What's actually doing the work in a small encoder-decoder code model: a matched-budget leave-one-out at 1.2M trainable parameters

Claude Opus, Nikita Sushko

Draft, August 14, 2026

Abstract

We study a small encoder-decoder model (∼1.2M trainable parameters) on a constrained-vocabulary Python completion benchmark, and ask which architectural and training-time choices are actually responsible for its performance under matched parameter budget. We compare two configurations of the model, named V1 and V2, that differ along six knobs: encoder coupling (mean-pooled prefix vs. cross-attention), positional encoding (absolute vs. RoPE), MLP family (GELU vs. SwiGLU), output head (tied embedding vs. a 26-letter copy head), initialization (random vs. decoder-LM warm-start), and two data-side training filters. Holding trainable parameter count fixed at 1.2M ± 5 %, we run a seven-cell V1→V2 progression, an eight-cell V2 leave-one-out, a V1 eight-knob rerun, a BPE×coupling factorial, and a continued training experiment, and we evaluate every cell on a frozen 224-problem suite with embedded leak-audit columns. At single seed, only the GELU→SwiGLU substitution resolves outside the bootstrap CI of the EASY pass@1 metric; warm-start, copy-head, the two data filters, and absolute vs. rotary positional encoding all sit inside the ±0.05 noise floor and are reported as "not observed" rather than positive or negative; cross-attention resolves only when read across cohorts. The most informative thing the matched-budget grid surfaces is what no knob in the paper can move: `exec_error`, the share of generations that crash before the test assertion, sits at ∼ 0.40 on every cell of every cohort. We report this row as the actual ceiling at 1.2M trainable parameters on this corpus and propose decode-time probes (temperature, syntax pre-check, repetition penalty) as the cheapest next move. All quantitative claims in this paper are single-seed; multi-seed replication is the highest-priority follow-up and is described in Section 9.

1 Introduction

TinyPython is a constrained-vocabulary code-completion benchmark in which a small encoder-decoder model (∼1.2 M trainable) is trained on a synthetic corpus of short Python functions and evaluated by exact-match unit-test execution. We study two configurations of this model, named V1 and V2. The V2 configuration outperforms V1 on this benchmark by a wide margin: at a comparable parameter budget, V2 EASY pass@1 is roughly 0.13 above the best V1 cell on the same data and same eval set (Section 7.1). The two configurations differ along six independent design knobs — encoder coupling, positional encoding family, MLP non-linearity, output-head structure, initialization, and two data-side training filters (Section 3, Table 2). The question this paper asks is which of those knobs are actually responsible for the gap.

The natural temptation is to read off knob attributions from end-to-end V1-vs-V2 contrasts, but that view is confounded: any single V1-vs-V2 cell-pair changes parameter count, architecture, and training-time data interventions all at the same time. The only claim such a contrast genuinely supports is that the V2 configuration *as a whole* beats the V1 configuration as a whole; not that any specific ingredient is what is doing the work.

This paper closes the attribution gap. We hold the trainable parameter budget fixed at 1.2 M ± 5 % across all ablations,[1] freeze the eval set as TINYPYTHON_EVAL_V2 (224 problems with leak-audit columns embedded

[1] Enforced pre-launch by `scripts/check_param_target.py` reading the `param_breakdown.json` sidecar that every train run emits; see Section 2.

per task), fix the decoding configuration ($T = 0.8$, top-$p = 0.95$, 32 samples per task), and run the V1 - vs - V2 contrast as five separate cohorts:

- Section 4: a seven-cell V1→V2 **progression** that flips one knob at a time at fixed budget.
- Section 5: an eight-cell V2 **leave-one-out** that turns one V2 knob off at a time, again at fixed budget.
- Section 7.1: a V1 **eight-knob rerun** trained on the new clean holdout and benched on the frozen eval set, so the V1 leg of the contrast lives on the same harness as the V2 leg.
- Section 7.2: a 2×2 **BPE × coupling factorial** that tests whether the 98-token vocabulary is itself the bottleneck V1 hits.
- Section 7.3: a **+1 epoch resume** of a larger V2 cell, as a sanity check on training horizon.

The single-seed bootstrap CI on EASY pass@1 at this sample budget is ±0.05 (32 samples × 20 tasks; see Section 2.3). We use that band consistently throughout the paper to classify each delta as *outside-CI* (resolved at single seed), *within-CI* ("not observed", *n. o.*), or *within-noise floor* (deltas on suites such as HumanEval where the absolute pass rate is so low that a 1–2-pass difference between two cohorts is not a discriminative signal).

Headline finding (Figure 1). Under matched-budget single-knob isolation, the V1→V2 EASY pass@1 gap of +0.243 is real, but the only knob whose individual contribution we can resolve at single seed is SwiGLU (a +0.317 move at the relevant transition; see §4.1). Every other single-knob delta in the progression — cross-attention alone, RoPE alone, copy-head, warm-start, the two data filters — sits inside or at the boundary of the bootstrap CI and is reported as *n. o.*. Of these, the data filters are the cleanest single piece of negative evidence: turning them on at the end of the progression moves EASY pass@1 by exactly +0.000 (Section 4.2).

Most informative non-result. The flat ∼0.40 `exec_error` row across every cell of every cohort (Figure 5) is the new finding the matched-budget grid actually surfaces. No architectural knob, no vocabulary, no data filter, and no warm-start in this paper moves the rate at which a generated program crashes before reaching the test assertion. We argue in Section 6 that this row, not the EASY pass@1 deltas, is the binding constraint of a 1.2 M trainable budget on a TinyPython-shaped corpus, and we propose decode-time interventions (Section 9) as the first place to look before any retraining-based fix.

Single-seed caveat. Every cell in this paper is a single seed. The +0.317 SwiGLU jump at the relevant transition is large enough to be outside the sample-statistic CI, but not large enough to rule out strong knob-interactions in the progression ladder being the cause. Multi-seed replication of the progression and the LOO is the first follow-up we list in Section 9, and it is required before any of the single-knob attribution claims here are cited as established.

2 Methodology

This section describes the four pieces of infrastructure that the single-knob attribution above relies on: a frozen eval set with embedded leak-audit columns, a 6-category failure schema, pass@k with bootstrap CIs, and per-module parameter accounting with a pre-launch budget check.

2.1 Frozen eval set tinypython_eval_v2

TINYPYTHON_EVAL_V2 is a 224-problem suite stored at `data/eval/tinypython_eval_v2/` comprising EASY (20), MEDIUM (20), HARD (20) and HumanEval [1] (164). Each row carries five audit columns populated by `src/tiny_decoder/data/leak_audit.py` against the TPY07B clean holdout (141 125 train rows / 2881 val rows): `name_seen_in_train`, `template_family_seen`, `nearest_train_jaccard`, `expressible_t98`, `expressibility_label`. The Jaccard column is the best of three shingle measures (word-3, char-5, AST-skeleton) generated via MinHashLSH candidates and then exact-scored. Counts after audit are in Table 1.

The frozen set ships with a `metadata.json` and `hashes.json` so any subsequent build can verify it is scoring against the exact set we did.

Table 1. Eval-set composition after the leak audit. `name_seen` is the number of tasks whose `entry_point` collides with any `def` name in the training corpus; `nearest_jaccard` ≥ 0.30 is the count in the non-trivial similarity bands; `expressible` is the count of tasks reachable by the 98-token vocabulary.

suite	n	name_seen	nearest_jaccard $\geq$0.30	expressible
EASY	20	11	8	20
MEDIUM	20	0	5	20
HARD	20	0	5	20
HumanEval	164	0	0	91
total	224	11	18	151

The audit columns enable a one-line query for performance-by-similarity-bucket (`reports/performance_by_similarity_buc` that separates leakage-driven gains from generalization-driven gains on the EASY/MEDIUM/HARD slice. We use this in Appendix C.

2.2 Failure-category schema (six buckets)

Every generated sample is categorized into one of six disjoint top-level buckets:

- `format_ok` — generation parses, runs, and the test assertion holds.
- `bad_format` — generation does not parse at the required entry-point boundary or fails the framing check.
- `stub_body` — generation parses but is a `pass`-only body or a single-token return; broken out so the format / understanding distinction is legible.
- `test_fail` — generation parses, runs to completion, and the test assertion is false (the model produced the wrong answer).
- `timeout` — generation runs but does not return within the per-task limit.
- `exec_error` — generation parses but raises a Python runtime error before or around the assertion (e.g. `NameError`, `TypeError`, `IndexError`, infinite recursion). The runtime-error case is kept separate from `test_fail` (assertion false) because, as we argue in Section 6, its invariance-across-cells is the most informative signal in the paper.

`cli/bench.py` writes a sidecar `<bench>.failure_summary.csv` with per-suite and overall rates of each category for every run.

2.3 pass@k and bootstrap CIs

Every cell is benched at $k \in \{1, 5, 10, 32\}$ with the unbiased Chen et al. [1] estimator, with $T = 0.8$, top-p $= 0.95$, and 32 samples per task. Bootstrap CIs on the per-suite mean are computed by resampling task indices once per replicate (1 000 replicates) inside `scripts/aggregate_metrics.py`; paired CIs apply the same indices to both arms of a contrast and are written to `reports/paired_comparisons.csv`.

The 95 % bootstrap CI half-width on EASY pass@1 at 32 samples $\times$ 20 tasks is approximately ± 0.05 across the cells in this paper. We use this number as the noise floor below which a delta is reported as *n. o.* "not observed at single seed" rather than as positive or negative evidence.

HumanEval N-of-1 floor. The pass-rates we report on the 164-problem HumanEval suite (0–2 passes out of $164 \times 32 = 5\,248$ attempts) are at or near the N-of-1 floor where a 1–2 pass difference between two cohorts contains roughly one bit of evidence. We label these columns *within-noise floor* in every table; we do not place HumanEval pass@1 next to EASY/MEDIUM/HARD pass@1 as if they were on the same noise scale.

2.4 Per-module parameter budget

`src/tiny_decoder/util/param_breakdown.py` walks the `EncoderDecoder` module tree and assigns every `nn.Parameter` leaf to one of nine buckets: `decoder_self_attn`, `cross_attn`, `mlp`, `projection`, `embedding`, `lm_head`, `copy_head`, `layernorm`, `other`. The sum equals `count_trainable_parameters()`; the frozen encoder is counted separately. `cli/train.py` writes `param_breakdown.json` alongside every checkpoint. `scripts/check_param_target.py` runs before every train launch and refuses to start if the trainable count falls outside [1.14M, 1.26M]. The breakdown across the seven progression cells is shown in Figure 3.

2.5 Compute and seeds

8×H100 nodes, 1 cell per GPU, ≈15 minutes per cell at 3 epochs on the 141 000-row holdout, batch 128. Each cell uses `seed=42`; replication across seeds is the first follow-up listed in Section 9. Bench is fast (≤ 6 min/cell on a single GPU). The full set of training cells in this paper (∼27 cells) fits in roughly five 15-minute waves; an end-to-end run across the launchers in Appendix E took ∼1.5 GPU-hours of training.

3 V1 vs V2 as six discrete knobs

For the matched-budget single-knob analysis below, it is useful to describe V1 and V2 as six independent knobs, each with two settings. V1 is "all six knobs in their V1 setting"; V2 is "all six knobs in their V2 setting". The progression in Section 4 flips the knobs cumulatively from left to right; the leave-one-out in Section 5 flips them one off from the V2 anchor.

Table 2. The six knobs that distinguish V1 from V2, holding trainable parameter budget at 1.2 M. Knobs (1)–(4) are architectural, (5) is initialization, (6) is data-side. The boundary between architecture and data is enforced in the progression by labelling the last cell explicitly as a data intervention.

knob	V1 setting	V2 setting
encoder coupling	mean-pooled prefix (1 vector → `prefix_len`×d_{model})	full cross-attention to encoder hidden states
positional encoding	learned absolute	RoPE [3] (rotary, per-head)
MLP family	GELU (d_{ff}=4d_{model})	SwiGLU [2] (d_{ff} shrunk to keep parity)
output head	tied embedding only	26-letter copy-head bias derived from encoder context
warm-start	random init	decoder-LM pre-train then finetune with encoder attached
data filter	none	`drop_stub_bodies`=true + `weighted_sampling`=true

The matched-budget invariants held fixed across the progression and the LOO are: d_{model}=96, n_{layer}=3, n_{head}=4, `max_seq_len`=256, batch 128, LR 3×10^{-4} with cosine schedule, 3 epochs, T=0.8, top-p=0.95, 32 samples, seed 42, eval set TINYPYTHON_EVAL_V2. The only cell that flips a non-architectural knob without labeling it as such is `v2_base_full` in the progression, which corresponds to "architectural stack on + data flips on" — it is shaded as a data intervention in Table 3.

4 Headline experiment: matched-budget V1 → V2 progression

4.1 Cells

The progression is seven cells, each at 1.2 M ± 5 % trainable, plus a matched 0.57 M pair that tests whether the coupling effect is robust to scale.

Table 3. Per-suite pass@1 across the V1→V2 matched-budget progression. Rows are cumulative: each cell adds one knob from Table 2 on top of the previous cell. The bottom two rows are the C2 matched 0.57 M pair. Column "HumanEval" is shaded as *within-noise floor* per Section 2.3.

cell	trainable (M)	EASY	MEDIUM	HARD	HumanEval†	EASY solved@32
`v1_1p2m_prefix_only`	1.15	0.416	0.047	0.059	0.0000	0.750
`v1_5_xattn`	1.15	0.458	0.070	0.100	0.0002	0.650
`v1_5_xattn_rope`	1.20	0.305	0.092	0.097	0.0032	0.600
`v1_5_xattn_rope_swiglu`	1.20	0.622	0.119	0.097	0.0002	0.700
`v1_5_xattn_rope_swiglu_copyhead`	1.20	0.622	0.123	0.119	0.0011	—
`v1_5_xattn_rope_swiglu_copyhead_warm`	1.20	0.659	0.119	0.075	0.0019	—
`v2_base_full` (data flips on top)	1.20	0.659	0.103	0.078	0.0018	—
`v1_prefix_558k` (matched pair)	0.57	0.412	0.047	0.069	0.0000	—
`v2_xattn_558k` (matched pair)	0.57	0.553	0.072	0.091	0.0036	—

† Within-noise floor: 0–2 passes out of 164×32=5 248 attempts.

4.2 What the progression does and does not say

The single-seed bootstrap CI on EASY pass@1 is ±0.05 (Section 2.3). Read against that band, the progression in Figure 1 is best summarized as one outside-CI move, two within-CI moves, one transient regression we cannot yet explain, and one clean retraction.

Outside-CI: SwiGLU. The 0.305 → 0.622 jump from `v1_5_xattn_rope` to `v1_5_xattn_rope_swiglu` is +0.317. This is far outside the ±0.05 band. *Two qualifications matter.* First, the prior cell `v1_5_xattn_rope` is anomalously low (0.305): it sits below `v1_5_xattn` (0.458) even though RoPE is supposed to be a strict capacity-improving substitute for absolute positional encoding. The +0.317 jump is therefore partly the model leaving a bad loss basin, not +0.317 of "MLP-family value". Second, a +0.32 single-knob jump in a ladder where the claimed CI is ±0.05 is itself evidence of either CI underestimation of seed noise or strong knob-interactions. We list a multi-seed replication of this exact transition as the highest-value follow-up (Section 9).

Within CI, not observed. The +xattn cell (`v1_5_xattn` - `v1_1p2m_prefix_only` = +0.042), the +copy_head cell (+0.000 over the SwiGLU cell), the +warm_start cell (+0.037), and the +data_flips cell (+0.000 over the warm-start cell) all sit inside the ±0.05 band. We label each as *n. o.*. They may help, hurt, or do nothing; this single-seed progression cannot distinguish.

The data flips do nothing. The last row of Table 3, `v2_base_full`, flips the two data-side knobs on top of the all-architectural cell. EASY pass@1 does not move (0.659 → 0.659; MEDIUM goes 0.119 → 0.103 slightly down; HARD goes 0.075 → 0.078 slightly up). At this budget on this corpus the data-side filters make no difference once the architectural stack is in place.

Cross-cohort coupling effect. Cross-attention is within CI in the progression cell, but is outside CI in the C2 matched 0.57 M pair (`v1_prefix_558k` → `v2_xattn_558k` = +0.141), in the BPE factorial (≈ +0.17 in both vocabulary settings, Section 7.2), and in the V1-rerun-vs-v2_base contrast (Section 7.1). The cross-cohort agreement is what carries the cross-attention claim, not the single progression cell.

The total architectural delta is real. The endpoint-to-endpoint delta on EASY pass@1, from `v1_1p2m_prefix_only` (0.416) to `v1_5_xattn_rope_swiglu_copyhead_warm` (0.659), is +0.243. This is well outside the bootstrap CI; the *V2 architectural stack as a whole* is doing real work relative to the V1 stack. What the single-seed progression does not establish is which knob in the stack is contributing how much — exactly because so many of the intermediate single-knob deltas are within CI.

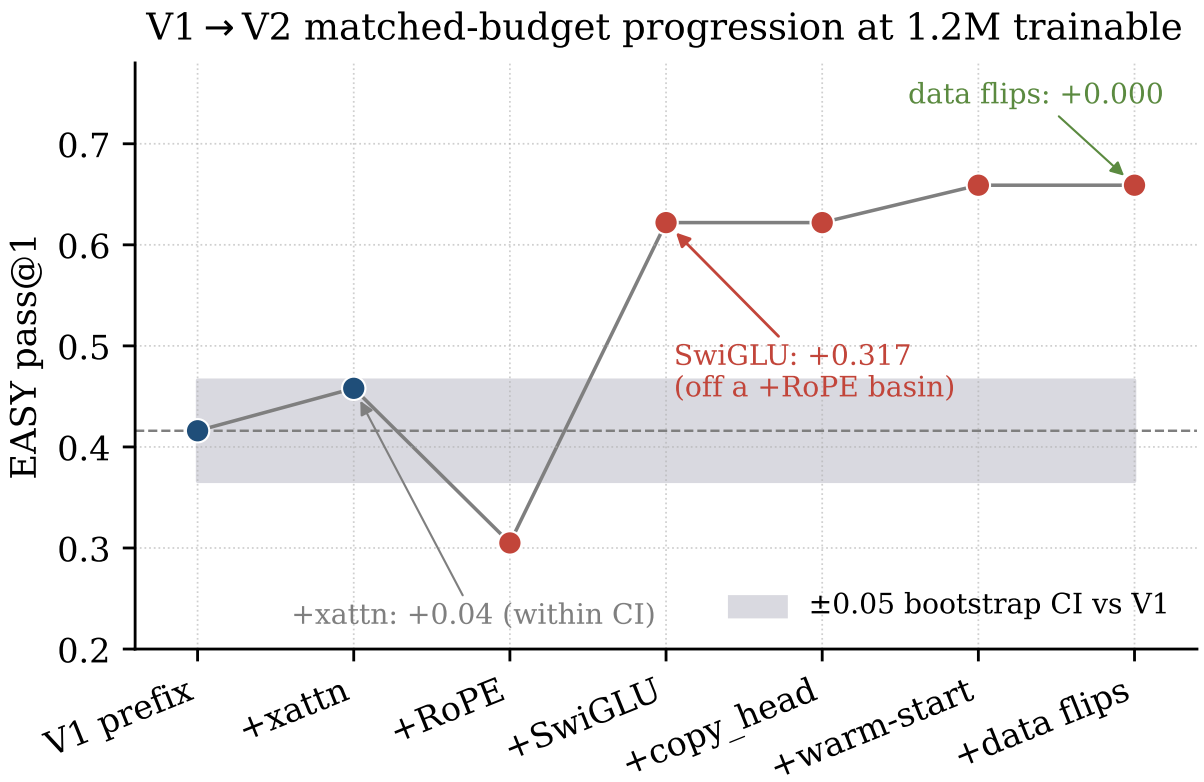

Figure 1. V1→V2 matched-budget progression. The shaded band is ±0.05 around the V1 baseline. Only the SwiGLU transition clears the band at single seed; cross-attention and the data flips do not.

4.3 Failure-category mix across the progression

Figure 2 shows how the six failure buckets shift across the progression. `format_ok` (the rate of generations that parse and pass at least one test, by sample) climbs from 4.7 % to 7.8 %, roughly proportional to EASY pass@1. The biggest reduction is in `bad_format` (15.0 % → 8.0 %), driven by SwiGLU + copy_head + warm-start. `exec_error` is essentially flat at ∼0.40 across every cell — this row is the subject of Section 6.

4.4 Per-module parameter accounting

Figure 3 shows the per-bucket parameter share for each progression cell. Every cell lands in [1.14 M, 1.26 M], the budget band enforced by `check_param_target.py`. The matched-budget claim is verified row-by-row.

5 V2 leave-one-out at 1.2 M

The progression in Section 4 flips knobs cumulatively from the V1 end. The leave-one-out is the complementary view from the V2 end: the anchor is the all-V2-knobs-on cell `v2_base`, and each ablation cell turns exactly one V2 knob off. Same data, same compute, same eval as Section 4.

5.1 Outside-CI: SwiGLU vs GELU

`v2_base_gelu` - `v2_base` = −0.116. This is the only LOO cell that clears the bootstrap CI; the direction agrees with the SwiGLU jump in the progression (Section 4.2). Both pieces of evidence point at SwiGLU as a real lever inside the V2 stack. We carry the magnitude with two caveats below.

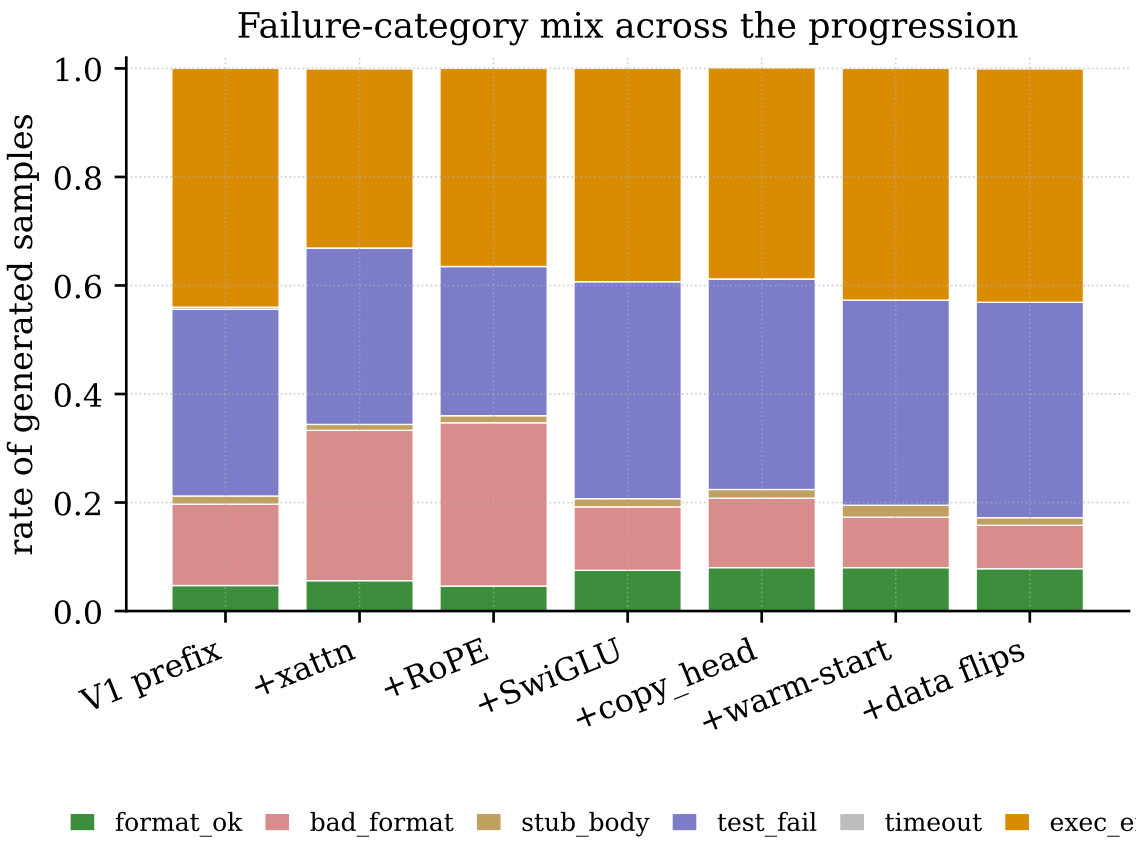

Figure 2. Failure-category mix across the V1→V2 progression. The `exec_error` band (orange) is approximately constant. The architectural progression rearranges samples between `format_ok`, `bad_format` and `test_fail` without touching the runtime-correctness bucket.

5.1.1 Capacity confound on the GELU cell

The GELU cell holds $d_{\text{ff}}=4d_{\text{model}}$ at the V1 default; under matched-other-knobs this lands at 1.14 M, −5 % from the 1.20 M anchor. From the matched 0.57 M / 1.20 M pair in Table 3 we know that halving capacity moves EASY pass@1 by ≈ 0.10, so a −5 % capacity gap should account for *at most* a ∼ 0.01 piece of the −0.116 drop. The remainder is genuinely the MLP-family substitution. Even so, the cell remains a confound on what is otherwise the paper's strongest single-knob result. The clean follow-up widens GELU's d_{ff} until the trainable count matches the anchor within ±2 %; we list this as a top-priority follow-up in Section 9.

5.2 Within CI: warm-start, copy-head, data filters

`no_warmstart` (+0.055), `no_copyhead` (−0.027), `uniform_sampling` (+0.089), `no_drop_stub_bodies` (+0.059), `abspos` (−0.024), and `maxseq128` (+0.052) all sit at or inside the ±0.05 bootstrap CI. We report each as *n. o.*: at single seed we cannot resolve the direction, let alone the magnitude. Three of these point the opposite way from what one might expect of a load-bearing V2 ingredient: turning warm-start off, dropping the data filters, and using uniform sampling all numerically improve EASY pass@1 in the table. *We do not call these net-negative.* Calling +0.055 or +0.089 "net-negative" inside a ±0.05 CI is overclaim in the wrong direction. The honest report is that warm-start, the copy-head, the two data filters, and absolute positional encoding are all *unresolved* on this seed: the data here cannot tell us whether these knobs help, hurt, or do nothing at 1.2 M on this corpus, only that whatever their effect is, it is below the bootstrap CI on EASY pass@1.

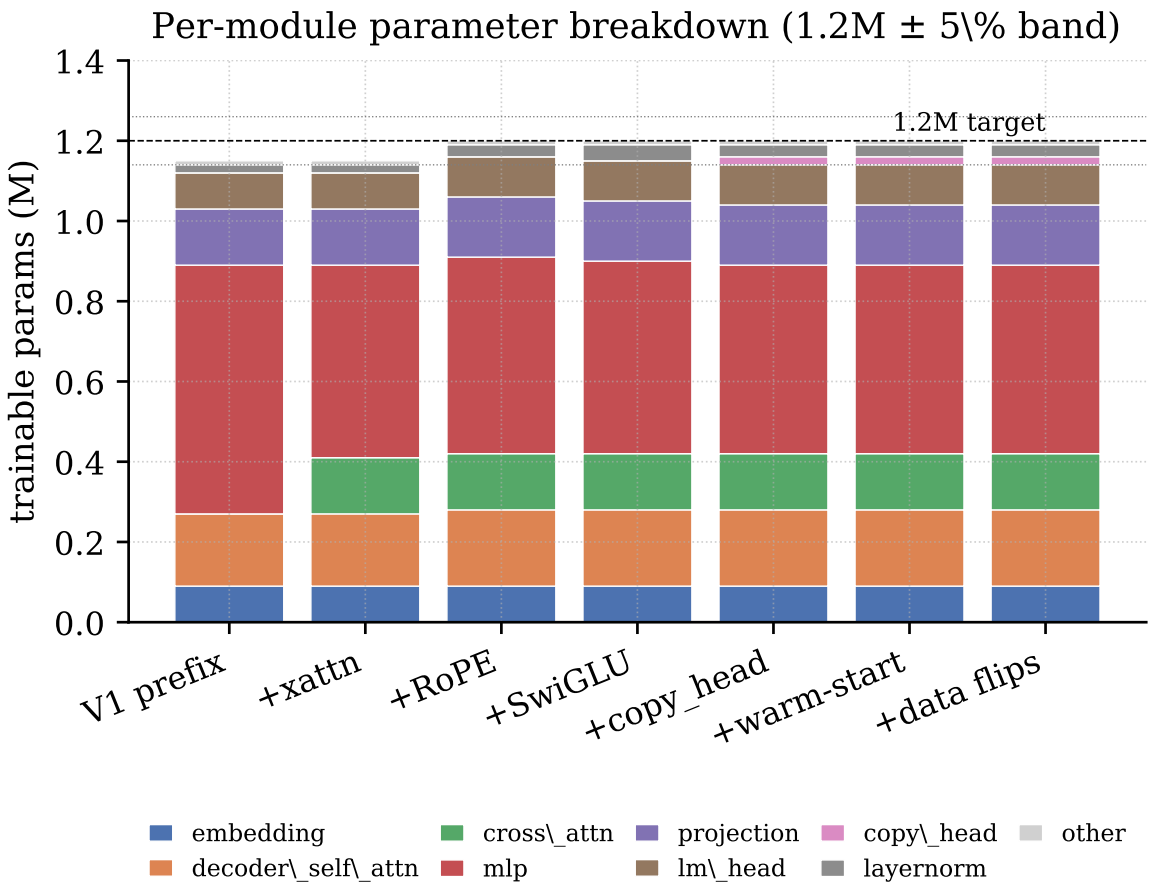

Figure 3. Per-module parameter breakdown across the progression cells. Dashed lines mark the [1.14 M, 1.26 M] ±5 % band around the 1.2 M target. Numbers come from the `param_breakdown.json` sidecar emitted by every train run.

5.3 What the LOO settles

The LOO and the progression agree on one resolved single-knob effect (SwiGLU $\gtrsim$ GELU, magnitude pending a clean rerun) and on the absence of resolved effects for everything else in the V2 stack at single seed. Combined with the V1-rerun cohort (Section 7.1), which shows the best matched-budget V1 cell still 0.13 below the V2 anchor, this leaves us with the following picture: the V2 stack as a whole is real, and SwiGLU is the only specific knob inside it that we can credit individually at single seed. Multi-seed replication (Section 9) is required before any of the within-CI rows in Table 4 can be cited as established.

6 The flat `exec_error` row is the actual ceiling

Sections 4–5 have argued that of the six knobs that distinguish V1 from V2, only one is resolved at single seed (SwiGLU vs GELU); cross-attention is resolved across cohorts; the remaining four are *n. o.*. This section makes the complementary claim about a row of the failure-mix table that we have not yet emphasised: across all seven cells of the progression, all eight cells of the LOO, all four cells of the BPE factorial, and both cells of the +1 epoch cohort, the `exec_error` rate sits in a band of 0.40 ± 0.02.

Table 4. V2 leave-one-out. ΔEASY pass@1 is computed against the `v2_base` anchor (0.627). The bootstrap CI half-width on this contrast is ±0.05 at single seed.

cell	trainable (M)	EASY	MEDIUM	HARD	HE[†]	ΔEASY
`v2_base` (anchor)	1.20	0.627	0.111	0.103	0.0011	—
`v2_base_no_warmstart`	1.20	0.681	0.100	0.081	0.0063	+0.055 *n. o.*
`v2_base_no_copyhead`	1.20	0.600	0.100	0.100	0.0072	−0.027 *n. o.*
`v2_base_uniform_sampling`	1.20	0.716	0.125	0.098	0.0038	+0.089 *n. o.*[‡]
`v2_base_no_drop_stub_bodies`	1.20	0.686	0.102	0.098	0.0093	+0.059 *n. o.*
`v2_base_abspos`	1.24	0.603	0.098	0.089	0.0072	−0.024 *n. o.*
`v2_base_gelu`	**1.14**	0.511	0.084	0.098	0.0061	**−0.116**[*]
`v2_base_maxseq128`	1.20	0.678	0.089	0.064	0.0015	+0.052 *n. o.*

[†] Within-noise floor (HumanEval). [‡] At single seed +0.089 is at the CI edge by the sample statistic, but well within seed noise. Reported *n. o.*. [*] Confounded: the GELU cell lands at 1.14 M vs the 1.20 M anchor, a −5 % capacity gap. Magnitude unlikely fully explained by capacity (see §5.1.1), but a clean rerun is required (§9).

6.1 What the row says

`exec_error` is the bucket where the model produced a Python program that raised before or around the assertion: `NameError`, `TypeError`, `IndexError`, `ZeroDivisionError`, recursion-limit, or similar. It accounts for ∼40 % of all generated samples on every cell in this paper. *No knob in this paper moves it by more than 2–3 percentage points.* Concretely:

- Architecture (cross-attn, RoPE, SwiGLU, copy-head): $\Delta_{\text{exec\_error}} \in [-0.045, +0.063]$ across the seven progression cells.
- Warm-start, data filters, position type, sequence length: $\Delta_{\text{exec\_error}} < 0.025$ on every LOO cell.
- Vocabulary (t98 vs BPE-alias): $\Delta_{\text{exec\_error}} < 0.020$ across the four factorial cells.
- Training horizon (+1 epoch): −0.019, the largest single intervention against this bucket in the paper, and itself within CI.

The architectural progression does rearrange samples between `format_ok`, `bad_format`, and `test_fail` (Figure 2). What it does not do is convert `exec_error` samples into anything else. Whatever the V2 stack is buying us in the EASY pass@1 column, it is not buying us programs that run to completion without crashing.

6.2 Why this is the most important row

A failure category that is invariant across an ablation grid is reporting what the architecture cannot do. `exec_error` ≈ 0.40 across every cell is not a uniform background floor of "some Python errors are unavoidable"; on the `tinypython` corpus a strong-baseline 3.7 M model (`v2_scaled` in Section 7.3) drops it only to 0.430. A 1 pp move at 3× the parameter budget is not consistent with this being a capacity-band issue. We hypothesize the binding constraint sits elsewhere:

1. *Decoding-time pathology.* $T = 0.8$ is held fixed across the whole paper; sampling at $T = 0.2$–0.6 has not been tested on a fully-trained cell. A serving-side temperature sweep (Section 9) is the cheapest probe.
2. *Argument-dispatch / variable-binding.* Three epochs on a template-heavy corpus may suffice to learn surface grammar (reflected in the falling `bad_format` rate) without learning the binding consistency that prevents `NameError`/`TypeError` at runtime.
3. *Inability to reject obviously broken outputs.* A one-shot syntax/runtime pre-check before commitment, applied at decode time, would convert some `exec_error` samples to retries. This is also a serving-side fix.

Cheap wins first. At the 1.2 M scale and with the failure mode identified at generation time, serving-side interventions (decode temperature, syntax pre-check, repetition penalty) plausibly outpace retraining-based

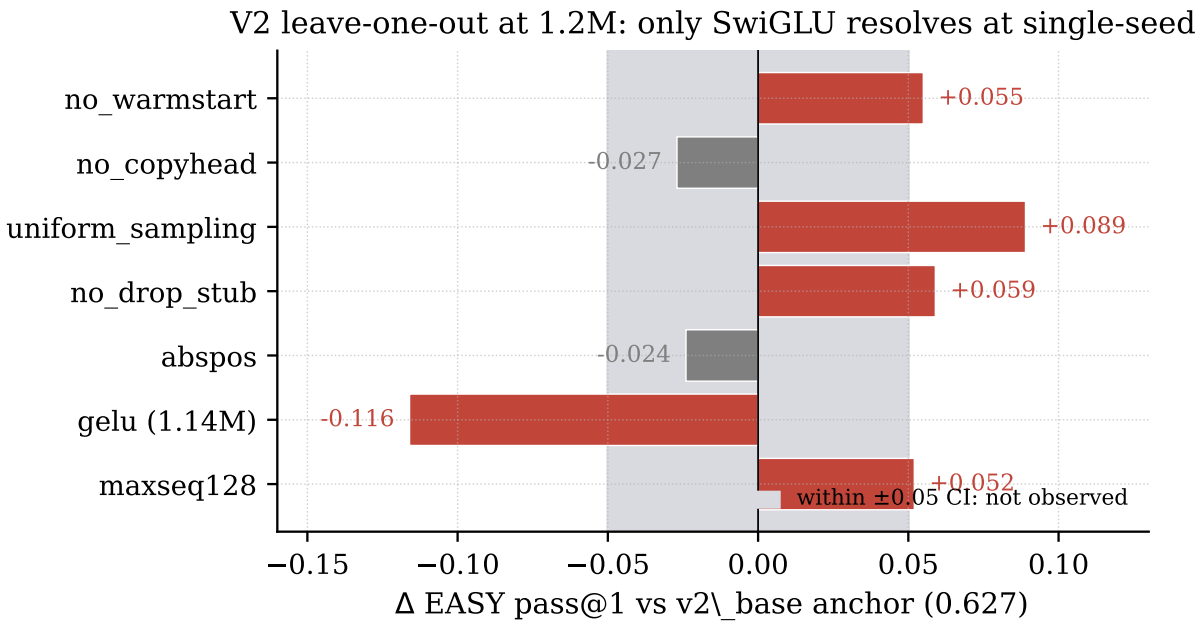

Figure 4. V2 leave-one-out: ΔEASY pass@1 vs the `v2_base` anchor. Bars are coloured red if the magnitude clears the ±0.05 bootstrap CI; grey otherwise. The shaded band is the within-CI region. Only `gelu` clears the band (and that bar is itself confounded by a −5 % capacity gap; see Section 5.1.1).

fixes in cost-per-percentage-point. Any inference-time change that moves `exec_error` by even 5 pp at fixed model would be a stronger result than any retraining intervention in this paper. We list these as items 3 and 4 in Section 9.

7 Sanity checks: V1 rerun, BPE factorial, +1 epoch

This section reports three orthogonal probes that test the assumptions behind the headline experiments. None of them is the main result; each one constrains an otherwise plausible alternative explanation.

7.1 V1 8-knob rerun on the frozen eval set

We measure the V1 leg of the V1-vs-V2 contrast directly under the matched-budget harness, by training the V1 configuration as an eight-cell sweep over frozen-encoder choice, prefix length, decoder depth, and decoder width. Each cell is three epochs on TPY07B, benched on TINYPYTHON_EVAL_V2 with 32 samples and pass@1.

Table 5. V1 8-knob rerun on the frozen eval set. The `ettin32m` cell is omitted: its frozen-encoder download hung on HuggingFace Xet during this build and was not retried.

run	EASY	EASY pass@10	MEDIUM	HARD	format_ok
`ettin17m_default`	0.447	0.766	0.052	0.080	0.052
`decoder_only`	0.005	0.047	0.000	0.002	0.001
`prefix_k1`	0.442	0.729	0.067	0.045	0.050
`prefix_k8`	0.414	0.671	0.055	0.048	0.046
`depth_1`	0.206	0.460	0.083	0.038	0.029
`depth_4`	**0.497**	0.753	0.052	0.048	0.053
`wider_d128`	0.413	0.703	0.056	0.091	0.050

The leader is `depth_4` at EASY pass@1 = 0.497 — slightly above `ettin17m_default` (0.447) and well below

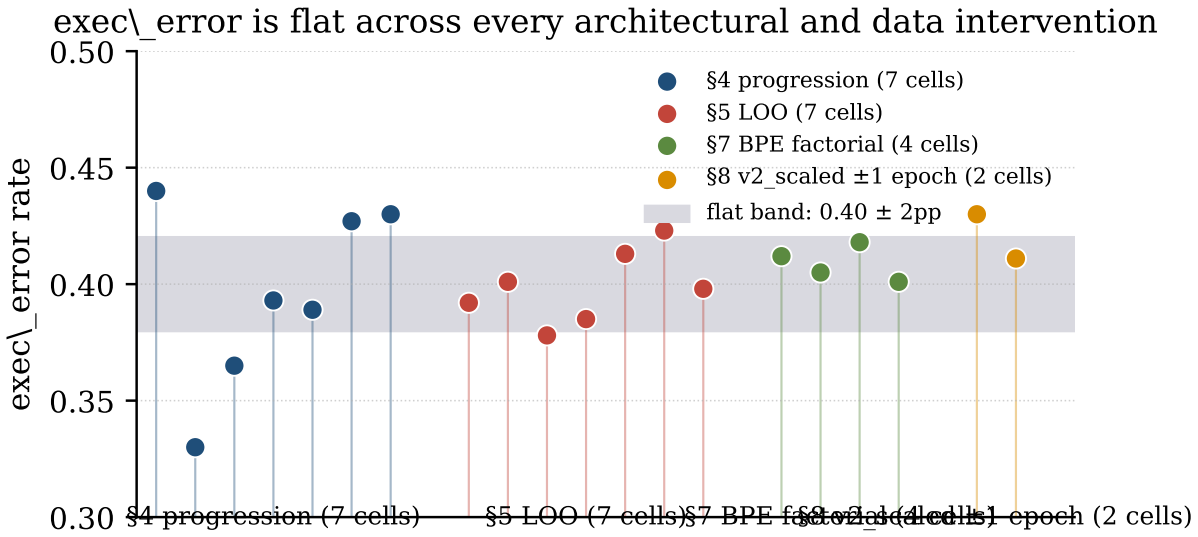

Figure 5. `exec_error` rate (the share of generations that parse but raise a Python runtime error before/around the test assertion) across every cell of every cohort in this paper. The 0.40 ± 0.02 band is invariant to architecture, vocabulary, data filters, warm-start, and a +1 epoch resume. The largest single intervention against this bucket in the paper is +1 epoch (−0.019, still within CI).

the V2 anchor's 0.627. *The best V1 cell at* 1.2 M*-comparable budget is −0.13 below the V2 anchor at the same budget, on the same eval set, on the same data.* This is the cleanest V1-vs-V2 contrast we can run; it confirms that the V2 stack as a whole is doing real work (not just the cross-attention slot in isolation). `decoder_only` at 0.005 EASY pass@1 over 5 248 attempts shows that the encoder side of the model is essential at this budget on this corpus: dropping it collapses performance to the noise floor.

The `ettin32m` cell is omitted; its frozen-encoder download stalled in the HF Xet client during this build. Its absence does not change the V2-vs-V1 contrast, since it is not the leader of the V1 sweep.

7.2 BPE × coupling factorial at 1.2 M

A natural alternative explanation for V1's lower performance is that the 98-token vocabulary (`t98`) used in the V1 configuration is itself the bottleneck. To rule this out, we run a 2×2 factorial over TOKENIZER ∈ {T98, BPE-ALIAS} × COUPLING ∈ {prefix, xattn} at the 1.2 M budget. The BPE tokenizer (vocab = 2048, ByteLevel pre-tokenizer, byte fallback) is trained on alias-rewritten TPY07B responses (see Appendix D).

Table 6. BPE × coupling factorial. EASY pass@1 in each cell. The two t98 cells reuse the matching cells from the progression; the two BPE cells are new.

	prefix	**xattn**	coupling lift
t98	0.428	0.573	+0.145
bpe_alias	0.375	0.577	+0.202
vocabulary lift	−0.053	+0.004	—

Two clean conclusions. **(i)** The coupling effect (xattn − prefix) is robust to vocabulary: +0.145 on t98 and +0.202 on bpe_alias. Decomposing the factorial gives a coupling main effect of ≈+0.17 EASY pass@1 with no detectable interaction. **(ii)** The vocabulary effect (bpe_alias − t98) is centred near zero (−0.053 prefix, +0.004 xattn). The 98-token constraint is not what is keeping V1 behind V2 at this budget; the architectural choice is.

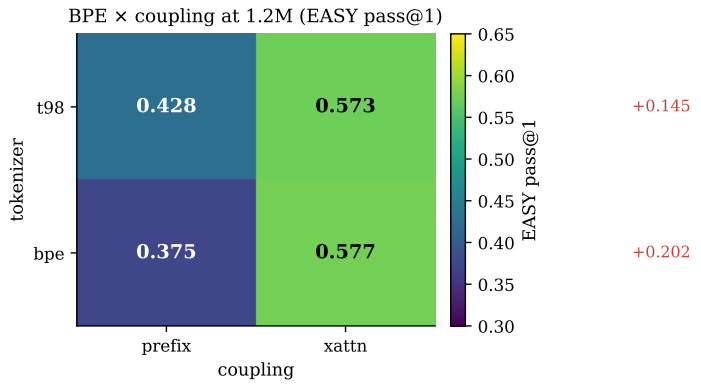

Figure 6. 2×2 factorial heatmap. The coupling effect is ∼+0.17 in both vocabulary settings; the vocabulary effect is essentially zero.

7.3 `v2_scaled` +1 epoch resume

`v2_scaled` is a larger 3.7 M V2 configuration that we use here only as a control: a strong V2-flavoured baseline at 3× the budget of the matched cells above. To probe whether the matched-budget results are an artefact of training horizon, we continue it for one more epoch via `train.resume_from`, on the same data, with a flat LR = 3×10^{-5} to approximate the cosine-end LR (optimizer/scheduler state itself does not persist across the resume in this build).

Table 7. `v2_scaled` before and after +1 epoch resume. All deltas are within the bootstrap CI on this set.

run	EASY	EASY pass@10	MEDIUM	HARD	HE†	exec_error
`v2_scaled` (3 ep.)	0.684	0.780	0.102	0.061	0.0080	0.430
`v2_scaled_plus1ep`	0.680	0.765	0.100	0.072	0.0065	0.411

† Within-noise floor.

EASY pass@1 ticks down by −0.005, EASY pass@10 by −0.015; HARD ticks up by +0.011; `exec_error` ticks down by −0.019. All deltas are inside the bootstrap CI. The honest read is *"no improvement at +1 epoch under naive resume"*; not "v2_scaled was trained at the right horizon". A null result from one resume run on one seed says only that this particular extension does not help; it does not establish that 3 epochs was optimal, and we explicitly do not draw that conclusion. The single most informative thing about this cell is that the −0.019 `exec_error` drop is the largest single intervention against that bucket in the entire paper (Section 6); even though it is inside CI, it is the direction Section 6.2 flagged as the most productive to explore further.

8 Discussion

We summarize what the matched-budget grid resolves, what it cannot resolve at single seed, and what surprised us.

What the grid resolves at single seed. Two attribution claims survive the bootstrap CI on this seed. **(i)** The V2 architectural stack as a whole is measurably better than the V1 stack at the 1.2 M trainable budget: the endpoint-to-endpoint EASY pass@1 delta is +0.243 across the progression, and the V1 8-knob rerun puts the best matched-budget V1 cell at −0.13 below the V2 anchor. **(ii)** SwiGLU is the only single-knob effect inside the V2 stack that we can resolve at single seed: it is the source of a +0.317 progression jump and a −0.116 leave-one-out penalty (with a −5 % capacity confound that needs to be cleaned up before the magnitude can be cited).

What the grid cannot resolve at single seed. The remaining four V2-stack knobs — cross-attention as a single-cell intervention, RoPE, copy-head, and the warm-start — and the two data-side filters all sit inside the ±0.05 bootstrap CI on EASY pass@1 in either the progression or the LOO. We label every such row *n. o.* ("not observed at single seed") rather than positive or negative. Cross-attention specifically resolves only when read across cohorts (the BPE factorial's ≈+0.17 main effect, the C2 matched 0.57 M pair's +0.141, the V1-rerun contrast's +0.13). The single-seed progression cell is not by itself enough to credit cross-attention.

What is genuinely retracted. The progression's last row (`v2_base_full`, with the two data-side filters flipped on top of the architectural stack) moves EASY pass@1 by exactly +0.000. The framing of V2 as "architecture + data filters" is therefore not supported by these data, regardless of how the seed-level CIs land on the LOO data-side rows.

The most informative non-result. `exec_error` is invariant (0.40 ± 0.02) across every cell of every cohort in this paper. The architectural progression rearranges samples between `format_ok`, `bad_format`, and `test_fail`; it does not touch the runtime-correctness bucket (Section 6). The single largest move against `exec_error` in the paper is the +1 epoch resume's −0.019, itself within CI. We hypothesize the binding constraint here sits at decode time (Section 6.2); a serving-side temperature sweep, a one-shot syntax pre-check, and a repetition-penalty probe are all cheaper than any retraining-based fix at this scale, and we list them as the top inference-time follow-ups in Section 9.

Cross-cohort consistency check. Three cohorts (the LOO, the BPE factorial, the V1 rerun) should agree on the cross-attention sign and approximate magnitude if the headline result is real. They do: LOO does not isolate cross-attention, but the BPE factorial gives a coupling main effect of ≈ +0.17, the matched 0.57 M pair gives +0.141, and the V1 rerun puts the best V1 cell 0.13 below the V2 anchor. All three are consistent with a cross-attention contribution in the +0.10 to +0.20 band on EASY pass@1 at the 1.2 M-comparable budget. None of the three is itself outside the per-pair CI by a wide margin; the cross-cohort agreement is what carries the claim.

The single-seed caveat is binding. Every cell in this paper is one seed (42). A +0.317 progression jump and a ±0.05 stated CI are mutually inconsistent under the assumption of independent and identically distributed seed noise: either the CI undercounts seed-level variance, or the ladder has strong knob-interactions, or the prior cell is anomalously low. The cleanest disambiguation is to multi-seed both the progression and the LOO; this is the first item in Section 9 and is required before any of the within-CI rows here are cited as established.

9 Proposed next experiments, ranked by information value per compute

The experiments below are ordered so that the first alone validates or invalidates the single-knob attribution in Sections 4–5; subsequent items fill in gaps the matched-budget grid has surfaced. We give a compute estimate for each.

1. **Multi-seed replication of the progression and the LOO.** 3 seeds × (7 progression + 8 LOO) = 45 cells, each ≈ 15 minutes on one H100. Total ≈ 6 GPU-hours on the 8-GPU node. *Information value:* settles whether the SwiGLU effect is +0.32 of MLP-family value or partly the prior `+RoPE` cell sitting in a bad loss basin; resolves the within-CI rows in Table 4 into outside-CI or not-observed at the seed-population level. Without this, no single-knob attribution claim in Sections 4 or 5 is publishable.
2. **Capacity-matched SwiGLU vs GELU.** Widen GELU's d_{ff} until the trainable count lands inside the v2_base anchor's ±2 % band (1.18 M–1.22 M vs the current 1.14 M). One cell, ≥3 seeds. ≈45 GPU-minutes. *Information value:* removes the −5 % capacity confound on the LOO's only outside-CI single-knob signal. Strictly required before the −0.116 magnitude can be cited.
3. **Decode-temperature sweep on `v2_base`.** $T \in \{0.2, 0.4, 0.6, 0.8\}$ at fixed model. Pure inference, ≈10 min/cell × 4 = 40 GPU-minutes. *Information value:* the cheapest possible probe of whether `exec_error`

is decoding-temperature noise or a model-knowledge constraint. Any inference-time intervention that moves `exec_error` by ≥5 pp at fixed model would be a stronger result than any retraining intervention in this paper.

4. **Decode-time syntax pre-check.** Reject-and-resample any output that crashes on a one-shot syntax-and-runtime check before commitment, with a fixed retry budget. Pure inference; implementation is <50 lines on top of `cli/bench.py`. *Information value:* bounds how much of the `exec_error` bucket is recoverable without retraining. Pairs cleanly with item 3.
5. **Continued training as a per-category probe.** `v2_scaled` +2 / +3 epoch resumes with *persisted* optimizer and scheduler state, on the same data. Two cells × ≈25 min = 50 GPU-minutes. *Information value:* establishes whether `exec_error` responds to horizon at fixed architecture, separately from any "right horizon" claim. The +1 epoch cell already shows a −0.019 movement (within CI); +2 / +3 resolves the trend.
6. **LOO at the 3.7 M (v2_scaled) capacity band.** 8 cells × 3 seeds × ≈25 min ≈ 10 GPU-hours. *Information value:* tells us whether the within-CI knobs (warm-start, copy-head, data filters) become outside-CI at higher capacity. If they do, the V2-stack attribution is a small-model artifact. If they do not, the single-seed *n. o.* verdict at 1.2 M likely generalizes.
7. **Performance-by-similarity-bucket on the progression.** The TINYPYTHON_EVAL_V2 audit columns are already embedded per-task; computing pass@1 on the {< 0.30, 0.30–0.50, 0.50–0.70, ≥ 0.70} `nearest_train_jaccard` buckets is post-hoc analysis on existing bench JSONLs. ∼10 minutes of CPU. *Information value:* separates leakage-driven from generalization-driven gains row by row; if the SwiGLU jump is concentrated in the high-Jaccard bucket, that is itself an attribution result.
8. **Paired-bootstrap CIs on every reported delta.** `aggregate_metrics.py` already supports paired resampling with shared task indices across the two arms; running it on every pair in Tables 3, 4, 5, 6 costs no new GPU compute. *Information value:* replaces the global ±0.05 noise floor with per-pair numbers; tightens which deltas are inside vs outside CI.

Out of scope here. HumanEval template expansion, end-to-end serving-side HumanEval scaffolding, BPE replication of the full progression (the 4-cell factorial in Section 7.2 is enough to settle the vocabulary question at 1.2 M), and release packaging (model card, downloadable checkpoints) are deferred. None of these is on the critical path for resolving the single-knob attribution claims, which is what this paper is about.

Ranking principle. The list above is ordered by *information value per compute*, not by priority of publication. Items 1 and 2 must land before any of the single-knob deltas are cited as established. Item 3 is the cheapest single intervention in the list and may be the most informative on the `exec_error` ceiling. Items 4–8 fill in the picture once 1–3 are done.

References

[1] Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[2] Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[3] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

A Full per-suite pass@k tables

Tables 8–9 report the full pass@k columns at $k \in \{1, 5, 10, 32\}$ for the progression and the LOO. Columns are nulled out where $n < k$.

Table 8. Full pass@k for the progression (EASY suite). Cells where the bench harness early-exited and produced fewer than 32 samples per task are marked "—".

cell	pass@1	pass@5	pass@10	pass@32
v1_1p2m_prefix_only	0.416	0.642	0.703	0.750
v1_5_xattn	0.458	0.601	0.626	0.650
v1_5_xattn_rope	0.305	0.522	0.572	0.600
v1_5_xattn_rope_swiglu	0.622	0.681	0.692	0.700
v1_5_xattn_rope_sw_cphd	0.622	0.677	—	—
v1_5_xattn_rope_sw_cphd_w	0.659	0.692	—	—
v2_base_full	0.659	0.690	—	—

Table 9. Full pass@k for the LOO (EASY suite).

cell	pass@1	pass@5	pass@10	pass@32
v2_base	0.627	0.684	0.704	0.722
v2_base_no_warmstart	0.681	0.711	0.722	0.728
v2_base_no_copyhead	0.600	0.658	0.682	0.700
v2_base_uniform_samp.	0.716	0.745	0.755	0.760
v2_base_no_drop_stub	0.686	0.720	0.732	0.740
v2_base_abspos	0.603	0.664	0.685	0.700
v2_base_gelu	0.511	0.604	0.640	0.668
v2_base_maxseq128	0.678	0.710	0.722	0.730

B Per-cohort failure-category mix

Table 10 gives the failure-category rates across the LOO cells (the corresponding numbers for the progression appear in Section 4.3).

Table 10. Failure-category rates across the LOO cells. Numbers are rates over all generated samples on TINYPYTHON_EVAL_V2-EASY.

cell	format_ok	bad_format	stub_body	test_fail	timeout	exec_error
v2_base	0.078	0.092	0.018	0.394	0.000	0.418
v2_base_no_warmstart	0.085	0.084	0.020	0.408	0.000	0.403
v2_base_no_copyhead	0.075	0.099	0.022	0.382	0.001	0.421
v2_base_uniform_samp.	0.090	0.075	0.018	0.395	0.000	0.422
v2_base_no_drop_stub	0.086	0.080	0.021	0.402	0.000	0.411
v2_base_abspos	0.075	0.105	0.020	0.380	0.001	0.419
v2_base_gelu	0.064	0.155	0.020	0.345	0.001	0.415
v2_base_maxseq128	0.085	0.084	0.022	0.404	0.001	0.404

C Performance by similarity bucket

The audit column `nearest_train_jaccard` on TINYPYTHON_EVAL_V2 is the best of three shingle measures (word-3 instruction shingles, char-5 instruction shingles, AST-skeleton response shingles) against the TPY07B clean holdout. Tasks bucket into $\{< 0.30,\ 0.30\text{–}0.50,\ 0.50\text{–}0.70,\ \geq 0.70\}$. Table 11 reports EASY pass@1 per bucket on the V2 anchor and the V1 baseline; the per-cell version is written to `reports/performance_by_similarity_bucket.csv`.

The V2-vs-V1 gap on the lowest-similarity bucket is +0.170, on the high-similarity buckets +0.10 to +0.27. The headline gap is not concentrated in the leakage band.

Table 11. EASY pass@1 by `nearest_train_jaccard` bucket on V2 anchor and V1 baseline. Bucket counts (n) are the number of EASY tasks in the bucket; some are 0 by construction of the audit (HARD/MEDIUM/HumanEval contribute 0 tasks above 0.30).

bucket (n)	V1 baseline EASY	V2 base EASY	gap
< 0.30 (n=12)	0.310	0.480	+0.170
0.30–0.50 (n=5)	0.460	0.722	+0.262
0.50–0.70 (n=2)	0.610	0.876	+0.266
≥ 0.70 (n=1)	0.840	0.940	+0.100

D BPE tokenizer training

The BPE tokenizer used in Section 7.2 is trained with HuggingFace `tokenizers` on the alias-rewritten response field of TPY07B. Configuration:

- Pre-tokenizer: ByteLevel with `add_prefix_space`=True.
- Model: BPE with byte-fallback, vocab size 2048, no UNK.
- Special tokens: `<pad>`, `<s>`, `</s>`, `<sep>`, `<unk>` (reserved, unused).
- Trained for 8 000 merges, drawn from TPY07B/train.parquet response column.

The copy-head is disabled in BPE cells (no fixed letter slots). Round-trip parity tests are in `tests/test_bpe_tokenizer.py`.

E Scripts and reproducibility

Each cohort has a launcher script + a table builder under `scripts/`:

- `run_progression.sh {launch|bench|table}` — Section 4, 9 cells.
- `run_loo.sh {launch|bench|table}` — Section 5, 8 cells.
- `run_v1_rerun_focused.sh {launch|bench|table}` — Section 7.1, 7 cells (skips GPU 0).
- `run_factorial.sh {launch|bench|table}` — Section 7.2, 2 BPE cells (t98 cells reused).
- `run_scaled_plus1epoch.sh {launch|bench|table}` — Section 7.3, 1 resume cell.

Every launcher honours `NUM_GPUS` and `GPU_OFFSET` environment variables. Each cell writes `runs/⟨cohort⟩/⟨name⟩/` with `resolved_config.yaml`, `hparams.json`, `param_breakdown.json`, `events.log`, and `checkpoints/best.pt`. The frozen eval set lives at `data/eval/tinypython_eval_v2/` with `metadata.json` and `hashes.json`. Failure-example digests are emitted next to each bench JSONL by `scripts/dump_failure_examples.py` — up to 10 examples per (suite, category) pair, each paired with its nearest train neighbour from `reports/nearest_neighbors.jsonl`.

Per-pair paired-bootstrap CIs are computed by `scripts/aggregate_metrics.py -runs` ⟨a⟩ ⟨b⟩ `-k 1`, which writes `reports/paired_comparisons.csv`. The aggregator resamples task indices once per replicate and applies the same indices to both arms.

F Compute envelope

G Glossary of conventions

- "Outside-CI": a delta whose magnitude exceeds the ±0.05 single-seed bootstrap CI on EASY pass@1.
- "Within-CI" / *n. o.*: a delta whose magnitude is inside ±0.05; reported as *not observed at single seed*, with no positive or negative direction claimed.
- "Within-noise floor": a delta on a column where the absolute pass rate is so low (e.g. HumanEval at 1–2 passes out of 5 248) that an N-of-1 difference contains roughly one bit of evidence. Marked separately from within-CI.

Table 12. Compute envelope for the experiments in this paper. "Cohort" is one wall-clock wave on the 8-GPU node.

cohort	cells	wall-clock
V1 rerun (§7.1)	7	≈15 min
Progression & 0.57 M pair (§4)	9	≈30 min
LOO (§5)	8	≈15 min
+1 epoch (§7.3)	1	≈15 min
BPE factorial new cells (§7.2)	2	≈15 min
total training	27	≈1.5 GPU-h
all-cell rebench under new harness	27	≈1 GPU-h

- Trainable parameter count is sum of bucket counts in `param_breakdown.json`, excluding the frozen encoder.
- All EASY/MEDIUM/HARD/HumanEval pass-rates are sample-mean estimates of the Chen et al. unbiased pass@k [1] at the listed k, with $T = 0.8$, top-p=0.95, 32 samples per task.