Inner Loop Inference for Pretrained Transformers:
Unlocking Latent Capabilities Without Training

Jonathan Lys[1] Vincent Gripon[1] Bastien Pasdeloup[1] Axel Marmoret[1]
Lukas Mauch[2] Fabien Cardinaux[2] Ghouthi Boukli Hacene[2]

[1]IMT Atlantique, Lab-STICC, UMR CNRS 6285, Brest, France
[2]Sony Europe Ltd., Stuttgart Technology Center, EUREC, Germany

EUSIPCO 2026 — SiG-DML-L1: Explainable and Interpretable Learning

Mechanistic interpretability point of view

Transformers perform iterative refinement of a propagated latent state.

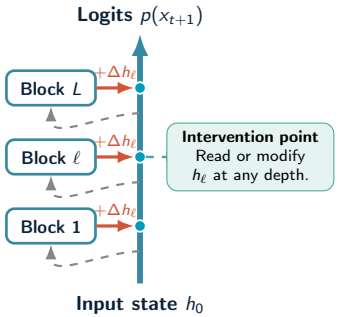
Residual computation

$$h_\ell = h_{\ell-1} + \Delta h_\ell$$

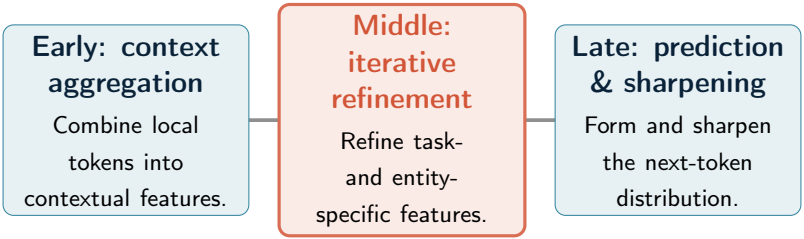
Each block reads the current state and writes an incremental update.

Mechanistic interpretability view

Probe, patch, ablate, or replace h_ℓ to understand **what each layer does**.



Motivation: MI suggests stages of inference

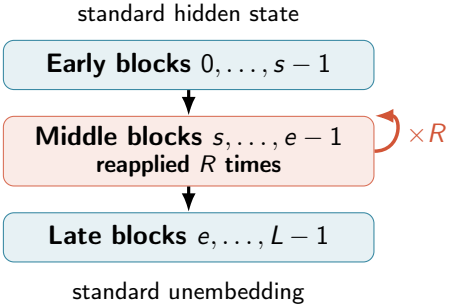


Our focus: the refinement layers

Find the refinement layers $\longrightarrow$ **Reuse them to refine** h $\longrightarrow$ **Improve accuracy**

Simplified from the four-stage hypothesis of Lad, Gurnee, and Tegmark (2024); stage boundaries are approximate.

Method: Middle looping



Hyperparameters

- s: first reused block
- e: first block after the loop
- R: number of loops

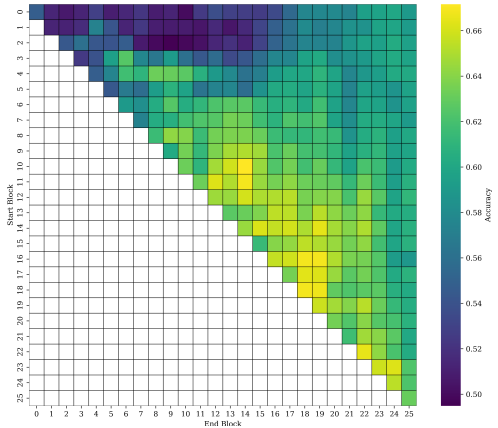
Added test-time compute

For loop length $N = e - s$:

$$K = L + (R-1)N$$

Weights remain unchanged.

A simple baseline: naive looping



Accuracy over loop start/end pairs; darker cells indicate stronger degradation.

Gemma 2–2B, $R = 3$, on the WinoGrande benchmark

Across the complete start–end sweep:

- every naive loop is **below the baseline**;
- severe settings approach chance accuracy.

Hypothesis

Reapplied blocks receive hidden states unlike those seen during pretraining. Repeated residual updates can then amplify the mismatch.

Method: loop anchoring as regularization

Regularized update

First apply the reused block range F:

$$h^{(t)} = F\left(\hat{h}^{(t-1)}\right)$$

Then interpolate cached loop states:

$$\boxed{\hat{h}^{(t)} = \sum_{i=0}^{t} \alpha_i h^{(i)}} \qquad \sum_{i=0}^{t} \alpha_i = 1$$

Here $h^{(0)}$ is the non-looped reference at the loop boundary.

Uniform — the most reliable variant

$$\hat{h}^{(t)} = \frac{1}{t+1} \sum_{i=0}^{t} h^{(i)}$$

Equal averaging damps the drift while retaining the new update.

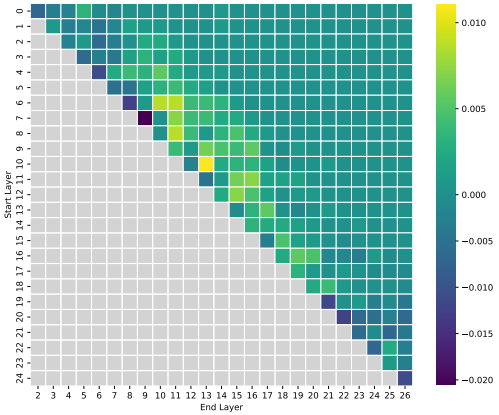
Two alternatives

Moving average: $\hat{h}^{(t)} = \eta h^{(0)} + (1-\eta) h^{(t)}$

Auto-alignment: softmax weights from $\langle h^{(0)}, h^{(i)} \rangle$

The loop is useful only when its trajectory stays tied to the pretrained computation.

Hyperparameter selection on a reference task



Gemma 2–2B: accuracy gain under uniform interpolation.

Selection protocol

1. Sweep (s, e) on **WinoGrande**.
2. Select one model-specific interval.
3. Zero-shot transfer the configuration and evaluate on other benchmarks.

What the sweep found

Gemma 2–2B peaks at layers **10–13** (+0.78 points). Across models, selected intervals fall around **40–60% depth**.

Models: Gemma 2–2B/9B; Llama 3–8B.

Tasks: WinoGrande, ARC-E/C, GSM8K, HellaSwag, MMLU.

Results: Gains depend on the model family and task

Best regularized method − baseline (percentage points)

	Gemma 2–2B	Gemma 2–9B	Llama 3–8B
WinoGrande	+0.78	+0.08	+0.48
ARC Easy	-0.08	+0.04	-0.08
ARC Challenge	+0.59	+0.34	+0.34
GSM8K	+1.36	+0.06	–
HellaSwag	+0.19	+0.12	+0.17
MMLU	+0.56	+0.25	-0.44

Best of uniform, moving average, and auto-alignment.

Pattern across tasks

Gemma 2: higher in **5/6** tasks for 2B and **6/6** for 9B.

Llama 3: higher in **3/5** reported tasks.

What matters

Performance mostly shows matches or slight gains, especially on the Gemma 2 models.

Results: Control and comparison

Noise control: Gemma 2–2B (GSM8K)

Standard forward	25.02	baseline
Matched noise	24.79	**-0.23**
Regularized loop	26.38	**+1.36**

Noise moves accuracy down; the structured loop moves it up.

Across the full table, the best regularized method also outperforms matched noise in every reported model–task case.

Normalization tracks robustness

- **Gemma 2:** pre- and post-normalization; gains are more consistent.
- **Llama 3:** pre-normalization only; gains vary by task.

Important qualification

This comparison is **suggestive, not causal**: model family, training, and normalization all change together.

Results: Visualization of the latent trajectory

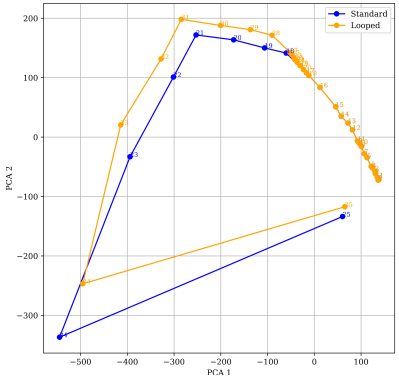


Figure: 2D PCA projection of hidden states for one sample.

What the projection shows

- The looped path deviates during the reused segment.
- The difference persists through later layers.
- The overall trajectory remains close to the baseline path.

Interpretation

A small structured change may be enough to alter final logit margins.

Conclusion: regularization enables extra depth

What the evidence supports

1. Naive layer reuse destabilizes frozen models.
2. Interpolation makes extra test-time depth usable.
3. Gemma shows small gains across most tasks; Llama is less reliable.

What remains open

- Can adaptive loop placement or stopping amplify the gains?
- Which architectural features make looping stable?
- When is another loop worth its added latency and compute?

A frozen Transformer can sometimes benefit from extra internal steps,
but only when the trajectory remains anchored.

Open question: how to scale the loop count?

This work

Anchored updates make a few extra internal steps usable.

Next challenge

Preserve stability as the number of loops R grows.

The broader question

Can loop count become a **reliable test-time compute budget**?

A promising direction

Chen et al. (2026) recast looping as **numerical integration** and use **Runge–Kutta updates** to improve stability, especially at larger loop counts.

Chen et al., *Training-Free Looped Transformers*, arXiv:2605.23872.

Inner Loop Inference for Pretrained Transformers

Questions?

`github.com/jonathanlys01/looped-transformer`

Appendix: complete benchmark table (1/2)

Benchmark	Model	Baseline	Uniform	Mov. avg.	Auto-align	Noise
WinoGrande	Gemma 2–2B	68.75 ± 1.30	$\mathbf{69.53} \pm 1.29$	69.06 ± 1.30	68.98 ± 1.30	68.51 ± 1.31
	Gemma 2–9B	77.35 ± 1.18	$\mathbf{77.43} \pm 1.17$	77.35 ± 1.18	77.35 ± 1.18	77.35 ± 1.18
	Llama 3–8B	76.16 ± 1.20	$\mathbf{76.64} \pm 1.19$	76.40 ± 1.19	76.09 ± 1.20	75.93 ± 1.20
ARC Easy	Gemma 2–2B	$\mathbf{80.30} \pm 0.82$	80.01 ± 0.82	80.18 ± 0.82	80.22 ± 0.82	79.84 ± 0.82
	Gemma 2–9B	87.84 ± 0.67	87.84 ± 0.67	$\mathbf{87.88} \pm 0.67$	87.84 ± 0.67	87.84 ± 0.67
	Llama 3–8B	$\mathbf{77.69} \pm 0.85$	77.31 ± 0.86	77.40 ± 0.86	77.61 ± 0.86	77.57 ± 0.86
ARC Challenge	Gemma 2–2B	52.82 ± 1.46	$\mathbf{53.41} \pm 1.46$	52.90 ± 1.46	52.99 ± 1.46	52.65 ± 1.46
	Gemma 2–9B	67.75 ± 1.37	$\mathbf{68.09} \pm 1.36$	$\mathbf{68.09} \pm 1.36$	67.58 ± 1.37	67.49 ± 1.37
	Llama 3–8B	59.73 ± 1.43	59.64 ± 1.43	59.73 ± 1.43	$\mathbf{60.07} \pm 1.43$	59.64 ± 1.43

Values are accuracies in percent as reported in the paper; bold marks the row maximum.

Appendix: complete benchmark table (2/2)

Benchmark	Model	Baseline	Uniform	Mov. avg.	Auto-align	Noise
GSM8K	Gemma 2–2B	25.02 ± 1.19	26.00 ± 1.21	$\mathbf{26.38} \pm 1.21$	26.08 ± 1.21	24.79 ± 1.19
	Gemma 2–9B	68.46 ± 1.28	$\mathbf{68.52} \pm 1.27$	68.39 ± 1.28	68.31 ± 1.28	68.33 ± 1.28
HellaSwag	Gemma 2–2B	74.51 ± 0.43	$\mathbf{74.70} \pm 0.43$	74.64 ± 0.43	74.60 ± 0.43	74.50 ± 0.43
	Gemma 2–9B	82.41 ± 0.38	$\mathbf{82.53} \pm 0.38$	82.42 ± 0.38	82.42 ± 0.38	82.42 ± 0.38
	Llama 3–8B	82.14 ± 0.38	82.30 ± 0.38	$\mathbf{82.31} \pm 0.38$	$\mathbf{82.31} \pm 0.38$	82.14 ± 0.38
MMLU	Gemma 2–2B	53.93 ± 3.57	$\mathbf{54.49} \pm 3.58$	54.35 ± 3.57	54.13 ± 3.57	53.97 ± 3.58
	Gemma 2–9B	72.17 ± 3.11	$\mathbf{72.42} \pm 3.10$	72.23 ± 3.11	72.18 ± 3.11	72.20 ± 3.11
	Llama 3–8B	$\mathbf{34.50} \pm 3.47$	33.95 ± 3.45	33.99 ± 3.45	34.06 ± 3.45	33.74 ± 3.45

Values are accuracies in percent as reported in the paper; bold marks the row maximum.

Appendix: the three interpolation rules

Uniform

$$\hat{h}^{(t)} = \frac{1}{t+1} \sum_{i=0}^{t} h^{(i)}$$

Equal weight for every cached state.

Moving average

$$\hat{h}^{(t)} = \eta h^{(0)} + (1-\eta) h^{(t)}$$

Direct interpolation with the baseline state.

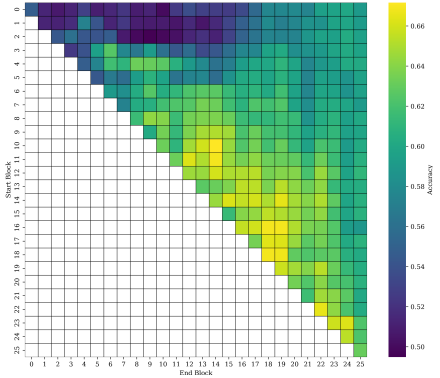
Auto-alignment

$$\alpha_i \propto \exp\Big(\langle h^{(0)}, h^{(i)} \rangle \Big)$$

Data-dependent weights favor baseline-aligned states.

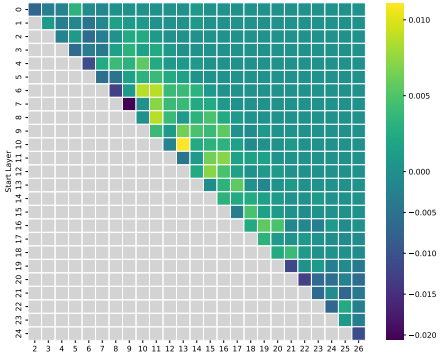
Uniform interpolation is the simplest and most reliable overall.

Appendix: naive and regularized layer sweeps



Naive, $R = 3$

Every interval is below baseline.



Uniform interpolation

Middle-layer intervals recover and sometimes improve accuracy.