Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Polyphase TTT: Reducing Chunk-Boundary Sensitivity in Frozen Test-Time Training Models

Dian Yu

dian.yu2@student.unsw.edu.au

UNSW Sydney https://orcid.org/0009-0002-0107-1014

Research Article

Keywords: test-time training, fast weights, recurrent neural networks, chunking, robustness, training-free inference

Posted Date: November 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11872967/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Polyphase TTT: Reducing Chunk-Boundary Sensitivity in Frozen Test-Time Training Models

Dian Yu[1*]

[1*]UNSW Sydney, Sydney, NSW, Australia.

Corresponding author(s). E-mail(s): dian.yu2@student.unsw.edu.au;

Abstract

Test-Time Training (TTT) layers use the parameters of an online learner as recurrent state. Shifting the learner's chunk boundaries can change the model's answer to the same text even when the chunk width stays fixed. We measure this sensitivity and introduce Polyphase TTT, an inference method for frozen checkpoints. Each pass uses a different placement of the same chunk grid, with local rotary positions and learning-rate indices aligned to that grid. The predictions from these passes make a bounded correction to the default prediction. A confidence cascade decides how many passes to run. On 384 held-out LAMBADA examples per checkpoint, the share of examples whose predictions differ across four grid origins drops from 16.15% to 11.72% at 350 million parameters and from 15.63% to 10.94% at 760 million. At the default origin, accuracy stays at 42.45% for the smaller checkpoint and changes from 42.97% to 43.49% for the larger one. On 243 four-choice questions derived from SQuAD and HotpotQA, five fewer examples change their answer across origins, and combined accuracy is unchanged. On LAMBADA, the cascade averages 1.39–1.44 passes and takes 1.51–1.65 times as long as one pass.

Keywords: test-time training, fast weights, recurrent neural networks, chunking, robustness, training-free inference

1 Introduction

Consider a language model asked to predict the final word of a passage. In one run, its fast-weight learner starts a new chunk after token 16. In another, the first cut falls after token 12. The passage and checkpoint are identical, yet the predicted word

can change. We study this dependence on where the chunk grid starts in Test-Time Training (TTT).

TTT layers use the parameters of a small online learner as recurrent state (Sun et al. 2025). The outer model's *slow weights* are learned offline and frozen at inference. The inner learner's *fast weights* change as the model processes a sequence. In TTT-MLP that learner is a two-layer perceptron trained online on a self-supervised reconstruction objective.

TTT-MLP groups adjacent tokens into fixed-width *chunks* for parallel execution. A cut determines when the fast-weight reference is refreshed. It also determines each token's local rotary position and the position-indexed factor in its inner learning rate.

Natural text has no boundary every 16 tokens. In Section 4.2, LaCT observes that "text sequences lack inherent chunk structures" and treats chunk size as a hyperparameter (Zhang et al. 2026). LaCT uses block-wise causal updates and local attention to handle larger chunks. We ask what happens when a checkpoint and its chunk width are already fixed. Does shifting the *origin* of the grid change its predictions?

Holding the token sequence, checkpoint, and chunk width fixed, moving the origin changes around one in six answers on held-out LAMBADA examples. We call this *chunk-origin instability*. Aggregate accuracy can conceal it when gains and losses across examples cancel.

We propose *Polyphase TTT*. A *phase* is one placement of the trained-width chunk grid. Its *calendar* specifies the chunk cuts and the local indices used by the rotary embedding and learning-rate schedule. Each phase runs the same checkpoint on exactly the same token sequence. Because the cuts determine when the fast-weight reference is refreshed, different calendars can produce different fast states and predictions.

The default calendar provides an *anchor* for combining predictions. We add a fixed fraction of the difference between the phase average and the anchor prediction. A confidence cascade starts at the default origin, evaluates the opposite origin when the anchor has a low confidence margin, and adds the two remaining origins if the first two predictions disagree.

Our main findings are:

- On 384 held-out LAMBADA examples per checkpoint, four origins yield different predictions on 16.15% and 15.63% of examples. A separate sweep over all 16 origins finds greater disagreement at larger offsets from the default origin.
- Polyphase TTT reduces these rates to 11.72% and 10.94%. A four-choice check using SQuAD and HotpotQA yields five fewer prediction flips across 243 questions.
- The cascade averages 1.39–1.44 passes and takes 1.51–1.65 times the single-pass latency. At the same pass budget, it reduces more flips than random allocation and performs close to offline confidence ranking.

2 Related Work

2.1 Test-time training and chunkwise sequence models

TTT-Linear and TTT-MLP use a linear map or a two-layer perceptron as recurrent state and update it on a self-supervised objective (Sun et al. 2025). Chunkwise execution is especially important for TTT-MLP because it turns many small online-learning operations into larger matrix operations.

Transformer-XL identifies context fragmentation at fixed segment boundaries and carries hidden activations across segments (Dai et al. 2019). A TTT fast-weight learner instead changes its update reference at a boundary. LaCT makes these dependencies explicit with block-wise causal masks and adds a local attention path in its language-model architecture (Zhang et al. 2026). These approaches change how a model carries information through a partition. Here the object of study is the origin of a fixed-width update grid after pretraining, with the same update operator used at each origin.

TNT trains across a hierarchy of chunk *widths* and briefly fine-tunes at the deployment width (Li et al. 2026). Recent preprints take complementary approaches. Modular TTT separates the learner, objective, and update organization (arXiv:2608.07110). E^2-TTT develops a chunk-parallel implementation of its per-token update dynamics (arXiv:2608.21308). Our intervention applies to released TTT-MLP checkpoints and varies where their existing width-16 grid begins.

2.2 Prediction marginalization and selective computation

Deep ensembles vary learned parameters (Lakshminarayanan et al. 2017). Test-time augmentation varies input views, and its aggregation and selection rules have been studied separately (Shanmugam et al. 2021; Kim et al. 2020). Polyphase TTT uses an ensemble over internal update schedules for a single checkpoint and token sequence. Its intervention moves update boundaries, rotary indices, and the position-indexed learning-rate schedule together. Selective evaluation shares the motivation of confidence-based early exit (Xin et al. 2020). ConsistentEE trains a policy to choose when a language model exits its stack of layers and uses parallel decoding to address cache updates (Zeng et al. 2026). Polyphase TTT retains the full model depth on every pass and allocates additional executions under different chunk origins. Uniform phase averaging serves as a fusion baseline.

2.3 Positional phases

Rotary position embeddings encode relative offsets by rotating queries and keys (Su et al. 2024). TTT-MLP indexes these rotations within each mini-batch. Each grid placement uses the same range of rotary indices and assigns them to different sequence positions.

3 Why Chunk Origin Changes Predictions

3.1 Slow state, fast state, and chunkwise computation

Let $x_{0:L-1}$ be a token sequence indexed from zero, and let W_t denote the fast parameters of a TTT layer after processing position t. Write W_{-1} for the initial state. A tokenwise update can be written as

$$y_t = f(x_t; W_{t-1}), \qquad W_t = \mathcal{U}(W_{t-1}, x_t). \tag{1}$$

An efficient mini-batch implementation groups b positions and evaluates a chunk operator

$$(y_{c:c+b-1}, W_{c+b-1}) = \mathcal{C}_b(x_{c:c+b-1}, W_{c-1}). \tag{2}$$

For the trained width $b = 16$, the default cuts are $0, 16, 32, \ldots$. Within a full chunk, indices $0, \ldots, 15$ select the rotary positions and the position-dependent learning-rate factors.

Tokens in one chunk use the same reference fast weights in the within-chunk update. The next chunk starts from a refreshed reference. Moving a cut changes which tokens share that reference and can affect the fast state carried forward.

3.2 Width and origin are different coordinates

The *width* b determines the nominal number of tokens per update. The *origin* r determines where blocks of that width are aligned. Width changes the operating scale of the online learner; origin translates the same grid. For a 32-token illustration with $b = 16$, two calendars are

$$\mathcal{K}_0 : [0{:}15] \mid [16{:}31], \tag{3}$$

$$\mathcal{K}_4 : [0{:}3] \mid [4{:}19] \mid [20{:}31]. \tag{4}$$

The second calendar has a short prefix and suffix, and each full chunk still has width b. Both calendars cover the same 32 tokens without padding. We vary the origin at inference with the trained width held fixed.

3.3 Internally consistent calendars

For an offset $r \in \{0, \ldots, b-1\}$, define a calendar $\mathcal{K}_r$ with cuts

$$0,\ r,\ r+b,\ r+2b, \ldots, L, \tag{5}$$

retaining only cuts strictly inside the sequence between the endpoints 0 and L. The local index at token position t is

$$\pi_r(t) = (t - r) \bmod b. \tag{6}$$

The index $\pi_r(t)$ selects both the rotary position and the position-dependent factor in the inner learning rate. Each pass starts from the checkpoint's initial fast state and

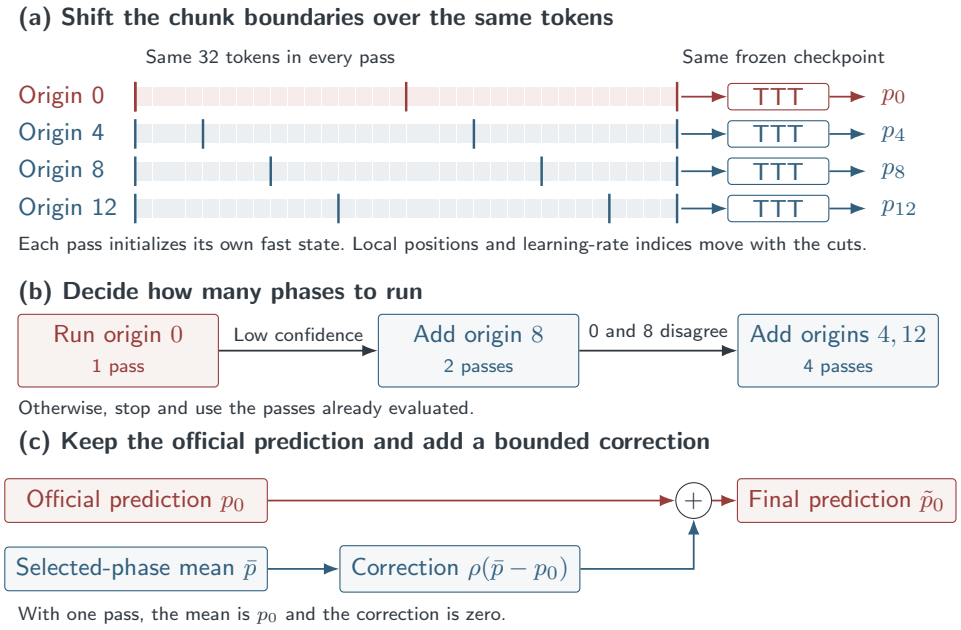

Fig. 1 Polyphase TTT at inference. (a) Each origin places width-16 cuts over the same tokens and runs the same checkpoint from its standard initial fast state. Dark ticks mark chunk boundaries. (b) The cascade evaluates one, two, or four origins. (c) The phase mean $\bar{p}$ yields a bounded residual around the official prediction p_0, producing $\tilde{p}_0$

carries the updated state from chunk to chunk. The passes maintain separate fast states.

We use four quarter phases,

$$\mathcal{R} = \{0, 4, 8, 12\}, \tag{7}$$

for $b = 16$ (Fig. 1). Phase 0 uses the checkpoint's default calendar. The other three phases shift it by a quarter, a half, and three quarters of a chunk.

3.4 Decision instability and its two forms

Let $\hat{y}_r(x)$ be the answer decision under calendar r and $c_r(x) \in \{0, 1\}$ its correctness. We measure two forms of instability over the phase set $\mathcal{R}$:

$$I_{\text{pred}}(x) = \mathbb{1}\big[\exists r, s \in \mathcal{R} : \hat{y}_r(x) \neq \hat{y}_s(x)\big], \tag{8}$$

$$I_{\text{corr}}(x) = \mathbb{1}\big[\exists r, s \in \mathcal{R} : c_r(x) \neq c_s(x)\big]. \tag{9}$$

A prediction flip means that two origins give different answers, even if both answers are wrong. A correctness flip means that at least one origin gives the correct answer

and another gives a wrong one. We compare both outcomes on the same examples before and after applying Polyphase TTT.

4 Polyphase TTT

4.1 Phase marginalization

For a target position, let $p_r \in \Delta^{V-1}$ be the vocabulary distribution produced by calendar r. The full polyphase mean is

$$\bar{p} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} p_r. \tag{10}$$

Uniform averaging gives the same prediction regardless of which of the four origins serves as the anchor. It gives equal weight to each phase, so it can change a confident and correct default prediction.

To limit these changes, we use the default output as the anchor and add a fraction of the difference between the phase mean and the anchor:

$$\tilde{p}_0 = p_0 + \rho(\bar{p} - p_0) = (1 - \rho)p_0 + \rho\bar{p}, \qquad 0 \le \rho \le 1. \tag{11}$$

We fix $\rho = 0.25$. Setting $\rho = 0$ returns the anchor distribution. Setting $\rho = 1$ returns the mean over the evaluated phases.

The output is a valid distribution. Let k be the number of evaluated origins, including the anchor, and let p_{alt} average the other $k - 1$ origins when $k > 1$. The anchor is present in the mean, so the effective weight on the other origins is $\alpha_k = \rho(k - 1)/k$. The displacement obeys

$$\|\tilde{p}_0 - p_0\|_1 = \alpha_k \|p_{\text{alt}} - p_0\|_1 \le 2\alpha_k. \tag{12}$$

At $\rho = 0.25$, the anchor retains 87.5% of the weight with two passes and 81.25% with four. With one pass the correction is zero. A convex mixture preserves any top prediction shared by all passes under the same tie-breaking rule. A new correctness flip can still arise if the passes originally gave different wrong answers and the mixture corrects only some of them.

At deployment the anchor is always phase 0. For the robustness audit, we also rotate the anchor through every $r \in \mathcal{R}$ and apply the same rule,

$$\tilde{p}_r = (1 - \rho)p_r + \rho\bar{p}. \tag{13}$$

With full evaluation, all anchors share the same mean, so $\tilde{p}_r - \tilde{p}_s = (1 - \rho)(p_r - p_s)$. This contracts the difference between the predictive distributions. The cascade can select different phase sets for different anchors. We therefore measure changes in the top prediction directly and report accuracy at the default origin separately.

Algorithm 1 Confidence-cascaded Polyphase TTT

Require: input x, frozen checkpoint F, phases $\{0, 4, 8, 12\}$, $\rho = 0.25$, $\tau = 0.04$

$p_0 \leftarrow F(x; \mathcal{K}_0)$
if $m(p_0) \geq \tau$ **then**
    return p_0
end if
$p_8 \leftarrow F(x; \mathcal{K}_8)$
if the predictions from p_0 and p_8 agree **then**
    $\bar{p} \leftarrow (p_0 + p_8)/2$
    return $(1 - \rho)p_0 + \rho\bar{p}$
end if
$p_4 \leftarrow F(x; \mathcal{K}_4)$ and $p_{12} \leftarrow F(x; \mathcal{K}_{12})$
$\bar{p} \leftarrow (p_0 + p_4 + p_8 + p_{12})/4$
return $(1 - \rho)p_0 + \rho\bar{p}$

4.2 Confidence cascade

Evaluating every phase on every example is unnecessary when the official prediction is already decisive. Let

$$m(p_0) = p_0^{(1)} - p_0^{(2)} \tag{14}$$

be the gap between the two highest probabilities for the first answer token in LAMBADA. For question answering, we compute the gap after normalizing scores over the four choices. We select the threshold $\tau = 0.04$ on development data.

When the margin reaches the threshold, the cascade returns the anchor. Otherwise it evaluates phase 8, which shifts the grid by half a chunk. In the original LAMBADA evaluation, the agreement check compares the predicted token sequences over the teacher-forced span. We separately replay this check using only the first answer token, so the decision to run extra phases uses only the passage prefix. In the origin audit, the offsets in Algorithm 1 are relative to the chosen anchor modulo 16. Timed runs use phase 0.

4.3 Implementation and cost

Each pass uses one origin across all TTT layers and heads. In a shifted pass, the short prefix uses the final indices of the width-16 schedule. Full chunks begin at local index zero, and the short suffix uses the leading indices. We run the passes serially and retain their output distributions for fusion.

For an input x, the number of passes $N(x)$ is one, two, or four. We report the mean pass count and measure elapsed time separately to capture the overhead of dispatch and fusion. Each pass releases its fast state before the next pass begins.

5 Experimental Protocol

5.1 Models

We use the public `ttt-mlp-350m-books-2k` and `ttt-mlp-760m-pile-8k` checkpoints (Sun et al. 2025). Parameters are frozen and the native width is 16. The checkpoints differ in both size and pretraining configuration, so the comparison tests replication across checkpoints. Phase 0 uses the official forward path.

5.2 Data and metrics

LAMBADA

LAMBADA tests final-word prediction from a discourse passage (Paperno et al. 2016). We split off the last whitespace-separated word and retain examples whose tokenization agrees across the split and whose total length is at most 512 tokens. A fixed seeded sample provides 384 test examples per checkpoint, excluding the earlier 64- and 96-example screens. At each answer position we score the next token given the true preceding tokens. An example is correct if every predicted token in the final word matches the target token sequence. The flip event compares those sequences across origins.

Four-choice question answering

We construct an external check from the SQuAD v1.1 and HotpotQA distractor development data (Rajpurkar et al. 2016; Yang et al. 2018). SQuAD supplies questions with a single supporting passage. HotpotQA supplies questions with annotated multi-document evidence. Each group contains four source questions with distinct single-token answer candidates. The SQuAD context concatenates the four passages under a 512-token cap. The HotpotQA context concatenates the four evidence bundles under a 1,024-token cap. Each prompt uses the shared context and one question. We select the highest-scoring candidate and compare it with the source answer. The QA cascade is evaluated by replaying the selection rule on saved scores.

The final QA check includes 61 SQuAD questions at 350M and 182 HotpotQA questions at 760M. The latter comprise 62 questions from the first held-out run and 120 from a later 144-question run. We remove source-question IDs already present in the development or earlier held-out set. Questions that share evidence bundles can still be dependent, so we treat the QA counts as descriptive.

For each method, *default accuracy* uses phase 0 as anchor. *Mean phase accuracy* averages accuracy over the four anchors, and *worst phase accuracy* reports the lowest of the four accuracies. For LAMBADA, we count how many flips the cascade removes or introduces on the same examples. Because the fusion rule preserves unanimous predictions, we report prediction flip counts descriptively. We use a two-sided exact sign test for the change in correctness flip events.

Table 1 Primary LAMBADA results on 384 held-out examples per checkpoint. Default accuracy uses phase 0. Mean, worst, and flip columns rotate the anchor over all four calendars. Accuracy and instability are proportions. Latency is measured for the single-pass baseline and the cascade. For the four-pass methods, the shared diagnostic audit is timed separately

Model	Method	Default acc.	Mean acc.	Worst acc.	Pred. flip	Corr. flip	Latency
350M	Uncorrected calendars	0.4245	0.4297	0.4193	0.1615	0.0833	1.00×
	Uniform four-phase	0.4297	0.4297	0.4297	0.0000	0.0000	—
	Anchored residual	0.4245	0.4310	0.4245	0.1172	0.0469	—
	Confidence cascade	**0.4245**	**0.4303**	**0.4245**	**0.1172**	**0.0469**	**1.51×**
760M	Uncorrected calendars	0.4297	0.4303	0.4271	0.1563	0.0807	1.00×
	Uniform four-phase	0.4375	0.4375	0.4375	0.0000	0.0000	—
	Anchored residual	0.4349	0.4310	0.4245	0.1068	0.0547	—
	Confidence cascade	**0.4349**	**0.4303**	**0.4219**	**0.1094**	**0.0547**	**1.65×**

5.3 Selection protocol

The final evaluation uses four evenly spaced origins and $\rho = 0.25$. On a separate pool of 162 constructed SQuAD questions, we scan confidence thresholds in increments of 0.01. We minimize mean phase count subject to matching or improving on the full residual's canonical accuracy, worst-origin accuracy and both flip rates. This selects $\tau = 0.04$, which is reused on the final evaluations. The complete origin sweep uses the earlier LAMBADA diagnostic pool.

5.4 Implementation and reproducibility

The evaluation holds token IDs, sequence length, and order fixed across origins. The phase-0 baseline calls the original forward directly. The timed LAMBADA cascade reproduces the phase-0 predictions obtained when its policy is replayed on four saved passes. We measure peak allocated CUDA memory and normalize elapsed time against the mean of the single-pass runs immediately before and after the cascade and full audit. Online Resource 1 contains seeds, identifiers, scores, and reproduction scripts. The workflow figure uses TikZ and the results figures use Matplotlib.

At each checkpoint, 226 of the 384 examples have multi-token answers. Using first-token agreement in place of full-span agreement produces the same phase allocations and predictions for every example and anchor at both checkpoints. We continue to score the full answer span with teacher forcing.

6 Results

6.1 Frozen TTT is sensitive to chunk origin

Across the four origins, predictions differ on 16.15% of examples at 350M and 15.63% at 760M. Correctness differs on 8.33% and 8.07%, respectively (Table 1). Accuracy varies much less across origins because some examples become correct while others become incorrect.

We also evaluate all 16 origins on a separate 96-example diagnostic set at 350M (Table 2). Across all origins, predictions differ on 20.83% of examples and correctness differs on 9.38%. Disagreement with phase 0 tends to increase with circular distance

Table 2 Complete origin sweep on 96 fixed 350M LAMBADA examples. Coverage is relative to the examples detected by all 16 origins

Origin set	Pred. flip	Corr. flip	Pred. coverage	Corr. coverage
$\{0, 4, 8, 12\}$	0.1354	0.0521	0.650	0.556
All 16 origins	0.2083	0.0938	1.000	1.000

Table 3 Paired flip counts for the uncorrected model and the cascade. For correctness flips, a two-sided exact sign test compares removed and introduced events

Model	Event	Removed	Introduced	Exact p
350M	Prediction flip	17	0	—
	Correctness flip	14	0	1.22×10^{-4}
760M	Prediction flip	18	0	—
	Correctness flip	10	0	1.95×10^{-3}

from that phase. The descriptive Pearson correlations are 0.905 for prediction disagreement and 0.872 for correctness disagreement. Phase 8, half a chunk away, changes 12 predictions and five correctness outcomes. The four selected phases capture 13 of the 20 prediction flips and five of the nine correctness flips found by the full sweep.

6.2 Polyphase residuals reduce decision instability

The cascade reduces prediction flips by 27.4% at 350M and 30.0% at 760M. Correctness flips fall by 43.8% and 32.3%, respectively. On the same examples, the cascade removes 17 and 18 prediction flips and 14 and 10 correctness flips. We observe no new correctness flips (Table 3).

At the default origin, accuracy stays at 42.45% at 350M and is 42.97% before and 43.49% after correction at 760M (Fig. 2). The latter difference corresponds to two more correct answers across 384 examples.

6.3 Ablation of fusion and phase allocation

Table 1 first compares two rules that both use four passes. Uniform averaging returns the same answer for every anchor and achieves the highest default-origin accuracy in these subsets. The anchored residual keeps most of the weight on the default prediction, which allows some variation between anchors. Applying the cascade to the anchored rule lets it return early.

At 350M, the cascade and the full residual both remove 17 prediction flips. At 760M the cascade removes 18 and the full residual removes 19. Their reductions in correctness flips are the same at both sizes. At the default origin, their predictions agree on all 384 examples for each checkpoint. At 760M, the cascade's worst-origin accuracy is 42.19%, compared with 42.71% for the uncorrected model.

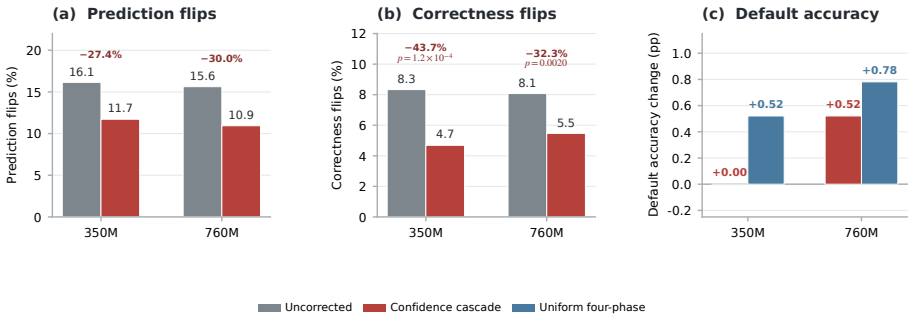

Fig. 2 Chunk-origin robustness and default-origin accuracy on held-out LAMBADA. Panels (a) and (b) show relative reductions in flip rates. Panel (b) also reports exact sign-test values. Panel (c) shows changes in accuracy, in percentage points, from the default-origin baseline

Table 4 Allocation at equal forward-pass budgets on held-out LAMBADA. Prediction and correctness flips are measured over the four anchors. Accuracy uses phase 0. All values are percentages

	350M			760M		
Allocation	Pred. flip	Corr. flip	Acc.	Pred. flip	Corr. flip	Acc.
Random (same quotas)	15.16	7.53	42.40	14.49	7.47	43.10
Margin rank (same quotas)	11.98	4.95	42.19	11.20	5.47	43.49
Margin rank (one or four)	12.24	5.21	42.45	11.20	5.73	43.49
Confidence cascade	11.72	4.69	42.45	10.94	5.47	43.49

The allocation controls use the same phases and fusion rule as the cascade at each pass count (Table 4). The first two controls match the number of one-, two-, and four-pass examples at each anchor. Random allocation draws an ordering of the examples and reuses it across the four anchors. We average over 10,000 random orderings. Offline margin ranking assigns four passes to the lowest-margin examples and two passes to the next lowest. A second margin control uses one or four passes and matches the mean budget in expectation. All controls use only prediction scores and the pass budget.

The cascade reduces prediction flips by 3.44 and 3.55 percentage points compared with matched random allocation. Both margin controls recover most of this gain. Compared with margin ranking at the same quotas, the cascade removes one additional prediction flip per checkpoint. It also decides when to stop for each example without ranking the evaluation batch.

The cascade stops after one pass on 288 of 384 examples at 350M and 286 at 760M. It uses two passes on 69 and 62 examples, and four on 27 and 36. Mean pass counts are 1.391 and 1.443, and measured latency is 1.51 and 1.65 times the single-pass reference (Fig. 3). Peak allocated memory rises by 0.38% and 0.03%. The separate four-pass audit takes 4.98–4.99 times the reference latency, including scoring several fusion rules and anchor rotations from the same forward passes.

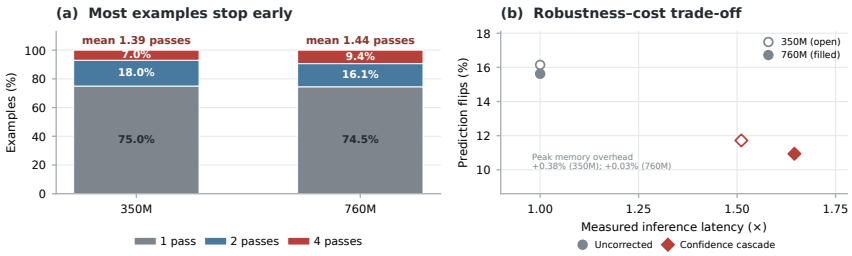

Fig. 3 Efficiency on LAMBADA. (a) Distribution of passes under the default anchor. (b) Prediction flips and measured latency for the single-pass reference and the cascade. For the uncorrected reference, flips are measured across origins and latency is measured for one pass. Timing of the shared four-pass diagnostic audit is reported separately in the text

Table 5 External four-choice question-answering checks. Each flip column reports uncorrected → cascade rates. "Rem./int." gives paired events removed and introduced

Dataset	n	Prediction flip	Correctness flip	Canonical accuracy	Pred. rem./int.	Corr. rem./int.
SQuAD	61	0.0820 → 0.0656	0.0492 → 0.0492	0.6066 → 0.6066	1/0	0/0
HotpotQA	182	0.0714 → 0.0495	0.0549 → 0.0330	0.6813 → 0.6813	4/0	4/0
Combined	243	0.0741 → 0.0535	0.0535 → 0.0370	0.6626 → 0.6626	5/0	4/0

6.4 External question-answering checks

On the four-choice tasks, the cascade removes one prediction flip on SQuAD and four on HotpotQA. The number of HotpotQA correctness flips also falls by four (Table 5). At the default origin, one corrected HotpotQA answer offsets one new error. Combined accuracy at that origin stays at 66.26%. The cascade selects 1.08 phases on average across questions and anchors. We report these small counts as descriptive results.

7 Discussion

At 760M, the uncorrected model changes its answer on 60 of 384 examples across origins, even though the accuracy of the four origins differs little. An execution that reproduces the same tokenwise recurrence at every origin would give identical predictions, provided its inputs and positional indices were held fixed. In TTT-MLP, our intervention moves the fast-weight refresh points, rotary indices, and learning-rate indices together. Their individual contributions remain to be measured.

In the full sweep, the four selected origins reveal 65% of the examples whose predictions vary across all 16 origins. The cascade reduces the number of passes needed to use those four origins, with some flips remaining. Its worst-origin accuracy at 760M falls slightly. Our results cover two frozen width-16 checkpoints and short, scored tasks. Free-running generation, longer contexts, and other model families remain open.

8 Conclusion

Shifting the width-16 chunk grid changes about one in six LAMBADA predictions in both frozen checkpoints. Pooling predictions from several grid placements reduces that

rate to about one in nine while keeping accuracy at the default origin close to baseline. The cascade uses the extra passes more effectively than random allocation and takes 1.51–1.65 times the single-pass latency. This makes chunk origin a measurable source of instability in frozen TTT-MLP. Randomizing the origin during pretraining would test whether checkpoints can learn to be less sensitive to it.

Statements and Declarations

Funding. No external funding was received for this study.

Competing interests. The author declares no competing interests.

Data availability. All evaluated datasets and pretrained checkpoints are publicly available from their original sources. Locked example manifests and raw result files used for the reported analyses are provided in Online Resource 1.

Code availability. Reproduction code, environment requirements, and figure-generation scripts are provided in Online Resource 1.

Author contributions. D.Y. conceived the method, implemented the experiments, analyzed the results, and wrote the manuscript.

References

Dai Z, Yang Z, Yang Y, et al (2019) Transformer-XL: Attentive language models beyond a fixed-length context. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp 2978–2988, https://doi.org/10.18653/v1/P19-1285

Kim I, Kim Y, Kim S (2020) Learning loss for test-time augmentation. In: Advances in Neural Information Processing Systems, pp 4163–4174

Lakshminarayanan B, Pritzel A, Blundell C (2017) Simple and scalable predictive uncertainty estimation using deep ensembles. In: Advances in Neural Information Processing Systems

Li Z, Behrouz A, Deng Y, et al (2026) TNT: Improving chunkwise training for test-time memorization. In: International Conference on Learning Representations, arXiv:2511.07343

Paperno D, Kruszewski G, Lazaridou A, et al (2016) The LAMBADA dataset: Word prediction requiring a broad discourse context. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp 1525–1534

Rajpurkar P, Zhang J, Lopyrev K, et al (2016) SQuAD: 100,000+ questions for machine comprehension of text. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp 2383–2392

Shanmugam D, Blalock D, Balakrishnan G, et al (2021) Better aggregation in test-time augmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision

Su J, Ahmed M, Lu Y, et al (2024) RoFormer: Enhanced transformer with rotary position embedding. Neurocomputing 568:127063

Sun Y, Li X, Dalal K, et al (2025) Learning to (learn at test time): RNNs with expressive hidden states. In: Proceedings of the 42nd International Conference on Machine Learning, Proceedings of Machine Learning Research, vol 267. PMLR, Vancouver, Canada, pp 57503–57522

Xin J, Tang R, Lee J, et al (2020) DeeBERT: Dynamic early exiting for accelerating BERT inference. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp 2246–2251, https://doi.org/10.18653/v1/2020.acl-main.204

Yang Z, Qi P, Zhang S, et al (2018) HotpotQA: A dataset for diverse, explainable multi-hop question answering. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp 2369–2380

Zeng Z, Chen Z, Zhuang H (2026) A training-inference consistent framework for early exiting in language models with parallel decoding. International Journal of Machine Learning and Cybernetics 17(3):121. https://doi.org/10.1007/s13042-025-02952-y

Zhang T, Bi S, Hong Y, et al (2026) Test-time training done right. In: International Conference on Learning Representations

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Figure1.png