Test-Time Training with KV Binding Is Secretly Linear Attention

Junchen Liu[1,2,3*], Sven Elflein[1,2,3], Or Litany[1,4], Zan Gojcic[1], Ruilong Li[1*]

[1]NVIDIA [2]University of Toronto [3]Vector Institute [4]Technion *Equal contribution

TTT-KVB

Each sequence modeling layer maintains *fast weights* f_θ (a small MLP) updated during training and inference.

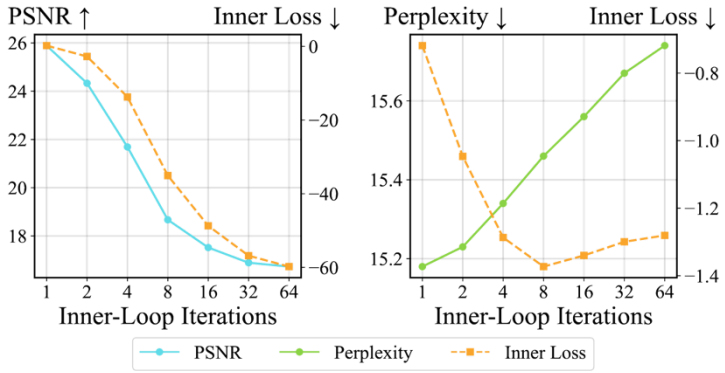
Tokens are projected into keys k, values v, and queries q.

1. **Update:** gradient descent on f_θ with KV loss: $\mathcal{L} = \|f_\theta(k) - v\|^2$
2. **Apply:** pass query q through the updated f_θ: $o = f_\theta(q)$

Commonly read as ***"memorizing"*** KV pairs, an interpretation we challenge.

Observation: Better Inner Loss Can Hurt

More inner iterations improve the inner loss, but degrade task performance.

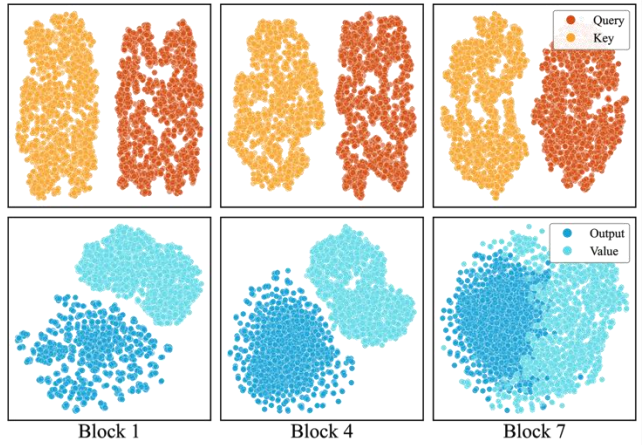


Gradient Ascent & Replace Q with K works!

Model	PPL ↓	PSNR ↑	Top-1 ↑
Baseline	16.43	25.94	79.34
Gradient ascent	16.19	25.85	79.61
Replace Q with K	16.18	25.95	79.18

Q and K Are Misaligned

Queries and keys live in different distributions, so the inner-loop function is not acting like a normal retrieval table.



TTT as Linear Attention

Consider a TTT model whose inner-loop function has a linear, bias-free final layer:

$$f(x) = \phi(x;\Theta)W$$

After one gradient descent step on an objective $\mathcal{L}$ with learning rate η, updating all trainable parameters:

$$(W_{t+1}, \Theta_{t+1}) = (W_t, \Theta_t) - \eta \nabla_{(W_t,\Theta_t)} \mathcal{L}(f_t(k))$$

The output for any query q can be written as:

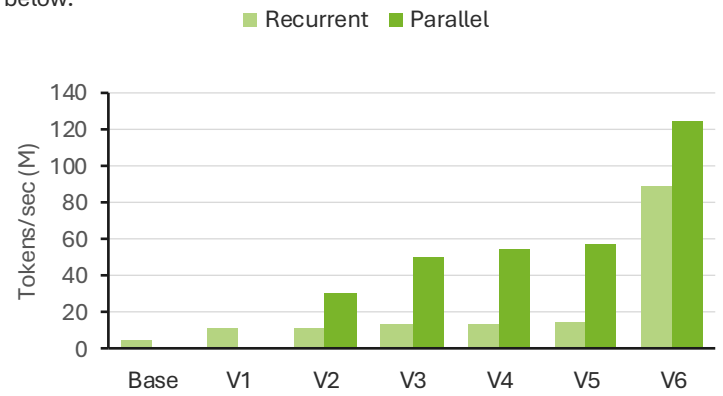
$$o = \phi_{t+1}(q)\left(W_t + \phi_t(k)^\top g_t(k)\right), \quad g_t(k) \triangleq -\eta \frac{\partial \mathcal{L}}{\partial f_t(k)}$$

This is the linear attention form, where:

$$\hat{q} = \phi_{t+1}(q), \quad \hat{k} = \phi_t(k), \quad \hat{v} = g_t(k), \quad S_0 = W_t$$

Practical: Simplify, Parallelize, Generalize

Recognizing TTT as linear attention is not merely theoretical — it yields concrete practical benefits. This perspective reveals that many commonly adopted design choices (per-token learning rates, weight normalization, deeper MLPs) are not essential, enabling substantial simplification. Moreover, it makes explicit that the seemingly recurrent inner-loop updates admit a fully parallel formulation, leading to significant efficiency gains. We progressively ablate each component below:



Variant	Design step	PPL ↓	TPS ↑
Base	LaCT / ViTTT	16.43	4.30M
V1	update only last layer	15.93	10.60M
V2	remove weight norm	16.31	30.18M
V3	MLP → single linear layer	16.23	49.69M
V4	Remove per-token LR	16.12	53.99M
V5	Remove momentum	15.97	57.28M
V6	Remove Muon orthog	16.80	124.6M