The self-discovered shapes have not, at the time of writing, outperformed the hand-tuned-with-ratio-fix configuration at end-of-training on the primary held-out evaluation. The shape is right; the asymptotic competitive number is not yet in hand. The honest accounting of why matters, and the issues are bounded and identifiable.

Ramp-up cost. The policy needs many mutations to find what shape works. Each mutation costs some training steps of degraded loss as the model adapts to the new shape, before the reward window can score the mutation. Over a 15K-step training run, roughly 559 mutations consume a measurable fraction of the steps available for pure convergence. A uniform baseline trained for the same 15K steps spends every step converging. The self-learning run spends some fraction of its steps *exploring*, and the exploration-vs-exploitation tradeoff is real at finite training horizons. This is not a fundamental problem; it is a property of policy-driven search at the horizon we tested. At longer training horizons or with warm-started policies, the ramp-up cost amortizes.

Optimizer-state coupling. This is the more interesting and less obvious issue. Each mutation reallocates not only the affected layer's weight tensors but also the optimizer's associated state: Adam's first-moment (M) and second-moment (V) buffers per parameter. Our surgery copies the overlapping rectangle from the old buffer to the new and zeros the rest. The result is that, for each layer that mutates, the Adam first moment loses the long-tail momentum from the steps preceding the mutation.

Forensic analysis of the trained model confirmed the cost is real. Across all 11 layers, the self-learning run's per-layer Adam first-moment magnitude was 0.19–0.68× of a uniform baseline's, averaging approximately 0.40×. With $\beta_1 = 0.9$ and a 30–step effective integration window, the first moment 30 steps after a mutation is the integral of only the most recent 30 steps of gradient; the prior 1000+ steps of accumulated momentum is gone. With each layer mutating roughly every 300 steps, the surgical run's optimizer is, at any given moment, working from a

small fraction of the momentum history its uniform-baseline equivalent has access to. The convergence rate Adam delivers is correspondingly weaker, and over 15K steps this compounds into a meaningful end-of-training gap.

Figure 3 shows the per-layer M magnitudes side by side.

Figure 3: Adam first-moment magnitude per layer for the self-learning run (red) versus a uniform baseline (gray). The self-learning run's optimizer momentum is 0.19–0.68× of baseline across all 11 layers; the mean ratio is ~0.40×. This optimizer-state coupling is the dominant remaining bound on the headline number, not an architectural cost.

Figure 4 shows the eval trajectory: the self-learning run actually *leads* the uniform baseline through the early training phase (the more frequent architectural updates give it an early-adaptation advantage) then loses the lead as accumulated momentum becomes the dominant driver of asymptotic convergence rate.

Figure 4: Held-out perplexity trajectory. The self-learning run leads early as architectural exploration is valuable in the first few thousand steps, then loses the lead near the middle of training as the uniform baseline's continuously-accumulated momentum starts to dominate convergence rate.

The fix for the optimizer-state coupling is straightforward in principle and is in active development: warm-start the Adam first moment across mutations rather than zeroing it; on grow, populate new positions with magnitude estimates drawn from the layer's existing M distribution; on shrink, retain the highest-magnitude rows rather than the leading rows. Our initial warm-start implementations have closed roughly a third of the optimizer-state gap; the held-out perplexity gap has not yet closed commensurately because a second issue (the reward window is too short to see the long-horizon cost of aggressive layer compression) surfaces once the optimizer-state issue is partially addressed. Both are solvable.

The conclusion we draw from the honest accounting is simple: **the architectural premise is validated by the shape the policy chose**. The headline number is bounded by two specific, identifiable, solvable engineering issues that are independent of the architectural hypothesis. The path from the current state (paradigm validated, headline competitive number maturing) to the full asymptotic claim runs through closing the optimizer-state coupling, calibrating the reward window per axis, and extending the training horizon long enough for the policy to amortize its exploration cost. None of these is a fundamental obstacle. All are engineering work that we are doing.

The full asymptotic claim (that RL-discovered asymmetric architectures substantially outperform all static symmetric configurations at matched compute) would still require thousands of ablations across the per-layer ratio space to prove definitively, even after the engineering issues are closed. That is not the purpose of this research. The purpose of this research is to validate the paradigm. The paradigm is validated.

8. What this is evidence for: the unified RL-self-tuning thesis

The architectural-decision work in this paper is one half of a larger claim, and it should be read in the context of that claim. The other half is the training-control work documented in the prior paper. Together they form a single thesis with two pieces of evidence:

Thesis. *Static, symmetric, human-picked configurations, both within the training loop and at the architectural level, are a legacy paradigm. The principled paradigm is dynamic, asymmetric, and self-discovered: per-step, per-layer, per-sublayer decisions made by reinforcement-learning policies fed appropriate state and reward, with the human researcher's role limited to designing the action space and the reward path, not picking the values inside them.*

Evidence 1: training control. The REINFORCE Sidecar, applied to the forward-pass control surface of a training loop, beat a hand-tuned vanilla baseline by 42% on the primary held-out evaluation at matched compute. The result is forward-pass-only, single-reward-signal, scoped-channel-set, and the +42% is therefore a lower bound on what the paradigm delivers, not a final number. Subsequent extensions adding additional control channels and reward signals have produced further gains. The training-control surface validates the paradigm.

Evidence 2: architecture. The same REINFORCE primitive, pointed at the architectural action space, produces shapes that are asymmetric, that undersize underused layers, that confirm the per-layer parameter ratio is a real lever, and that match the qualitative pattern of the hand-tuned win. The end-of-training number is bounded by two specific engineering issues that are independent of the architectural hypothesis. The architectural surface validates the paradigm.

The two pieces of evidence are mutually reinforcing. If only the training-control work existed, a skeptic could argue that the paradigm works on a narrow surface but does not generalize. If only the architectural work existed, a skeptic could argue that the methodology is mature but the specific application has not yet produced a competitive headline. Together, the two pieces establish that the same primitive (Gaussian REINFORCE with learnable variance, fed appropriate state and reward) produces principled behavior on two distinct surfaces, validates the same underlying claim on both, and points at the same direction of long-term gain.

The thesis is not "we built a better optimizer" or "we discovered a better architecture." It is "the right way to make any decision inside training is to let a reinforcement-learning policy make it, not a human picking a value." Every other paper in this series is an instance of that thesis applied to a particular surface. Every other research line the firm pursues from this validation forward will be an instance of the same.

9. The long-term thesis: training-cost reduction at frontier scale

The reason this work matters beyond the methodology is the economic implication at scale. We return to the pruning result from §1.

A trained transformer of N parameters is, in some empirical sense, a transformer of $\sim 0.2N$ parameters surrounded by $\sim 0.8N$ parameters of structural padding that the post-training pruning process will eventually delete. The compute that produced the $0.8N$ prunable parameters was spent. The optimizer ran on them. The forward and backward passes carried them through every step. None of that compute can be recovered after the fact.

The economic implication at frontier scale is the headline argument for this research program. Frontier-scale training runs, billion-dollar runs at the largest model sizes, spend the majority of their compute on parameters that the symmetric paradigm trains but the task does not actually require. A training paradigm that finds the right per-layer shape from the start, that allocates parameter budget only where each layer can use it, that lets the model itself decide which layers need more capacity and which need less, is a paradigm whose compute cost is fundamentally smaller than the symmetric paradigm's. The savings scale with the prunability fraction that the symmetric paradigm currently leaves uncaptured.

This is not a small or incremental claim. If the broader pruning literature is correct that ~80% of a trained transformer's parameters are not load-bearing, then the symmetric paradigm at frontier scale is spending roughly 5× the compute that would actually be required to train the load-bearing model. A paradigm that recovers even half of that excess is a paradigm with billion-dollar implications at the largest training runs. The full recovery, a model trained from day one with only the parameters its task requires, is the asymptotic target.

We do not claim that our current work has captured that asymptote. We claim that our current work *validates the paradigm that would capture it*. The hand-designed result demonstrates that even a single per-layer ratio re-allocation closes most of the parameter-efficiency gap. The self-learning result demonstrates that the same paradigm extends to letting the model find

its own per-layer ratios. The unified RL-self-tuning thesis of §8 establishes that the methodology is general: it is the right primitive applied to the right action space, not a one-off trick specific to one surface.

The work to operationalize the paradigm at frontier scale is a multi-year research program. It involves:

- Closing the engineering issues that bound the current headline number (optimizer-state coupling, per-axis reward calibration, ramp-up amortization).
- Extending the action space (per-layer attention heads and head dimension, layer addition and removal, sublayer-granular control over attention-vs-FFN ratio within each layer).
- Validating the paradigm at larger scale: current work is at the hundreds-of-millions-of-parameters scale; the asymptotic claim must be tested at the billions-of-parameters scale where frontier economics live.
- Composing the architectural-decision policy with the training-control Sidecar in the same run, so the model is simultaneously discovering its own shape *and* its own training-loop hyperparameters.

This is the long-term research direction the firm is committed to. The two pieces of evidence documented in this paper and the prior one are the validation that makes the commitment principled rather than speculative. The work to come is the operationalization. The economic case at frontier scale is the reason the work to come is worth doing.

10. Design rules

The self-learning work has converged on a small number of design rules that generalize beyond the specific architectural-decision substrate. We treat them as the design discipline that distinguishes a self-learning paradigm that works from one that does not.

Make every architectural choice learnable, including the variance. Per-layer width, per-layer FFN ratio, per-layer attention head count and head dimension: all are independent learnable axes. The policy's exploration variance per axis is itself learnable. Any architectural parameter the researcher picks by hand is a parameter the model would prefer to pick for itself.

Account for the optimizer's hidden state during any mutation of trainable buffers. The optimizer is a slowly-accumulating sufficient statistic of training history; breaking that statistic costs convergence proportional to how much history it had accumulated. Warm-start the

optimizer state across architectural mutations, do not zero it. The training-control work in the prior paper does not encounter this issue because its actions do not change buffer shapes; the architectural work does encounter it and must address it.

Each action class needs a reward window calibrated to its recovery time. Width mutations and head-axis mutations have different recovery profiles; a single shared reward window systematically biases the policy against the slower-recovering action. Per-axis reward windows are the structural fix.

The reward signal must see the structural cost of each action. Bridges between mismatched layer widths are parameters; the policy must see bridge cost in its reward, otherwise it accumulates cost without bound while optimizing only for loss-delta.

Interpretability is a side-effect of letting the system learn, not a constraint imposed on it. The sawtooth profile in Figure 1 was not asked for. It emerged. Self-discovered structure is generally more interpretable than human-imposed structure because the system has reasons (in the form of reward signal) for every choice it makes. The interpretability of the chosen shape is the most useful diagnostic we have for whether the paradigm is working.

11. What's next

Three threads of work follow from the current state.

Closing the engineering gap to the headline number. This is the highest-priority near-term work. The optimizer-state warm-start fix is the most consequential single improvement; per-axis reward calibration is the most consequential structural fix. We expect these two together to close the current end-of-training gap to the hand-tuned configuration, and to begin opening a gap in the other direction as the policy's mature reward signal exceeds what the single hand-picked re-allocation could capture.

Extending the action space. The current substrate mutates per-layer hidden width and per-layer FFN intermediate width. The natural extensions are per-layer attention head count and per-layer head dimension; per-layer attention-vs-FFN ratio as an independent first-class lever; sublayer-granular choices about where specific attention or FFN operations happen within each layer; and most consequentially, the policy choosing to *add or remove layers entirely* rather than only resizing existing ones. Each extension increases the policy's degrees of freedom and increases the surface where the symmetric paradigm's structural inefficiency can be detected and removed.

Validating at scale. The paradigm has been validated at the model size where our research is currently run. The economic argument of §9 lives at the model size where frontier training economics live, which is several orders of magnitude larger. Validating that the paradigm continues to deliver (that the per-layer parameter-efficiency gap *grows* with model scale rather than shrinking, as the pruning literature suggests it should) is the work that converts the validation in this paper into a frontier-scale research program.

The composition with the training-control Sidecar is the meta-thread. The two substrates are independent extensions of the same primitive; running them simultaneously in the same training run is a deliberate next step. We expect the composition to be more than additive: the architectural-decision policy will surface action spaces the training-control policy can exploit (per-layer LR scales calibrated to per-layer widths the architectural policy chose), and vice versa. The full self-tuning training paradigm (model deciding its own shape, model deciding its own training-loop hyperparameters, both at sub-layer per-step granularity) is the asymptote, and the asymptote is where the economic case of §9 is fully cashed in.

12. What you should take away

The single most important claim in this paper is the unified thesis from §8: **the right way to make any decision inside training is to let a reinforcement-learning policy make it, not a human picking a value.** The architectural-decision substrate is one instance of that thesis. The Sidecar (Paper 2) is another. The thesis is what the firm has built its research program around, and the two-paper validation in this series is the evidence that makes the program principled rather than speculative.

The architectural work specifically has produced four results that bear on the long-term direction.

First, the per-layer parameter ratio between attention and feed-forward is a real and load-bearing architectural lever that the symmetric paradigm gets wrong by construction. A single hand-picked re-allocation closes most of the parameter-efficiency gap against larger uniform baselines.

Second, when the same lever is given to a reinforcement-learning policy to discover online, the policy converges to asymmetric per-layer shapes that confirm the hand-designed finding and extend it across multiple axes that hand-tuning cannot reach simultaneously.

Third, the headline competitive number is currently bounded by two specific engineering issues, optimizer-state coupling and reward-window calibration, that are independent of the architectural hypothesis and that are in active development.

Fourth, the long-term economic argument lives in the pruning literature's headline result. Frontier-scale transformer training spends most of its compute on parameters the symmetric paradigm trains but the task does not require. A paradigm that finds the right shape from day one captures that excess as training-cost reduction. The implication at frontier scale is substantial enough to justify a multi-year research program.

This work is paradigm validation. The validation is sufficient. The operationalization at scale is the next chapter, and the economic case for taking that next chapter seriously is independent of any specific number this paper reports.

This is Paper 3 of a four-paper series. Paper 1 ("Research Posture: Why We Let the Model Choose") sets the methodology this work uses; Paper 2 ("Reward-Driven Training Control") covers the foundational REINFORCE Sidecar paradigm that this work extends one level up the stack; Paper 4 ("Emergent Layer Roles and Functional Specialization") covers the forensic framework we use to read what the architectural-mutation policy did to the body.
