IMPROVING FEW-STEP LANGUAGE FLOWS WITH UNTIED SELF-CONDITIONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-matching language models refine all token positions in parallel and can trade sampling steps for latency, yet generation quality still degrades sharply with few sampling steps. We trace a source of this degradation to how self-conditioning reuses predictions during sampling. Training constructs the conditioning input from the current noisy state. During sampling, self-conditioning ties two distinct uses to the same clean prediction: advancing the latent through the solver, and conditioning the next model evaluation. A correction that improves generation when supplied to the solver can become harmful when also used for self-conditioning. We introduce Untied Self-Conditioning, a training-free method that unties these two uses by adapting a shared prediction to their different requirements, using only one model evaluation per sampling step. The past prediction already contributes to the latent, we therefore scale down its conditioning components along directions shared by the model's input projections. For the solver update, we instead use prediction history to estimate how the clean prediction changes during a sampling step and scale the resulting correction using empirical trajectory statistics. Across several pretrained language flow baselines, our method consistently improves generation quality: with only 8 function evaluations (NFE), it reduces generative perplexity on OpenWebText from 531 to 61 ($8.7\times$) for LangFlow, and from 71 to 44 for ELF-B. Under an adapted Arena-Hard-Auto v2 evaluation on LangFlow trained on OpenWebText, our generations win 96.3% of non-tied comparisons with the official sampler at the same NFE. Improvements hold from 8 to 256 sampling steps across all tested models and datasets, suggesting that our method offers a promising direction for efficient sampling in continuous diffusion language models.

1 INTRODUCTION

Flow matching (Lipman et al., 2023) has enabled high-quality parallel generation in continuous domains, with recent progress extending to text (Li et al., 2022; Gong et al., 2023; Strudel et al., 2022; Gulrajani & Hashimoto, 2023; Hu et al., 2024; Chen et al., 2026; Hu et al., 2026). These models embed discrete tokens in a continuous latent space and refine all positions simultaneously, approaching the generation quality of autoregressive language models (Chen et al., 2026; Hu et al., 2026). Reducing the number of sampling steps makes this parallelism a promising route to substantially higher throughput than autoregressive language models. However, generation quality of these models degrades sharply at low numbers of function evaluations (NFE).

A standard technique in these models is self-conditioning (Chen et al., 2023). At each sampling step, the model receives its own clean prediction from the preceding step as an additional input. Here, a clean prediction estimates the noise-free token representations from the current noisy latent. Reusing these predictions lets the model refine earlier estimates as denoising proceeds, substantially improving both generation quality and likelihood in continuous diffusion language models (Strudel et al., 2022; Karimi Mahabadi et al., 2024; Gulrajani & Hashimoto, 2023; Yuan et al., 2024; Liu et al., 2024a; Hu et al., 2026; Chen et al., 2026).

However, standard self-conditioning ties the solver update and the next conditioning input to the same prediction. During sampling, the solver uses this prediction to advance the latent, and the same prediction is passed as the conditioning input to the next model evaluation. The prediction therefore reaches the model in two ways: through its contribution to the current latent and through

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

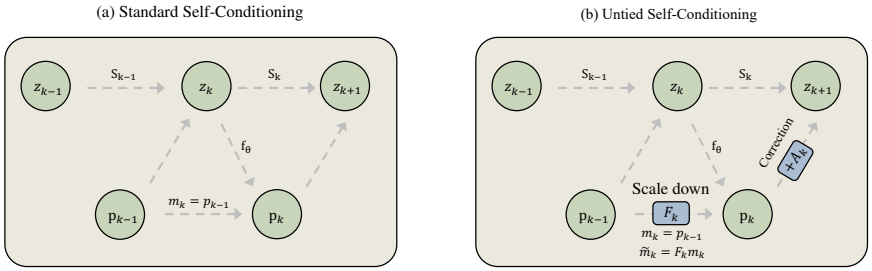

Figure 1: **Past predictions serve both conditioning and integration.** (a) Standard sampling reuses p_{k-1} to form the current latent and to condition its next prediction. (b) Untied Self-Conditioning adjusts these two uses: F_k scales down conditioning along shared input directions, and A_k uses prediction history to refine the solver update. We store the raw prediction p_k; the next model evaluation reads $F_{k+1}p_k$ as conditioning.

the explicit conditioning input. Our analysis shows that these two uses impose different requirements. For conditioning, the appropriate weight depends on the prediction's correlation with the current noisy latent, even when the quality of each input is unchanged. For the solver update, the relevant quantity is the average clean prediction over the sampling interval, while a single model evaluation provides only the prediction at the beginning of that interval. A correction that benefits the solver can impair generation when also reused as conditioning.

Building on this diagnosis, we introduce Untied Self-Conditioning, a training-free method that unties the conditioning and solver inputs derived from a shared model prediction (Figure 1). We retain the raw prediction for subsequent self-conditioning and construct a corrected prediction for the solver. Since the past prediction already contributes to the latent, we adjust how strongly it enters through self-conditioning. The model's learned input projections identify directions represented through both inputs, and we scale down the conditioning components along these directions. For the solver update, we use histories of raw predictions and predictions supplied to the solver to estimate how the clean prediction changes across the sampling interval and scale the resulting correction using statistics from reference trajectories.

Empirically, Untied Self-Conditioning substantially improves the generation quality of ELF and LangFlow, with the largest gains at low NFE. For LangFlow on OpenWebText at 8 NFE, it reduces generative perplexity (GenPPL) from 531 to 61 ($8.7\times$); at the same NFE, ELF-B improves from 71 to 44. At 32 NFE on ELF-B, it reaches a GenPPL of 21. In an adapted Arena-Hard-Auto v2 evaluation of LangFlow on OpenWebText, our generations win 96.3% of non-tied comparisons with the official sampler at 8 NFE.

Our contributions are:

- We analyze the different requirements of conditioning and solver inputs. We show that input correlation can change the sign of the optimal conditioning weight even when each input's quality is unchanged, and derive the step-average prediction required by the solver.
- We introduce Untied Self-Conditioning, a training-free method that unties the conditioning and solver inputs derived from a shared model prediction. It combines a conditioning transform based on input-projection geometry with a solver correction built from histories of raw predictions and predictions supplied to the solver.
- We conduct extensive experiments with three ELF model sizes on OpenWebText and with LangFlow on OpenWebText and LM1B, showing consistent improvements across sampling budgets. Pairwise LLM-judge evaluation shows a clear preference for our generated text. Ablations reveal that corrections that improve generation when applied to the solver can become harmful when also used for self-conditioning.

2 PRELIMINARIES

A language flow embeds discrete tokens into a continuous space and learns a velocity field that transports noise to clean representations. Let x denote a clean representation and $\epsilon \sim \mathcal{N}(0, I)$ noise. For a linear interpolant, the noisy latent is $z_t = (1-t)\epsilon + tx$, with $t = 0$ corresponding to noise and $t = 1$ to clean data. The model produces a clean prediction p of x, which defines the velocity $(p - z_t)/(1-t)$ for $t < 1$. Sampling integrates the resulting ODE from z_0.

Self-conditioning (Chen et al., 2023) supplies the prediction network f_θ with its own clean prediction as an additional input m. During training, on randomly selected examples, it first computes $m = \mathrm{sg}[f_\theta(z, 0, t)]$ and then uses m to predict from the same noisy state z at the same time t. Other examples use $m = 0$. The operator sg stops the gradient through this first prediction. During sampling, the conditioning input is the previous step's prediction, $m_k = p_{k-1}$, with $m_0 = 0$.

For a sampling grid $0{=}t_0{<}\cdots{<}t_K{=}1$, let z_k denote the latent at t_k. Each step evaluates the model and advances the latent:

$$p_k = f_\theta(z_k, m_k, t_k), \qquad z_{k+1} = S_k(z_k, p_k), \qquad m_{k+1} = p_k, \tag{1}$$

where S_k is the numerical solver step. For example, Euler integration of the linear flow gives

$$z_{k+1} = z_k + \frac{t_{k+1} - t_k}{1 - t_k}(p_k - z_k).$$

The prediction p_k therefore contributes to the updated latent and is also passed directly as m_{k+1} to the next model evaluation.

3 ANALYZING CONDITIONING AND SOLVER INPUTS

We analyze how a prediction should be used for conditioning and for the solver update. For conditioning, we examine how its relationship to the current latent affects its value as an additional input. For the solver, we determine what prediction is needed to advance over a finite interval as the latent and time evolve.

3.1 INPUT CORRELATION CHANGES THE OPTIMAL CONDITIONING WEIGHT

To distinguish the effect of input dependence from individual input quality, we consider a scalar estimation problem. We represent the latent and conditioning input as two noisy estimates of a common clean target $Y \sim \mathcal{N}(0, \tau^2)$:

$$Z = Y + \sigma_z \epsilon_z, \qquad M = Y + \sigma_m \epsilon_m, \qquad \mathrm{Corr}(\epsilon_z, \epsilon_m) = \rho, \tag{2}$$

where ϵ_z, ϵ_m are zero-mean, unit-variance, jointly Gaussian, and independent of Y, with $0 < \sigma_z < \sigma_m$. We estimate Y by $\widehat{Y} = a_\rho Z + b_\rho M$, choosing the coefficients to minimize mean squared error. The coefficient b_ρ is the weight given to the conditioning input. The correlation ρ controls how much noise the two channels share. Varying ρ leaves both marginals $P(Y, Z)$ and $P(Y, M)$ unchanged. Only the joint $P(Z, M)$ changes.

The optimal conditioning weight is

$$b_\rho = \frac{\tau^2(\sigma_z^2 - \rho\sigma_z\sigma_m)}{(\tau^2 + \sigma_z^2)(\tau^2 + \sigma_m^2) - (\tau^2 + \rho\sigma_z\sigma_m)^2}. \tag{3}$$

This coefficient decreases with ρ. The denominator is positive, so its sign changes at $\rho_* = \sigma_z/\sigma_m$. Above this threshold, noise cancellation calls for a negative weight on M; retaining a positive weight increases MSE relative to the best estimate from Z alone. Thus, a fixed conditioning weight can help at one correlation level and hurt at another, even when each input's quality is unchanged. Appendix A.1 gives the calculation.

In a controlled ELF-B experiment at 8 NFE, a fixed conditioning-projection update fit under independent corruptions loses its benefit as corruption correlation rises. The guided-velocity MSE ratio (modified/original) increases from 0.92 to 1.07 between $\rho = 0$ and $\rho = 1$ (Figure 4, Section 5.4).

3.2 Overlap between the input projections

The scalar example concerns a single conditioning weight. For a vector-valued conditioning input, we also need a criterion for choosing which components to adjust. We use the overlap between the model's input projection subspaces as a geometric criterion for this choice.

In self-conditioned flow-matching language models, the latent and self-conditioning input enter the network through a single learned projection that operates on their concatenation. Its linear part is $W_z z + W_m m$, where W_z and W_m are the effective input-projection matrices. Each matrix responds only to the component of its input in its row space. Let P_z and P_m be the orthogonal projectors onto these row spaces. To measure this overlap for a conditioning input m, define the projection energy

$$\mathcal{E}(m) = \tfrac{1}{2}\|P_z P_m m\|^2. \tag{4}$$

This is the quadratic form $\mathcal{E}(m) = \frac{1}{2}\langle m, Km\rangle$ with

$$K = P_m P_z P_m. \tag{5}$$

On the self-conditioning subspace, the eigenvectors q_i of K are principal directions, and the eigenvalues $\rho_i^2 \in [0,1]$ are the squared cosines of the principal angles between the two row spaces. A large ρ_i^2 indicates stronger overlap between the two projection subspaces.

A simple way to reduce this energy is the map $(I - aK)$ with $0 < a \leq 1$.It retains fraction $1 - a\rho_i^2$ along direction q_i, attenuating conditioning components more strongly where the projections overlap more, while components unique to self-conditioning pass through. Section 4.1 uses this ordering of directions in the sampler's conditioning transform.

3.3 The solver should approximate the step-average prediction

Self-conditioning supplies information for a model evaluation, while the solver uses the resulting prediction to advance the latent over a finite interval. Consider a step $[t_k, t_{k+1}]$ with the self-conditioning input held fixed at m_k. A single model evaluation provides the prediction at the beginning of the step, but the prediction generally changes as the latent and time evolve. The solver must account for how these predictions contribute over the interval.

To isolate the contribution of the clean prediction, we express the affine flow in normalized latent coordinates y and solver time u, which absorb its time-dependent coefficients: $\mathrm{d}y/\mathrm{d}u = p(u) - y$. For the linear interpolant above, $y = z$ and $u = -\log(1-t)$ for $t < 1$. Here $p(u) = f_\theta(z(u), m_k, t(u))$ is the model's clean prediction along the same step. Writing $p_k = p(u_k)$ and $h_k = u_{k+1} - u_k > 0$, integration gives the prediction required by the exact update:

$$\bar{p}_k = \frac{1}{1 - e^{-h_k}} \int_0^{h_k} e^{-(h_k - s)} p(u_k + s)\, \mathrm{d}s. \tag{6}$$

The correction target is therefore $C_k^* = \bar{p}_k - p_k$. For a smooth prediction path, expanding $p(u_k + s)$ around u_k gives $C_k^* = (h_k/2)\dot{p}_k + O(h_k^2)$. The latent update multiplies this correction by $1 - e^{-h_k} = O(h_k)$, making its contribution second order in step width. Reference averages computed with 32 substeps per interval on frozen ELF-B at 8, 16, and 32 NFE give fitted scaling exponents of 1.04 and 2.02 for the correction and its latent contribution, respectively (Appendix C.7).

Directly estimating $\bar{p}_k$ would require additional model evaluations. If a direction B_k satisfies $\langle C_k^*, B_k\rangle > 0$, applying a sufficiently small positive multiple of it reduces the squared one-step integration error compared with using p_k alone (Appendix A.3).

Section 4.2 constructs the correction direction from recent prediction changes and the solver-weighted history of predictions used in latent updates. Offline trajectory statistics set the correction's mean and scale.

4 Untied Self-Conditioning

At each step, Untied Self-Conditioning transforms the self-conditioning input, evaluates the model once, and constructs a correction from prediction history. The raw prediction updates self-conditioning; the corrected prediction advances the solver.

4.1 SELF-CONDITIONING TRANSFORM

The goal is to reduce $\mathcal{E}(m)$ from Section 3.2 at each step. Principal directions Q and squared cosines ρ_i^2 are computed once from frozen weights; the transform retains fraction $\ell_{k,i}$ along direction q_i:

$$\begin{aligned} F_k &= I - Q(I - \mathrm{Diag}(\ell_{k,1}, \ldots, \ell_{k,r}))\, Q^\top, \\ \widetilde{m}_k &= F_k m_k. \end{aligned} \tag{7}$$

We scale down components more strongly along directions with larger ρ_i^2, following the ordering of the $(I{-}aK)$ transform (Section 3.2). The base exponent $\lambda_R^{(0)}$ is derived from the model's response geometry (Appendix C.4); $\eta_R \geq 0$ scales it, giving effective exponent $\lambda_R = \lambda_R^{(0)}\eta_R$ to control attenuation through the retention factors $\ell_{k,i}$.

The step-dependent strength reflects agreement between two prediction-history residuals (Section 4.2). Let $\bar{\delta}_k$ be their clipped cosine distance, averaged over tokens and reference trajectories, and κ_k the update weight of the solver-weighted history. The strength varies across steps via

$$a_k = \kappa_{k-1}(1 - \bar{\delta}_{k-1}), \tag{8}$$

At the same solver-history weight, stronger average agreement gives stronger scaling down of conditioning. These statistics and coefficients are fixed per model/NFE configuration before generation. The retention factors are

$$\begin{aligned} b_{k,i} &= \max\{1 - a_k\rho_i^2, \epsilon_{\mathrm{fp}}\}, \\ g_k &= \left(\prod_{i=1}^{r} b_{k,i}\right)^{1/r}, \\ \ell_{k,i} &= (g_k b_{k,i})^{\lambda_R}. \end{aligned} \tag{9}$$

Here $r \geq 1$ is the number of principal directions, and the floor $0 < \epsilon_{\mathrm{fp}} \leq 1$ prevents complete suppression of a direction. The factor $g_k^{\lambda_R}$ scales all principal directions equally. At the first step no prior prediction exists, so the input passes unchanged ($a_0 = 0$). One evaluation with this input produces $p_k = f_\theta(z_k, \widetilde{m}_k, t_k)$.

4.2 SOLVER-STEP CORRECTION

On the current trajectory, we maintain two prediction histories to construct the solver correction. The EMA of raw predictions evolves as

$$E_{k+1} = (1 - \alpha)E_k + \alpha p_k, \qquad E_0 = 0. \tag{10}$$

The residual $H_k = p_k - E_k$ compares the current raw prediction with this history and serves as the primary correction direction.

Solver-weighted history. The second history lets the correction account for past solver updates, including corrections already applied to the latent. Each solver step $S_k(z_k, q_k)$ depends linearly on the prediction q_k it receives (before correction $q_k{=}p_k$; with correction $q_k{=}p_k{+}A_k$, see Section 4.3). We track the running weighted average of predictions that entered the solver:

$$O_{k+1} = (1 - \kappa_k)\, O_k + \kappa_k\, q_k, \qquad O_0 = 0, \tag{11}$$

where $\kappa_k \in [0, 1]$ is the update weight of this normalized history (Appendix A.2).

Correction direction. The residual $p_k - O_k$ compares the current prediction with the solver-weighted history. We match its norm to that of H_k to compare the directions of the two residuals:

$$G_k = \begin{cases} \|H_k\| \dfrac{p_k - O_k}{\|p_k - O_k\|} & \text{if } \|H_k\|, \|p_k - O_k\| > \epsilon, \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

After norm matching, $H_k - G_k$ measures directional disagreement. We use it to extrapolate the EMA residual away from the solver-history residual, keeping H_k as the primary direction:

$$U_k = H_k + \eta_O\, \Delta u_k\, (H_k - G_k), \tag{13}$$

where $\Delta u_k \in [0, 1]$ is a normalized step-width feature (Appendix B.3) and $\eta_O \geq 0$ weights the disagreement between the two residuals. Section 5.3 compares this direction with using H_k alone under the same normalization.

Algorithm 1 Untied Self-Conditioning sampling. The corrected prediction updates the solver states (z, O); the raw prediction updates the self-conditioning states (m, E).

Input: frozen model f_θ, base solver S_k, time grid $\{t_k\}_{k=0}^{K}$, initial noise state z_0 from the base sampler
Precomputed: overlap basis Q, cosines $\{\rho_i^2\}$, base transform exponent $\lambda_R^{(0)}$, average cosine distance $\bar{\delta}_k$, per-step correction mean μ_k and centered RMS σ_k
Hyperparameters: η_R, η_T, η_O (nonnegative strength coefficients); $\alpha \in (0, 1]$ (EMA weight on the current prediction)
Initialize: $m_0, E_0, O_0 \leftarrow 0$; $\kappa_{-1} \leftarrow 0$, $\bar{\delta}_{-1} \leftarrow 0$
for $k = 0, \ldots, K-1$ **do**

1. $\widetilde{m}_k \leftarrow F_k m_k$	*self-conditioning transform*
2. $p_k \leftarrow f_\theta(z_k, \widetilde{m}_k, t_k)$	*one model evaluation*
3. $H_k \leftarrow p_k - E_k$; compute G_k, U_k by equation 12 and equation 13	
4. Compute A_k by equation 14	*solver correction*
5. $z_{k+1} \leftarrow S_k(z_k, p_k + A_k)$; $O_{k+1} \leftarrow (1-\kappa_k)O_k + \kappa_k(p_k + A_k)$	*corrected → solver*
6. $m_{k+1} \leftarrow p_k$; $E_{k+1} \leftarrow (1-\alpha)E_k + \alpha p_k$	*raw → self-conditioning*

end for. Decode z_K with the base model's final evaluation and return the decoded tokens (Appendix A.2).

Normalizing with offline statistics. The direction U_k comes from the current trajectory. Its mean and scale are set from history-based corrections applied by a fixed reference sampler: a per-step token mean μ_k and centered RMS σ_k, estimated on disjoint offline trajectory banks (Appendix B.4). Let C_{tok} center a tensor by subtracting its mean over active token positions. The normalized correction, with overall strength η_T, is

$$\Pi_k(U_k) = \mu_k + \sigma_k \frac{C_{\text{tok}} U_k}{\text{RMS}(C_{\text{tok}} U_k)}, \qquad A_k = \eta_T \Pi_k(U_k). \tag{14}$$

This replaces the scale and mean of U_k while preserving its centered direction; inactive positions and zero-norm cases follow Appendix B.5. Appendix C.7 evaluates the resulting direction against high-resolution step averages.

4.3 State update

Each correction applies to its own pathway:

$$\boxed{\begin{aligned} z_{k+1} &= S_k(z_k, p_k + A_k), & m_{k+1} &= p_k, \\ O_{k+1} &= (1-\kappa_k)O_k + \kappa_k(p_k + A_k), & E_{k+1} &= (1-\alpha)E_k + \alpha p_k. \end{aligned}} \tag{15}$$

The raw p_k updates self-conditioning and the EMA; the corrected $p_k + A_k$ advances the solver and its state. The correction targets the solver's step-average prediction. Keeping it out of the stored prediction lets F_{k+1} determine the next conditioning input separately (Section 5.3).

Algorithm 1 summarizes the loop.

5 Experiments

We evaluate Untied Self-Conditioning across models and sampling budgets, then examine its conditioning and solver corrections through controlled experiments.

5.1 Experimental setup

Data and generation. ELF-B, ELF-M, and ELF-L are evaluated on OpenWebText with NFE ranging from 8 to 64. LangFlow is evaluated on OpenWebText and LM1B with NFE from 8 to 256. OpenWebText generations contain 1,024 tokens; LangFlow LM1B generations contain 128.

Evaluation. We report generative perplexity (GenPPL) under GPT-2 Large and measure token diversity with unigram entropy H. The results in Tables 3 and 4 average six seeds, each with 1,024 generations. Comparisons use the same pretrained model, base solver, time grid, and initial latents.

5.2 Generation quality improves at every tested NFE

Untied Self-Conditioning improves generation quality relative to the official sampler in every tested configuration (Figure 2). At 8 NFE, Untied Self-Conditioning reduces LangFlow GenPPL on OpenWebText from 531.2 to 60.8 ($8.7\times$) and on LM1B from 264.0 to 70.4 ($3.7\times$). On ELF-B at 8 NFE it reaches 43.90 versus the official sampler's 70.63. At 32 NFE on ELF-B it reaches 21.0 versus the official sampler's 23.7 on OpenWebText. Figure 3(a) compares OpenWebText GenPPL with published few-step language models.

For LangFlow on OpenWebText, Untied Self-Conditioning also maintains unigram entropy near the data reference across sampling budgets. At 8, 16, and 32 NFE on LangFlow, none of the 6,144 paired samples at each budget has higher NLL under Untied Self-Conditioning (Appendix C.11). Across six workloads the sampler retains 95.5% geometric-mean throughput (Appendix C.10).

Pairwise LLM-judge evaluation. Figure 3(b) provides a direct quality comparison independent of perplexity using an adapted Arena-Hard-Auto v2 protocol (Li et al., 2025). Each NFE pairs 1,024 generations from the official sampler and Untied Self-Conditioning on the same latent; each pair is judged blind in both presentation orders by DeepSeek V4 Pro, yielding 8,192 judgments in total. Untied Self-Conditioning wins 96.3% of non-tied comparisons at 8 NFE and retains a clear advantage at 64 NFE.

Untied Self-Conditioning also outperforms several training-free sampling baselines on ELF-B at 8 NFE (Table 13).

5.3 Correction placement and prediction history

Table 1: **Correction placement and method components.** We evaluate ELF-B at 8 NFE using one run of 1,024 matched initial latents. Panel (a) tests where to apply the correction with the conditioning transform F fixed; the top left entry uses F alone. Next SC denotes the stored prediction before applying F. Panel (b) tests the conditioning transform and solver correction separately and together. Entries report GenPPL and unigram entropy H.

(a) Correction placement

	Solver: p_k	Solver: $p_k + A_k$
Next SC: p_k	67.27 / 5.23	**45.63 / 5.14**
Next SC: $p_k + A_k$	157.77 / 5.35	334.19 / 5.33

(b) Components

SC transform	Correction off	Correction on
Off	74.18 / 5.27	50.04 / 5.16
On	67.27 / 5.23	**45.63 / 5.14**

Correction placement. Table 1(a) tests where to apply the correction by varying whether it is supplied to the solver and the next self-conditioning input, with the conditioning transform and correction rule fixed on ELF-B at 8 NFE. Applying it only to the solver gives the best GenPPL among the four assignments. Adding the correction to the solver lowers GenPPL from the transform-only baseline's 67.27 to 45.63. Also supplying the corrected prediction as self-conditioning raises GenPPL to 334.19. Applying the correction to conditioning alone gives 157.77.

A comparison using the same correction rule across five additional settings also favors supplying the correction to the solver (Appendix C.8).

Contributions of prediction history and conditioning. Table 1(b) tests the contribution of each component by enabling the conditioning transform and solver correction separately and together. Both improve GenPPL on their own, and the complete method outperforms either component alone. Without the conditioning transform, adding the solver-weighted history term to the EMA residual improves GenPPL from 56.90 to 50.04 under the same mean/RMS normalization. The complete history-based direction also outperforms a random direction with the same RMS (71.45; Appendix C.9).

Scaling down conditioning components aligned with projection overlap lowers GenPPL from 64.27 to 55.64, outperforming reversed and random direction assignments with matched eigenvalues (Table 8, Appendix C.6).

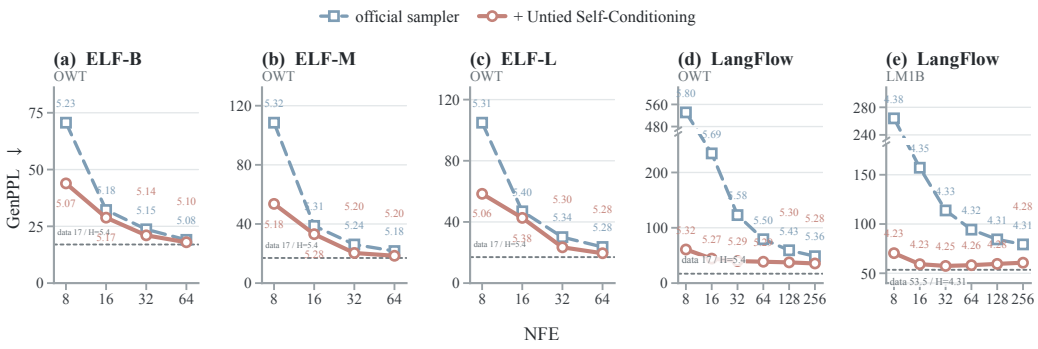

Figure 2: **Untied Self-Conditioning improves generation quality, especially at low NFE.** Panels (a)–(e) compare the official sampler (dashed) with Untied Self-Conditioning (solid) for each model–dataset pair. Numbers beside the curves report unigram entropy. Dotted lines mark data references taken from Hu et al. (2026) (OWT) and Yoo et al. (2026) (LM1B).

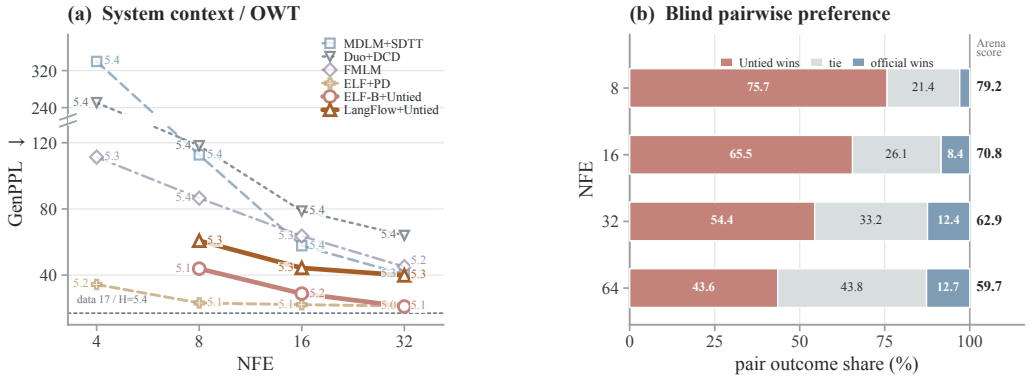

Figure 3: **Untied Self-Conditioning improves generation quality on OpenWebText.** (a) GenPPL comparison with published few-step models; external curves taken from Hu et al. (2026). (b) The LLM judge prefers LangFlow generations from Untied Self-Conditioning over those from the official sampler, with 1,024 pairs per NFE. Bars show percentages of all pairs; labels on the right give the Arena score.

5.4 Conditioning gains depend on noise correlation

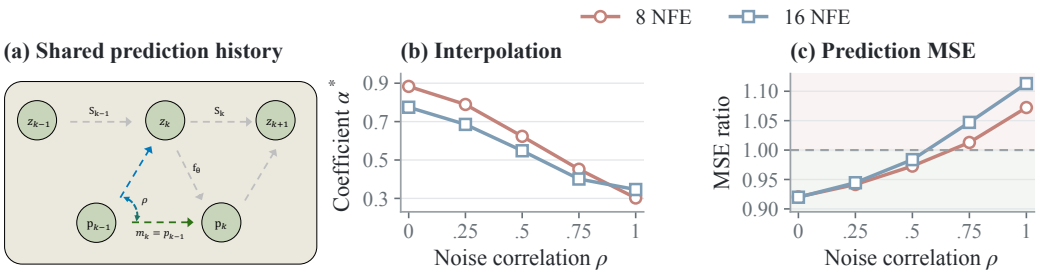

Figure 4: **Correlated noise reduces the benefit of a fixed projection modification.** (a) A past prediction reaches the next evaluation through the latent update and the conditioning input, motivating a test of input dependence. (b) MSE-minimizing interpolation coefficient. (c) Guided-velocity MSE ratio (modified/original). The curves vary Gaussian corruption correlation with fixed marginals; the modification is a low-rank conditioning-projection update fit at $\rho = 0$.

We test whether a conditioning modification learned with independent corruptions retains its benefit when those corruptions become correlated. We vary the correlation between Gaussian corruptions at successive model evaluations, preserving each corruption's Gaussian marginal. The original model and a copy whose conditioning projection carries a fixed low-rank update, fit under independent corruptions (ρ=0; Appendix C.5), receive the same latent and time. We evaluate five correlations from 0 to 1 at 8 and 16 NFE.

For each setting, we interpolate between the two predictions and compute the coefficient minimizing MSE against the same guided velocity target. Coefficients zero and one recover the original and modified predictions, respectively. The optimal coefficient decreases from 0.88 to 0.30 at 8 NFE and from 0.77 to 0.35 at 16 NFE. The MSE ratio of the modified to the original prediction rises from 0.92 to 1.07 and from 0.92 to 1.11, respectively.

A projection fit for independent corruptions loses its benefit as the corruptions become correlated. This supports the Gaussian analysis of Section 3.1: the benefit of conditioning depends on how its errors relate to those of the current input. Appendix C.5 gives the construction, target, and controls.

6 Related Work

Self-conditioned language flows. Continuous diffusion and flow models for text embed discrete tokens into a continuous space and refine all positions in parallel (Li et al., 2022; Strudel et al., 2022; Gulrajani & Hashimoto, 2023; Han et al., 2023; Karimi Mahabadi et al., 2024; Tae et al., 2025; Gong et al., 2023; Yuan et al., 2024; Liu et al., 2024a; Yang et al., 2026; Hu et al., 2024; Chen et al., 2026; Hu et al., 2026). Previous-prediction self-conditioning (Chen et al., 2023) supplies the model with its own clean prediction from the preceding step; variants appear across embedding, simplex, and contextual formulations (Strudel et al., 2022; Karimi Mahabadi et al., 2024; Gulrajani & Hashimoto, 2023; Yuan et al., 2024; Liu et al., 2024a; Hu et al., 2026). We study models in which a frozen network receives the current latent and a previous clean prediction through separate linear input projections. ELF and LangFlow are two instances of this interface.

Solvers and multi-step methods. Advanced solvers reuse past model evaluations to improve numerical integration of the probability-flow ODE (Zhang & Chen, 2023; Lu et al., 2025; Zhao et al., 2023). Separately, several methods aggregate past clean predictions to stabilize or refine the current prediction supplied to the solver (Wizadwongsa et al., 2024; Qian et al., 2024; Li & zeng, 2026; Sadat et al., 2026; Li et al., 2026).

Improving self-conditioning. Several studies diagnose train–inference mismatch in the self-conditioning recurrence and address it through modified training (Shabalin et al., 2025; Liu et al., 2024b; Cong et al., 2026). Other methods expose the model to its own intermediate states during training (Savinov et al., 2022), specialize through post-training (Cardei et al., 2026), or distill recurrent self-conditioning iterations (Yoo et al., 2026). At inference time, Analog Bits explores self-guidance using a second model evaluation at the current step (Chen et al., 2023), and ACE addresses repetition in the self-conditioning feedback loop (Zhang et al., 2026). Each prediction serves both the current solver update and the next conditioning input.

7 Conclusion

We introduced Untied Self-Conditioning, a training-free method for improving few-step sampling in continuous diffusion language models. It adapts a shared prediction separately to the requirements of conditioning and solver updates, since a correction that improves generation when applied to the solver can become harmful when also used for conditioning. Across ELF and LangFlow, it improves generation quality at every tested NFE. With eight evaluations it reduces LangFlow GenPPL on OpenWebText from 531 to 61, with gains confirmed by pairwise LLM-judge preference and per-sample external-evaluator likelihood. We hope this work encourages closer examination of prediction reuse in diffusion sampling.

AI USE STATEMENT

In this work, we used generative AI tools to assist in checking code and verifying mathematical derivations and proofs. We have not used generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, designing research methodology or experiments, implementing methods, translating, cleaning datasets, qualitative data analysis, or interpreting results. Additionally, we used generative AI tools to edit and polish the manuscript for readability, for literature retrieval, discovery, and organization of related work, and to draft sections of the paper. The authors reviewed all AI-assisted text and verified it against original sources and experiment artifacts. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies inference for pretrained generative language models and introduces no new dataset or human-participant study. Deployments should retain the access controls and output safeguards appropriate for the underlying models.

REPRODUCIBILITY STATEMENT

Section 2 defines the baseline sampling recurrence. Section 3 analyzes how input correlation and the finite sampling interval affect conditioning and solver inputs. Section 4 develops the self-conditioning transform, history-based solver correction, and untied state update. Appendices A and B provide proofs, solver-specific recurrences, precomputation rules, and numerical conventions. Section 5.1 and Appendix C report the evaluated checkpoints and sampling budgets, sample counts, matched controls, and evaluation procedures.

REFERENCES

Michael Cardei, Huu Binh Ta, and Ferdinando Fioretto. Simple self-conditioning adaptation for masked diffusion models, 2026. URL https://arxiv.org/abs/2604.26985.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3itjR9QxFw.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. Langflow: Continuous diffusion rivals discrete in language modeling, 2026. URL https://arxiv.org/abs/2604.11748.

Dat Nguyen Cong, Tung Kieu, and Hoang Thanh-Tung. FastDiSS: Few-step match many-step diffusion language model on sequence-to-sequence generation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17572–17592, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.870. URL https://aclanthology.org/2026.findings-acl.870/.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=jQj-_rLVXsj.

Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 16693–16715. Curran Associates, Inc., 2023. doi: 10.52202/075280-0730. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/35b5c175e139bff5f22a5361270fce87-Paper-Conference.pdf.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.647. URL https://aclanthology.org/2023.acl-long.647/.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. Elf: Embedded language flows, 2026. URL https://arxiv.org/abs/2605.10938.

Vincent Hu, Di Wu, Yuki Asano, Pascal Mettes, Basura Fernando, Björn Ommer, and Cees Snoek. Flow matching for conditional text generation in a few sampling steps. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 380–392, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-short.33. URL https://aclanthology.org/2024.eacl-short.33/.

Rabeeh Karimi Mahabadi, Hamish Ivison, Jaesung Tae, James Henderson, Iz Beltagy, Matthew Peters, and Arman Cohan. TESS: Text-to-text self-conditioned simplex diffusion. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2347–2361, St. Julian's, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-long.144. URL https://aclanthology.org/2024.eacl-long.144/.

Shigui Li and Delu zeng. DiFA: Inference-time forward-process alignment for diffusion models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=YGyWLM0OEW.

Tiancheng Li, Weijian Luo, Zhiyang Chen, Liyuan Ma, and Guo-Jun Qi. Self-guidance: Boosting flow and diffusion generation on their own. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(1):781–791, 2026. doi: 10.1109/TPAMI.2025.3611831.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 34209–34231. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/li25h.html.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343. Curran Associates, Inc., 2022. doi: 10.52202/068431-0313. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/1be5bc25d50895ee656b8c2d9eb89d6a-Paper-Conference.pdf.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Pan Liu, Xiaohua Tian, and Zhouhan Lin. Enable fast sampling for Seq2Seq text diffusion. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 8495–8505, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.497. URL https://aclanthology.org/2024.findings-emnlp.497/.

Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Text diffusion with reinforced conditioning, 2024b. URL https://arxiv.org/abs/2402.14843.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22(4):730–751, June 2025. ISSN 2731-5398. doi: 10.1007/s11633-025-1562-4. URL http://dx.doi.org/10.1007/s11633-025-1562-4.

Yurui Qian, Qi Cai, Yingwei Pan, Yehao Li, Ting Yao, Qibin Sun, and Tao Mei. Boosting diffusion models with moving average sampling in frequency domain. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8911–8920, June 2024.

Seyedmorteza Sadat, Farnood Salehi, and Romann M. Weber. HiGS: History-guided sampling for plug-and-play enhancement of diffusion models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=cyQUZDMpg3.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=T0GpzBQ1Fg6.

Alexander Shabalin, Viacheslav Meshchaninov, Egor Chimbulatov, Vladislav Lapikov, Roman Kim, Grigory Bartosh, Dmitry Molchanov, Sergey Markov, and Dmitry Vetrov. Tencdm: Understanding the properties of the diffusion model in the space of language model encodings, 2025. URL https://arxiv.org/abs/2402.19097.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. Self-conditioned embedding diffusion for text generation, 2022. URL https://arxiv.org/abs/2211.04236.

Jaesung Tae, Hamish Ivison, Sachin Kumar, and Arman Cohan. TESS 2: A large-scale generalist diffusion language model. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21171–21188, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1029. URL https://aclanthology.org/2025.acl-long.1029/.

Suttisak Wizadwongsa, Worameth Chinchuthakun, Pramook Khungurn, Amit Raj, and Supasorn Suwajanakorn. Diffusion sampling with momentum for mitigating divergence artifacts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=HXc5aXeoc8.

Zhihan Yang, Wei Guo, Shuibai Zhang, Subham Sekhar Sahoo, Yongxin Chen, Arash Vahdat, Morteza Mardani, and John Thickstun. Continuous diffusion scales competitively with discrete diffusion for language, 2026. URL https://arxiv.org/abs/2605.18530.

Jaehoon Yoo, Wonjung Kim, Floor Eijkelboom, Chanhyuk Lee, Nicholas M. Boffi, Seunghoon Hong, and Jinwoo Kim. Self-conditioned flow map language models via fixed-point flows, 2026. URL https://arxiv.org/abs/2607.00714.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. Text diffusion model with encoder-decoder transformers for sequence-to-sequence generation. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 22–39, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.2. URL https://aclanthology.org/2024.naacl-long.2/.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Loek7hfb46P.

Shuai Zhang, Zijie Chen, Hongliang He, Lun Du, and Zhenzhong Lan. Low perplexity is repetition: A one-dimensional self-conditioning attractor in continuous diffusion lms, 2026. URL https://arxiv.org/abs/2607.00588.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 49842–49869. Curran Associates, Inc., 2023. doi: 10.52202/075280-2170. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/9c2aa1e456ea543997f6927295196381-Paper-Conference.pdf.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 55502–55542. Curran Associates, Inc., 2023. doi: 10.52202/075280-2423. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ada8de994b46571bdcd7eeff2d3f9cff-Paper-Conference.pdf.

A Derivations and Proofs

A.1 Gaussian conditioning calculation

Let $T = \tau^2$, $a = \sigma_z$, and $b = \sigma_m$. Gaussian conditioning on (Z, M) gives the coefficient on M as

$$b_\rho = \frac{T(a^2 - \rho ab)}{\Delta_\rho}, \qquad \Delta_\rho = (T + a^2)(T + b^2) - (T + \rho ab)^2 > 0. \tag{16}$$

Neither marginal $P_{Y,Z}$ nor $P_{Y,M}$ depends on ρ. For $0 < a < b$, direct differentiation gives

$$\frac{\mathrm{d}b_\rho}{\mathrm{d}\rho} = -\frac{Tab\left[(b^2 - a^2)(T + a^2) + a^2b^2(\rho - a/b)^2\right]}{\Delta_\rho^2} < 0. \tag{17}$$

Thus b_ρ vanishes at $\rho = a/b$ and changes sign there. At $\rho = 1$, the shared-noise observations permit exact cancellation, $Y = (bZ - aM)/(b - a)$. The conditional fusion rule can therefore change sign while both feature–target marginals remain fixed.

A.2 Exact average of past predictions using the solver coefficients

For ELF, distinguish the time s_k of the model evaluation (after any re-noising) from the target time t_{k+1}. Write the latent at that evaluation as

$$z_k = (1 - s_k)\nu_k + s_k O_k, \tag{18}$$

where ν_k contains the complementary noise component and O_k is the normalized contribution of predictions previously supplied to the solver. Suppose the solver uses q_k to advance the latent from s_k to t_{k+1}, with

$$\beta_k = \frac{t_{k+1} - s_k}{1 - s_k}. \tag{19}$$

Substitution into the affine solver update gives

$$O_{k+1} = \frac{(1 - \beta_k)s_k O_k + \beta_k q_k}{t_{k+1}} = (1 - \kappa_k)O_k + \kappa_k q_k, \qquad \kappa_k = \frac{\beta_k}{t_{k+1}}. \tag{20}$$

The identity $(1 - \beta_k)s_k + \beta_k = t_{k+1}$ makes the two coefficients sum to one. The coefficient of q_k in the full latent is β_k, whereas $\kappa_k = \beta_k/t_{k+1}$ is its update share inside normalized O_{k+1}. With $O_0 = 0$ and $s_0 = 0$, the first normalized update has $\kappa_0 = 1$ and $O_1 = q_0$, even though the full-latent coefficient is only $\beta_0 = t_1$. Re-noising changes s_k but preserves this algebraic separation.

For a LangFlow transition from negative log-SNR γ_t to γ_s, the clean-prediction component after dividing by the signal coefficient obeys

$$O_s = e^{(\gamma_s - \gamma_t)/2} O_t + \left(1 - e^{(\gamma_s - \gamma_t)/2}\right) q_t. \tag{21}$$

Thus equation 11 holds with $\kappa = 1 - e^{(\gamma_s - \gamma_t)/2}$. Additive stochastic terms belong to the complementary component of the latent and do not enter O_k.

After the K updates in Algorithm 1, the final evaluation decodes z_K, using zero conditioning for ELF and $F_K m_K$ for LangFlow. ELF uses $K = \mathrm{NFE}$ denoising evaluations followed by its decoder; LangFlow uses $K = \mathrm{NFE} - 1$ update evaluations and one final decoding evaluation.

A.3 WEIGHTED PREDICTION OVER A SOLVER STEP AND STEP-SIZE ORDERS

Hold the self-conditioning input fixed between two model evaluations, and let $y(u)$ denote the normalized latent coordinate driven by the clean prediction. For self-conditioned language flows whose solver updates the latent affinely in the clean prediction, a change of time coordinate gives

$$\frac{\mathrm{d}y}{\mathrm{d}u} = p(u) - y(u), \qquad p(u) = f_\theta(z(u), m_k, t(u)). \tag{22}$$

Here u is the increasing solver time coordinate (ELF uses $u = -\log(1-t)$ in its active region; LangFlow uses $u = -\gamma/2$) and $h_k = u_{k+1} - u_k > 0$ is the width of step k in this coordinate. Multiplying equation 22 by e^u gives

$$\frac{\mathrm{d}}{\mathrm{d}u}\left(e^u y(u)\right) = e^u p(u). \tag{23}$$

Integration over $[u_k, u_k + h]$ yields

$$y(u_k + h) = e^{-h} y(u_k) + \int_0^h e^{-(h-s)} p(u_k + s)\,\mathrm{d}s. \tag{24}$$

Defining the step-average prediction

$$\bar{p}_k(h_k) = \frac{1}{1 - e^{-h_k}} \int_0^{h_k} e^{-(h_k - s)} p(u_k + s)\,\mathrm{d}s, \tag{25}$$

the exact endpoint is

$$y(u_k + h_k) = e^{-h_k} y(u_k) + (1 - e^{-h_k})\bar{p}_k(h_k). \tag{26}$$

If the prediction path is three times differentiable,

$$p(u_k + s) = p_k + s\dot{p}_k + \frac{s^2}{2}\ddot{p}_k + O(s^3). \tag{27}$$

Substitution into equation 25 gives

$$C_k^*(h) = a_1(h)\dot{p}_k + \frac{a_2(h)}{2}\ddot{p}_k + O(h^3), \tag{28}$$

where

$$\begin{aligned} a_1(h) &= \frac{h - 1 + e^{-h}}{1 - e^{-h}} = \frac{h}{2} + O(h^2), \\ a_2(h) &= \frac{h^2 - 2h + 2 - 2e^{-h}}{1 - e^{-h}} = \frac{h^2}{3} + O(h^3). \end{aligned} \tag{29}$$

Since $1 - e^{-h} = h + O(h^2)$, the true correction is first order and its contribution to the endpoint is second order. For any candidate B_k, substituting $p_k + \eta B_k$ for the interval prediction in equation 26 gives

$$y_{k+1}^{(\eta B)} - y_{k+1}^* = (1 - e^{-h_k})(\eta B_k - C_k^*). \tag{30}$$

The squared endpoint error therefore differs from the squared correction error by the positive factor $(1 - e^{-h_k})^2$. For a nonzero direction B_k applied with scale $\eta > 0$, the change in squared correction error is

$$\|C_k^* - \eta B_k\|^2 - \|C_k^*\|^2 = \eta^2 \|B_k\|^2 - 2\eta \left\langle C_k^*, B_k \right\rangle. \tag{31}$$

Thus, if $\langle C_k^*, B_k \rangle > 0$, any scale $0 < \eta < 2\langle C_k^*, B_k \rangle / \|B_k\|^2$ reduces the squared one-step integration error relative to the uncorrected update.

The same weighted-average representation holds for a general scalar affine flow

$$\frac{\mathrm{d}y}{\mathrm{d}u} = a(u)y(u) + b(u)p(u). \tag{32}$$

If

$$\varphi(v, u) = \exp\left(\int_u^v a(s)\,\mathrm{d}s\right), \tag{33}$$

variation of constants weights $p(u+s)$ by $\varphi(u+h, u+s)b(u+s)$. Normalizing this weight gives the corresponding step-average prediction whenever its integral is nonzero. ELF uses $u = -\log(1-t)$ on finite active intervals; the LangFlow update after dividing by the signal coefficient uses $u = -\gamma/2$ and $h = (\gamma_t - \gamma_s)/2$. The ELF terminal endpoint has an infinite unregularized log-clock width; the sampler uses the bounded interval feature defined in Appendix B.3 for this step.

A.4 Correction for a finite, zero-initialized EMA

For an affine sequence in the model-evaluation index, $p_i = p_0 + iv$, with $E_{i+1} = (1-\alpha)E_i + \alpha p_i$, $E_0 = 0$, and $\alpha \in (0, 1]$, let $r = 1 - \alpha$. Direct summation gives $w_n p_n - E_n = L_n v$, where $w_n = 1 - r^n$ and

$$L_n = \alpha \sum_{j=1}^{n} j\, r^{j-1} = \frac{1 - (n+1)r^n + nr^{n+1}}{\alpha}. \tag{34}$$

The raw residual satisfies $p_n - E_n = r^n p_n + L_n v$, so the bias-corrected increment $\widehat{v} = (w_n p_n - E_n)/L_n$ recovers v exactly for $n \geq 1$. For $\alpha \in (0, 1)$, the lag L_n is strictly positive and increases monotonically toward $1/\alpha$; at $\alpha = 1$, $L_n = 1$. This identity concerns increments per model evaluation. Converting them to derivatives in solver time also requires the time-grid spacing and prediction curvature. The sampler uses the raw residual as a history-based direction.

B Method Details

B.1 Principal-angle directions for self-conditioning

Let W_z and W_m be the effective frozen input-projection matrices for the latent and self-conditioning inputs. Orthonormal row-space bases define projectors P_z and P_m. On the self-conditioning row space,

$$P_m P_z P_m Q = Q\,\mathrm{Diag}(\rho_1^2, \ldots, \rho_r^2), \qquad 0 \leq \rho_i \leq 1. \tag{35}$$

The basis Q and squared cosines ρ_i^2 are computed once from frozen model weights.

At step k, the transform uses

$$a_k = \kappa_{k-1}(1 - \bar{\delta}_{k-1}), \tag{36}$$

where κ_{k-1} is the solver's update weight from equation 11 and $\bar{\delta}_{k-1}$ is the average clipped cosine distance defined in Appendix B.4. The state initialization $\kappa_{-1} = 0$, $\bar{\delta}_{-1} = 0$ makes $F_0 = I$. With $\lambda_R^{(0)} \geq 0$ and $\lambda_R = \lambda_R^{(0)} \eta_R$, the retention factors follow equation 9. The resulting transform is

$$F_k = I - Q(I - \mathrm{Diag}(\ell_{k,1}, \ldots, \ell_{k,r}))Q^\top. \tag{37}$$

It is symmetric, preserves the orthogonal complement of Q, and multiplies principal direction i by $\ell_{k,i}$. The floor ϵ_{fp} prevents complete suppression of any direction. The common factor $g_k^{\lambda_R}$ adds a uniform contraction within the principal subspace; because $0 < b_{k,i} \leq 1$ and $b_{k,i}$ is nonincreasing in ρ_i^2, the transform scales down components more along directions with greater overlap, as the $(I - aK)$ family of Section 3.2 does. Setting this factor to one isolates the direction-dependent scaling in the experiment reported in Appendix C.6.

B.2 Input-projection geometry across models

Table 2 reports the effective input-projection matrices used to compute the principal-angle basis. For ELF, the latent and self-conditioning inputs pass through a shared bottleneck projection before the first transformer block, producing effective matrices of shape 128×512 (rank 128 in a 512-dimensional input space). The measured principal-angle spectrum is non-trivial: squared cosines range from near zero to 0.92 with a well-spread distribution that is stable across checkpoints.

For LangFlow, the self-conditioning projection is 768×1536 and splits into two 768×768 blocks. Both are full rank, so $P_z{=}P_m{=}I$ and all squared cosines equal one. The self-conditioning transform therefore applies uniform scaling in this architecture.

Table 2: Geometry of the input projections. ρ_i is the cosine of the ith principal angle between the two row spaces. ELF projections have rank 128, with ρ_i varying across directions. LangFlow projections have full rank, giving $\rho_i = 1$ and uniform scaling.

Model	Shape	Rank	$\rho_{\min}$	ρ_{med}	$\rho_{\max}$	Mean ρ_i^2	$\lvert\rho_i{\geq}0.9\rvert$	$\lvert\rho_i{\leq}0.1\rvert$
ELF-B	128×512	128	0.001	0.594	0.959	0.391	18	14
ELF-M	128×512	128	0.001	0.581	0.950	0.381	17	13
ELF-L	128×512	128	0.004	0.577	0.952	0.375	13	13
LangFlow	768×768	768	full rank; all $\rho_i{=}1$					

B.3 Prediction history and solver-step width

The EMA uses weight $\alpha \in (0, 1]$ on the current prediction, with $E_0 = 0$. The history residual used by the sampler is

$$H_k = \mathbf{1}[\sigma_k > 0](p_k - E_k), \tag{38}$$

and is active on steps with nonzero precomputed statistics. The zero-initialized EMA introduces a startup bias that decays as $(1{-}\alpha)^k$; a bias-corrected variant $\widehat{v}_k{=}(w_k p_k{-}E_k)/L_k$ recovers the increment of an affine prediction sequence (Appendix A.4). The sampler uses the raw residual.

A second direction estimate compares p_k with the running weighted average O_k and is rescaled samplewise to match $\|H_k\|_F$:

$$G_k = \begin{cases} \|H_k\|_F \dfrac{p_k - O_k}{\|p_k - O_k\|_F} & \text{if } \|H_k\|_F, \|p_k - O_k\|_F > \epsilon, \\ 0 & \text{otherwise,} \end{cases} \tag{39}$$

as in equation 12. The history-based correction direction is

$$U_k = H_k + \eta_O \, \Delta u_k \, (H_k - G_k). \tag{40}$$

It reduces to H_k when the two estimates agree or the step width vanishes. For ELF, let t_k^{start} be the official solver-grid start before any re-noising and define

$$c_\epsilon(t) = \begin{cases} -\log(1-t), & t \leq 1-\epsilon, \\ -\log\epsilon + \dfrac{t-(1-\epsilon)}{\epsilon}, & t > 1-\epsilon, \end{cases} \qquad \widetilde{c}_\epsilon(t) = \frac{c_\epsilon(t)}{c_\epsilon(1)}. \tag{41}$$

The feature used by the sampler is $\Delta u_k = \widetilde{c}_\epsilon(t_{k+1}) - \widetilde{c}_\epsilon(t_k^{\text{start}})$. LangFlow uses $\Delta u_k = (\gamma_k - \gamma_{k+1})/(\gamma_{\max} - \gamma_{\min})$. Both lie in $[0, 1]$ on their sampling grids. The exact solver-time widths are $-\log(1 - t_{k+1}) + \log(1 - s_k)$ and $(\gamma_k - \gamma_{k+1})/2$, respectively (Appendix A.3).

B.4 Empirical statistics from reference corrections

The offline statistics are computed from a fixed reference recurrence that applies a correction during the middle portion of sampling and ramps smoothly. The specific constants below define this reference and are not varied per configuration. The reference EMA uses $\alpha = 3/4$, and its conditioning transform uses the base exponent with multiplier one. For LangFlow, this reference exponent is 0.075. The transform uses the clipped disagreement at each token from the preceding reference step in place of $\bar{\delta}$. Let s_k^{ref} be a step-dependent activation schedule, $g_T^{(0)}$ a fixed base scale (0.075 for LangFlow), and $\lambda_k^{\text{ref}} = 3.5\, g_T^{(0)} s_k^{\text{ref}}$. On each reference trajectory, the correction is

$$\begin{aligned} \widetilde{H}_k &= \lambda_k^{\text{ref}}(p_k - E_k), \\ \widetilde{G}_k &= \begin{cases} \|\widetilde{H}_k\|_F \dfrac{p_k - O_k}{\|p_k - O_k\|_F}, & \|\widetilde{H}_k\|_F, \|p_k - O_k\|_F > \epsilon, \\ 0, & \text{otherwise,} \end{cases} \\ \widetilde{U}_k &= \widetilde{H}_k + \Delta u_k \, \widetilde{\delta}_k \odot (\widetilde{H}_k - \widetilde{G}_k). \end{aligned} \tag{42}$$

The activation schedule is $s_k^{\text{ref}} = \mathbf{1}[0.40 < r_k \leq 0.95]\sqrt{(r_k - 0.40)/0.55}$ for normalized sampling progress r_k; inactive tokens are zeroed throughout. During generation, Equation 40 uses $H_k =$

$p_k - E_k$ on every step with $\sigma_k > 0$. For active token n, $\widetilde{\delta}_{k,n}$ is $\min\{1, 1 - \cos(\widetilde{H}_{k,n}, \widetilde{G}_{k,n})\}$ when both vectors are nondegenerate and zero otherwise. The reference recurrence supplies $p_k + \widetilde{U}_k$ to the solver and stores the raw p_k for the next self-conditioning input.

For a correction tensor V on active positions I, let $\mathsf{M}_I(V)$ be its token mean and $C_I V = V - \mathsf{M}_I(V)$ its centered component. Two disjoint frozen trajectory banks $\mathcal{B}_\mu$ and $\mathcal{B}_\sigma$ define

$$\begin{aligned} \bar{\delta}_k &= \mathbb{E}_{\xi \sim \mathcal{B}_\mu} \left[\frac{1}{|I_k^\xi|} \sum_{n \in I_k^\xi} \widetilde{\delta}_{k,n}^\xi \right], \\ \mu_k &= \mathbb{E}_{\xi \sim \mathcal{B}_\mu} [\mathsf{M}_{I_k^\xi}(\widetilde{U}_k^\xi)], \\ \sigma_k^2 &= \frac{\sum_{\xi \in \mathcal{B}_\sigma} \|C_{I_k^\xi} \widetilde{U}_k^\xi\|_F^2}{\sum_{\xi \in \mathcal{B}_\sigma} |I_k^\xi| D}, \end{aligned} \tag{43}$$

where D is the prediction width. Thus μ_k and σ_k summarize the correction used by the fixed reference recurrence. The true correction C_k^* enters separately through the endpoint condition in equation 31. As with empirical model statistics in training-free solvers (Zheng et al., 2023), the update form is fixed analytically and the frozen model and grid supply its coefficients. Here $\mathbb{E}_{\xi \sim \mathcal{B}}$ denotes a uniform empirical average over the stored bank. For LangFlow, the collected moments are converted to the effective units in equation 47; the disagreement statistic is used directly.

B.5 Matching the empirical mean and RMS

For active positions I_k, let C_{tok} center over I_k and RMS_{I_k} compute the root-mean-square over active positions and width. If $\mathrm{RMS}(C_{\mathrm{tok}} U_k) > 0$, the unscaled projection $\Pi_k(U_k)$ in equation 14 satisfies

$$\mathsf{M}_{I_k}(\Pi_k) = \mu_k, \qquad \mathrm{RMS}_{I_k}(C_{\mathrm{tok}} \Pi_k) = \sigma_k. \tag{44}$$

Among all tensors sharing these two moments, the centered component uniquely maximizes alignment with $C_{\mathrm{tok}} U_k$ (by Cauchy–Schwarz). The outer strength η_T scales both moments, so the final correction A_k has mean $\eta_T \mu_k$ and centered RMS $\eta_T \sigma_k$. Inactive positions receive zero correction. If the centered direction has zero norm, the implementation retains only $\eta_T \mu_k$.

The affine endpoint condition in equation 31 now applies with $B_k = \Pi_k(U_k)$. Whenever $\langle C_k^*, \Pi_k(U_k) \rangle > 0$, a nonempty interval of positive η_T strictly improves the endpoint obtained from p_k alone.

To separate the two contributions, write $N_k = |I_k|$ and $\overline{C}_k^* = \mathsf{M}_{I_k}(C_k^*)$. When the centered target is nonzero, θ_k denotes the angle between $C_{\mathrm{tok}} C_k^*$ and $C_{\mathrm{tok}} U_k$. Orthogonality of the token-mean and centered components gives the exact decomposition

$$\begin{aligned} \langle C_k^*, \Pi_k(U_k) \rangle_F &= N_k \left\langle \overline{C}_k^*, \mu_k \right\rangle + \sigma_k \frac{\langle C_{\mathrm{tok}} C_k^*, C_{\mathrm{tok}} U_k \rangle_F}{\mathrm{RMS}_{I_k}(C_{\mathrm{tok}} U_k)} \\ &= N_k \left\langle \overline{C}_k^*, \mu_k \right\rangle + \sigma_k \sqrt{N_k D}\, \|C_{\mathrm{tok}} C_k^*\|_F \cos\theta_k. \end{aligned} \tag{45}$$

The first term depends on how well the precomputed mean μ_k aligns with the true correction's mean; the second depends on the angle θ_k between the centered correction direction and the centered target. Frozen reference trajectories set μ_k and σ_k; the current trajectory determines θ_k through U_k. The endpoint is strictly improved whenever $2 \langle C_k^*, \Pi_k(U_k) \rangle_F > \eta_T \|\Pi_k(U_k)\|_F^2$.

C Additional Experimental Details and Results

C.1 Evaluation protocol

Every matched generation comparison uses the same frozen checkpoint, initial latent and stochastic noise, base solver, time grid, sequence length, and number of model evaluations.

ELF uses its official random logit-normal grid. Its SDE noise scale is 2.0, 2.0, 1.5, and 1.0 at 8, 16, 32, and 64 NFE. The official self-conditioning scale is 3.0 for all evaluated NFE. Evaluation uses six

independent runs, each with 1,024 generations of length 1,024. Standard errors are computed across runs.

LangFlow uses its official time grid, obtained from quantiles of the proposal distribution between $1 - 10^{-5}$ and 10^{-5}. Each of six independent runs produces 1,024 generations per setting. OpenWebText samples have length 1,024 and LM1B samples have length 128.

ELF computes generative perplexity with frozen GPT-2 Large after applying its official filter to remove empty decoded samples, then retokenizes decoded text for unigram entropy. LangFlow computes entropy from generated token IDs and perplexity after decoding. Official and Untied Self-Conditioning configurations use the same evaluator convention in each comparison.

C.2 Pairwise LLM-judge evaluation protocol

The pairwise evaluation adapts the Arena-Hard-Auto v2 protocol (Li et al., 2025) for unconditional text generation. Every NFE in $\{8, 16, 32, 64\}$ produces 1,024 text pairs by matching official and Untied Self-Conditioning generations on the same initial latent (LangFlow on OpenWebText, 1,024-token sequences). Each pair is presented twice to the judge with the answer order exactly reversed, yielding 8,192 judgments.

The judge (DeepSeek V4 Pro, thinking enabled) receives neither method names nor NFE labels. It scores grammatical fluency, discourse coherence, semantic content, and degeneration artifacts, then returns a five-level verdict: $A \gg B$, $A > B$, $A = B$, $B > A$, $B \gg A$. Significant verdicts ($\gg$) contribute three binary observations to the Arena score; slight verdicts ($>$) and ties contribute one.

Pair-level outcomes merge the two games per pair: a pair is a candidate win if the weighted score exceeds 0.5, an official win if below 0.5, and a tie otherwise. The 95% confidence interval uses 10,000 paired bootstrap resamples. No judgments were excluded or reassigned.

C.3 Complete endpoint results

Tables 3 and 4 give the absolute GenPPL, uncertainty, and unigram entropy values corresponding to Figure 2.

Table 3: Untied Self-Conditioning improves generation quality across ELF model sizes on OpenWebText, with the largest gains at 8 NFE. We report GenPPL and unigram entropy H as mean $\pm$ standard error over six seeds, each with 1,024 samples.

		GenPPL ↓		H	
Model	NFE	Official	Ours	Official	Ours
ELF-B	8	70.63±1.73	43.90±2.66	5.233±.017	5.074±.011
	16	32.18±0.75	28.82±1.06	5.184±.008	5.165±.010
	32	23.69±0.27	21.03±0.19	5.149±.005	5.139±.005
	64	19.04±0.25	17.99±0.24	5.075±.006	5.097±.007
ELF-M	8	108.51±7.42	53.59±3.34	5.320±.030	5.185±.009
	16	38.71±1.45	33.11±1.65	5.307±.006	5.280±.008
	32	26.13±0.24	20.37±0.22	5.239±.004	5.198±.003
	64	21.68±0.21	18.49±0.15	5.181±.005	5.204±.005
ELF-L	8	104.90±5.34	58.39±3.71	5.308±.101	5.063±.126
	16	46.94±1.74	42.65±2.25	5.399±.005	5.376±.007
	32	30.07±0.30	23.41±0.34	5.342±.003	5.296±.005
	64	23.61±0.21	19.53±0.15	5.279±.004	5.284±.005

Table 4: Untied Self-Conditioning improves generation quality for LangFlow on OpenWebText and LM1B. We report GenPPL and unigram entropy H as mean $\pm$ standard error over six seeds, with 1,024 samples per seed. Sequences contain 1,024 tokens for OpenWebText and 128 for LM1B.

		GenPPL ↓		H	
Data	NFE	Official	Ours	Official	Ours
OWT	8	531.19±3.38	60.77±0.21	5.805±.001	5.317±.001
	16	234.48±0.84	44.32±0.11	5.686±.001	5.271±.001
	32	122.96±0.47	39.84±0.14	5.583±.001	5.290±.001
	64	79.31±0.27	38.62±0.12	5.497±.001	5.293±.002
	128	59.38±0.15	37.46±0.09	5.426±.002	5.305±.002
	256	48.61±0.16	35.51±0.08	5.362±.002	5.283±.002
LM1B	8	263.98±1.33	70.42±0.14	4.379±.001	4.229±.001
	16	157.14±0.64	59.23±0.18	4.349±.001	4.227±.001
	32	113.79±0.68	57.44±0.12	4.333±.001	4.247±.001
	64	94.30±0.30	58.20±0.06	4.322±.001	4.265±.001
	128	84.38±0.11	59.53±0.12	4.314±.001	4.275±.001
	256	79.29±0.19	60.72±0.14	4.309±.001	4.282±.001

Table 5: Word 4-gram repetition rate (word Rep-4) for LangFlow generations on OpenWebText and LM1B. Entries are mean $\pm$ standard error over six seeds.

Data	NFE	Official	Ours
OWT	8	$(0.11\pm0.01)\times10^{-3}$	$(1.90\pm0.05)\times10^{-3}$
	16	$(0.51\pm0.02)\times10^{-3}$	$(3.94\pm0.13)\times10^{-3}$
	32	$(1.54\pm0.06)\times10^{-3}$	$(5.56\pm0.20)\times10^{-3}$
	64	$(3.95\pm0.08)\times10^{-3}$	$(8.51\pm0.22)\times10^{-3}$
	128	$(8.38\pm0.19)\times10^{-3}$	$(12.04\pm0.35)\times10^{-3}$
	256	$(16.91\pm0.41)\times10^{-3}$	$(19.54\pm0.47)\times10^{-3}$
LM1B	8	$(0.07\pm0.01)\times10^{-3}$	$(0.48\pm0.02)\times10^{-3}$
	16	$(0.18\pm0.02)\times10^{-3}$	$(0.70\pm0.04)\times10^{-3}$
	32	$(0.34\pm0.04)\times10^{-3}$	$(0.74\pm0.05)\times10^{-3}$
	64	$(0.57\pm0.05)\times10^{-3}$	$(0.92\pm0.06)\times10^{-3}$
	128	$(0.81\pm0.05)\times10^{-3}$	$(1.14\pm0.09)\times10^{-3}$
	256	$(1.11\pm0.10)\times10^{-3}$	$(1.41\pm0.09)\times10^{-3}$

Figure 3(a) places the OpenWebText results next to published few-step systems. The external curves are taken from Hu et al. (2026): ELF+PD is a progressive-distillation model with additional training, and MDLM+SDTT, Duo+DCD, and FMLM are different model families evaluated under their own protocols. Without retraining, Untied Self-Conditioning on ELF-B and on LangFlow has lower plotted GenPPL than MDLM+SDTT, Duo+DCD, and FMLM at every shared NFE.

C.4 Precomputation and inference-time settings

The principal-angle directions for the latent and self-conditioning input projections are computed once from the frozen input weights. The per-step statistics in this subsection use only latent and prediction tensors. For ELF, finite differences measure the first-order model responses to perturbing the self-conditioning input and the prediction supplied to the solver. Their 2×2 Gram matrix determines the base transform exponent $\lambda_R^{(0)}$ and the reference correction scale $g_T^{(0)}$. For LangFlow, the full rank of the input projections gives uniform conditioning scaling (Appendix B.2). We express its coefficients in effective units as defined below. For ELF, denote the response Gram matrix by G, the total strength by s_0, and the relative solver weight by φ. Define $v = (1-\varphi, \varphi)^\top$ and $\mathbf{1} = (1,1)^\top$. The resulting scales are

$$\chi = \sqrt{\frac{\mathbf{1}^\top G \mathbf{1}}{v^\top G v}}, \qquad \lambda_R^{(0)} = s_0\chi(1-\varphi), \qquad g_T^{(0)} = s_0\chi\varphi. \tag{46}$$

We set $s_0 = \varphi = 0.75$, fixing the relative first-order response norm of the two interventions.

The empirical per-step statistics in equation 43 use independent trajectory banks. ELF combines four sets of 1,024 trajectories for each model and NFE. Disjoint halves estimate the token mean and average clipped cosine distance on one side and the centered RMS on the other. LangFlow uses 128 independent latents with a disjoint $64/64$ split. These quantities summarize the correction in the fixed reference recurrence. High-resolution step averages are used only in the separate diagnostics of Appendix C.7.

For LangFlow, let $b = 0.075$ be the base gain used in the reference recurrence and denote the resulting statistics by μ_k^{ref} and σ_k^{ref}. Table 6 uses

$$\begin{aligned} \eta_R &= b\,\eta_R^{\mathrm{raw}}, \quad \lambda_R^{(0)} = 1, \\ \eta_T &= b\,\eta_T^{\mathrm{raw}}, \quad \mu_k = \mu_k^{\mathrm{ref}}/b, \qquad \sigma_k = \sigma_k^{\mathrm{ref}}/b. \end{aligned} \tag{47}$$

These definitions preserve the transform exponent and final correction. The reference trajectories use $g_T^{(0)} = b$; the moments are rescaled after collection. The ELF coefficients multiply the base quantities computed above. Table 6 gives the coefficients used in the main evaluation; its LangFlow entries use the effective units defined above.

C.5 Correlated Gaussian corruptions

The experiment uses 256 fresh OpenWebText examples with frozen ELF-B at 8 and 16 NFE. For each example, standardized Gaussian innovations define $\epsilon_j = \rho\epsilon_{j-1} + \sqrt{1-\rho^2}\,\xi_j$ for $\rho \in \{0, 0.25, 0.5, 0.75, 1\}$, with a standard Gaussian initial noise. At query time t_j, the latent is $z_j = t_j x + (1-t_j)\sigma\epsilon_j$, where x is the clean embedding and σ is the denoiser noise scale. Each corruption keeps its Gaussian marginal as ρ varies.

The evaluation target is constructed using the original frozen model. Let v_{clean} be the clean velocity target, v_0 the prediction with zero conditioning, and v_{same} the prediction conditioned on a clean prediction computed from the same noisy latent, all expressed as velocities. At guidance scale g, the target is

$$y = v_{\mathrm{clean}} + (1 - g^{-1})(v_{\mathrm{same}} - v_0). \tag{48}$$

The original model is compared with a copy whose self-conditioning projection carries a frozen rank-16 low-rank update; all other weights are identical. The update was fit beforehand for 800 steps on OpenWebText to the same target y, on query states whose memory came from the model's predictions at earlier query times under independent Gaussian corruptions ($\rho = 0$), mixed with same-state, zero-memory, and decoder rows. Both models receive the same latent and time; each carries its own memory, obtained by evaluating earlier corruptions with that model and feeding its raw prediction into the next evaluation.

For predictions v_{base} and v_{modified}, define $d = v_{\mathrm{modified}} - v_{\mathrm{base}}$ and $e = v_{\mathrm{base}} - y$. The coefficient minimizing empirical MSE is

$$\alpha^*(\rho) = -\frac{\sum \langle e, d\rangle}{\sum \|d\|^2}, \qquad v(\alpha) = v_{\mathrm{base}} + \alpha d. \tag{49}$$

Coefficients zero and one correspond to the original and modified predictions. Table 7 reports the continuous optimum and the MSE ratio at coefficient one. The differences between the coefficients at $\rho = 0$ and $\rho = 1$ are 0.58 at 8 NFE and 0.43 at 16 NFE, with 95% intervals $[0.55, 0.62]$ and $[0.40, 0.46]$.

Controls using a clean prediction computed from the current noisy state have risk ratios near 1.02; controls without self-conditioning have ratios within 10^{-4} of 1. The quadratic risk decomposition is exact to within 10^{-8}, providing a numerical check of the coefficient calculation.

C.6 Comparing self-conditioning transforms

The experiment in Table 8 disables the solver correction and uses 256 paired ELF-B generations at 8 NFE. Every configuration supplies the raw prediction to the solver and stores it for the next

Table 6: Inference-time coefficients (Algorithm 1). Coefficients shared across consecutive rows are shown once.

System	NFE	η_R	η_T	η_O	α
ELF-B	8	2	2	1	$\frac{2}{3}$
	16		0.5		0.75
	32, 64		1		
ELF-M	8	2	2	6	0.75
	16		1		
	32		2	1	1
	64				0.75
ELF-L	8	2	2	1	0.5
	16		1		0.75
	32		2		1
	64	1			0.75
LangFlow OWT	8	2	2	1	1
	16, 32	1			
	64	2			0.75
	128, 256	1			
LangFlow LM1B	8	2	3	1	1
	16	1			0.875
	32, 64, 128, 256				0.75

Table 7: Effect of Gaussian noise correlation on a fixed modification of the conditioning projection in ELF-B. ρ is the noise correlation. The interpolation weight α^* minimizes MSE against the guided velocity target when combining the original and modified velocity predictions. The MSE ratio is modified divided by original.

	8 NFE		16 NFE	
ρ	α^*	MSE ratio	α^*	MSE ratio
0.00	0.88	0.92	0.77	0.92
0.25	0.79	0.94	0.69	0.94
0.50	0.62	0.97	0.55	0.98
0.75	0.45	1.01	0.40	1.05
1.00	0.30	1.07	0.35	1.11

Table 8: **Scaling aligned with projection overlap gives the lowest GenPPL.** We compare three assignments of the same scaling factors to input directions on ELF-B at 8 NFE, using 256 samples with the solver correction disabled. Alignment is the cosine between the change in conditioning and $-\nabla\mathcal{E}$. $\Delta\mathcal{E}/\,\mathrm{RMS}$ is the change in projection energy divided by the RMS of the change in conditioning.

Assignment	Alignment ↑	$\Delta\mathcal{E}/\,\mathrm{RMS}$ ↓	GenPPL ↓
Official	–	–	64.27
Overlap-aligned	**0.94**	-40.35**k**	**55.64**
Reversed	0.51	-22.94k	73.95
Random	0.74	-32.15k	59.59

model evaluation. The overlap-aligned, reverse, and random transforms have exactly the same set of eigenvalues and mean strength. Their only difference is the assignment of these eigenvalues to the principal directions.

For a self-conditioning input m, define

$$\mathcal{E}(m) = \frac{1}{2} \sum_{n \in I_k} \sum_i \rho_i^2 \langle m_n, q_i \rangle^2 , \tag{50}$$

where q_i is principal direction i. Table 8 reports the cosine similarity between the self-conditioning modification and $-\nabla\mathcal{E}$, the change in $\mathcal{E}$ divided by the modification RMS, and GenPPL. The overlap-aligned assignment gives the highest cosine similarity, the most negative normalized change in $\mathcal{E}$, and the lowest GenPPL.

Overlap alignment with solver correction. In a separate control, Table 9 tests whether the overlap-aligned transform still helps with a fixed history-based solver correction. The three transforms have the same retention eigenvalues but assign them to different input directions. The overlap-aligned assignment gives the strongest alignment with $-\nabla\mathcal{E}$, the largest reduction in projection energy per unit modification RMS, and the lowest GenPPL: 34.93, compared with 38.17 for the solver correction alone.

Table 9: Scaling aligned with projection overlap gives the lowest GenPPL with a fixed solver correction (ELF-B, 8 NFE; 256 paired generations). The three transforms use the same retention eigenvalues, assigned to different input directions. Alignment is the cosine between the change in conditioning and $-\nabla\mathcal{E}$; $\Delta\mathcal{E}/\,\mathrm{RMS}$ is the change in projection energy divided by the RMS of the change in conditioning.

Configuration	Alignment ↑	$\Delta\mathcal{E}/\,\mathrm{RMS}$ ↓	GenPPL ↓
Official SC, no solver correction	–	–	65.23
Official SC, solver correction	–	–	38.17
Overlap-aligned	**0.94**	-53.62**k**	**34.93**
Reversed	0.54	-32.52k	40.21
Random	0.75	-43.69k	36.92

Comparison with uniform scaling. We also compare the self-conditioning transform with uniform scaling on 256 fresh ELF-B samples at 16 NFE, with the solver correction disabled (Table 10). Overlap-aligned scaling gives lower GenPPL than both the official sampler and uniform scaling, with or without the common scaling factor in equation 9.

Table 10: Comparison of scaling by projection overlap and uniform scaling on ELF-B at 16 NFE. We use 256 samples with the solver correction disabled. The transform aligned with projection overlap multiplies the component along principal direction i by $\max(1 - a_k\rho_i^2, \epsilon_{\mathrm{fp}})$ before global scaling. We compare versions with and without global scaling. H denotes unigram entropy.

Configuration	GenPPL ↓	GPT-2 NLL ↓	H
Official	29.44	3.39	5.13
Overlap-aligned, no global scaling	**26.56**	**3.28**	5.06
Overlap-aligned, method scaling	27.42	3.32	5.08
Uniform scaling	29.90	3.40	5.13

C.7 Reference averages over one solver step

Subdividing each solver step into 32 substeps provides a high-resolution reference for the step-average prediction. Across 8, 16, and 32 NFE, the correction C_k^* scales as first order in step width (fitted exponent 1.04, 95% CI $[1.02, 1.05]$) and its contribution to the latent update scales as second order (fitted exponent 2.02, 95% CI $[2.01, 2.04]$), matching the predicted orders from Section 3.3. The EMA-based estimate from past predictions outperforms equal-RMS permutations.

The scaling experiment uses frozen ELF-B at 8, 16, and 32 NFE. Every setting contains 128 fresh trajectories. The first 64 fit one nonnegative coefficient for each estimate derived from prediction history; the remaining 64 evaluate normalized MSE and equal-RMS permuted controls. Confidence intervals use 4,000 sample bootstrap resamples.

C.8 WHERE TO APPLY THE SOLVER CORRECTION ACROSS CHECKPOINTS

The main experiment on where to apply the solver correction uses 1,024 matched ELF-B generations at 8 NFE. A second experiment applies the same fixed solver correction to five additional model and NFE settings with 64 samples each and no parameter changes. As shown in Table 11, applying the correction in the solver outperforms the official sampler on both GPT-2 Large NLL and OpenWebText bigram NLL. Adding the same correction to the next self-conditioning input raises the bigram NLL in all five settings.

Table 11: Correction placement across ELF model sizes and sampling budgets, using 64 paired samples per setting. Entries are NLL differences (nats/token) measured by GPT-2 Large and a bigram model trained on OpenWebText (OWT). Δ_{solver} subtracts official NLL from the NLL with correction only in the solver; negative values indicate lower NLL. $\Delta_{\text{next SC}}$ measures the additional NLL change when the corrected prediction is also used for conditioning; positive values indicate higher NLL.

Model	NFE	GPT-2 Δ_{solver}	OWT Δ_{solver}	OWT $\Delta_{\text{next SC}}$
ELF-B	16	-0.07	-0.16	$+3.00$
ELF-M	8	-0.38	-0.31	$+2.51$
ELF-M	16	-0.17	-0.18	$+3.45$
ELF-L	8	-0.27	-0.25	$+1.63$
ELF-L	16	-0.14	-0.14	$+2.75$

C.9 ADDITIONAL ABLATIONS

The remaining tables use ELF-B at 8 NFE with the official SDE noise scale 2.0 and one run of 1,024 matched generations per configuration.

Table 12: Effect of the correction direction on ELF-B at 8 NFE. The first four rows use the same conditioning transform and correction mean. The "Token mean only" variant omits the centered component; the permuted and random variants preserve its RMS. H denotes unigram entropy.

Solver correction	GenPPL ↓	H
History-based direction with mean/RMS scaling	**45.63**	5.14
Token mean only	57.21	4.98
History-based direction permuted across samples	49.27	4.99
Random centered direction, equal RMS	71.45	5.02
Official (no correction)	74.18	5.27

Table 12 changes the correction direction. A random centered direction with equal RMS nearly recovers official GenPPL, showing the benefit of the history-based direction. Removing the centered component raises GenPPL to 57.21; permuting it across samples gives 49.27, also above the full method's 45.63. The permuted and random controls preserve the correction's mean and centered RMS.

Table 13 compares matched samplers on ELF-B at 8 NFE. Untied Self-Conditioning reaches 45.63 GenPPL versus 56.90 for the HiGS (Sadat et al., 2026), 74.72 for Momentum (Wizadwongsa et al., 2024), and 74.18 for the official sampler. The EMA control uses our mean/RMS normalization, so its comparison with the full method measures the additional solver history and conditioning transform.

C.10 THROUGHPUT

Table 14 reports total samples per second and their ratio. The ratio ranges from 0.91 to 0.98, with geometric mean 0.96: the sampler retains over 95% of the baseline throughput without requiring additional model evaluations, since all auxiliary quantities are derived from tensors already available in the sampling loop.

Table 13: Sampler comparison on ELF-B at 8 NFE. We compare the official sampler, Momentum (Wizadwongsa et al., 2024), HiGS (Sadat et al., 2026), ACE (Zhang et al., 2026), and Untied Self-Conditioning using 1,024 generations from the same initial latents and time grid. Untied Self-Conditioning achieves the lowest GenPPL; H denotes unigram entropy.

Sampler	GenPPL ↓	H
Official	74.18	5.27
Momentum	74.72	5.28
ACE	115.43	5.43
HiGS	56.90	5.18
HiGS + ACE	65.84	5.24
Untied Self-Conditioning	**45.63**	5.14

Table 14: Sampling throughput of the official sampler and Untied Self-Conditioning. Retained is the throughput of Untied Self-Conditioning divided by that of the official sampler.

Model	NFE	Official (samples/s)	Untied Self-Conditioning (samples/s)	Retained
ELF-B	8	72.98	67.59	92.6%
	16	39.39	35.93	91.2%
ELF-M	8	22.20	21.47	96.7%
	16	11.78	11.38	96.6%
ELF-L	8	10.42	10.21	98.0%
	16	5.53	5.42	98.1%
			Geometric mean retained	**95.5%**

C.11 Per-sample quality on LangFlow

Table 15: Paired NLL comparison for LangFlow on OpenWebText, using GPT-2 Large. Generations are paired by initial latent across six seeds, with 1,024 samples per seed and NFE. "Higher NLL" counts pairs where Untied Self-Conditioning has higher NLL than the official sampler. Mean ΔNLL is the average NLL of Untied Self-Conditioning minus that of the official sampler, in nats per token.

NFE	Paired samples	Ours higher NLL	Fraction	Mean ΔNLL
8	6,144	0	0.00%	−2.1680
16	6,144	0	0.00%	−1.6658
32	6,144	0	0.00%	−1.1269
64	6,144	13	0.21%	−0.7194
128	6,144	111	1.81%	−0.4608
256	6,144	346	5.63%	−0.3140

Table 15 reports per-sample NLL comparisons on LangFlow on OpenWebText. At 8, 16, and 32 NFE, none of the 6,144 paired samples at each budget has higher NLL under Untied Self-Conditioning.

C.12 Illustrative text comparisons

We show one matched pair for each model family to illustrate the decoded behavior.

LangFlow–OpenWebText example. 64 NFE, $\Delta\text{NLL} = -1.66$. The Ours generation removes the repeated phrase "second place in second place" and sustains article-like segments for longer.

Official

<|endoftext|>

All of that in mind, of course, was that calling grand-peat–where Monaco start the Elite line at home–would improve to second place in second place overall in 2017. Ratings were spoiled from the first rankings, because Pabennis tried 19 points outrightly, and so far away to his best for competitive titles, but we quickly never predict the end of a steep, hot year within Giga Champions and one of this year's McFOR favourites.

So for traveling to Valencia the rest of the year–every, for example–that's my question of why, despite Sevan had so many promising moments in 2013, every player still felt the same cliche, wondrous pace about his own country, when in the Multiverse, it was Fernandez as a good roommate, not just my own wife and teenagers who had a dedicated, tactical approach.

For as long I went, the confidence I'd found throughout the entire season (that optimism happened now just as soon as I can imagine) was just real for the ideal year for anything. Valencia is there, we're ahead, and it still feels bad to see it coming yet quickly.<|endoftext|>DSP Spaces, the Jaaharabad Airways Ltd, is building the Qatar port and will connect it to Ahmedabad Airport for the cross-border services.

The port will begin at Mokintabad International Airport and will then connect to Panurbiba National Airport.

Although the airport will be closed, the port will also connect back to Disa, Ag-a-Palad and a public waychard known as Zone Na.

The Panurbiba Center will also serve the so-called Lid Ban log-airlets and outlets as part of the project.

The Lid Ban Center will also serve the conservation of water provided for the port's special water deck.

Besides the airport, DSP will move around the two-lane branches with an M/6 system and serve up to 2 hundred passengers. The campus or 251 passengers.

Commenting at the project, DSP Chairman, Gindas Kumar, said the project would be "donural home for Ahmedabad people."

"With the Qatar port terminal, Zone Na will be Ahmedabad's major link at the border. All investment investments will be made into this airport to make a much smoother environment for Indians," said Kumar.

"The port will be extended from Zone Na to the Ahmedabad Air location without further delays for shuttle operators to no services from Ahmedabad International Airport. This will be a good haven for noncoperative passenger services and the port will have a fast link to its Qatar border."

Zone Na Director of visual Group Joan Engmave told reporters that "The port, the arch from Mecca to Danda, was handed over to Qatar, and the frame will soon be out."

Refaj Airport, Ahmedabad Airport and Matinadh Airport are the new Ahmedabad airports to take advantage of the Qatar proper. The project is another off-use airport such as the Public Satellite airport and well-lands.<|endoftext|>Two weeks ago I waited for Nigel Farage's half-day residence near Queen Square in Goldstone Paul's, came on a passport "thatks our rise toward the independence vote" and sealed the decision to "born living Europeans that I didn't tell voters". "To assure you that this time round, or any other time of year, I must abide by the clear code of fact that I pass along for the visit," cracked alily. The passport had turned me over to the City. I stood confronting myself on Yorkshire Street, in Goldstone Paul's, suddenly mentioning Nigel Farage's MPs as if he had decided to call them "friends" in "Hab that Wh nearby". I was deeply informed of this distinction and certainly obliged to hypothesise about it from a source I would only suspect at Ukip if unspecified. It is still unsurprising, though, that the case of Ukip in general was derived from his previously made Southampton speculation. And there are some recent facts to examine.

But what Ukip's main milons, or the compassionate people themselves have told us about their spin, Nigel Farage's change implies an overt and strengthened leadership has almost nothing to do with people. Indeed it is true that many times net immigration left a devaluous stock of workers and brings in desperate people to reject it for access. The standards proposed by EU states concerning net immigration have made it clear that we should never let so many migrants to leave, because of the interests of the migrant and internally rather than the overseas taxpayers; which is precisely why it is common interest that Ukip's extreme leadership has then rightly sent us up into Europe. That past national day, in France, Mr Farage periodically holed himself and European countries<|endoftext|>

Ours

<|endoftext|>

All of that in mind, of course, was a telling no-brainer, because Monaco won the Champions League in 2014 and slipped back to second place in the Champions League in 2017. Ratings were spoiled from the first round, because the Bavarian crashed 19 points outrightly, and so nobody seemed to know what for sure, because instead they struggled to predict the end of a very, big year within Borussia Dortmund and one of the world's top F1 runners.

So for me anyway, the out of the year (2016, for example) wasn't my question of why, the Bavarian had so many great moments in 2016, every league move, the same flirting, the aerrolic event, etc, but for the Bavarian, it was at last a great day, not just my own wife and children who had a great, great experience.

Despite how close I was, the passion I'd seen throughout the entire season (that has happened here just as often as I can imagine) was just personal for the sake of the thing. Monaco is special, they're strong, and it would be bad to see it come true...<|endoftext|>Hamid Karzai, the Haqqani Taliban headliner, indicated that the United States would have to bury Mr. Karzai for the cease-fire talks.

The new peace agreement, an apparent move by Mr. Karzai to meet with the United States, has been in force after the disputed elections of Sunday. The military has also engaged in an effort to preserve Mr. Karzai's authority holding hold as early as Fallujah.

But on the Islamic State's announcement, Mr. Hamid Karzai, president of the American Parliament, said the U.N. "absolutelyprised" that the United States would have to intervene.

Photo

Mr. Karzai added that the U.N. announcement about the renewed peace talks should allow the United States to take further steps and exchange its obligations as well as with the United Nations.

Advertisement Continue reading the main story

Mr. Karzai, who had called the peace talks "stalled" after Afghan fighters had rallied hard to take the coup push against Mr. Karzai, said he didn't know if the United States was trading in or if Karzai was able to bury the peace agreements just after carrying out the reconciliation.

Yol. Karzai signed off a deal to rein Talibanate Mr. Karzai's forces to keep Mr. Karzai in power. But Mr. Karzai has also formally cred the agreement and says he won't sign a final peace agreement.

Newsletter Sign Up Continue reading the main story Please verify you're not a robot by clicking the box. Invalid email address. Please re-enter. You must select a newsletter to subscribe to. Sign Up You will receive emails containing news content , updates and promotions from The New York Times. You may opt-out at any time. You agree to receive occasional updates and special offers for The New York Times's products and services. Thank you for subscribing. An error has occurred. Please try again later. View all New York Times newsletters.

Mr. Karzai said the agreement was "a good step toward the peace process" and called the peace agreement "a step solely that I didn't talk about."

On a day that saw conditions in rural Afghanistan triple-levels of temperature, the United States assessed a clear line of terms and amred along to Mr. Karzai's presidency. The decision was turned head over to the United States. Heads of Peace on Sunday dismissed reports that Mr. Sharif's forces had asked Mr. Karzai for talks to prevent his departure. A statement on Sunday said the Afghan government had "confirmed that Mr. Karzai's forces had been in force."

Advertisement Continue reading the main story

Mr. Karzai issued a statement hours after he amred the agreement, rejecting the fact that although the talks had ended, "extraordinary changes had been made for the recent developments."

The Afghan government's main said that, although the United States would resume talks before the parties signed, the agreement's effect dissolved in October and urged Mr. Karzai to stop giving discussions. But it also insisted that the United States would reach a bargaining gap between the parties before talks in Kabul and to compromise its efforts to pass a peace decree by the end of escalating disputes.

"We will never really allow both sides to negotiate," Mr. Karzai's spokesman said. "You don't negotiate a bad relationship when there's clear security."

Mr. Karzai and his troops were, in fact, supporters of both the United States and NATO.<|endoftext|>

ELF-B example. 8 NFE, $\Delta\text{NLL} = -2.04$. The Ours generation starts with connected sentences about tank storage, while the official sample begins with fragmented tokens.

Official

In conclusion conclusion,,,,:, conclusion,. maxim maxim. >> din., din, din, din., am, >>,..,,.. >>,., din din.,,,,,, din. din din,., din...,, din.. summarize.,,,,.

Ours

In this article, a panel asked to determine the tank of the tank that is holding and the amount of tank that the tank is holding. Specifically the insights are determined by the size of the size of the tank tank, the location of the tank containers are holding and temperature, of the forced pressures of the tank held. The study is as described in the last part of the paper. In this article, again, calculations computes different features of tank storage, including 11 percent of the tank, half percent size included, 3/4 of the tank capacity, and the total amount of tank. If you want to use an increase in the tank, what is the total that you need? What can you use to determine how much tank you need? In this case, when you create a package so that once the maximum maximum capacity, the maximum can be estimated as 330 million USD or less because there are 20 storage stations to deliver the total, you will need to add an additional amount of storage per 20 storage containers to deliver the total. As example, if you create a platform tank, it will deliver the total of about 20 tanks all in a day. If you want to create the tank in a specific location, you would also need to create a platform tank. You have to have the tank that stores the containers holding the tank. The platform platform also has a locked pit which dumps the garbage and the device to hold out tank to the tank. This way the whole tank must be in a tight underground pit of the water, and then the tank tank must be filled in water so that it can run properly. As long as there is not enough water to hold the tank, the decay is required to get the perfect tank of the tank. But the major advantage of a platform holding is you use it to store the storage and water and to transporte the data flows to foreign systems with a specific storage point. First and foremost, you ensure that the platform storage is exactly that is fit for the system you need. Platform holding is a staple of easily defined and easily integrated into your platform platform. Once you have a platform platform ready, you can create your own platform platform. Platform holding brings in some of the most advanced storage technologies available, and it includes countless different features and technical details that you can use. There are plenty of apps that go to you of how you want to to launch your own platform platform, and you can use them often when you need platform holding. The fundamentals of platform holding are one of the fundamental principles of platform holding. The features of platform holding are easily defined and include many different elements, including popular software, programable design tools, intelligent design plugins, color charts, color tables, graphs, data, and customized background effects. They can be specially customized and implemented in custom graphics, some that adapt custom design features for customized applications, such as charts and maps. They can also be specially customized with apps used to create user-based playlists. If you want an tank that is a platform platform, there are features to be used. For example, I remember my reading an avatar book. I remember I reading the Oldboy game, when I was 16, and I was screaming and the kid started to think and I was kill me, and then he pulled her on the floor and shot me down. I didn't want to see her, but I was 13, and I didn't want to look at that much. So I'm shut to sleep, and waste my life fighting and take the life, and let her go. I mean, she's really insane, and I can't stand her. And then I'm all out of love! So this is a great comic book in my life, and I have so much to share. I feel really wasted for reading, too. While there is still more to read, and I'm getting really excited, I'll make some some cool ideas that I find worth reading and reading. If you'd like to share your insight, tell us your thoughts and inspiration, or let us take part. The chance to explore it today, you we'll get to explore all of the beautiful locales where you'll want to hang together, walk the streets, walk the streets, and enjoy amazing games and competitions. Next, you'll have a new sport, our global video game, where you can raise your children, inspire family and friends, and show their faith and heart. In great respect to Andrew Berger, Tettarian anscientist, is a surgid and author of writing the Prophet Creid and War Naraous Abiql Abiq is a Jewish man who has been out of his own home in Iraq-occupied Islamic Arabia, for for the purpose of the killings of the Taliban while he undertakes the military operations in the nearby Iraq Arab militants forces.