Test-Time Generation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Test-time training methods compress a growing context into fast weights, but most read the state through deterministic regression: a query is mapped to a single key-conditioned value in one or a few steps. This endpoint view averages multimodal bindings, entangles reusable value geometry with instance-specific variation, and weakens retrieval when nearby keys should recover distant values. We propose Test-Time Generation (**TTG**), a generative fast-weight framework that treats memory readout as key-conditioned transport from latent noise to target values. Our analysis casts fast weights as conditional associative memory and shows that rectified-flow denoising augments key-space similarity with a value-space posterior term whose retrieval margin grows along the trajectory. **TTG** implements this posterior with a fast-weight denoiser: updates supervise noisy versions of observed bindings, while reads integrate for a controllable number of steps. In novel view synthesis and long-context language modeling, **TTG** improves over LaCT-style deterministic readout and exposes accuracy–compute tradeoffs through stochastic update repeats and variable-depth generation.

1 Introduction

Biological memory is constructive, cue-conditioned, and plastic rather than archival. Episodic recall recombines partial traces, uses pattern separation and completion to recover structure from partial cues, and can be updated after retrieval [13, 19, 28, 31, 34, 35, 48]. This suggests a computational principle for machine memory: recall need not be a one-shot lookup. A cue can initialize a trajectory that first recovers shared structure and then progressively resolves instance-specific uncertainty.

Long-context models face the same tension in a compressed form. Recurrent state-space, fast-weight, and test-time training (TTT) methods write a growing stream into a fixed-size online state [3–5, 14, 18, 22, 23, 27, 36, 40–42, 46, 47, 49, 50]. This gives linear-time memory, but it also forces many rare key-value bindings to remain retrievable through a small recurrent state. Existing TTT memories usually update fast weights by deterministic target fitting: given a key, regress the target value in one or a few steps, then read the state through that endpoint map [3–5, 14, 22, 42, 49, 50]. Under squared loss, multimodal bindings collapse to the conditional mean; under finite capacity, shared value geometry, key routing, and residual variation compete inside the same endpoint map [11, 16, 17, 25, 51].

We instead treat fast-weight memory as conditional generation: learn the distribution $p(v \mid k, h)$ induced by the recurrent state, not only its mean. The state h supplies a key-conditioned binding prior, while latent noise carries residual value-space uncertainty that deterministic regression would average away [7, 38]. Figure 1 illustrates this shift from endpoint prediction to cue-conditioned sampling.

The next sections make this construction explicit. We first view a fast-weight state as a conditional associative memory over effective bindings. Under a Gaussian bridge from noise to values, Bayes' rule yields a posterior whose denoiser is the conditional mean and whose retrieval margin decomposes into a key-prior term plus a value-space denoising term. This analysis explains why iterative recall

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

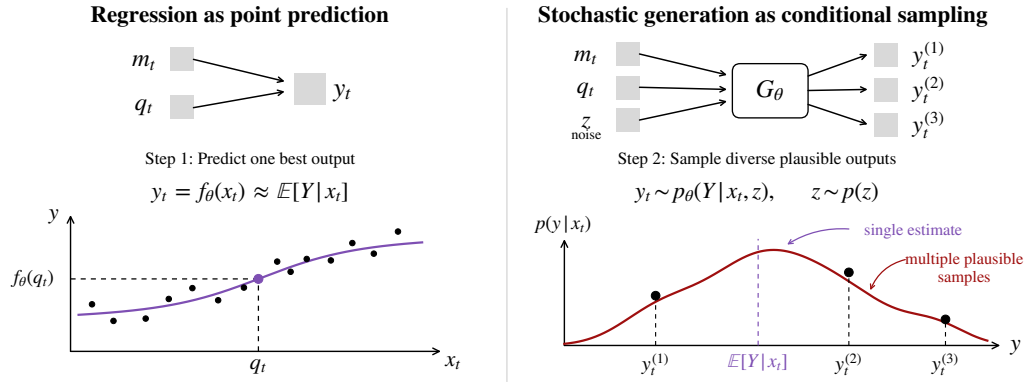


Figure 1: Regression versus conditional sampling for test-time memory. Given recurrent state h and query key k, deterministic fast-weight regression returns one value $f_\theta(k; h) \approx \mathbb{E}[V \mid k, h]$, so multimodal or residual uncertainty in the bindings is averaged into a single point estimate. Conditional generation augments the cue with latent noise ε and reads through $G_\theta(k, h, \varepsilon)$, modeling a distribution over plausible values and letting recall refine ambiguous bindings through a trajectory.

helps: early noisy states can share vector-field structure across bindings, while later key-conditioned steps separate individual memories as trajectory signal-to-noise increases [2, 6, 8, 9, 21, 44, 45].

Building on this posterior view, we propose **TTG**, a generative fast-weight framework for key-value binding. **TTG** instantiates the bridge with flow matching [1, 20, 30, 37, 39]: observed bindings supervise noisy value states during the online update, and read queries integrate a key-modulated denoising trajectory at test time. The same formulation gives two practical compute controls, stochastic repeats during update and integration depth during readout. We evaluate **TTG** in novel view synthesis and long-context language modeling, compare against deterministic fast-weight baselines, and study the objective, noise schedule, adaptive conditioning, and readout depth that determine recurrent memory quality.

2 Related Work

2.1 Test-Time Training

Long-context models trade explicit retrieval for recurrent compression: linear-attention, fast-weight, and state-space methods store history in compact states rather than full key-value caches [18, 25, 27, 36, 46, 47]. Test-time training further adapts hidden states, fast weights, or auxiliary parameters during inference [3–5, 14, 22, 40, 42, 50]. Our work keeps this compressed-memory setting, but replaces deterministic endpoint readout with generative recall conditioned on a key or query.

2.2 Generative Models

Generative models represent uncertainty with latent variables or iterative denoising and transport processes, including VAEs, diffusion models, score models, flow matching, and stochastic interpolants [1, 20, 26, 30, 33, 37, 39, 44]. Recent work connects these dynamics to associative memory: Hopfield retrieval, key-value attention, and denoising or flow trajectories can all be viewed as content-based recall [2, 6, 16, 21, 32, 45]. We transfer this view from data-space generation to fast-weight memory, where sampling becomes iterative test-time recall.

3 Preliminary

The introduction motivates replacing one-shot fast-weight regression with a generative readout. This section gives the common notation used by **TTG**: a recurrent state induces a key-conditioned prior over bindings, and a denoising trajectory turns that prior into posterior recall.

3.1 Fast Weights as Conditional Associative Memory

Let h be the recurrent fast-weight state after a prefix, abstracted as effective key-value bindings

$$\mathcal{D}_h = \{(k_i, v_i)\}_{i=1}^N, \qquad k_i \in \mathbb{R}^{d_k}, \quad v_i \in \mathbb{R}^{d_v}.$$

For a query key k, the state induces a binding prior and empirical conditional value distribution

$$\pi_i(k;h) := P(i \mid k, h), \qquad \sum_{i=1}^N \pi_i(k;h) = 1, \qquad \hat{p}(v \mid k, h) = \sum_{i=1}^N \pi_i(k;h)\, \delta(v - v_i).$$

Here $\pi_i(k;h)$ is the key-space evidence available before generation, while $\hat{p}(v \mid k, h)$ is the value distribution that a deterministic readout would collapse. This notation covers transformer key-value memories, modern Hopfield retrieval, and diffusion-style associative memory under the same conditional-memory view [2, 16, 21, 32].

3.2 Gaussian Bridge and Posterior Retrieval

To make the empirical conditional distribution readable by a generative model, bridge latent noise and values by

$$x_t = \alpha_t \varepsilon + \beta_t v, \qquad \varepsilon \sim \mathcal{N}(0, I), \qquad t \in [0, 1],$$

where $\alpha_0 = 1$, $\beta_0 = 0$, $\alpha_1 = 0$, $\beta_1 = 1$; linear flow matching uses $\alpha_t = 1 - t$ and $\beta_t = t$. Conditioned on binding i,

$$p_t(x \mid i) = \mathcal{N}(x; \beta_t v_i, \alpha_t^2 I),$$

so Bayes' rule gives

$$w_i(x, t, k; h) = \frac{\pi_i(k;h)\, \mathcal{N}(x; \beta_t v_i, \alpha_t^2 I)}{\sum_{j=1}^N \pi_j(k;h)\, \mathcal{N}(x; \beta_t v_j, \alpha_t^2 I)}. \tag{1}$$

$$m_t(x, k; h) := \mathbb{E}[V \mid x_t = x, k, h] = \sum_{i=1}^N w_i(x, t, k; h)\, v_i. \tag{2}$$

Proposition 3.1 (Bayes-optimal denoiser and velocity)**.** *Under* (1)*, the Bayes-optimal denoiser is*

$$D_t^\star(x, k; h) = m_t(x, k; h).$$

For the linear bridge $x_t = (1 - t)\varepsilon + tv$, the velocity is

$$u_t^\star(x, k; h) = \frac{m_t(x, k; h) - x}{1 - t}. \tag{3}$$

Proof. The minimizer of $\mathbb{E}[\|D(x, k) - V\|_2^2 \mid x, k, h]$ is the conditional mean $\mathbb{E}[V \mid x, k, h]$, which is (2) under the discrete posterior weights in (1). For the linear bridge, $x_t = (1 - t)\varepsilon + tv$ has constant target velocity $v - \varepsilon$. Since $\varepsilon = (x_t - tv)/(1 - t)$, the conditional velocity target is $\mathbb{E}[(V - \varepsilon) \mid x_t = x, k, h] = (m_t(x, k; h) - x)/(1 - t)$. □

Denoising and flow matching therefore retrieve a posterior mean whose weights depend on both the key and the current latent state. This is the mathematical object **TTG** will parameterize with fast weights, and it matches recent associative-memory and closed-form FM analyses [2, 6, 21].

3.3 Why Flow Matching Instead of Direct Regression

Theorem 3.2 (Deterministic regression collapses multimodal bindings)**.** *For squared loss on (K, V), the Bayes predictor is $f^\star(k) = \mathbb{E}[V \mid K = k]$ and the optimal risk is $\mathbb{E}[\mathrm{Var}(V \mid K)]$. Thus a deterministic one-shot regressor averages non-degenerate bindings instead of modeling their conditional distribution.*

Proof. Condition on $K = k$ and expand $\mathbb{E}[\|f(k) - V\|_2^2 \mid K = k] = \|f(k) - \mathbb{E}[V \mid K = k]\|_2^2 + \mathrm{tr}\,\mathrm{Var}(V \mid K = k)$. The first term is minimized by the conditional mean, leaving the conditional variance term. Averaging over K gives the stated risk and shows that any non-degenerate conditional distribution is collapsed to one endpoint. □

This is the endpoint failure described in the introduction: direct regression spends fast-weight capacity on a single conditional mean. Modeling the conditional distribution avoids that collapse [7, 38], and flow matching adds shared-to-specific dynamics:

Proposition 3.3 (Shared-to-specific retrieval dynamics)**.** *If $\beta_t\|v_i - v_j\| \ll \alpha_t$ for many pairs (i, j), the Gaussian factors in* (1) *overlap and*

$$m_t(x, k; h) \approx \sum_{i=1}^{N} \pi_i(k; h) v_i.$$

As α_t/β_t decreases, the posterior sharpens toward memory-specific recall, consistent with FM mean-to-cluster-to-point phases [45].

Proof. When $\beta_t\|v_i - v_j\| \ll \alpha_t$, the likelihood ratio between any two nearby Gaussian factors in (1) is close to one over the relevant latent region, so the normalized weights are dominated by the prior $\pi_i(k; h)$. As α_t/β_t decreases, the exponent in the Gaussian likelihood increasingly penalizes values whose $\beta_t v_j$ is far from the current latent x, sharpening the posterior toward value-specific bindings. □

Proposition 3.4 (Retrieval margin and denoising SNR)**.** *For two bindings i and j, define*

$$\Delta_{ij}(x, t, k; h) := \log \frac{w_i(x, t, k; h)}{w_j(x, t, k; h)}.$$

Then

$$\Delta_{ij}(x, t, k; h) = \log \frac{\pi_i(k; h)}{\pi_j(k; h)} - \frac{\|x - \beta_t v_i\|^2 - \|x - \beta_t v_j\|^2}{2\alpha_t^2}. \tag{4}$$

If $x_t = \beta_t v_i + \alpha_t \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, I)$, then

$$\mathbb{E}[\Delta_{ij}(x_t, t, k; h) \mid i, k, h] = \log \frac{\pi_i(k; h)}{\pi_j(k; h)} \; + \; \frac{\beta_t^2}{2\alpha_t^2}\|v_i - v_j\|^2.$$

Proof. Taking the log ratio of (1) cancels the shared normalization and the Gaussian constants, giving (4). If $x_t = \beta_t v_i + \alpha_t \varepsilon$, then $\|x_t - \beta_t v_i\|^2 = \alpha_t^2\|\varepsilon\|^2$ and $\|x_t - \beta_t v_j\|^2 = \|\beta_t(v_i - v_j) + \alpha_t\varepsilon\|^2$. The cross term has zero expectation, yielding the stated expected margin. □

The margin separates key-prior evidence from a value-space denoising term amplified by β_t^2/α_t^2 [6, 8, 9, 44]. Thus sharper key conditioning raises the prior margin, while flow integration adds value-space separation along the trajectory. **TTG** uses the first term through query-conditioned fast weights and the second term through iterative denoising readout.

4 TTG

The preliminary section defines the posterior that a generative memory should read from a fast-weight state. **TTG** turns that posterior into an online transformer block: the state h_b is stored in fast MLP weights, observed bindings supervise denoising along the bridge, and read queries run a short generation trajectory through the current state.

4.1 Generative Fast-Weight Readout

Section 3.1 views h as memory $\mathcal{D}_h = \{(k_i, v_i)\}_{i=1}^N$. In **TTG**, h is not materialized as an explicit table; it modulates a fast MLP that approximates the posterior denoiser in Proposition 3.1. A read query supplies the key k, and latent noise supplies the unresolved value-space component. Instead of a deterministic endpoint map $f(k; h)$, **TTG** learns a stochastic denoising readout $G_{\theta,S}(k, h, \varepsilon)$. With $\rho = 1 - t$, the bridge becomes $x_\rho = \rho\varepsilon + (1 - \rho)v$. Let

$$D_\theta(x, \rho, k; h) : \mathbb{R}^{d_v} \times [0, 1] \times \mathbb{R}^{d_k} \to \mathbb{R}^{d_v}$$

denote the fast-weight denoiser, which takes the current latent, noise level, read key, and recurrent state.

At test time, readout starts from normalized Gaussian noise and repeatedly denoises:

$$x_{\rho_0} = \varepsilon, \qquad \rho_0 = 1 > \rho_1 > \cdots > \rho_S = 0, \qquad \hat{v} = x_{\rho_S}. \tag{5}$$

At step s, the fast-weight MLP predicts $\tilde{v}_s = D_\theta(x_{\rho_s}, \rho_s, k; h)$ and updates

$$x_{\rho_{s+1}} = (1 - \lambda_s) x_{\rho_s} + \lambda_s \tilde{v}_s, \qquad \lambda_s = \frac{\rho_s - \rho_{s+1}}{\rho_s}, \tag{6}$$

with $\lambda_s = 1$ for the final step. This update is the operational readout used by the applications below: early steps share coarse value structure, and later steps let the key-conditioned posterior sharpen. Under the linear bridge with $t = 1 - \rho$, it follows

$$u_\theta(x, t, k; h) = \frac{D_\theta(x, 1 - t, k; h) - x}{1 - t},$$

whose Bayes target is (3). The number of steps S is therefore a test-time compute knob rather than a separate training objective.

4.2 Trajectory-Level Test-Time Objective

The write objective uses the same bridge as the readout, so every observed binding trains the posterior that will later be queried. For an observed binding (k, v), sample ρ and $\varepsilon \sim \mathcal{N}(0, I)$, then form

$$x_\rho = \rho\varepsilon + (1 - \rho)v.$$

The clean-value denoising loss is

$$\mathcal{L}_{\text{denoise}}(\theta) = \mathbb{E}_{(h,k,v),\rho,\varepsilon}\left[\|D_\theta(x_\rho, \rho, k; h) - v\|_2^2\right]. \tag{7}$$

For $\rho > 0$, this also supervises

$$u_\theta(x, \rho, k; h) = \frac{D_\theta(x, \rho, k; h) - x}{\rho}.$$

Training uses continuous ρ or schedule-matched discrete levels, often with multiple noise draws per binding. Unlike deterministic TTT regression, these repeats expose the same value through different corrupted states, giving trajectory-level supervision while keeping readout in denoiser form. After $t = 1 - \rho$, the target matches Proposition 3.1.

4.3 Online State Updates and Cached Inference

The abstract state h becomes an online sequence state h_b. The model processes chunks C_b with local window attention, reads the prefix state by (5)–(6), and applies a shared slow MLP. The resulting features provide write bindings $\{(k_r, v_r)\}_{r \in C_b}$ for the next state.

After an update chunk, **TTG** writes the state by one learned TTT step:

$$h_{b+1} = U_\phi(h_b, \{(k_r, v_r, \eta_r)\}_{r \in C_b}), \tag{8}$$

where η_r are tokenwise learning rates. Each memory block stores gated-MLP fast weights; denoising residuals are backpropagated through them, scaled by η_r, accumulated across tokens and noise repeats, and applied to the master weights. Normalized low-precision copies are cached for reads.

Thus slow denoiser parameters are shared across time, while h_b carries recurrent memory across chunks. Caching does not change the conditional memory $\hat{p}(v \mid k, h_b)$ from Section 3.1; it only amortizes posterior-mean readout and boundary updates.

4.4 Applications to LVSM

For LVSM-style novel view synthesis [24], posed images and target poses are patchified into tokens, as instantiated in Figure 2. Observed tokens write a scene memory $\mathcal{D}_{h_b} = \{(k_r^{\text{upd}}, v_r)\}_{r \in C_b}$, where k_r^{upd} encodes pose, patch position, and local context, and v_r is the reconstruction latent or residual feature.

The update path samples $x_{\rho,r} = \rho\varepsilon_r + (1 - \rho)v_r$, applies the denoising loss in (7), and writes h_{b+1} with (8); the read path queries this state with target-camera tokens. For read key k_q^{read}, (1)–(2) define the posterior over observed bindings, while (5)–(6) implement $\hat{v}_q = G_{\theta,S}(k_q^{\text{read}}, h_{b+1}, \varepsilon_q)$ with target (3). Thus **TTG** keeps the feed-forward posed-token interface of transformer NVS [24, 50], but replaces deterministic apply $f_W(q)$ with posterior generative readout; noise repeats strengthen the scene update, and readout steps refine target features with fixed fast weights.

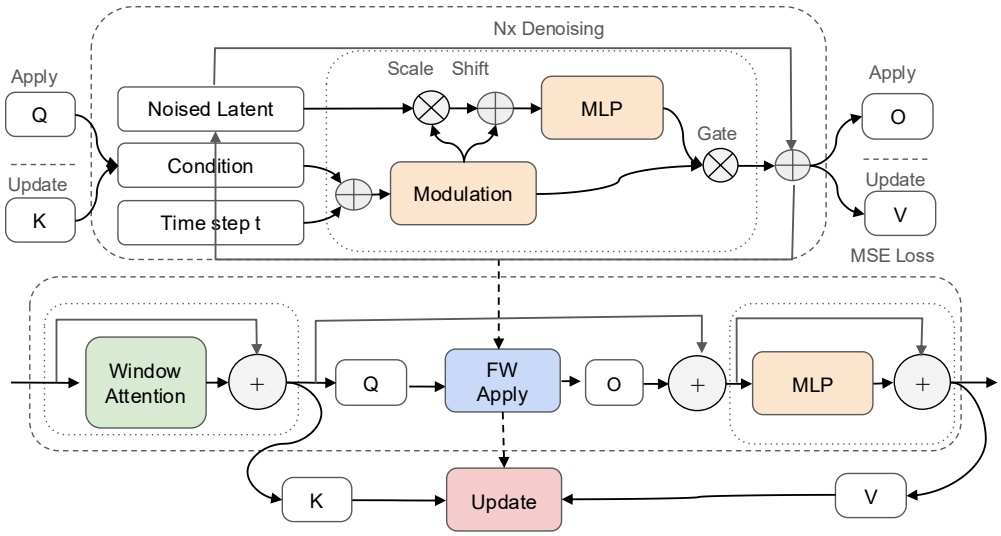


Figure 2: LVSM instantiation of **TTG** used in Section 4.4. Local window attention forms target-camera read queries, while observed posed-image tokens provide update keys and value targets that supervise the fast-weight state through an MSE denoising loss. At apply time, the generative readout refines a noised latent for S steps using key, time, and modulation signals before returning the target-view value.

4.5 Applications to Language Modeling

Autoregressive language models predict $p_\theta(x_n \mid x_1, \ldots, x_{n-1})$. Since text has no natural chunk, **TTG** treats chunk size as a hyperparameter and follows LaCT's shifted apply-before-update order: Figure 3 uses sliding-window attention inside C_b and fast weights for compressed evidence from earlier chunks.

Before writing $C_b = \{n_b, \ldots, n_b + B - 1\}$, h_b stores previous bindings $\mathcal{D}_{h_b} = \{(k_i^{\mathrm{upd}}, v_i)\}_{i<n_b}$; RMS-normalized hidden states produce RoPE-augmented read keys, update keys, and value targets. Apply reads only the prefix state, $\hat{v}_n = G_{\theta,S}(k_n^{\mathrm{read}}, h_b, \varepsilon_n)$ for $n \in C_b$, so no future chunk token enters the fast-weight path; (1)–(2) give the posterior read and (5)–(6) the denoising trajectory. After next-token prediction, examples $x_{\rho,n} = \rho\varepsilon_n + (1-\rho)v_n$ train the update in (7) and (8), preserving LaCT's recurrent TTT interface for long-context language models [22, 41, 42] while replacing one-step recall $f_W(q)$ with variable-depth generative readout.

5 Experiments

5.1 Setup

Novel View Synthesis Setup We compare LaCT and **TTG** on incremental novel view synthesis (NVS), where posed context views must support reconstruction from new camera poses, stressing recurrent retrieval over dense visual tokens. Object-level models train on Objaverse [10] and evaluate on the GSO Instant3D split [12]; scene-level models train on filtered DL3DV-10K [29] and evaluate on its benchmark split. We train with 12 views, evaluate with 24 views, and report PSNR. Each 256^2 image is tokenized into 8×8 patches (1,024 tokens per view); alternating posed-image context and pose-only target tokens yields 22,528 training tokens and 47,104 evaluation tokens. Both methods use the same 8-layer, width-512 backbone and 20K-step AdamW setup (learning rate 4×10^{-4}, weight decay 0.05, 2.5K warmup, gradient clipping at 1.0, L2 loss) on 8 A100 80G GPUs with global batch size 128.

Language Modeling Setup Language modeling predicts the next-token distribution from the preceding context, so long sequences directly test whether a recurrent state can preserve token-level evidence across many chunks. We compare LaCT and **TTG** on Long-Data-Collections [43], training

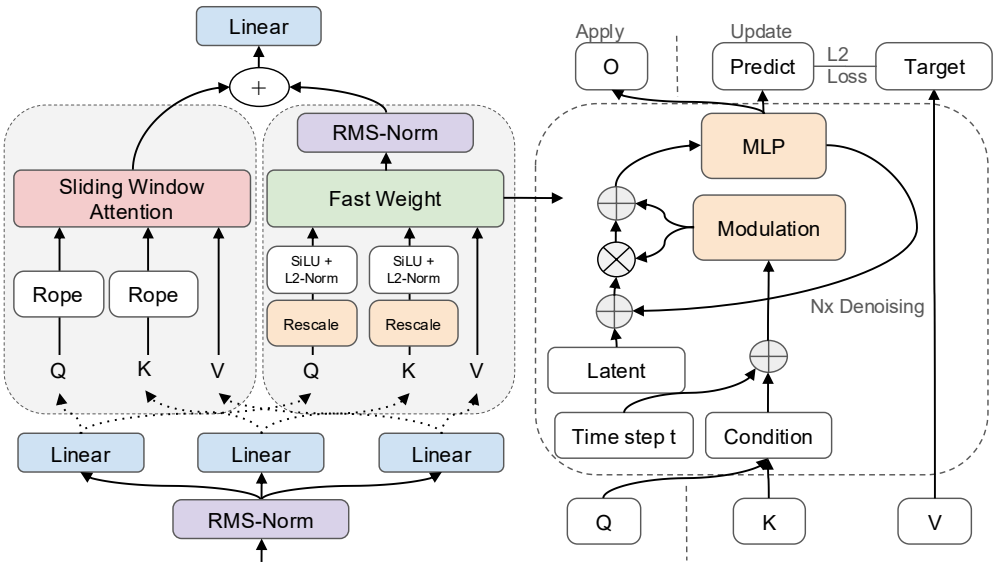
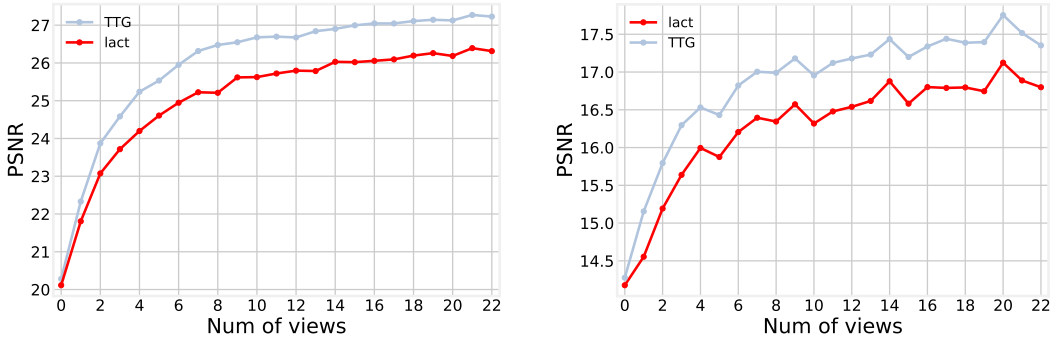


Figure 3: Language-model instantiation of **TTG** used in Section 4.5. Sliding-window attention handles causal local context inside the current chunk, while the fast-weight branch stores compressed evidence from previous chunks. Apply reads the prefix state through an S-step denoising module conditioned on the read key and noise level; after prediction, update keys and value targets train the next state with an L2 denoising loss.



(a) Main NVS benchmark.

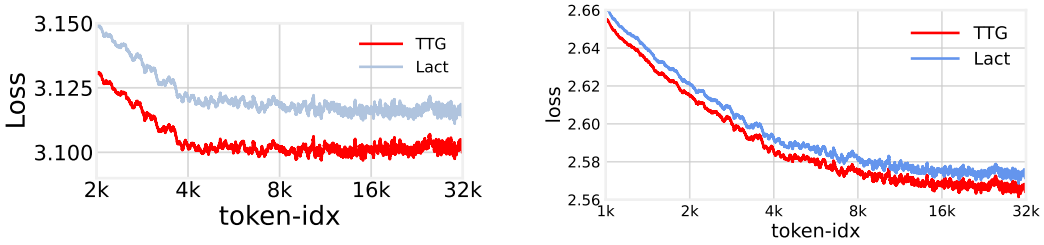
(b) DL3DV.

Figure 4: Novel view synthesis results as the number of target views increases. We compare LaCT and **TTG** on the main NVS benchmark and DL3DV; the y-axis reports final evaluation PSNR.

on up to 60B tokens from the 68.8B-token corpus with the Mistral tokenizer. To preserve parallelism while exposing the models to long histories, both methods process 4K-token chunks and use gated attention for recurrent-state readout. We evaluate three model scales: 124M parameters trained on 10B tokens, and 760M parameters trained on 40B tokens. All language models use RoPE base 1M; the three scales use global batch sizes of 32, and 32 on 8, and 32 A100 80G GPUs, respectively. We report Book-3 validation loss [15] on 32K-token sequences averaged over 10,240 evaluation steps.

5.2 Main Results

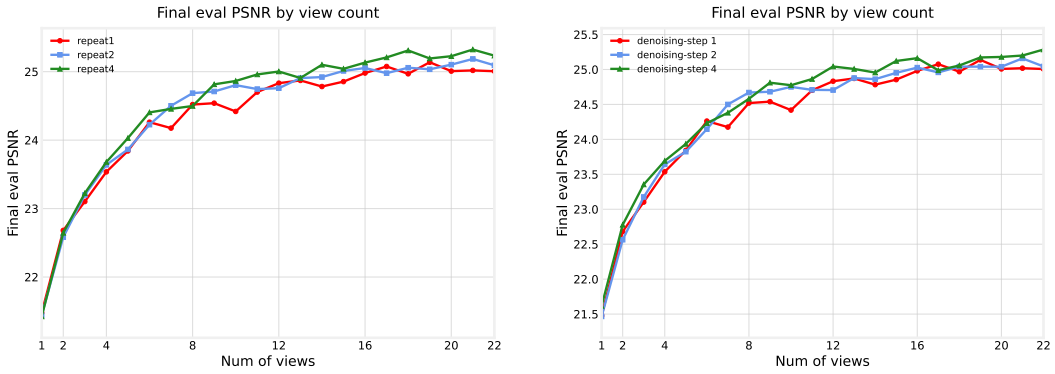
Novel View Synthesis Figure 4 tests the retrieval failure mode from Section 3.3: as more target views are queried, the recurrent state must keep many camera-conditioned bindings separable rather than average them into an endpoint map. On the object-level benchmark, **TTG** improves average PSNR from 25.09 to 26.00 and improves the last evaluated target view from 26.31 to 27.23. On DL3DV, the gain is larger: average PSNR increases from 16.27 to 17.46 and the final target-view



(a) Per-token loss comparison.

(b) Log-scaled token positions.

Figure 5: Per-token validation loss in long-context language modeling. This setting tests whether a fixed recurrent fast-weight state can keep token-level key-value evidence sharply retrievable as the context grows. Across both evaluation views, **TTG** maintains lower loss than LaCT through the shown context positions up to 32K tokens, consistent with cue-conditioned generative retrieval preserving compressed evidence more effectively than deterministic target fitting; lower is better.



(a) Repeat scaling.

(b) Denoising-step scaling.

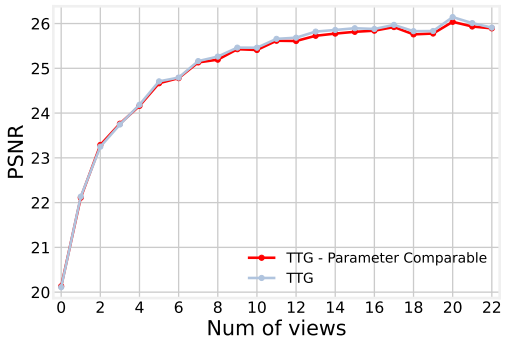
Figure 6: Test-time scaling for diffusion-based **TTG** on novel view synthesis. Here r denotes the number of stochastic repeats per observed binding and d denotes the number of denoising steps used during readout; the y-axis reports final evaluation PSNR as the number of target views increases.

PSNR from 16.80 to 18.64. The widening gap on the scene-level benchmark is consistent with the margin view in (4): when pose keys alone are not enough to disambiguate texture and occlusion, value-space denoising contributes additional separation during readout.

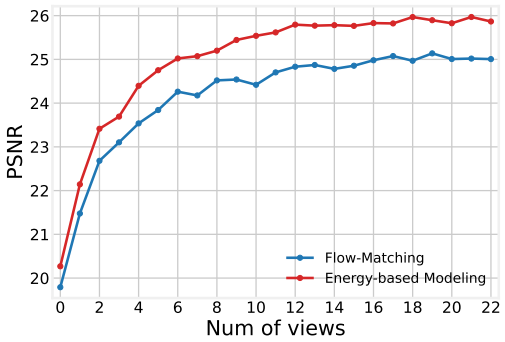
Language Modeling Figure 5 evaluates the same endpoint-versus-trajectory question in text. On Book-3 validation, **TTG** keeps a lower per-token loss across the full plotted context range. The absolute loss gap is small, but it is persistent across token positions, which is the expected signature for a recurrent memory improvement rather than a short-context modeling change. Together with the NVS results, these language-modeling results support the paper's central claim: generative readout helps the compressed state remain useful when one-shot deterministic recall would blur many bindings together.

5.3 Test-Time Scaling

TTG exposes two test-time compute knobs in novel view synthesis: stochastic update repeats r, which revisit observed bindings with fresh noise, and readout depth d, which runs more denoising steps for each query. Figure 6 shows that both knobs improve the PSNR envelope: increasing r strengthens the adapted fast-weight memory, while increasing d refines target-view features with the updated memory fixed.



(a) Parameter-comparable comparison.



(b) Generative objective design choice.

Figure 7: Novel view synthesis comparisons for strict fairness and generative modeling design. The left panel reports the Section 5.4 parameter-comparable comparison, and the right panel reports the Section 5.5 flow-matching versus energy-based modeling comparison. Both plot final evaluation PSNR as the number of target views increases.

5.4 Strict Fair Comparisons

Parameter Comparable Comparison We next separate the effect of generative readout from parameter allocation inside the memory block. Figure 7(a) compares the default **TTG** model with a parameter-comparable variant. Both models use the same data, backbone, training schedule, and evaluation protocol, so this control targets memory-block allocation rather than optimization or dataset differences. The curves nearly overlap across target views: average PSNR changes from 24.92 to 24.95 and final target-view PSNR changes from 25.86 to 25.89. This suggests that the gains in Figure 4 are not explained by a larger or easier memory parameterization, but by the trajectory-level read and write objective.

5.5 Design Choice on Generative Models

The framework is not tied to a single generative objective. The analysis and default implementation use flow matching because it gives the closed-form bridge and velocity target in Proposition 3.1. Figure 7(b) compares this choice with an energy-based modeling variant under the same NVS protocol. Energy-based modeling gives higher PSNR in this configuration, improving average PSNR from 24.16 to 24.99 and final target-view PSNR from 25.01 to 25.87. We interpret this as evidence that the key idea is the conditional generative memory interface, while the best objective can be task dependent. The important requirement is that the fast-weight state support a conditional distribution over values, rather than a single deterministic point estimate. Flow matching remains attractive for its direct trajectory supervision and compute-controlled readout; energy-based objectives may provide a stronger inductive bias for visual reconstruction.

6 Conclusion

This paper reframes test-time memory readout as conditional generation rather than deterministic endpoint regression. Starting from a fast-weight state interpreted as a conditional associative memory, we showed that denoising retrieval decomposes into a key-prior term and a value-space posterior term whose margin grows along the trajectory. **TTG** implements this idea by writing noisy versions of observed bindings into fast weights and reading queries through a short denoising process. Across novel view synthesis and long-context language modeling, this trajectory-level interface improves over LaCT-style recall and exposes useful compute controls through stochastic update repeats and readout depth. The main message is the same one introduced at the start: compressed memory need not recall by averaging to a single endpoint; it can use the cue to start a generative trajectory that progressively resolves the binding.

References

[1] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic Interpolants: A Unifying Framework for Flows and Diffusions, 2023. URL https://arxiv.org/abs/2303.08797. _eprint: 2303.08797.

[2] Luca Ambrogioni. In search of dispersed memories: Generative diffusion models are associative memory networks, November 2023. URL http://arxiv.org/abs/2309.17290. arXiv:2309.17290 [stat].

[3] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to Memorize at Test Time, December 2024. URL http://arxiv.org/abs/2501.00663. arXiv:2501.00663 [cs].

[4] Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. ATLAS: Learning to Optimally Memorize the Context at Test Time, May 2025. URL http://arxiv.org/abs/2505.23735. arXiv:2505.23735 [cs].

[5] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested Learning: The Illusion of Deep Learning Architectures, December 2025. URL http://arxiv.org/abs/2512.24695. arXiv:2512.24695 [cs].

[6] Quentin Bertrand, Anne Gagneux, Mathurin Massias, and Rémi Emonet. On the Closed-Form of Flow Matching: Generalization Does Not Arise from Target Stochasticity, 2025. URL https://arxiv.org/abs/2506.03719. _eprint: 2506.03719.

[7] Christopher M. Bishop. Mixture density networks. Technical Report 4288, Aston University, 1994. URL https://research.aston.ac.uk/en/publications/mixture-density-networks.

[8] Sergio Calvo-Ordonez, Matthieu Meunier, Alvaro Cartea, Christoph Reisinger, Yarin Gal, and Jose Miguel Hernandez-Lobato. Weighted Conditional Flow Matching, 2025. URL https://arxiv.org/abs/2507.22270. _eprint: 2507.22270.

[9] Ho Kei Cheng and Alexander Schwing. The Curse of Conditions: Analyzing and Improving Optimal Transport for Conditional Flow-Based Generation, 2025. URL https://arxiv.org/abs/2503.10636. _eprint: 2503.10636.

[10] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. doi: 10.1109/CVPR52729.2023.01263. URL https://arxiv.org/abs/2212.08051.

[11] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is Not All You Need: Pure Attention Loses Rank Doubly Exponentially with Depth, March 2021. URL https://arxiv.org/abs/2103.03404. arXiv:2103.03404.

[12] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2022. doi: 10.1109/ICRA46639.2022.9811809. URL https://arxiv.org/abs/2204.11918.

[13] Yadin Dudai. The Restless Engram: Consolidations Never End. *Annual Review of Neuroscience*, 35:227–247, 2012. doi: 10.1146/annurev-neuro-062111-150500. URL https://www.annualreviews.org/doi/10.1146/annurev-neuro-062111-150500.

[14] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A. Efros. Test-Time Training with Masked Autoencoders, September 2022. URL http://arxiv.org/abs/2209.07522. arXiv:2209.07522 [cs].

[15] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020. URL https://arxiv.org/abs/2101.00027. arXiv:2101.00027 [cs.CL].

[16] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021. URL https://aclanthology.org/2021.emnlp-main.446/.

[17] Vincent Gripon and Claude Berrou. A Massively Parallel Associative Memory Based on Sparse Neural Networks, 2013. URL https://arxiv.org/abs/1303.7032. arXiv:1303.7032.

[18] Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, May 2024. URL http://arxiv.org/abs/2312.00752. arXiv:2312.00752 [cs].

[19] Demis Hassabis, Dharshan Kumaran, and Eleanor A. Maguire. Using Imagination to Understand the Neural Basis of Episodic Memory. *Journal of Neuroscience*, 27(52):14365–14374, 2007. doi: 10.1523/JNEUROSCI.4549-07.2007. URL https://www.jneurosci.org/content/27/52/14365.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, December 2020. URL http://arxiv.org/abs/2006.11239. arXiv:2006.11239 [cs].

[21] Benjamin Hoover, Hendrik Strobelt, Dmitry Krotov, Judy Hoffman, Zsolt Kira, and Duen Horng Chau. Memory in Plain Sight: Surveying the Uncanny Resemblances of Associative Memories and Diffusion Models, 2023. URL https://arxiv.org/abs/2309.16750. _eprint: 2309.16750.

[22] Jinwu Hu, Zhitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. Test-Time Learning for Large Language Models, May 2025. URL http://arxiv.org/abs/2505.20633. arXiv:2505.20633 [cs].

[23] Kazuki Irie and Samuel J. Gershman. Fast weight programming and linear transformers: from machine learning to neurobiology, January 2026. URL http://arxiv.org/abs/2508.08435. arXiv:2508.08435 [cs].

[24] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. LVSM: A Large View Synthesis Model with Minimal 3D Inductive Bias, April 2025. URL http://arxiv.org/abs/2410.17242. arXiv:2410.17242 [cs].

[25] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

[26] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, December 2022. URL http://arxiv.org/abs/1312.6114. arXiv:1312.6114 [stat].

[27] Aakash Lahoti, Kevin Y. Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved Sequence Modeling using State Space Principles, March 2026. URL http://arxiv.org/abs/2603.15569. arXiv:2603.15569 [cs].

[28] Jonathan L. C. Lee, Karim Nader, and Daniela Schiller. An update on memory reconsolidation updating. *Trends in Cognitive Sciences*, 21(7):531–545, 2017. doi: 10.1016/j.tics.2017.04.006. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC5605913/.

[29] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision, 2023. URL https://arxiv.org/abs/2312.16256. arXiv:2312.16256 [cs.CV].

[30] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling, February 2023. URL http://arxiv.org/abs/2210.02747. arXiv:2210.02747 [cs].

[31] Karim Nader, Glenn E. Schafe, and Joseph E. Le Doux. Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval. *Nature*, 406(6797):722–726, 2000. doi: 10.1038/35021052. URL https://www.nature.com/articles/35021052.

[32] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, David Kreil, Michael K. Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield Networks is All You Need, 2021. URL https://arxiv.org/abs/2008.02217. _eprint: 2008.02217.

[33] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models, May 2014. URL http://arxiv.org/abs/1401.4082. arXiv:1401.4082 [stat].

[34] Daniel L. Schacter, Kenneth A. Norman, and Wilma Koutstaal. The Cognitive Neuroscience of Constructive Memory. *Annual Review of Psychology*, 49:289–318, 1998. doi: 10.1146/annurev.psych.49.1.289. URL https://www.annualreviews.org/doi/10.1146/annurev.psych.49.1.289.

[35] Daniel L. Schacter, Donna Rose Addis, and Randy L. Buckner. Remembering the past to imagine the future: The prospective brain. *Nature Reviews Neuroscience*, 8(9):657–661, 2007. doi: 10.1038/nrn2213. URL https://www.nature.com/articles/nrn2213.

[36] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear Transformers Are Secretly Fast Weight Programmers, June 2021. URL http://arxiv.org/abs/2102.11174. arXiv:2102.11174 [cs].

[37] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, November 2015. URL http://arxiv.org/abs/1503.03585. arXiv:1503.03585 [cs].

[38] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28*, 2015.

[39] Yang Song, Jascha Sohl-Dickstein, Durk P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations, 2020. URL https://arxiv.org/abs/2011.13456. _eprint: 2011.13456.

[40] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts, July 2020. URL http://arxiv.org/abs/1909.13231. arXiv:1909.13231 [cs].

[41] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (Learn at Test Time): RNNs with Expressive Hidden States, August 2025. URL http://arxiv.org/abs/2407.04620. arXiv:2407.04620 [cs].

[42] Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-End Test-Time Training for Long Context, December 2025. URL http://arxiv.org/abs/2512.23675. arXiv:2512.23675 [cs].

[43] Together AI. Long-data-collections. https://huggingface.co/datasets/togethercomputer/Long-Data-Collections, 2024. Hugging Face dataset.

[44] Yonglong Tong, Nitish Mukherjee, Xitong Sun, Ching Chun Park, Yujia Lee, Alexander X. Li, Jason D. Lee, and Kwang-Sung Jun. Conditional Flow Matching: Simulation-Free Dynamic Optimal Transport, 2023. URL https://arxiv.org/abs/2302.00482. _eprint: 2302.00482.

[45] Zhengchao Wan, Qingsong Wang, Gal Mishne, and Yusu Wang. Elucidating Flow Matching ODE Dynamics with Respect to Data Geometries and Denoisers, 2025. URL https://arxiv.org/abs/2412.18730. _eprint: 2412.18730.

[46] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated Linear Attention Transformers with Hardware-Efficient Training, December 2023. URL https://arxiv.org/abs/2312.06635v6.

[47] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated Delta Networks: Improving Mamba2 with Delta Rule, December 2024. URL https://arxiv.org/abs/2412.06464v3.

[48] Michael A. Yassa and Craig E. L. Stark. Pattern Separation in the Hippocampus. *Trends in Neurosciences*, 34(10):515–525, 2011. doi: 10.1016/j.tins.2011.06.006. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC3183227/.

[49] Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to Discover at Test Time, January 2026. URL http://arxiv.org/abs/2601.16175. arXiv:2601.16175 [cs].

[50] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-Time Training Done Right, May 2025. URL http://arxiv.org/abs/2505.23884. arXiv:2505.23884 [cs].

[51] Shu Zhong, Mingyu Xu, Tenglong Ao, and Guang Shi. Understanding Transformer from the Perspective of Associative Memory, May 2025. URL http://arxiv.org/abs/2505.19488. arXiv:2505.19488 [cs].

A Limitations

TTG adds compute to deterministic fast-weight memories. Stochastic update repeats increase the cost of writing observed bindings, and deeper denoising readout increases per-query latency. Section 5.3 shows that these knobs improve NVS quality, but the best operating point depends on the task, hardware budget, and acceptable latency. Our current experiments also use a limited set of datasets and mostly single training runs, so the reported trends should be read as evidence for the generative-memory mechanism rather than a full scaling law. Finally, the theory analyzes an empirical conditional memory with Gaussian bridges; real transformer states only approximate this abstraction, and different objectives, schedules, or key parameterizations can change the practical retrieval margin.

B Broader Impacts

The contribution is a general memory mechanism for generative and long-context models. Positive impacts may come from more efficient use of compressed context in vision and language systems. Negative impacts are indirect and inherit from downstream deployments: stronger long-context language models and higher-quality view synthesis can be misused for misleading content, surveillance, or copyright-sensitive generation if deployed without domain safeguards. We therefore view release controls, dataset provenance checks, and downstream misuse evaluation as application-specific responsibilities for systems built with **TTG**.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the generative fast-weight readout claim, the posterior-margin analysis, and the two empirical settings; Section 5.3 and Appendix A scope the compute and generalization claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Appendix A discusses added update/readout compute, latency tradeoffs, limited dataset and run coverage, and the gap between the Gaussian-bridge analysis and real transformer states.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes].

Justification: Section 3.1–3.3 defines the memory model and bridge assumptions, and each theorem or proposition is followed by a proof or proof sketch in the main text.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: The method section specifies the online update and readout objectives, and the experimental setup reports datasets, tokenization, sequence construction, model sizes, optimizer settings, training lengths, and evaluation metrics.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No].

   Justification: The current draft cites the datasets and benchmarks, but it does not yet include code, checkpoints, or exact reproduction commands in the paper or supplemental material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: Section 4 describes the model update and readout, while the setup section reports the relevant data splits, tokenization, optimizer, learning rate, warmup, gradient clipping, batch sizes, model scales, and evaluation protocols.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No].

   Justification: The current draft reports single-run curves and tables without error bars, confidence intervals, or multi-seed significance tests; this is acknowledged in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The setup reports GPU type, memory class, GPU counts, and batch sizes, but it does not yet report wall-clock time or total compute for each run.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work is an algorithmic study using existing datasets and benchmarks, with no human-subject data collection or deployment claims in the paper.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

Justification: Appendix B discusses positive efficiency/context-use implications and indirect misuse risks for downstream language and view-synthesis systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A].

Justification: The paper does not currently claim to release new models, datasets, or checkpoints that would require release safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No].

Justification: The main datasets and benchmarks are now cited, but explicit license names and terms of use are not yet documented in the paper or supplement.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper does not introduce or release a new dataset, benchmark, model checkpoint, or other asset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The work does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: The core method does not use an external LLM as an important, original, or non-standard component; language models are an experimental domain.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.