
Where Is the Adaptive-Compute Headroom in Next-Concept Models? A Calibrated Study of Early Termination and Selective Refinement on Sudoku

Han Shaun Lee
Stanford University
hanl@stanford.edu

Abstract

Next-Concept Models (NCM) generate a sequence as a coarse-to-fine hierarchy of discrete codes, one parallel transformer pass per level, and their authors propose adapting the amount of computation to example difficulty. We ask the concrete question such a proposal implies: can a stopping rule that observes only inference-available signals terminate the hierarchy early on easy inputs while preserving task quality and reducing real wall-clock cost? We instrument the released NCM Sudoku checkpoints with a stopping decision *inside* the level loop, decode valid intermediate reconstructions through the frozen tokenizer (proving that level- k outputs do not depend on unseen future codes and documenting an aliasing trap in the release), and build fixed-depth and oracle earliest-successful-depth curves on 2,000 board-setting pairs, 0 boards are solved, or even constraint-valid, before the final level (one-sided 95% bound 0.050%), although full depth solves 99.4% of easy boards. Encoding the *ground-truth* boards with the tokenizer and truncating at level 10 yields 87.9% cell accuracy and no complete board, so the limit is a property of the residual tokenizer, not of the generator: the last level is indispensable, and calibrated stopping rules, a constraint verifier, and compute-matched baselines all collapse to full depth on held-out boards. Following the pre-registered contingency, we then study the opposite direction, *selective refinement*: extra full-depth stochastic samples only for boards whose output fails the inference-available Sudoku verifier. On held-out boards this lifts the hard model from 85.0% to 93.0% solved (+8.0 pp [6.2, 10.0]) for 21.46 instead of 12 mean level passes (saturating at 95.9% with 24 samples), and the easy model on hard boards by +22.0 pp, measured end to end on a Tesla T4 at 128.1 instead of 93.4 ms per board at batch 1 (p95 311 ms) when the extra samples are drawn in parallel, versus 174.8 ms (p95 882 ms) when drawn sequentially; compute-matched gates that do not consult the verifier gain far less; a calibrated predictor that aborts doomed samples early is dominated by the plain verifier gate despite large oracle headroom. Latencies are measured on a Tesla T4 with CUDA synchronisation at batch sizes 1, 32 and 128. All numbers come from recorded runs; code, data manifests and raw results are released.

1 Introduction

Next-Concept Modeling [Kumbong et al., 2026] tokenises a sequence with a multiscale residual vector quantiser into K levels of discrete codes of growing length ($\ell_k = 1, 2, 4, \dots, 89$ for Sudoku)

and trains a transformer to predict level $k+1$ from the reconstruction of levels $\leq k$ and a prefix. Generation therefore costs one parallel transformer pass per level, and the accompanying blog post explicitly suggests adapting computation to example difficulty. Early exit is a mature idea [Teerapittayanon et al., 2016, Graves, 2016, Elbayad et al., 2020, Xin et al., 2020, Schuster et al., 2022]; what is *not* known is whether NCM’s coarse-to-fine hierarchy actually offers usable intermediate outputs, how much a calibrated rule could save once intermediate decoding and the rule itself are paid for, and how the answer behaves under a difficulty shift. This paper is an empirical, pre-registered answer for the released Sudoku checkpoints, which are the only NCM task with an exact, input-only verifier.

Our contributions are:

1. **A faithful instrumented inference path.** We add a per-level stopping callback inside `NextConceptTransformer.generate`, decode intermediate full-length reconstructions through the frozen tokenizer, compact the batch as rows exit, and time generation, decoding and the controller separately. We verify bit-exact agreement with the unmodified release on all 2,000 boards per setting and on GPU, prove that level- k reconstructions are invariant to arbitrary (including all-zero) future codes, and show that the release’s `idx_to_zhats` returns K references to one in-place accumulator, so a naive collector would silently decode full-depth information at every “intermediate” level.
2. **A zero-headroom negative result with a mechanism.** Fixed-depth and oracle earliest-successful-depth curves show no board solved before the last level in any of three settings, and a tokenizer-capacity diagnostic on ground-truth boards locates the cause in the residual quantiser. Calibrated logistic/isotonic rules, a constraint verifier, entropy/stability rules and compute-matched random stopping are all evaluated under a frozen protocol on a held-out split and under easy→hard shift; none saves computation without destroying accuracy.
3. **Selective refinement as the pre-specified alternative.** Using the verifier as a gate, extra full-depth stochastic samples buy +8.0 pp on hard/hard at 21.46 mean passes per board; we compare against uniform, random, entropy and calibrated no-verifier gates at matched budgets and show that a calibrated early-abort predictor is dominated, although an oracle abort would cut the compute to 13.67 passes.
4. **Cost accounting.** Per-level transformer, decoder and controller costs and end-to-end latencies are measured on a Tesla T4 with warm-up, CUDA synchronisation and randomised repeats at batch sizes 1, 32 and 128, including the online refinement policies; drawing refinement samples in parallel exploits the launch-bound batch-1 regime and cuts the p95 latency of refinement by roughly two thirds.

2 Related work

Early exit and adaptive depth. BranchyNet [Teerapittayanon et al., 2016] and Adaptive Computation Time [Graves, 2016] introduced confidence- and halting-based early exit; Depth-Adaptive Transformers [Elbayad et al., 2020] and DeeBERT [Xin et al., 2020] apply it to sequence models. CALM [Schuster et al., 2022] is the closest prior work: it exits per token from intermediate layers of a language model using calibrated confidence thresholds with a statistical consistency guarantee. We do not claim to invent early exit; the question here is NCM-specific: whether the *level* hierarchy (not the layer stack) admits exits, what the exact execution cost of checking is, and whether calibration transfers across difficulty.

Multiscale residual quantisation. NCM’s tokenizer follows the next-scale residual design of VAR [Tian et al., 2024]. Our capacity diagnostic (§6.2) is a direct measurement of what such a residual hierarchy can reconstruct when truncated.

Calibration and risk control. We use Platt-style logistic calibration [Platt, 1999], isotonic regression [Zadrozny and Elkan, 2002], expected calibration error [Guo et al., 2017], and a Learn-then-Test style threshold selection on a separate calibration split [Angelopoulos et al., 2021].

3 Background: NCM inference on Sudoku

The released easy and hard Sudoku models share one architecture: a frozen VQ-VAE tokenizer (14.9M parameters, $d=256$, 12 levels of 16 codes each, learned level-0 down/up-sampling cascade, per-level residual convolutions Φ_k) and an 8-layer, $d=512$ generator (25.5M parameters) with block-diagonal attention over a 91-token raw puzzle prefix and the current level. A board is 89 tokens (81 cells, 8 row separators; blank 0, digits 1–9, separator 10, BOS 11). Decoding is greedy (temperature 0) in the release configuration. The generator predicts level 0 from learned queries and then, for $k = 0, \dots, 10$, folds the level- k codes into a running reconstruction $\hat{z}_k = \sum_{j \leq k} \Phi_j(\text{up}(E_j c_j)) \in \mathbb{R}^{89 \times 256}$, area-pools it to length ℓ_{k+1} , and runs the transformer once to predict c_{k+1} . The evaluation script finally decodes $\hat{z}_{11}$ with the one-block tokenizer decoder and compares strings with the stored solution.

4 Method

4.1 Stopping inside the level loop

We re-implement the eager generation loop (the released configuration asserts against the KV-cache path when `query_level0` is set) and insert, after the level- k codes are folded into $\hat{z}$, (i) a cloned snapshot $\hat{z}_k$, (ii) an optional decode $b_k = \arg \max \text{Dec}(\hat{z}_k)$, (iii) feature extraction, (iv) a policy decision per batch row, and (v) compaction of the batch to the still-active rows before the next transformer pass. A row that stops at level k executes exactly $k+1$ level passes. We decode the *full-length* snapshot rather than the area-pooled z -level used by the release’s multiscale visualiser, because the Sudoku decoder must emit 89 tokens; the tokenizer was trained with a “fine-drop” regulariser that feeds the decoder such truncated full-length reconstructions half of the time, so this is the decoder’s intended input.

Correctness tests. We test (all tests are in `tests/`): exact equality of codes and boards with the unmodified release under temperature 0 on CPU and GPU; that stopping at k returns the level- k board of the full run and executes $k+1$ passes; that rows exiting at different levels (compaction) reproduce their full-run boards; that $\hat{z}_k$ computed from a code prefix is bit-identical when the future codes are randomised or set to zero (zero is a valid code, not padding); that our snapshots match the release’s own intermediate z -levels up to pooling; and that the release’s `idx_to_zhats` returns aliased tensors (`data_ptr` identical for all K entries).

4.2 Inference-available features and policies

At level k a row exposes: the level index; code-distribution statistics of the just-sampled level (mean/max entropy, mean/min max-probability, mean/min top-1/top-2 margin, mean/min log-probability of the chosen codes, and their running aggregates); relative representation change $\|\hat{z}_k - \hat{z}_{k-1}\|/\|\hat{z}_k\|$; decoded-output statistics (decoder confidence, number of cells changed since the previous level); and Sudoku validity of b_k against the *input* puzzle only: separator positions, blanks, non-digits, givens mismatches, and the number of missing (row, digit), (column, digit) and (box, digit) pairs. A board is *verified* when it is complete, has zero violations and preserves all givens. Ground-truth solutions are never visible to any policy; they are used only to label development boards and to evaluate.

Policies: full hierarchy; every fixed truncation depth; a verifier rule (stop at the first verified board, from a tuned minimum level); entropy/stability and entropy/representation-change rules (thresholds tuned on dev+cal); calibrated stopping (logistic regression with a per-level intercept, optionally isotonic-recalibrated, fit on dev; stop when the estimated probability that the level- k board is solved exceeds θ ; θ is the smallest value whose calibration-split solved-rate loss is ≤ 1 percentage point, frozen before the test split is read); random stopping drawn from the calibrated policy’s calibration-split stop-level histogram (compute-matched); and the oracle earliest level at which the board is solved (diagnostic only).

4.3 Selective refinement

If no intermediate level is usable, adaptive computation can only go the other way: spend *more* on hard inputs. Our refinement policy runs the deterministic pass, verifies the output, and only when verification fails draws full-depth stochastic samples (temperature T chosen on dev, top- k off) until one verifies or a budget of extra level passes is exhausted; the verifier selects the output in every variant. Gates that decide *which* boards receive extra compute are compared at matched expected compute: the verifier gate, a random gate, a gate on the deterministic pass’s final-level code entropy, and a calibrated gate that predicts failure from the deterministic trajectory’s features without any verifier. Inside resamples we additionally test *calibrated early abort*: a logistic predictor of “this trajectory will verify”, fit on dev trajectories from generator-only features (free) or generator+decode features (costing one decode per level), aborts a sample at the first level where the estimate falls below θ_a ; θ_a is the largest threshold whose calibration-split false-abort rate among eventually-verified samples is $\leq 5\%$. An oracle abort (doomed samples cut at level 0) bounds the achievable saving.

5 Experimental setup

Data and splits. The release ships 2,000 boards per difficulty. The solution files are byte-identical across easy, medium and hard: the same 2,000 unique solution grids are masked to 40 (easy) or 30 (hard) givens, and the hard givens are a subset of the easy givens of the same index. The independent unit is therefore the board index. One seeded partition (dev 800 / calibration 400 / test 800) is shared by all difficulties, so the easy→hard shift never scores a grid whose easy variant was used for fitting or thresholding.

Settings. Easy model on easy boards (the release pairing), easy model on hard boards (input shift), and hard model on hard boards (release pairing); the calibrators and thresholds are always fit on the easy/easy dev and cal splits and applied unchanged elsewhere.

Hardware and timing. Quality is computed on CPU (2 threads; bit-identical codes to the release) and on a Tesla T4 via Modal; latencies are measured on the T4 (2.14.1+cu130, TF32 disabled) after warm-up, with `torch.cuda.synchronize()` at every section boundary, at batch sizes 1, 32 and 128, on the 800 test boards, reporting mean and p95 per board. Peak GPU memory was 831 MB. Compute is also reported as level passes and as transformer token-passes ($\sum_k (91 + \ell_k) = 1414$ for full depth).

Statistics. Paired bootstrap over boards (10,000 resamples for the main tables, 4,000–5,000 elsewhere) for differences against the release behaviour; one-sided Clopper–Pearson bounds for zero counts. Three seeds for every learned calibrator, five for random baselines.

Cost. All GPU work ran on Modal T4 instances at the list price of \$0.59/h; the total spend is reported in the reproducibility statement.

6 Results

6.1 Fixed-depth and oracle curves: no early solution exists

Table 1 gives the fixed-depth curve on the test split. Cell accuracy rises monotonically with depth (86.7% at level 10 for easy/easy) but the solved rate is exactly zero at every level below the last in all three settings: 0, 0 and 0 of 800 test boards each, and 0 of all 6000 board–setting pairs over the full 2,000 boards (one-sided 95% upper bound 0.050%). The level-10 boards still contain on average 27.3 missing (unit, digit) pairs on easy/easy, so the input-only verifier fires at the final level or never; it never accepts a wrong board (0, 0, 0 verified-but-unsolved cases). The oracle earliest-successful depth is therefore 12 levels for every board, and no stopping rule, however well calibrated, can save a single pass without giving up the board. Full depth solves 99.4% [98.8, 99.9] (easy/easy), 42.1% [98.8, 45.6] (easy model on hard boards) and 85.0% [82.5, 87.5] (hard/hard) of test boards.

Table 1: Fixed-depth quality on the 800-board test split (stopping after level k , decoding the full-length reconstruction). Solved % is exact match with the stored solution; cell % is per-cell accuracy. No board is solved before the final level in any setting.

k	ℓ_k	easy model / easy		easy model / hard		hard model / hard	
		solved%	cell%	solved%	cell%	solved%	cell%
0	1	0.0	14.8	0.0	14.6	0.0	14.5
1	2	0.0	14.8	0.0	14.5	0.0	14.5
2	4	0.0	17.1	0.0	16.6	0.0	16.9
3	6	0.0	20.8	0.0	20.1	0.0	20.2
4	9	0.0	20.7	0.0	20.0	0.0	20.2
5	13	0.0	29.1	0.0	27.6	0.0	27.8
6	18	0.0	36.4	0.0	33.9	0.0	33.4
7	26	0.0	49.0	0.0	45.3	0.0	45.5
8	37	0.0	65.2	0.0	59.4	0.0	62.1
9	50	0.0	81.7	0.0	73.7	0.0	76.0
10	67	0.0	86.7	0.0	78.3	0.0	81.9
11	89	99.4	100.0	42.1	92.4	85.0	98.1

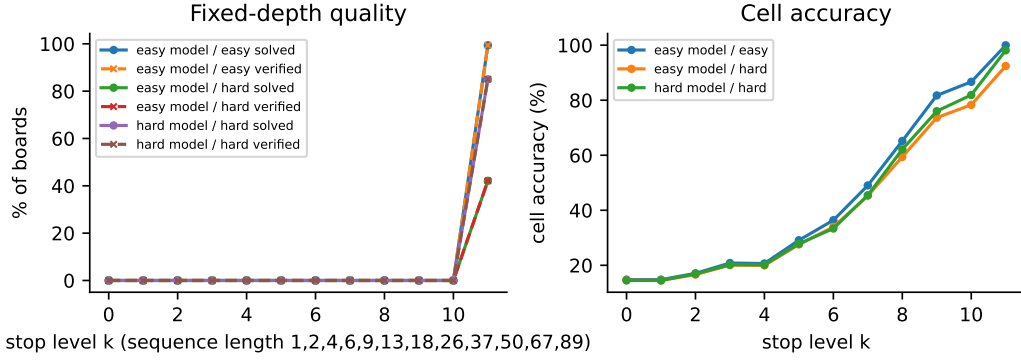


Figure 1: Fixed-depth curves on the test split: solved rate (left) and cell accuracy (right) after stopping at level k .

6.2 The tokenizer, not the generator, sets the limit

To separate “the generator predicts poor coarse codes” from “the tokenizer cannot express a complete board before its last level”, we encode the ground-truth solutions with the frozen tokenizer, truncate the code hierarchy at level k , and decode (Table 2). Even with the true codes, level 10 reaches only 87.9% cell accuracy with 25.1 violations per board and 0.0% solved boards. With 16 codes per level the first eleven levels carry at most $4 \sum_{k \leq 10} \ell_k = 932$ bits spread by interpolation over 89 positions, and the residual design places the finest corrections in the last, full-resolution level. The absence of early solutions is thus structural; it would persist for any generator trained on this tokenizer.

6.3 Held-out evaluation of stopping policies

Table 3 evaluates the frozen policies on the test split. Every calibrated variant (all feature sets, with and without isotonic recalibration, three seeds), the verifier rule and compute-matched random stopping select full depth for every board, so they reproduce the release accuracy with zero saving in all three settings; this is the correct behaviour of a risk-controlled rule when no safe exit exists, and the calibrators are accordingly very well calibrated ($ECE < 0.01$ over all levels in every setting). The only rules that ever stop early are the uncalibrated entropy/stability heuristics tuned on dev+cal: on hard/hard the entropy+stability rule runs 11.74 levels on average and loses 2.6 pp [1.6, 3.7] of solved boards; the entropy+representation rule, tuned on easy boards, collapses under the shift. The fixed-depth rows quantify the price of any early exit: −99.4 pp for saving one pass.

Table 2: Tokenizer capacity: ground-truth boards encoded by the frozen tokenizer, truncated at level k and decoded. Cell accuracy in % (first row per tokenizer) and mean number of missing (unit, digit) pairs.

tokenizer / truncation level k	0	1	2	3	4	5	6	7	8	9	10	11
easy tokenizer (GT boards)	14.6	15.1	17.4	21.1	20.9	29.2	36.8	49.9	65.9	82.5	87.9	100.0
violations	105	102	100	99	100	97	86	69	54	32	25	0
hard tokenizer (GT boards)	14.7	14.6	17.2	20.6	20.6	28.3	33.8	46.5	63.5	77.7	83.8	100.0
violations	99	95	89	86	86	91	84	73	58	42	33	0

Table 3: Held-out (test split) evaluation of stopping policies frozen on the easy dev/cal splits. Δ is the paired solved-rate loss relative to the full hierarchy in percentage points with a 95% bootstrap interval; “levels” is the mean number of executed level passes (12 = full depth). Oracle = diagnostic (uses ground truth).

Policy	easy model / easy data			easy model / hard data			hard model / hard data		
	solved%	Δ [CI]	levels	solved%	Δ [CI]	levels	solved%	Δ [CI]	levels
Full hierarchy (release)	99.4	+0.0 [0.0,0.0]	12.00	42.1	+0.0 [0.0,0.0]	12.00	85.0	+0.0 [0.0,0.0]	12.00
Fixed depth $k=9$	0.0	+99.4 [98.8,99.9]	10.00	0.0	+42.1 [38.8,45.5]	10.00	0.0	+85.0 [82.5,87.5]	10.00
Fixed depth $k=10$	0.0	+99.4 [98.8,99.9]	11.00	0.0	+42.1 [38.8,45.5]	11.00	0.0	+85.0 [82.5,87.5]	11.00
Verifier rule (stop when valid)	99.4	+0.0 [0.0,0.0]	12.00	42.1	+0.0 [0.0,0.0]	12.00	85.0	+0.0 [0.0,0.0]	12.00
Entropy+stability rule	98.9	+0.5 [0.1,1.0]	11.96	42.1	+0.0 [0.0,0.0]	12.00	82.4	+2.6 [1.6,3.7]	11.74
Entropy+representation rule	98.5	+0.9 [0.4,1.6]	11.92	42.1	+0.0 [0.0,0.0]	12.00	0.0	+85.0 [82.5,87.5]	3.13
Random, compute-matched	99.4	+0.0 [0.0,0.0]	12.00	42.1	+0.0 [0.0,0.0]	12.00	85.0	+0.0 [0.0,0.0]	12.00
Calibrated (all features)	99.4	+0.0 [0.0,0.0]	12.00	42.1	+0.0 [0.0,0.0]	12.00	85.0	+0.0 [0.0,0.0]	12.00
Calibrated (generator-only)	99.4	+0.0 [0.0,0.0]	12.00	42.1	+0.0 [0.0,0.0]	12.00	85.0	+0.0 [0.0,0.0]	12.00
Oracle depth (diagnostic)	99.4	+0.0 [0.0,0.0]	12.00	42.1	+0.0 [0.0,0.0]	12.00	85.0	+0.0 [0.0,0.0]	12.00

6.4 What stopping would have cost to check

Table 4 reports the per-level cost of a transformer level pass, one intermediate decode and the controller (feature extraction, host transfer and validity check) on the T4. At batch 1 every level pass costs about 6.3–6.5 ms regardless of its length: inference is launch-bound, a full board costs 76.3 ms of generator time (98.9 ms end-to-end), and an intermediate decode plus check adds 0.80+0.56 ms per level, i.e. roughly 20% of a level pass. At batch 32 the pass cost grows from 50.1 ms at level 0 to 95.3 ms at the full-length level, so the late levels that an exit would skip are also the expensive ones; the decode and check are amortised to 1.91+0.86 ms per batch. Measured end-to-end latencies (Table 5; three repeats per policy in seeded random order with a warm-up before every cell, paired bootstrap CIs over boards at batch 1 and over batches at batch 32) are consistent with the level-pass accounting: fixed depth 10 and 9 cut the batch-1 latency by -9.7% and -17.9% (batch 32: -13.0% and -23.5%), which is the saving an exit would have bought had its boards been usable. Policies that decode and check every level pay for it and gain nothing: the verifier rule costs +16.6% [16.0, 17.1] at batch 1 and +3.5% at batch 32, the entropy+stability rule +13.6%. Board-level CIs understate run-to-run drift: three policies that execute exactly the full-depth computation (generator-only calibrated, compute-matched random and oracle, none of which ever stops early) land between -1.7% and +3.7% of the full row at batch 1 (-0.3% to +0.1% at batch 32), and the full row itself ranges 97.2–102.9 ms over repeats. We therefore read only batch-1 differences outside this empirical null band as effects.

6.5 Selective refinement: where the adaptive-compute headroom actually is

Because no exit is safe, we evaluate the pre-specified alternative (§4.3) on recorded GPU trajectories: for every board and setting, one deterministic pass and eight full-depth stochastic samples at each of $T \in \{0.5, 1.0\}$, with every level’s features and decoded board stored (51 passes $\times$ 2,000 boards). Policies are replayed exactly on these recordings; the temperature, the abort calibrators and their thresholds, and the no-verifier gate are fit on each setting’s dev split and thresholded on its calibration split before the test split is read, and an easy-fit variant tests the easy→hard shift. The dev choice was $T^* = 1.0$ for both hard settings and $T^* = 0.5$ on easy boards. Across all 51 recorded passes of all three settings the verifier never accepted a wrong board, so the verifier-selected output is correct whenever any candidate verifies.

Table 10 and Figure 2 give the test-split frontier. On easy/easy only 5 of 800 test boards fail, so refinement can add at most a fraction of a point (+0.2 pp). On the hard release pairing, the verifier

Table 4: Per-level component cost on a Tesla T4 (ms per call, mean of repeated synchronised trials): the transformer pass that produces the codes of level k , one decoder call on the full-length snapshot, and the controller (features, host transfer, validity check).

batch	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	decode	controller
1	6.3	6.4	6.4	6.4	6.4	6.1	6.4	6.4	6.2	6.4	6.4	6.5	0.80	0.56
32	50.1	51.2	50.7	51.8	53.1	55.9	58.6	59.5	66.9	76.4	80.8	95.3	1.91	0.86
128	189.7	192.2	196.5	202.4	200.5	216.0	219.0	245.1	267.8	307.1	339.0	372.4	6.44	1.68

Table 5: Measured end-to-end latency per board on the T4 (easy/easy test boards: 400 at batch 1, 800 at batch 32; ms, mean of three repeats run in seeded random order), with the relative difference to full depth and its paired 95% bootstrap CI, and the policy’s solved rate. Differences inside the empirical null band (see text) are run-order drift, not effects.

policy	T4 bs=1	T4 bs=32	solved%
full	98.9 (+0.0% [+0.0,+0.0])	24.94 (+0.0% [+0.0,+0.0])	99.5
fixed9	81.2 (-17.9% [-18.3,-17.5])	19.08 (-23.5% [-23.6,-23.4])	0.0
fixed10	89.3 (-9.7% [-10.1,-9.3])	21.70 (-13.0% [-13.1,-12.9])	0.0
verifier kmin0	115.3 (+16.6% [+16.0,+17.1])	25.80 (+3.5% [+3.4,+3.6])	99.5
entropy stability	112.4 (+13.6% [+12.9,+14.3])	25.82 (+3.5% [+3.3,+3.7])	99.0
entropy rep	98.4 (-0.5% [-1.4,+0.2])	24.82 (-0.5% [-0.9,-0.2])	98.5
calibrated full s0	98.0 (-0.9% [-1.2,-0.5])	25.04 (+0.4% [+0.3,+0.5])	99.5
calibrated generator only s0	102.5 (+3.7% [+3.0,+4.4])	24.97 (+0.1% [+0.0,+0.2])	99.5
random matched	100.3 (+1.4% [+0.9,+1.9])	24.90 (-0.2% [-0.3,-0.1])	99.5
oracle	97.2 (-1.7% [-2.1,-1.4])	24.86 (-0.3% [-0.4,-0.2])	99.5

gate turns the 120 failing boards (15.0%) into a steep gain: +3.1 pp [2.0, 4.4] for 13.80 mean level passes per board (vs. 12), and +8.0 pp [6.2, 10.0] (85.0%→93.0%) for 21.46 mean passes, i.e. 21.46/12 of the release compute on average. Under the input shift (easy model on hard boards) the same rule adds +22.0 pp [19.1, 24.9] at 55.56 passes. The cost is concentrated in the tail: at budget 96 the p95 board runs 108 level passes on hard/hard, so selective refinement trades mean compute for a heavy latency tail (measured below).

Measured end-to-end latency. We implemented the verifier-gated policy online (deterministic pass, verification, re-batching of the failing rows, one stochastic sample per round, verification, until all rows verify or the budget is spent; `src/ncm_es/refine_online.py`) and timed it on the T4 on the 800 test boards of both hard settings with fresh samples, two repeats in randomised order, at batch 1 and 32 (Table 6). The online policy reproduces the replayed accuracy (hard/hard, budget 96: +8.2 pp [6.4, 10.1] measured vs. +8.0 pp replayed). At batch 1 the measured latency ratio to the release tracks the level-pass ratio closely (1.84 vs. 1.78; easy model on hard boards 4.89 vs. 4.67), so level passes are a faithful proxy for mean latency in the launch-bound regime: the mean cost of +8.2 pp is 174.8 instead of 95.2 ms per board, and the p95 board waits 882 ms (release 118 ms). The component-sum estimates used in Table 10 are *optimistic in absolute terms* (138.9 vs. 174.8 ms) because they omit per-call setup and host-side work. At batch 32 refinement is less efficient than its pass count suggests: failing rows are re-batched into ever smaller batches that under-use the GPU, so the throughput cost ratio is 2.17 against a pass ratio of 1.78, and the whole batch waits for its slowest board (p95 batch latency 2203 ms vs. 813 ms).

Parallel sampling exploits the launch-bound regime. Because a batch-1 level pass costs the same for one row as for several, a failing board can draw w samples at once as one small batch and keep the first verified one. Measured in the same container as the sequential policy (Table 7), width 4 or 8 at budget 96 keeps the accuracy of sequential refinement on hard/hard (+8.0 pp [6.2, 9.9] for $w=8$ vs. +8.2 pp) while cutting mean latency from 173.5 to 128.1 ms per board (release 93.4 ms) and p95 latency from 875 to 311 ms, at the price of more level passes (26.40 vs. 21.38). With budget 192 the width-8 policy reaches +9.7 pp at 142.1 ms. For the easy model on hard boards the saving is larger (477.8 → 217.5 ms mean, p95 961 → 315 ms). Level passes are therefore a good proxy for *throughput* but not for *latency*: once the device is not saturated, extra samples are nearly free in wall-clock terms. At batch 32 the device is compute-bound and width 8 brings no latency benefit (56.0 ms per board vs. 54.4 ms sequential in the round-1 container).

Table 6: Measured end-to-end latency of online verifier-gated refinement on a Tesla T4 (test split, 800 boards, fresh samples at $T^* = 1.0$, mean of two repeats run in randomised order). ms/board is total wall time divided by boards; ratios are relative to budget 0 (release); p95 batch ms is the 95th percentile of per-batch latency (= per-board latency at batch 1); est. ms is the component-sum estimate used in Table 10.

setting	bs	budget	solved%	Δ [CI]	passes	ms/board	lat. ratio	pass ratio	p95 batch ms	est. ms
hard model / hard data	1	0	85.0	+0.0 [0.0,0.0]	12.00	95.2	1.00	1.00	118	77.6
hard model / hard data	1	24	89.4	+4.4 [3.2,5.6]	15.27	125.5	1.32	1.27	295	98.5
hard model / hard data	1	96	93.2	+8.2 [6.4,10.1]	21.38	174.8	1.84	1.78	882	138.9
hard model / hard data	32	0	85.0	+0.0 [0.0,0.0]	12.00	25.0	1.00	1.00	813	–
hard model / hard data	32	24	89.7	+4.7 [3.4,6.0]	15.22	34.2	1.37	1.27	1232	–
hard model / hard data	32	96	93.1	+8.1 [6.4,9.9]	21.41	54.4	2.17	1.78	2203	–
easy model / hard data	1	0	42.1	+0.0 [0.0,0.0]	12.00	95.3	1.00	1.00	120	77.6
easy model / hard data	1	24	52.0	+9.9 [8.2,11.6]	25.16	205.9	2.16	2.10	311	162.5
easy model / hard data	1	96	62.4	+20.3 [17.8,22.9]	55.99	465.6	4.89	4.67	923	359.5
easy model / hard data	32	0	42.1	+0.0 [0.0,0.0]	12.00	25.0	1.00	1.00	817	–
easy model / hard data	32	24	51.5	+9.4 [7.8,11.2]	25.18	54.0	2.15	2.10	1933	–
easy model / hard data	32	96	62.3	+20.2 [17.7,22.8]	56.27	123.4	4.93	4.69	4686	–

Table 7: Sequential ($w=1$) vs. parallel (width w) online verifier-gated refinement on a Tesla T4 (test split, 800 boards, fresh samples, two repeats in randomised order, all cells of a setting in one container). Δ is relative to the release (budget 0) in the same container.

setting	bs	budget	width	solved%	Δ vs release [CI]	mean passes	ms/board	p95 batch ms
hard model / hard data	1	0	1	85.0	+0.0 [0.0,0.0]	12.00	93.4	97
hard model / hard data	1	96	1	93.2	+8.2 [6.4,10.1]	21.38	173.5	875
hard model / hard data	1	96	4	93.2	+8.3 [6.4,10.1]	23.37	127.7	340
hard model / hard data	1	96	8	93.0	+8.0 [6.2,9.9]	26.40	128.1	311
hard model / hard data	1	192	8	94.7	+9.7 [7.7,11.7]	33.12	142.1	530
hard model / hard data	32	96	8	93.2	–	26.40	56.0	2407
easy model / hard data	1	0	1	42.1	+0.0 [0.0,0.0]	12.00	96.8	115
easy model / hard data	1	96	1	62.4	+20.3 [17.8,22.9]	55.99	477.8	961
easy model / hard data	1	96	4	62.6	+20.5 [17.9,23.1]	60.78	221.2	351
easy model / hard data	1	96	8	61.7	+19.6 [17.1,22.1]	67.56	217.5	315
easy model / hard data	1	192	8	68.6	+26.5 [23.8,29.3]	104.34	303.2	531
easy model / hard data	32	96	8	62.9	–	67.56	141.6	5445

Where the frontier saturates. The eight recorded samples per temperature leave 53.3% of failing hard/hard boards rescued and the frontier still rising, so we recorded 16 more samples at the frozen $T^* = 1.0$ for both hard settings (24 in total) and re-evaluated the frozen verifier gate up to budget 288 (Table 8). On hard/hard the gain keeps growing but saturates: +10.0 pp [8.0, 12.1] at budget 192 (27.34 mean passes) and +10.9 pp at 288 (85.0%→95.9%, 31.73 mean passes); within 8, 16 and 24 samples 53.3%, 66.7% and 72.5% of the failing boards are rescued, and 33 of 120 are never rescued. The easy model on hard boards saturates near half of its failures (49.7% within 24 samples; +28.7 pp at 116.37 mean passes). Sampling noise is small relative to board-level uncertainty: across 200 random re-orderings of each board’s recorded samples the solved rate at budget 96 varies with a standard deviation of 0.39 pp (hard/hard) and 0.87 pp (easy model on hard boards).

The verifier is the signal. At matched budget, gates that do not consult the verifier when allocating compute are clearly worse: on hard/hard at budget 24 a random gate of the same size gains +0.6 pp [0.1, 1.2], the final-level entropy gate +1.6 pp, and a calibrated gate from generator-only features +2.1 pp [1.2, 3.1], against +4.7 pp [3.4, 6.2] for the verifier gate. Refining every board (“uniform”) reaches the same accuracy as the verifier gate only by spending 26.01 instead of 15.22 passes.

Calibrated early abort does not pay. The oracle abort shows large headroom: if doomed samples could be identified at level 0, budget 96 would cost 13.67 instead of 21.46 mean passes at the same +8.0 pp. The calibrated abort predictor, held to $\leq 5\%$ false aborts on calibration (test: 4.3%), cuts 46.0% of doomed samples on hard/hard, but each false abort discards one of the few samples that would have verified, and the resulting policy is dominated by the plain verifier gate at matched compute (e.g. +3.1 pp at 14.68 passes vs. +3.1 pp at 13.80 passes). Decode-based features add nothing over generator-only features while costing a decode per level. The easy-fit abort transfers to hard/hard but collapses under the easy-model input shift (+2.1 pp at budget 24 vs. +10.0 pp without

Table 8: Verifier-gated refinement with 24 recorded samples at $T^* = 1.0$ (test split). Budgets up to 288 extra level passes (24 samples); sample-order sd is the standard deviation of the solved rate over 200 seeded permutations of each board’s samples (sampling noise, not board noise).

setting	budget	solved%	Δ vs det [CI]	mean passes	p95 passes	sample-order sd (pp)
hard model / hard data	12	88.1	+3.1 [2.0,4.4]	13.80	24	0.44
hard model / hard data	24	89.8	+4.7 [3.4,6.2]	15.22	36	0.45
hard model / hard data	48	90.9	+5.9 [4.2,7.5]	17.61	60	0.45
hard model / hard data	96	93.0	+8.0 [6.2,10.0]	21.46	108	0.39
hard model / hard data	144	94.0	+9.0 [7.1,11.0]	24.60	156	0.30
hard model / hard data	192	95.0	+10.0 [8.0,12.1]	27.34	204	0.26
hard model / hard data	288	95.9	+10.9 [8.8,13.1]	31.73	205	0.00
easy model / hard data	12	48.6	+6.5 [4.9,8.3]	18.95	24	0.68
easy model / hard data	24	52.1	+10.0 [8.0,12.1]	25.11	36	0.78
easy model / hard data	48	56.6	+14.5 [12.0,17.0]	36.39	60	0.84
easy model / hard data	96	64.1	+22.0 [19.1,24.9]	55.56	108	0.87
easy model / hard data	144	67.4	+25.2 [22.2,28.3]	72.14	156	0.65
easy model / hard data	192	69.2	+27.1 [24.1,30.2]	87.48	204	0.51
easy model / hard data	288	70.9	+28.7 [25.6,32.0]	116.37	300	0.00

Table 9: Stronger early-abort predictors (test split). Δ vs verifier is the paired solved-rate difference to the plain verifier gate at the same budget (pp, 95% CI); “saved” is the mean level passes per board saved relative to it. AUC is on the calibration split at levels 0 and 5. All fits on dev, thresholds on cal, frozen before test.

setting	predictor	threshold	cal AUC $k=0/5$	Δ vs verifier, b24	saved	Δ vs verifier, b96	saved
hard model / hard data	logistic	$FA \leq 5\%$	0.70 / 0.89	-1.6 [-2.5,-0.9]	0.54	-3.0 [-4.2,-1.9]	2.34
hard model / hard data	logistic	compute-targeted	0.70 / 0.89	-1.1 [-1.9,-0.5]	0.37	-1.9 [-2.9,-1.0]	1.09
hard model / hard data	boosted trees	$FA \leq 5\%$	0.67 / 0.89	-1.7 [-2.7,-0.9]	0.58	-3.4 [-4.7,-2.1]	2.80
hard model / hard data	boosted trees	compute-targeted	0.67 / 0.89	-1.1 [-1.9,-0.5]	0.39	-2.1 [-3.1,-1.2]	1.65
easy model / hard data	logistic	$FA \leq 5\%$	0.59 / 0.81	-1.5 [-2.4,-0.7]	1.42	-2.7 [-3.9,-1.6]	6.29
easy model / hard data	logistic	compute-targeted	0.59 / 0.81	-0.2 [-0.6,0.0]	0.31	-0.5 [-1.0,-0.1]	0.91
easy model / hard data	boosted trees	$FA \leq 5\%$	0.58 / 0.81	-2.0 [-3.0,-1.1]	1.45	-3.7 [-5.1,-2.5]	9.20
easy model / hard data	boosted trees	compute-targeted	0.58 / 0.81	+0.0 [-0.4,0.4]	0.36	-0.2 [-0.6,0.0]	1.98

abort). The intermediate levels that cannot be used as outputs (§6.1) are also too weakly predictive to prune samples profitably.

Is the abort negative a weak-predictor artifact? We pre-registered a stronger test (Table 9): gradient-boosted trees with fixed hyperparameters against the logistic model, each with the false-abort rule above and with a compute-targeted threshold (minimise calibration-split compute at budget 96 subject to at most 0.5 pp accuracy loss against the verifier gate). The signal is genuinely weak where it would matter: the calibration AUC for “this sample will verify” is only 0.70 at level 0 on hard/hard (0.59 for the easy model on hard boards) and reaches 0.89 only by level 5, after almost half of a sample’s level passes are spent. Boosted trees do not improve on it, and no predictor/threshold combination beats the plain verifier gate at any budget in either setting: the closest variant on hard/hard at budget 96 (logistic, compute-targeted) still loses -1.9 pp [-2.9, -1.0] while saving only 1.09 passes per board (boosted trees: -2.1 pp for 1.65 passes).

7 Ablations

Feature groups. Removing entropy, output-stability or validity features from the calibrated stopping rule changes nothing on test (all variants select full depth), because the label is constant below the last level; this is visible in Table 3 and in the frozen policy file. The ablation that matters is instead on the *label*: using “matches the full-depth output” instead of “solved” gives the identical answer, since no intermediate board equals the final one either (same-as-final rate 0 at every $k < 11$).

Decode overhead included vs. excluded. With overhead excluded, the verifier policy would be free; with it included it costs 115.3 vs. 98.9 ms per board at batch 1 (+16.6%, Table 5), a pure loss.

Per-item vs. batch-level exits. The batch-level variant of the calibrated policy behaves identically here because no row ever exits.

Table 10: Selective refinement on the test split (800 boards per setting). Budgets are extra level passes per board (12 = one extra full sample); “passes” is the mean total level passes per board (release = 12). Δ is the paired gain over the deterministic release output (pp, 95% bootstrap CI, 10,000 resamples). All thresholds frozen on dev/cal; oracle abort uses ground truth (diagnostic).

Policy	easy model / easy data			easy model / hard data			hard model / hard data		
	solved%	Δ [CI]	passes	solved%	Δ [CI]	passes	solved%	Δ [CI]	passes
Deterministic full depth (release)	99.4	+0.0 [0.0,0.0]	12.00	42.1	+0.0 [0.0,0.0]	12.00	85.0	+0.0 [0.0,0.0]	12.00
Verifier gate, budget 12	99.4	+0.0 [0.0,0.0]	12.07	48.6	+6.5 [4.9,8.3]	18.95	88.1	+3.1 [2.0,4.4]	13.80
Verifier gate, budget 24	99.6	+0.2 [0.0,0.6]	12.15	52.1	+10.0 [8.0,12.1]	25.11	89.8	+4.7 [3.4,6.2]	15.22
Verifier gate, budget 48	99.6	+0.2 [0.0,0.6]	12.24	56.6	+14.5 [12.0,17.0]	36.39	90.9	+5.9 [4.2,7.5]	17.61
Verifier gate, budget 96	99.6	+0.2 [0.0,0.6]	12.42	64.1	+22.0 [19.1,24.9]	55.56	93.0	+8.0 [6.2,10.0]	21.46
Uniform (every board), budget 24	99.6	+0.2 [0.0,0.6]	24.18	52.1	+10.0 [8.0,12.1]	31.68	89.8	+4.7 [3.4,6.2]	26.01
Random gate (no verifier), budget 24	99.4	+0.0 [0.0,0.0]	12.09	47.8	+5.6 [4.1,7.3]	23.49	85.6	+0.6 [0.1,1.2]	13.35
Entropy gate (no verifier), budget 24	99.4	+0.0 [0.0,0.0]	12.12	49.6	+7.5 [5.6,9.4]	25.14	86.6	+1.6 [0.8,2.5]	13.74
Calibrated gate (no verifier), budget 24	99.5	+0.1 [0.0,0.4]	12.16	51.1	+9.0 [7.0,11.0]	25.56	87.1	+2.1 [1.2,3.1]	14.10
Verifier gate + abort (gen. feats), 24	99.6	+0.2 [0.0,0.6]	12.13	50.7	+8.6 [6.8,10.6]	23.70	88.1	+3.1 [2.0,4.4]	14.68
Verifier gate + abort (+decode feats), 24	99.6	+0.2 [0.0,0.6]	12.14	50.7	+8.6 [6.8,10.6]	23.70	88.1	+3.1 [2.0,4.4]	14.67
abort fit on easy (shift), 24	—	—	—	44.2	+2.1 [1.1,3.2]	21.85	88.6	+3.6 [2.4,5.0]	15.21
Verifier gate + abort (gen. feats), 96	99.6	+0.2 [0.0,0.6]	12.37	61.5	+19.4 [16.6,22.1]	49.33	90.0	+5.0 [3.6,6.5]	19.13
Verifier gate + oracle abort, 96 (diagnostic)	99.6	+0.2 [0.0,0.6]	12.06	64.1	+22.0 [19.1,24.9]	18.05	93.0	+8.0 [6.2,10.0]	13.67

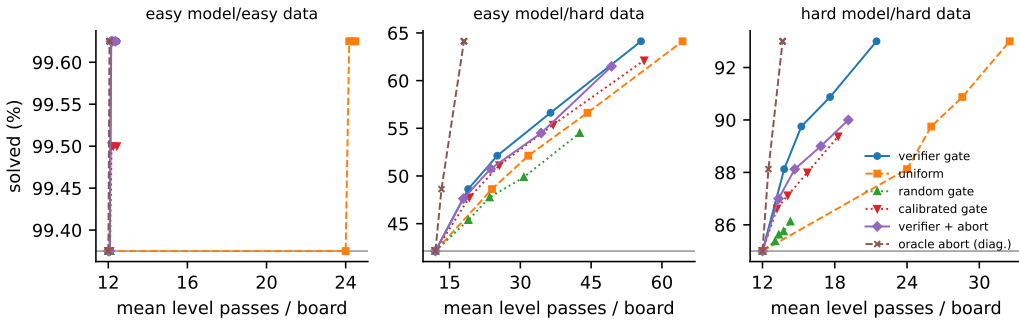


Figure 2: Quality/compute frontier of selective refinement (test split). Points are budgets 0, 12, 24, 48, 96 extra passes; grey line = deterministic release accuracy.

Easy→hard shift. Calibrators fit on easy boards remain calibrated on hard boards ($ECE \leq 0.008$) only because their predictions are near zero below the last level and near the full-depth accuracy at the last level; the heuristic rules, by contrast, transfer poorly (Table 3).

8 Limitations

The study covers one task family (Sudoku), the only released NCM task with an exact input-only verifier; text tasks (XSum, WMT14) would need an input-grounded proxy whose latency must be charged to the controller, and their intermediate reconstructions are coarse sketches rather than candidate outputs. The negative result is specific to this tokenizer design (16 codes per level, residual hierarchy ending at full resolution); tokenizers with larger coarse codebooks or non-residual hierarchies could behave differently, and we did not retrain tokenizers (training code is not released). GPU latencies are from one accelerator (T4) and the eager PyTorch path; compiled or CUDA-graph execution would shrink the launch-bound regime and change the batch-1 picture, though not the level-pass accounting. Refinement latencies in Table 10 are component-sum estimates; end-to-end measurements (Table 6) exist only for the verifier gate at three budgets on one GPU, and the refinement study records eight samples per temperature (24 at T^* for the hard settings), so budgets beyond 288 extra passes are untested. Refinement trades mean compute for a heavy latency tail. All calibrators are small logistic models; richer predictors could sharpen the early-abort frontier but cannot change the zero-headroom result.

9 Reproducibility statement

Upstream code is pinned to `HazyResearch/ncm@c702742` (MIT), checkpoints to `hazyresearch/ncm-checkpoints@d342ce8` and boards to

hazyresearch/ncm-datasets@6580ba4 (both MIT); SHA-256 of every weight and data file is verified at load time and recorded in `sources.lock.json`. Splits are seeded and stored; every policy threshold is frozen in `configs/frozen_policy.json` and `configs/frozen_refinement.json` before the test split is read. Every table and figure is generated by a script from raw run records under `results/` (`scripts/make_tables.py`, `scripts/make_figures.py`); the macros in this document are emitted by the same script. The cost ledger lists every Modal job with its measured duration and list-price estimate. Tests (pytest tests) cover release agreement, prefix consistency, batch compaction, future-code independence, the aliasing trap and cost accounting.

References

- Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv preprint arXiv:2110.01052*, 2021.
- Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations (ICLR)*, 2020.
- Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.
- Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017.
- Hermann Kumbong, Ishan S. Khare, Iddah Mlauzi, Andrew Woen, Charles Ding, E. Kelly Buchanan, Simran Arora, Azalia Mirhoseini, and Christopher Ré. Next-concept modeling: Fast, parallel coarse-to-fine text generation, 2026. URL <https://arxiv.org/abs/2610.14162>.
- John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*. MIT Press, 1999.
- Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *International Conference on Pattern Recognition (ICPR)*, 2016.
- Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.
- Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.