DESIGNING TRANSFORMERS FOR CONTINUOUS DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

What should a Transformer architecture preserve, learn, and feed back when language is generated by continuous denoising? In embedding-space diffusion, the denoiser connects a continuous corruption process to categorical prediction, while its outputs determine both state updates and auxiliary inputs. We study architecture through these distinct computational roles and formulate three design principles: separate feature-scale control from noise-dependent modulation; separate the forward weighting of geometric token evidence from its gradients to the codebook; and distinguish clean-state estimation from self-conditioning. We instantiate these principles in a LangFlow-style model through normalization, shortcut gradient routing and calibration, and noise and confidence conditioning. Elementary analyses make the scope of the design explicit: normalized attention limits its dependence on raw query–key magnitudes, the Gaussian observation model explains the analytic shortcut, and shrinking a posterior mean need not improve clean-state estimation. The architecture retains the baseline representation, objective, and inference sampler. A seven-stage OpenWebText-1024 ablation provides the core experimental test, alongside comparisons with FLM, RePlaID, LangFlow, and ELF implementations. Mechanism controls and transfer evaluations distinguish improvements to one implementation from evidence for a reusable CDLM architecture.

1 INTRODUCTION

Continuous diffusion offers a way to model language through the progressive refinement of real-valued representations. Recent models make different choices about the representation, objective, and dynamics, ranging from one-hot flows to learned token embeddings and latent spaces (Lee et al., 2026; Yang et al., 2026; Chen et al., 2026; Hu et al., 2026; Meshchaninov et al., 2026; Guo et al., 2026; Liang et al., 2026). These choices define the generative process. The denoising architecture determines how that process is learned and how token information is recovered along its trajectory. A central architectural question is therefore: *which computations should be shared across noise levels, and which quantities should have separate roles in prediction, optimization, and iterative refinement?*

In embedding-space language diffusion, the Transformer is part of a coupled system. The learned codebook defines the clean states; the noisy states provide evidence about token identity; and the predicted token distribution determines a continuous estimate used by the solver. Self-conditioning reuses this prediction as an input to subsequent computation. Consequently, a design choice at the vocabulary head can also affect representation learning, and a change to a feedback signal can alter what the denoiser learns to rely on. Understanding these interactions is necessary for developing an architecture that can serve as a reusable starting point for CDLM research.

We organize this problem around three distinctions. The first is between *feature scale and noise information*. Magnitude variation in shared attention projections need not be the only way to communicate corruption level; an explicit conditioning pathway can supply noise-dependent modulation. The second is between *using a representation and updating it*. An analytic token-likelihood branch can contribute to the forward prediction without receiving the same gradient paths as the contextual network. The third is between *estimating a clean state and constructing an auxiliary input*. A pos-

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.

terior mean is a natural continuous estimate, but the decision to expose that estimate to a learned conditioning pathway is a separate architectural choice.

These distinctions lead to a concrete design. Backbone and query–key RMSNorm control feature and attention-logit magnitudes, while a wider noise embedding increases modulation capacity. At the vocabulary interface, stop-gradient removes the shortcut's direct codebook update, and a bounded multiplier adjusts its forward contribution. An entropy-based gate controls the strength of bootstrap self-conditioning during training, without changing the posterior-mean estimate used by the inference sampler. The resulting model uses familiar building blocks, but makes their different roles explicit and individually testable.

We connect these choices to elementary properties of the denoising computation. The Gaussian observation model explains the shortcut inherited from Plaid and LangFlow (Gulrajani & Hashimoto, 2023; Chen et al., 2026), while the chain rule specifies the codebook gradient removed by detachment. Conditional-mean identities distinguish the role of an embedding estimate from that of a confidence-weighted input. This analysis provides precise questions for the experiments: how forward weighting and gradient routing interact, and whether confidence association matters beyond generic attenuation.

Our study builds on prior architectural work. RePlaID aligns a likelihood-based model with modern diffusion backbones (Yang et al., 2026); latent and decoding models examine representation learning and token recovery (Meshchaninov et al., 2026; Shen et al., 2026; Du & Ma, 2026). We contribute a role-based design framework, a concrete architecture within a fixed LangFlow-style formulation, and a sequence of controlled tests. The seven-stage ladder measures incremental gains, while mechanism controls and transfer evaluations test the broader architectural claims.

OpenWebText generation at sequence length 1024 provides the core testbed, including comparisons with local FLM, RePlaID, LangFlow, and ELF implementations. Establishing a reusable backbone further requires matched-capacity controls, variation across training seeds, transfer across model sizes and compatible formulations, and quality–cost comparisons. These evaluations are specified separately from the existing ladder. Our longer-term objective is an architectural foundation for CDLMs; the present model is a concrete candidate whose generality must be established through those tests.

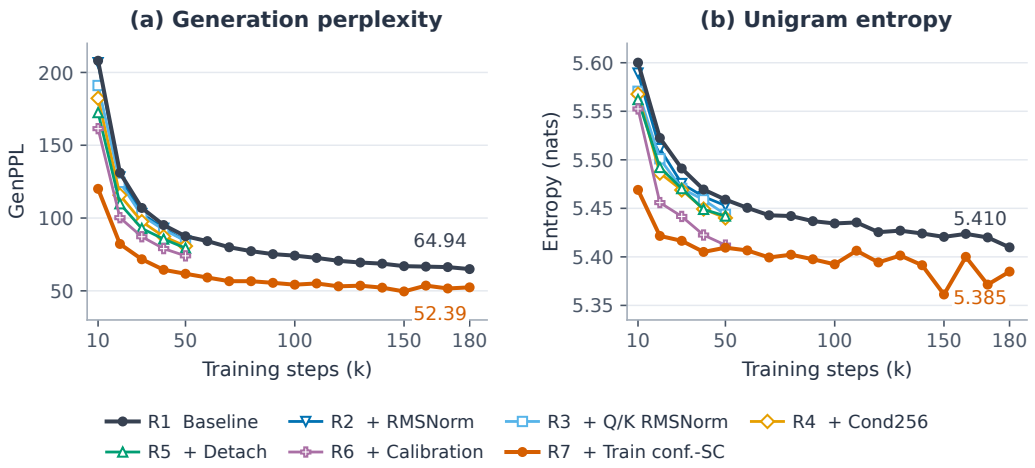

Figure 1: **Architectural ablations during training on OWT-1024.** GenPPL (left) and mean per-sample unigram entropy (right) at 128 NFE, using 1,024 samples and EMA weights. R1 (baseline) and R7 (full model) are shown through 180k updates; R2–R6 are shown through 50k. Points are unsmoothed logged measurements; endpoints are labeled at 180k. Legend additions are cumulative, with wider conditioning preceding shortcut detachment (Table 4). Lower GenPPL is accompanied by lower unigram entropy; these inline evaluations do not replace the final-checkpoint comparison.

2 Related Work

Our focus is the denoiser and its interfaces: how continuous observations, token predictions, learned representations, and feedback interact. This is complementary to advances in diffusion objectives, latent spaces, and sampling algorithms.

Continuous language diffusion and flow. Diffusion-LM introduced continuous denoising of word vectors for controllable generation (Li et al., 2022), and self-conditioned embedding diffusion extended this approach to language modeling (Strudel et al., 2022). CDCD formulated diffusion for categorical data (Dieleman et al., 2022), while Plaid developed likelihood-based training and studied compute scaling (Gulrajani & Hashimoto, 2023). More recent methods explore alternative representations and dynamics. FLM distills token-space flows into flow maps (Lee et al., 2026); LangFlow combines learned embeddings with information-based noise scheduling (Chen et al., 2026); and RePlaID modernizes Plaid's architecture (Yang et al., 2026). Hyperspherical flows constrain the dynamics to a sphere (Deschenaux & Gulcehre, 2026), whereas ELF performs flow matching over embeddings followed by discrete decoding (Hu et al., 2026). Our work studies the denoiser architecture and its interfaces while retaining a fixed LangFlow-style formulation.

Latent representations and the decoding interface. Several approaches modify the interface between continuous states and discrete tokens. CoDAR uses a contextual autoregressive decoder (Shen et al., 2026). LDLM jointly trains an encoder, denoiser, and decoder while regulating their gradient interactions (Meshchaninov et al., 2026). Cola-DLM combines a Text VAE with a continuous latent prior and conditional decoding (Guo et al., 2026), and AURORA-LM retains high-capacity latents while restricting the noisy-input pathway (Liang et al., 2026). Analysis of ELF further links token recovery to decoder margins and denoising trajectories (Du & Ma, 2026). Our modifications act within the existing codebook and prediction head: detachment changes a gradient path, and calibration changes the contribution of the analytic shortcut.

Backbone normalization and conditioning. DiT introduced adaptive-normalization Transformer blocks for diffusion (Peebles & Xie, 2023), and EDM2 examined magnitude control in image-denoiser training (Karras et al., 2024). RMSNorm normalizes feature scale without re-centering (Zhang & Sennrich, 2019), while nGPT normalizes both representations and weights (Loshchilov et al., 2025). Q/K normalization addresses attention-logit scale (Henry et al., 2020); related work studies attention-entropy collapse (Zhai et al., 2023) and uses Q/K RMSNorm in rectified-flow Transformers (Esser et al., 2024). We adopt established normalization operations and examine their incremental effects in continuous language denoising, together with the capacity of noise-dependent modulation.

Self-conditioning and trajectory consistency. Analog Bits demonstrated the benefit of self-conditioning for continuous diffusion of discrete data (Chen et al., 2022). Embedding diffusion and LangFlow likewise condition the denoiser on previous predictions (Strudel et al., 2022; Chen et al., 2026). AURORA-LM uses a trajectory-consistency objective to address differences between independently corrupted training states and iterative inference (Liang et al., 2026). Our confidence gate operates on the auxiliary bootstrap input during training, preserving the baseline loss.

Discrete diffusion and generation evaluation. MDLM simplifies masked diffusion training (Sahoo et al., 2024), while LLaDA and iLLaDA scale bidirectional masked denoising to large language models (Nie et al., 2025; 2026). Duo connects uniform-state discrete diffusion to an underlying Gaussian process (Sahoo et al., 2025). Comparisons across diffusion families show that likelihood rankings can differ from rankings based on sampling quality and cost (Sahoo et al., 2026). Generative perplexity can also decrease as a model loses distributional coverage (Franca & Tong, 2026). We therefore evaluate generated text using a common evaluator and reference set, reporting perplexity together with diversity, repetition, MAUVE (Pillutla et al., 2021), and sampling budget.

3 A Role-Based Architecture for Continuous Denoising

We view the denoiser as a shared Transformer with three architectural interfaces: noise-dependent computation, vocabulary prediction, and self-conditioning. Table 7 summarizes the design princi-

ples and their implementation. Our baseline, denoted A009, is a local LangFlow-style model (Chen et al., 2026). We preserve its generative formulation and use the resulting variants to test the proposed design, rather than attributing changes to a different objective or sampler.

3.1 Overall Architecture

Figure 2 summarizes Improved CDLM, the final architecture used in this study. The denoiser receives a sequence of noisy token embeddings, a noise-level embedding, and an optional self-conditioning estimate. A bidirectional Transformer processes the sequence, and a learned vocabulary head produces contextual token logits. An analytic shortcut adds token evidence computed directly from the noisy state and the codebook. The resulting distribution defines a clean embedding estimate for the next denoising step. The exact experiment identifier, including the `_250k` training-budget suffix, is given in Appendix C.

The architecture has 12 Transformer blocks, hidden width 768, 12 attention heads of dimension 64, and feed-forward width 3072. Relative to A009, the complete model uses backbone and query–key RMSNorm, a detached and calibrated shortcut, a 256-dimensional noise-conditioning vector, and confidence-weighted self-conditioning inputs during training. These choices alter how signals are normalized, combined, and differentiated, while preserving the Transformer depth, hidden width, diffusion representation, and inference procedure.

Forward and feedback pathways. Figure 2(a) expands a single Transformer block within the complete denoiser, read from bottom to top; $C = s\widetilde{E}$ abbreviates the clean-embedding table. The input projection forms $h_0 = z_\gamma + P[z_\gamma; x_{\mathrm{SC}}]$, where P is a zero-initialized, bias-free linear map from 1536 to 768 dimensions. Each block applies RMSNorm and noise-dependent modulation before its attention and feed-forward sublayers, with a separate gated residual connection around each sublayer. Within attention, queries and keys are RMS-normalized before rotary position encoding; values bypass these two operations. After the 12 blocks, an RMS-normalized, noise-modulated head maps hidden states to vocabulary logits using weights distinct from C. The evidence branch instead computes $z_\gamma\,\mathrm{sg}[C]^\top$, weighted by $a_\psi(\gamma)c_{\mathrm{skip}}(\gamma)$. The two branches are added before softmax. Stop-gradient affects only the explicit codebook factor of this branch: C remains trainable through the construction of z_γ and the resulting denoising loss.

Figure 2(b) separates the training and sampling rules for self-conditioning. During training, a bootstrap call to the same complete denoiser, with zero self-conditioning and without gradient tracking, supplies a distribution p_{boot}. Two branches compute its posterior mean $\mu = p_{\mathrm{boot}}C$ and token-wise entropy confidence $w(p_{\mathrm{boot}})$; their product becomes the self-conditioning input for the main call. Both calls use the same noisy state and noise level. During sampling, no bootstrap call or confidence gate is added: the current unweighted posterior mean is used by the solver and passed to the next denoising step. The first sampling step uses zero self-conditioning. This distinction is part of the specified architecture, rather than an inference-time variant chosen after evaluation.

Diffusion Transformer block. A 256-dimensional sinusoidal encoding of the noise level γ passes through a $256 \to 256 \to 256$ MLP with an intermediate SiLU, followed by a second SiLU, to produce c_γ. This vector feeds the Transformer blocks, the vocabulary head, and the shortcut calibrator. Following DiT (Peebles & Xie, 2023), each block maps it to shifts, scales, and residual gates for the attention and feed-forward branches. For a normalization operator N, the block has the form

$$u = h + g_a(c_\gamma) \odot \mathrm{Attn}(\mathrm{mod}(N(h); c_\gamma))\,, \tag{1}$$

$$h' = u + g_m(c_\gamma) \odot \mathrm{MLP}(\mathrm{mod}(N(u); c_\gamma))\,. \tag{2}$$

Here, mod is condition-dependent affine modulation, and g_a and g_m are residual gates. The baseline uses bias-free LayerNorm for N; the complete model uses RMSNorm and additionally normalizes queries and keys inside attention. Both use rotary position embeddings, fused QKV projections, GELU feed-forward layers, and zero-initialized adaptive-normalization projections. A separate zero-initialized projection incorporates self-conditioning from the concatenation of the noisy state and the estimated clean embedding.

3.2 Embedding Diffusion and Token Prediction

Continuous representation. Let $\mathcal{V}$ denote the vocabulary and $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ a learned codebook. Each token $y \in \mathcal{V}$ is represented by an embedding normalized to norm $\sqrt{d}$ and scaled by $s = 0.1$:

$$\widetilde{E}_y = \sqrt{d}\frac{E_y}{\|E_y\|_2}, \qquad x_0(y) = s\widetilde{E}_y. \tag{3}$$

The forward process corrupts the clean embedding with Gaussian noise. Parameterizing noise level by the log noise-to-signal ratio γ, we define

$$\alpha_\gamma^2 = \mathrm{sigmoid}(-\gamma), \qquad \sigma_\gamma^2 = \mathrm{sigmoid}(\gamma), \qquad z_\gamma = \alpha_\gamma x_0(y) + \sigma_\gamma \epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \tag{4}$$

A learned Gumbel schedule determines the distribution of γ, following LangFlow's information-based scheduling approach (Chen et al., 2026). All variants retain the baseline token-level cross-entropy objective and entropy-matching loss for the schedule. Self-conditioning is used on 25% of training examples, with a bootstrap prediction computed without gradient tracking (Chen et al., 2022; Strudel et al., 2022).

Vocabulary prediction. The baseline combines a learned prediction head with an analytic shortcut:

$$\ell_\theta(z_\gamma, \gamma) = \ell_{\mathrm{raw},\theta}(z_\gamma, c_\gamma) + c_{\mathrm{skip}}(\gamma)\, z_\gamma (s\widetilde{E})^\top, \qquad c_{\mathrm{skip}}(\gamma) = \frac{\alpha_\gamma}{\sigma_\gamma^2}. \tag{5}$$

For the equal-norm codebook and Gaussian corruption, the shortcut is the tokenwise observation log-likelihood up to a token-independent constant. This parameterization follows Plaid and LangFlow (Gulrajani & Hashimoto, 2023; Chen et al., 2026). Its contribution is linearly warmed up over the first 5,000 optimization steps. Section 3.4 studies how to use and learn this existing likelihood branch.

Sampling. Generation starts from Gaussian noise and uses the baseline Euler–EDM sampler in embedding space, with the EDM noise-to-signal parameterization (Karras et al., 2022). At each non-final step, the predicted distribution $p_\theta = \mathrm{softmax}(\ell_\theta)$ gives the clean embedding estimate

$$\widehat{x}_0 = \sum_{v \in \mathcal{V}} p_\theta(v)\, s\widetilde{E}_v, \tag{6}$$

This estimate determines both the solver update and the self-conditioning input to the next step. All variants retain this inference procedure, including the model trained with confidence-gated self-conditioning. The final prediction is decoded by token-wise argmax.

3.3 Feature Scale and Noise-Dependent Computation

Noise level changes the statistical problem presented to the denoiser, but it need not determine attention strength through uncontrolled feature norms. Our first principle is to give scale control and noise adaptation explicit, complementary roles. Normalization constrains magnitudes at shared projections; the conditioning vector supplies learned shifts, scales, and gates. This follows the broader motivation of magnitude control in diffusion and attention training (Karras et al., 2024; Zhai et al., 2023), without assuming that all norm information is irrelevant.

Backbone RMSNorm. LayerNorm subtracts the feature mean before normalizing magnitude. To examine whether this re-centering is needed in embedding-space denoising, we replace the two pre-normalization layers in each Transformer block and the normalization layer in the vocabulary head with RMSNorm (Zhang & Sennrich, 2019):

$$\mathrm{RMSNorm}(x) = g \odot \frac{x}{\sqrt{d^{-1}\sum_{j=1}^{d} x_j^2 + \varepsilon}}, \qquad \varepsilon = 10^{-6}. \tag{7}$$

The mean-square reduction is computed in FP32, and the output is cast to the projection dtype. Both normalization layers use a learned gain without a bias, so the replacement preserves the parameter count. RMSNorm rescales the input without removing its mean component. The ablation tests the effect of this change while retaining the baseline adaptive modulation.

Query–key RMSNorm. Normalizing the input to attention does not directly constrain the magnitudes produced by the learned query and key projections. In dot-product attention, changes in $\|q\|\|k\|$ alter logit scale independently of directional similarity. We therefore apply RMSNorm to each 64-dimensional query and key vector after the fused QKV projection and head reshape, before rotary position embeddings:

$$\widehat{q}_{i,h} = \mathrm{RMSNorm}(q_{i,h}), \qquad \widehat{k}_{i,h} = \mathrm{RMSNorm}(k_{i,h}). \tag{8}$$

Queries and keys have separate learned gains, each shared across heads within a layer; values are unchanged. The design follows QK normalization (Henry et al., 2020) and its RMSNorm implementation in rectified-flow Transformers (Esser et al., 2024). For head dimension d_h and gain vectors g_q, g_k, the normalized dot-product logit obeys

$$\left| \frac{(R_i\widehat{q})^\top (R_j\widehat{k})}{\sqrt{d_h}} \right| \le \sqrt{d_h}\, \|g_q\|_\infty \|g_k\|_\infty, \tag{9}$$

where R_i, R_j are the orthogonal rotary-position transformations. The bound removes raw input norms from the right-hand side; it does not fix a temperature or bound learned gains over training. Appendix A.1 gives the derivation. Whether this scale control improves optimization or generation remains the role of the ablation and attention diagnostics.

Explicit noise adaptation. Normalization does not replace knowledge of the noise level. We retain adaptive modulation throughout the network and increase its conditioning width from 128 to 256, as detailed in Section 3.5. The design hypothesis is that feature-scale control and sufficient modulation capacity work together. A normalization-by-conditioning control is needed to distinguish complementarity from the separate effects of the two changes.

3.4 Shortcut Gradient Routing and Calibration

The analytic shortcut provides observation evidence without passing through the Transformer. For fixed equal-norm embeddings and conditionally independent Gaussian corruption, posterior log odds decompose into a tokenwise observation term and a term conditioned on the other noisy positions (Appendix A.2). This explains the shortcut already present in the baseline. The learned head, however, sees the full noisy sequence; it is not constrained to represent only the context term. The two logit branches therefore have a useful architectural interpretation without being statistically identifiable.

Our second principle is to control two decisions separately: how much geometric evidence enters a prediction, and which gradients update the codebook that supplies it. The forward likelihood identity alone does not resolve either decision for a finite, jointly trained predictor.

Shortcut gradient detachment. We stop gradients through the codebook in the shortcut branch:

$$\ell_\theta = \ell_{\mathrm{raw},\theta} + c_{\mathrm{skip}}(\gamma)\, z_\gamma\big(s\ \mathrm{sg}[\widetilde{E}]\big)^\top. \tag{10}$$

Here, sg denotes stop-gradient. The operation leaves the forward logits unchanged for fixed parameters and adds no parameters. It removes the gradient through the shortcut's codebook factor, without freezing the codebook or blocking gradients through its other uses, including the construction of the noisy state. Appendix A.3 writes this removed contribution explicitly. Detachment therefore changes the optimization graph while preserving the forward model class. Its value depends on whether that gradient contribution is useful; the existence of a shared codebook does not itself establish gradient conflict. Related latent models also regulate representation–denoiser gradient interactions (Meshchaninov et al., 2026).

Learned shortcut calibration. Detachment changes how the shortcut is learned, but not its forward weighting. The analytic coefficient depends only on the corruption process, whereas the combined logits also depend on a learned contextual head. To let the model adjust the relative contribution of these two sources, we multiply the shortcut coefficient by a bounded function of noise level:

$$a_\psi(\gamma) = \exp\big(\mathrm{clip}\big(f_\psi(c_\gamma), -\log 2, \log 2\big)\big), \tag{11}$$

$$\ell_\theta = \ell_{\mathrm{raw},\theta} + a_\psi(\gamma) c_{\mathrm{skip}}(\gamma)\, z_\gamma\big(s\ \mathrm{sg}[\widetilde{E}]\big)^\top. \tag{12}$$

The function f_ψ is a $256 \to 32 \to 1$ MLP with a SiLU activation, using the wider conditioning vector described in Section 3.5. Its final layer is zero-initialized, giving $a_\psi = 1$ and preserving the uncalibrated shortcut at initialization. The interval $[0.5, 2]$ limits departures from the analytic coefficient. Calibration changes the forward parameterization; detachment changes its direct codebook gradient.

An unrestricted learned head could absorb a change in shortcut weighting. Calibration is therefore a structured adjustment for a finite model, not a claim that the Bayes likelihood coefficient is incorrect or that a larger function class is always required. Comparisons with a learned constant multiplier and a detach-by-calibration control test the noise dependence and interaction of these choices.

3.5 Conditioning Capacity and Uncertainty-Aware Feedback

The noise embedding identifies the corruption level, while self-conditioning supplies a previous prediction. The former supports the scale-control principle in Section 3.3. For the latter, our third principle is to distinguish the state estimate required by the solver from an auxiliary signal used to train the denoiser.

Noise-conditioning capacity. The conditioning vector supplies shifts, scales, and gates to all 12 Transformer blocks and modulation parameters to the vocabulary head. We increase its width from 128 to 256, changing the timestep MLP from $256 \to 128 \to 128$ to $256 \to 256 \to 256$. All adaptive-normalization projections receive the wider vector. This increases the capacity for noise-dependent modulation without changing the Transformer hidden width or feed-forward dimensions. The change adds 7.36M parameters, or 2.25% of the baseline; its ablation therefore includes a capacity increase and is not a parameter-matched test of conditioning alone.

Confidence-gated training inputs. Standard self-conditioning uses the posterior mean embedding in Equation 6. For a predicted distribution p, let $\mu(p) = \sum_v p(v) s\widetilde{E}_v$. Under the equal-norm codebook, the embedding variance under p is $s^2 d - \|\mu(p)\|_2^2$ (Appendix A.4). This quantity depends on codebook geometry as well as token probabilities. Predictive entropy alone is therefore not an exact measure of uncertainty in the continuous estimate.

We nevertheless use normalized entropy as a simple, parameter-free confidence proxy for the auxiliary training input:

$$w(p) = \mathrm{clip}\left(1 - \frac{H(p)}{\log|\mathcal{V}|}, 0, 1\right), \qquad x_{\mathrm{SC}} = w(p) \sum_{v \in \mathcal{V}} p(v) s\widetilde{E}_v. \tag{13}$$

Here, $H(p) = -\sum_{v \in \mathcal{V}} p(v) \log p(v)$. The gate is zero for a uniform distribution and one for a one-hot distribution. It adds no parameters and is applied token-wise to the bootstrap prediction computed without gradients on self-conditioned training examples. The purpose is to vary the strength of an auxiliary input, not to replace the supervised target or the sampler's clean-state estimate.

The distinction between estimation and conditioning is substantive. Define $\mathcal{R}_p(a) = \mathbb{E}_{Y \sim p} \|s\widetilde{E}_Y - a\|_2^2$. Replacing $\mu(p)$ by $w\mu(p)$ changes this squared-error risk by

$$\mathcal{R}_p(w\mu(p)) - \mathcal{R}_p(\mu(p)) = (1 - w)^2 \|\mu(p)\|_2^2. \tag{14}$$

Thus shrinkage cannot improve mean-squared error under that same distribution. For a learned p, this is not a guarantee about risk under the data distribution. The gate instead tests whether weakening uncertain bootstrap inputs helps train the denoiser.

At inference, the designated combined model uses the unweighted mean for both the Euler–EDM update and self-conditioning. A low-entropy prediction can be wrong, and attenuation can remove useful information. Controls that preserve mean gate strength but remove its association with confidence are needed to attribute a gain to uncertainty awareness rather than generic input shrinkage.

3.6 Architecture Variants

We evaluate the complete design through seven cumulative configurations: A009, followed by backbone RMSNorm, Q/K RMSNorm, increased conditioning width, shortcut detachment, learned calibration, and training-time confidence gating. Each row of Table 2 adds one modification to the

preceding configuration, with a separate training run from scratch. The final row defines IMPROVED CDLM, the architecture in Figure 2, with 334.45M parameters and a 250,000-update training budget.

4 EXPERIMENTS

The empirical study separates three questions: whether the complete architecture improves generation, which interventions explain any improvement, and whether the design transfers beyond the original configuration. The OWT-1024 system comparison and seven-stage ladder form the core experiment. Mechanism controls and broader transfer tests extend this evidence; their uncompleted results remain explicitly separate.

4.1 EXPERIMENTAL SETUP

Data and tokenizer. We train on OpenWebText using the baseline's GPT-2 boundary normalization and document-level BOS/EOS markers. The data pipeline includes packing, shuffling with a 10,000-document buffer, and random cropping to 1024 tokens. All ablation models use the Qwen3 tokenizer with a vocabulary of 151,936 tokens and are initialized from scratch.

Optimization. Each ablation variant is trained for 250,000 optimizer steps with a global batch size of 512. We use AdamW with weight decay 0.01 and gradient clipping at 1.0. The learning rate increases to 3×10^{-4} over 5,000 warmup steps, then decays to 3×10^{-5} following a cosine schedule. Training uses bfloat16, dropout 0.1, random seed 42, and an exponential moving average (EMA) of the weights with decay 0.999. All seven variants share the data order, objective, optimization schedule, and sampler. Ablation comparisons use EMA checkpoints from the same training step; external baselines retain their method-specific configurations.

Evaluation. The primary evaluation uses 5,120 unconditional generations of 1024 model tokens and a fixed reference set of 5,120 OWT samples drawn with seed 12,345. For each method, we report its sampling budget in network function evaluations (NFE), explicitly accounting for additional decoding calls. The controlled ablation uses 1024 denoiser evaluations, temperature 1, and final argmax decoding. A secondary comparison at 128 NFE assesses sensitivity to the sampling budget.

4.2 ARCHITECTURAL ABLATION

Table 2 presents a single seven-stage cumulative ladder, with the measured 50k-step GenPPL. Each variant is trained from scratch and adds one modification to the preceding configuration. Data order, training objective, and optimization schedule are shared; checkpoint step and sampler are matched within each evaluation setting. RMS denotes backbone and final-head RMSNorm; QK denotes query–key RMSNorm; Detach denotes stop-gradient on the shortcut codebook; Calib. denotes the learned shortcut multiplier; and Train Conf.-SC denotes training-time confidence weighting. Wider conditioning is introduced at R4 and shortcut detachment at R5. Adjacent rows measure incremental effects conditional on the components already present.

4.3 ANALYSIS OF THE ARCHITECTURAL CHANGES

The following diagnostics are planned to test the explanations in Section 3. They provide evidence about model behavior beyond the final generation scores.

Feature and attention scale. For the baseline and the two normalization variants, we examine residual activation RMS, query and key norms before and after normalization, attention entropy, and learned gain magnitudes. Measurements are grouped by layer and noise-level quantile. The comparison must distinguish removal of raw-norm dependence from growth in learned gains. A factorial control crossing Q/K normalization with conditioning width tests the proposed complementarity between scale control and noise adaptation.

5 CONCLUSION

We have developed a role-based approach to Transformer design for continuous diffusion language models. The architecture separates feature-scale control from noise modulation, forward short-

Table 1: Published language-modeling results from Table 2 of Chen et al. (2026), arXiv v3. All values are reported by that source, not measured in this study. PPL denotes validation perplexity (an upper bound for diffusion models), not GenPPL. A dash denotes an unreported value. ‡ preserves the source's marker for baselines retrained by the LangFlow authors.

	LM1B ($L = 128$)		OWT ($L = 1024$)	
Model	GenPPL↓	PPL↓	GenPPL↓	Entropy
Autoregressive				
Transformer	66.7‡	22.8‡	35.9	17.5
Absorbing-state diffusion				
D3PM Absorb	–	76.9	–	–
DiffusionBert	–	63.8	–	–
SEDD Absorb	115.9‡	32.0‡	–	–
MDLM	103.9‡	31.0‡	104.9	–
Uniform-state / continuous diffusion				
D3PM Uniform	–	137.9	–	–
Diffusion-LM	–	118.6	–	–
Plaid	77.3	32.4	–	–
SEDD Uniform	–	40.3	103.6	–
UDLM	99.8‡	33.8‡	–	–
Duo	97.6‡	33.6‡	77.6	–
FLM (1024 steps)	96.9	–	62.2	–
LangFlow	92.2	30.0	36.5	–
Controlled Comparisons				
LangFlow (Our Reproduce, 170K Iters)	–	–	41.98	5.24
LangFlow + Improved Arch (Our Reproduce, 170K Iters)	–	–	**28.75**	5.15
RePlaid (Our Reproduce, 50K Iters)	–	–	52.31	5.42
RePlaid + Improved Arch (Our Reproduce, 50K Iters)	–	–	**33.91**	5.26

Table 2: Cumulative architectural ablation on OWT-1024, from R1 (A009) to R7 (the complete model). Each successive row adds one component, with wider conditioning before shortcut detachment. The 50k-step column reports unsmoothed GenPPL using 128 NFE, 1,024 generated samples, temperature 1. The 170k-step column reports unsmoothed GenPPL using 1024 NFE, 1,024 generated samples, and temperature 1.

Cumulative architecture								50k steps 128 NFE	170k steps 1024 NFE	
Run	RMS	QK	c=256	Detach	Calib.	Train Conf.-SC	Params	GenPPL↓	GenPPL↓	Ent.
R1							327.09M	87.48	41.97	5.24
R2	✓						327.09M	84.41	-	-
R3	✓	✓					327.09M	84.37	-	-
R4	✓	✓	✓				334.44M	80.74	-	-
R5	✓	✓	✓	✓			334.44M	79.35	-	-
R6	✓	✓	✓	✓	✓		334.45M	74.03	-	-
R7	✓	✓	✓	✓	✓	✓	334.45M	**61.73**	**28.74**	5.15

cut weighting from direct codebook gradients, and state estimation from auxiliary feedback. A LangFlow-style implementation makes these distinctions concrete while preserving the generative formulation. The accompanying analysis specifies what each intervention changes, and the experimental design distinguishes generation gains, mechanism evidence, and transfer. The broader objective is a reusable architectural basis for CDLMs, with the OWT-1024 ladder as its first controlled test and cross-setting validation as the basis for stronger claims.

AI USE STATEMENT

OpenAI Codex assisted with literature search, configuration inspection, manuscript and mathematical drafting, synthetic checks of analytical identities, and preparation of the LaTeX source and architecture diagram. Experiments and evaluation artifacts are produced by the authors' training and evaluation code. The authors are responsible for verifying the analytical claims, numerical results, and interpretation of cited work before submission.

REPRODUCIBILITY STATEMENT

Section 4.1 describes the data, optimization, and evaluation protocol. Appendix A gives the analytical derivations, Appendix B specifies the model configuration, Appendix C identifies the ablation runs, and Appendix D records reporting requirements. A synthetic CPU check accompanies the analytical identities; it is separate from the language-model evaluations. The planned release includes configurations, model implementations, checkpoints, generated samples, and evaluation metadata.

ETHICS STATEMENT

This study uses OpenWebText and involves no new human-subject data collection. Models trained on web text may reproduce harmful content, personal information, or social biases. The generation metrics used here do not assess these risks. The planned release will document data provenance, intended research use, known limitations, and the procedure used to select qualitative examples.

REFERENCES

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog Bits: Generating Discrete Data using Diffusion Models with Self-Conditioning, 2022. URL https://arxiv.org/abs/2208.04202.

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.

Justin Deschenaux and Caglar Gulcehre. Language Modeling with Hyperspherical Flows, 2026. URL https://arxiv.org/abs/2605.11125.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous Diffusion for Categorical Data, 2022. URL https://arxiv.org/abs/2211.15089.

Zhicheng Du and Lan Ma. Continuous Language Diffusion as a Decoder-Interface Problem, 2026. URL https://arxiv.org/abs/2606.08810.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. In *Proceedings of the International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2403.03206.

Antonio Franca and Alexander Tong. Hacking Generative Perplexity: Why Unconditional Text Evaluation Needs Distributional Metrics, 2026. URL https://arxiv.org/abs/2606.08417.

Ishaan Gulrajani and Tatsunori B. Hashimoto. Likelihood-Based Diffusion Language Models, 2023. URL https://arxiv.org/abs/2305.18619.

Hongcan Guo, Qinyu Zhao, Yian Zhao, Shen Nie, Rui Zhu, Qiushan Guo, Feng Wang, Tao Yang, Hengshuang Zhao, Guoqiang Wei, and Yan Zeng. Continuous Latent Diffusion Language Model, 2026. URL https://arxiv.org/abs/2605.06548.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-Key Normalization for Transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4246–4253, 2020. doi: 10.18653/v1/2020.findings-emnlp.379. URL https://aclanthology.org/2020.findings-emnlp.379/.

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.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models, 2022. URL https://arxiv.org/abs/2206.00364.

Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and Improving the Training Dynamics of Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24174–24184, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Karras_Analyzing_and_Improving_the_Training_Dynamics_of_Diffusion_Models_CVPR_2024_paper.html.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. Flow Map Language Models: One-step Language Modeling via Continuous Denoising, 2026. URL https://arxiv.org/abs/2602.16813.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM Improves Controllable Text Generation, 2022. URL https://arxiv.org/abs/2205.14217.

Jiajun Liang, Yucheng Liao, Yukang Cao, Jiazhe Wei, Ken Li, Wende Tan, Jiankun Zhang, ZY Cui, Jingkang Yang, Liucheng Guo, Shiqi Yang, B. Yang, Caifeng Shan, Ziwei Liu, and Chenyang Si. AURORA-LM: Autoencoding Unified Representation for Continuous-Latent Diffusion Language Modeling, 2026. URL https://arxiv.org/abs/2608.02602.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized Transformer with Representation Learning on the Hypersphere. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=se4vjm7h4E.

Viacheslav Meshchaninov, Alexander Shabalin, Egor Chimbulatov, Nikita Gushchin, Ilya Koziev, Alexander Korotin, and Dmitry Vetrov. How to Train Your Latent Diffusion Language Model Jointly With the Latent Space, 2026. URL https://arxiv.org/abs/2605.07933.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large Language Diffusion Models, 2025. URL https://arxiv.org/abs/2502.09992.

Shen Nie, Qiyang Min, Shaoxuan Xu, Zihao Huang, Yuxuan Song, Yong Shan, Yankai Lin, Wayne Xin Zhao, Chongxuan Li, and Ji-Rong Wen. Improved Large Language Diffusion Models, 2026. URL https://arxiv.org/abs/2606.25331.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. URL https://arxiv.org/abs/2212.09748.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers, 2021. URL https://arxiv.org/abs/2102.01454.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models, 2024. URL https://arxiv.org/abs/2406.07524.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T. Chiu, and Volodymyr Kuleshov. The Diffusion Duality. In *Proceedings of the International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9P9Y8FOSOk.

Subham Sekhar Sahoo, Jean-Marie Lemercier, Zhihan Yang, Justin Deschenaux, Jingyu Liu, John Thickstun, and Ante Jukic. Scaling Beyond Masked Diffusion Language Models, 2026. URL https://arxiv.org/abs/2602.15014.

Junzhe Shen, Jieru Zhao, Ziwei He, and Zhouhan Lin. CoDAR: Continuous Diffusion Language Models are More Powerful Than You Think, 2026. URL https://arxiv.org/abs/2603.02547.

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.

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.

Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Josh Susskind. Stabilizing Transformer Training by Preventing Attention Entropy Collapse, 2023. URL https://arxiv.org/abs/2303.06296.

Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1910.07467.

A Elementary Architectural Analysis

This appendix records the assumptions and derivations underlying the method. The statements are elementary consequences of normalization, Gaussian likelihoods, the chain rule, and conditional-mean estimation. They clarify the interventions; they are not new convergence results or proofs of a generation-quality advantage. We work in real arithmetic, with nonzero codebook rows and finite noise levels such that $\sigma_\gamma > 0$.

A.1 Scale Dependence of Normalized Attention

For a head vector $q \in \mathbb{R}^{d_h}$, write $\widehat{q} = g_q \odot q/\sqrt{\|q\|_2^2/d_h + \varepsilon}$. Since $\varepsilon > 0$,

$$\|\widehat{q}\|_2^2 \le \|g_q\|_\infty^2 \frac{\|q\|_2^2}{\|q\|_2^2/d_h + \varepsilon} \le d_h \|g_q\|_\infty^2. \tag{15}$$

The analogous bound holds for $\widehat{k}$. Rotary transformations preserve Euclidean norm, so Cauchy–Schwarz gives Equation 9. The result applies to the dot-product term before softmax; it does not bound a separate additive attention bias.

For fixed gains, the bound is independent of the raw query and key norms. Positive rescaling is exactly removed only in the idealized $\varepsilon = 0$ case with nonzero inputs; with $\varepsilon > 0$, the invariance is approximate away from very small norms. Learned gains can themselves grow, and attention can remain concentrated. The result therefore motivates scale diagnostics, not an assertion that Q/K RMSNorm prevents all instability or entropy collapse (Henry et al., 2020; Zhai et al., 2023).

A.2 Observation Evidence and Contextual Prediction

Let $e_v = s\widetilde{E}_v$ and hold the codebook fixed. For token v at position i, Equation 4 gives

$$\log p(z_{\gamma,i} \mid y_i = v, \gamma) = -\frac{\|z_{\gamma,i} - \alpha_\gamma e_v\|_2^2}{2\sigma_\gamma^2} + C_\gamma \tag{16}$$

$$= \frac{\alpha_\gamma}{\sigma_\gamma^2} z_{\gamma,i}^\top e_v - \frac{\alpha_\gamma^2}{2\sigma_\gamma^2}\|e_v\|_2^2 + C(z_{\gamma,i}, \gamma). \tag{17}$$

All codebook entries have squared norm $s^2 d$, so the second term is independent of token identity. The remaining token-dependent term is precisely the baseline shortcut. This likelihood parameterization is used in Plaid and LangFlow (Gulrajani & Hashimoto, 2023; Chen et al., 2026); it is not introduced here.

Suppose the forward noise is independent across positions conditional on the clean sequence and γ. Write $\mathbf{z}_{\gamma,-i}$ for the noisy sequence without position i, and $p_\star$ for the distribution induced by the data and the forward process. For tokens with positive conditional probability, Bayes' rule yields

$$\log \frac{p_\star(y_i = v \mid \mathbf{z}_\gamma, \gamma)}{p_\star(y_i = u \mid \mathbf{z}_\gamma, \gamma)} = \frac{\alpha_\gamma}{\sigma_\gamma^2} z_{\gamma,i}^\top (e_v - e_u) \tag{18}$$

$$+ \log \frac{p_\star(y_i = v \mid \mathbf{z}_{\gamma,-i}, \gamma)}{p_\star(y_i = u \mid \mathbf{z}_{\gamma,-i}, \gamma)}. \tag{19}$$

The first term is observation evidence at the current position; the second uses the other noisy positions and includes the token prior. No uniform token-prior assumption is needed. Unequal embedding norms would require retaining the norm-dependent term in Equation 17.

The learned head is not restricted to depend on $\mathbf{z}_{\gamma,-i}$ alone. Consequently, interpreting it as an exact context-only prior would be unjustified. Moreover, an unrestricted head could absorb calibration by replacing ℓ_{raw} with $\ell_{\mathrm{raw}} + (a_\psi - 1)c_{\mathrm{skip}} z_\gamma (s\widetilde{E})^\top$. Calibration is a change in parameterization for a finite model, not a uniquely identifiable decomposition of its logits.

A.3 The Gradient Path Removed by Detachment

Consider one position after the shortcut warmup. Let $S = s\widetilde{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$, let $\beta = a_\psi(\gamma) c_{\mathrm{skip}}(\gamma)$, and write the logits as a column vector $\ell = r + \beta S z$. For an unweighted token cross-entropy loss,

$\delta = \partial\mathcal{L}/\partial\ell = p_\theta - \mathrm{onehot}(y)$. The direct shortcut contribution to the gradient of S is $\beta\delta z^\top$. If $J_S = \partial\,\mathrm{vec}(S)/\partial\,\mathrm{vec}(E)$, then

$$\nabla_{\mathrm{vec}(E)}\mathcal{L} - \nabla^{\mathrm{sg}}_{\mathrm{vec}(E)}\mathcal{L} = J_S^\top\,\mathrm{vec}\big(\beta\delta z^\top\big)\,. \tag{20}$$

Both gradients are evaluated at the same parameters and inputs, with identical computation except for stop-gradient on the shortcut's S factor. Contributions through r, through z when the noisy state depends on the codebook, and through any other retained path occur in both gradients. They are not removed by this operation. Batch averaging and token weights multiply the corresponding terms; the warmup factor can be included in β.

The identity follows by differentiating the matrix product and applying the chain rule through codebook normalization. It establishes exactly which contribution is removed, not that this contribution is harmful. In particular, a large c_{skip} alone does not imply a large gradient because δ also depends on the prediction. Measuring relative gradient norms and alignment with the retained gradient is needed to investigate an optimization-based explanation.

A.4 Posterior Means, Geometry, and Feedback

For any token distribution p, let $Y \sim p$, $X = e_Y$, $\mu(p) = \mathbb{E}_p[X]$, and $\mathcal{R}_p(a) = \mathbb{E}_p\|X - a\|_2^2$. Expanding the square and using $\mathbb{E}_p[X - \mu(p)] = 0$ gives

$$\mathcal{R}_p(a) = \mathcal{R}_p(\mu(p)) + \|a - \mu(p)\|_2^2. \tag{21}$$

Substituting $a = w\mu(p)$ proves Equation 14. Since $\|e_v\|_2^2 = s^2 d$ for every token,

$$\mathrm{tr}\,\mathrm{Cov}_p(X) = \mathcal{R}_p(\mu(p)) = s^2 d - \|\mu(p)\|_2^2. \tag{22}$$

Neither identity assumes that p is calibrated. They describe moments under p; they become statements about conditional data risk when p equals the true posterior.

Entropy alone does not determine Equation 22. For example, take equal-norm embeddings $e_1 = (r, 0)$, $e_2 = (0, r)$, and $e_3 = (-r, 0)$. The distributions $(1/2, 1/2, 0)$ and $(1/2, 0, 1/2)$ both have entropy $\log 2$, but their embedding variances are $r^2/2$ and r^2, respectively. Confidence gating is therefore a heuristic for auxiliary-input weighting, not a geometry-exact uncertainty estimator.

In the designated combined model, gating changes only the training bootstrap input. The inference solver and its self-conditioning input both retain the unweighted mean. The risk identity does not prove that an inference gate would be harmful in a learned iterative system; it explains why better conditioning and better state estimation should be tested as separate claims.

B Implementation Configuration

The overall architecture is described in Section 3.1, with component changes in Sections 3.3–3.5. All RMSNorm reductions use FP32 with $\varepsilon = 10^{-6}$. Q/K RMSNorm is applied after projection and head reshape, before RoPE. Each layer has separate query and key gain vectors shared across heads. Confidence gating is applied to the bootstrap input on self-conditioned training examples. At inference, both self-conditioning and the ODE update use the unweighted posterior mean embedding.

C Experiment Identifiers

The seventh identifier is the designated final model for this manuscript and Figure 2. It uses the explicit 250k-update configuration, not the similarly named run without this suffix, the 1M-update variant, or the early-stopped variant. In both tables, R4 has 256-dimensional conditioning without shortcut detachment, and R5 adds detachment at that width. Each measured score in the 50k-step column of Table 2 is the GenPPL scalar recorded at optimizer step 50,000, not a minimum or average over earlier checkpoints. The final-evaluation columns use the same run mapping and remain blank until the corresponding terminal-checkpoint evaluations have been validated.

Table 3: Shared configuration for the cumulative architectural ablation.

Item	Value
Data	OpenWebText; document BOS/EOS; packed random crops
Tokenizer / vocabulary	Qwen3 tokenizer; 151,936 tokens; no pretrained LM weights
Sequence length	1,024
Backbone	width 768; 12 blocks; FFN 3,072; 12 MHA heads; head dimension 64
Position / dropout	RoPE base 10,000; dropout 0.1
Diffusion state	normalized learned token embedding scaled by 0.1
Self-conditioning	25% of training examples
Training	250k updates; global batch 512; bfloat16; seed 42
Optimizer	AdamW; weight decay 0.01; gradient clip 1.0
Learning rate	3×10^{-4} to 3×10^{-5} cosine; 5k warmup
Checkpoint weights	EMA, decay 0.999
Inline evaluation	every 10k updates; 1,024 samples; 128 NFE
Primary evaluation	5,120 samples; length 1,024; 1,024 NFE
Evaluator	frozen GPT-2 Large; corpus-level GenPPL

D Evaluation and Reporting Details

For each reported result, the evaluation record will include the checkpoint step, weight type, sampler, NFE accounting, sample identifiers and seeds, decoded JSONL, evaluator and reference-set checksums, and complete metric JSON. Comparable ablation results require the following:

- identical checkpoint steps, EMA conventions, generation seeds, temperatures, NFE, and sample counts;
- complete sample sets at the specified sequence length, with no failed or empty outputs;
- corpus-level GenPPL aggregation as defined in Equation **??**;
- consistent tokenization and boundary handling for diversity and repetition metrics;
- the fixed 5,120-sample OWT reference and a common feature extractor for MAUVE;
- qualitative examples selected using predetermined sample identifiers.

Logged-scalar extraction. The trajectory table reads scalar events directly from the seven designated output directories. The recorded hyperparameters specify EMA weights, 1,024 samples of length 1,024, 128 NFE, temperature 1, seed 0, and the same GPT-2 Large scorer. GenPPL exponentiates the token-aggregated mean negative log-likelihood; unigram entropy averages sample-level token-frequency entropies rather than pooling all samples into a single distribution. The exporter retains exact step indices and raw scalar precision in CSV files, records source-event paths and snapshot checksums, and rounds only the displayed table. Missing TensorBoard entries remain blank even if other evaluations are available. These training-progress logs are kept separate from the terminal-checkpoint reporting requirements above.

E Additional Results

Table 4 reports the complete available 128-NFE training trajectories underlying Figure 1 and the 50k-step column of Table 2, retaining the full 10k–250k grid and leaving unrecorded measurements blank.

Additional results pending. Diagnostics by noise level, matched final-checkpoint comparisons at 128 and 1024 NFE, and runtime measurements remain to be completed. Fixed-ID qualitative generations from the baseline and the separate 1M-budget run are provided in Appendix G.

Table 4: Training trajectories from TensorBoard inline evaluation on OWT-1024 at 128 NFE. Each entry is *GenPPL / mean unigram entropy* (nats), using 1,024 generated samples, EMA weights, temperature 1, and generation seed 0. Rows give optimizer steps; blank entries mean that no scalar was recorded in the extracted logs. Values are unsmoothed and rounded to two and three decimal places, respectively. These are not the 5,120-sample final evaluations reserved in Table 1 and the final-evaluation columns of Table 2.

Step	R1	R2	R3	R4	R5	R6	R7
10k	208.04/5.600	206.72/5.589	191.03/5.571	182.24/5.568	172.45/5.562	161.40/5.552	120.06/5.469
20k	131.09/5.523	128.19/5.512	124.49/5.501	115.72/5.486	109.82/5.492	100.25/5.456	82.21/5.422
30k	106.93/5.491	103.12/5.476	101.65/5.471	97.56/5.469	92.85/5.470	87.17/5.442	71.69/5.417
40k	95.20/5.469	93.00/5.462	91.92/5.459	87.31/5.449	85.58/5.449	79.01/5.423	64.44/5.405
50k	87.48/5.459	84.41/5.453	84.37/5.444	80.74/5.440	79.35/5.442	74.03/5.412	61.73/5.409
60k	84.16/5.451	80.20/5.446	79.75/5.436		77.72/5.440		59.14/5.407
70k	79.94/5.443	78.08/5.442	77.03/5.433		73.35/5.424		56.63/5.399
80k	77.27/5.442	74.03/5.434	73.37/5.421		70.23/5.417		56.62/5.402
90k	75.22/5.437	74.17/5.442	71.89/5.426				55.50/5.398
100k	74.19/5.434	71.67/5.431	70.12/5.424				54.23/5.392
110k	72.61/5.435	71.54/5.435	68.57/5.422				55.07/5.406
120k	70.62/5.425	68.97/5.429	66.83/5.412				53.11/5.394
130k	69.45/5.427	68.16/5.422	66.77/5.419				53.54/5.401
140k	68.68/5.424	68.19/5.426	64.82/5.412				52.19/5.391
150k	66.96/5.421	66.77/5.421	63.83/5.408				49.59/5.361
160k	66.63/5.424	65.02/5.407	62.49/5.405				53.60/5.400
170k	66.31/5.420	64.65/5.415	61.47/5.403				51.63/5.371
180k	64.94/5.410	63.44/5.409	60.06/5.401				52.39/5.385
190k	65.09/5.409	63.03/5.407	59.07/5.392				
200k	64.23/5.418	63.22/5.411	59.24/5.397				
210k	66.04/5.426	62.65/5.411	58.23/5.388				
220k	64.78/5.421	61.93/5.403	57.71/5.386				
230k	65.01/5.418	60.79/5.395	58.24/5.390				
240k	64.80/5.416	61.26/5.403	58.19/5.394				
250k	64.26/5.416	60.76/5.396	57.86/5.387				

R1: A009; R2: + RMSNorm; R3: + Q/K RMSNorm; R4: + condition width 256; R5: + shortcut detach; R6: + learned shortcut calibration; R7: + training confidence-gated SC (IMPROVED CDLM). Additions are cumulative and follow the same order as Table 2. Exact run identifiers are given in Appendix C.

F CONTROLS FOR ARCHITECTURAL ATTRIBUTION

The following comparisons extend, rather than replace, the seven requested runs. They are planned controls with no results reported in this draft.

Use matched data exposure and optimization schedules for component comparisons, and separately report compute-matched comparisons where parameter counts differ. Any hyperparameter tuning must use comparable search budgets. Define settings and selection criteria before inspecting final test generations. Sample-level bootstrap intervals do not replace independently trained seeds.

Sampler provenance. Record the checkpoint, exact model class, code revision or source hashes, sampler implementation, and feedback rule together. Experiment names alone do not identify sampling behavior. The combined confidence-gated model in this ladder overrides the training-bootstrap conversion but inherits the default unweighted sampling self-conditioning. A separate confidence-gated model class may override the sampling hook; evaluations of that class must not be merged into the designated row.

Failure cases. Report settings in which a modification has no benefit or degrades quality. Reduced feature variance need not imply better generation; a detached gradient need not have been harmful; and confidence gating may suppress useful uncertain inputs. These outcomes constrain the scope of an architectural principle rather than invalidating the reporting protocol.

Table 5: Controls needed to interpret architectural gains. These are experimental specifications, not completed findings.

Question	Control	What it distinguishes
Does a component matter in the final model?	Remove each component from the complete configuration; retain required interfaces.	Final-configuration contribution versus an effect specific to the ladder order.
Do normalization and conditioning interact?	Q/K RMSNorm on/off crossed with condition width 128/256.	Complementarity versus separate main effects.
Is wider conditioning the right use of capacity?	Allocate a comparable parameter increase to the backbone; report actual FLOPs.	Conditioning placement versus additional capacity and computation.
Do shortcut changes interact?	Detachment on/off crossed with calibration on/off at fixed condition width.	Forward weighting versus gradient routing and their interaction.
Is noise-dependent calibration necessary?	Fixed unit, learned constant, and learned noise-dependent multipliers.	Adaptation to noise versus global logit rescaling.
Does the gate use meaningful confidence?	No gate, constant attenuation, and within-noise-bin shuffled gates.	Token-confidence association versus generic shrinkage.
Does the effect persist across seeds?	Independently trained paired baseline/modified runs with the same seed set.	Training variability versus sample-estimation uncertainty.

G QUALITATIVE GENERATIONS

We show eight complete unconditional generations: sample IDs 0 and 1 from each combination of two models and two sampling budgets. Selection uses these fixed indices, without ranking outputs by perplexity or perceived quality. The text is reproduced from the raw decoded records without correction, truncation, or removal of document-boundary tokens. Line wrapping and spacing are adjusted for typesetting. These are model outputs, not factual accounts; generated names, quotations, and bracketed citation-like strings are not evidence or references supplied by the authors.

Models and checkpoints. The extended-schedule model is `A009_RMSNORM_QK_RMSNORM_LOGIT_SHORTCUT_DETACH_COND256_LEARNED_SHORTCUT_CALIB_CONF_GATED_SC_1M`, evaluated at 170k updates. The suffix specifies its planned 1M-update schedule, not a completed 1M-step checkpoint. The baseline is the exact A009 run listed as R1 in Appendix C, evaluated at 180k updates. Each model uses the same checkpoint at 128 and 1024 NFE. The extended-schedule run is distinct from the `CONF_GATED_SC_250k` configuration in Table 2; the different training steps and schedules preclude interpreting these examples as a matched architectural ablation.

Table 6: Source settings for the qualitative examples. Each source file contains 1,024 generations; only IDs 0 and 1 are displayed. All outputs use EMA weights, length 1,024 model tokens, temperature 1, the LangFlow Euler sampler, and final argmax decoding. The generation seed equals the sample ID.

Run	Planned updates	Checkpoint	NFE	Sample IDs
Extended-schedule CDLM	1M	170k	128	0, 1
Extended-schedule CDLM	1M	170k	1024	0, 1
A009 baseline	250k	180k	128	0, 1
A009 baseline	250k	180k	1024	0, 1

Reading the examples. The 128- and 1024-NFE outputs for a given model and sample ID share the recorded integer seed; they are separate completed generations, not intermediate states of a single trajectory. Initial noise tensors were not archived, and cross-model IDs do not define a shared semantic prompt. We retain special tokens such as `<|im_end|>` and `<|endoftext|>` so that document transitions remain visible. Abrupt starts and endings, repetitions, inconsistent entities, and factual errors are left intact. These examples support qualitative inspection, not claims of typical quality, factual reliability, or monotonic improvement with NFE. Source-file hashes, line numbers, original token IDs, and the selection rule are preserved in the accompanying sample export.

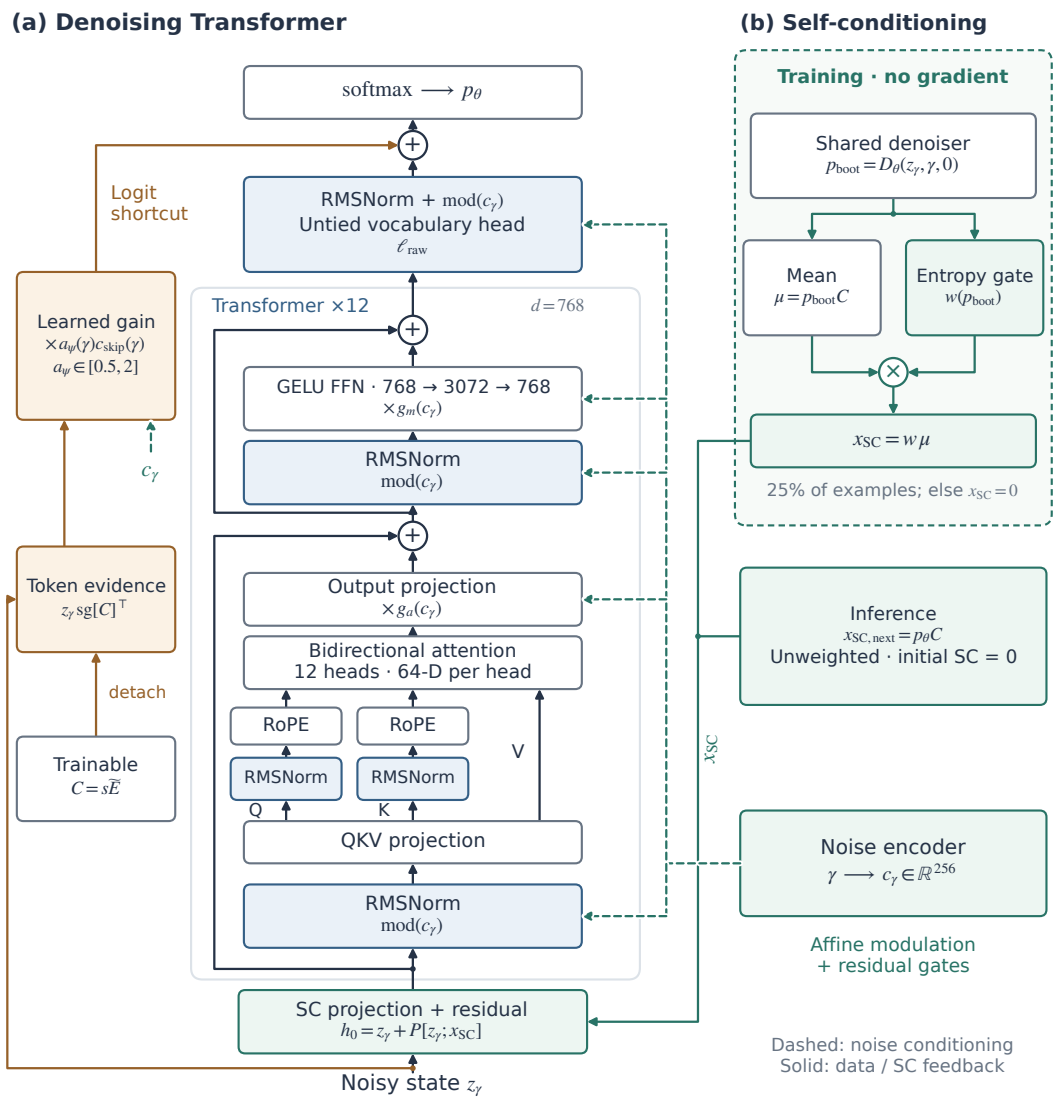

Figure 2: **Transformer architecture and self-conditioning in IMPROVED CDLM. (a)** The denoiser is read bottom to top, with one of 12 blocks expanded. RMSNorm precedes each modulated sublayer; headwise Q/K RMSNorm precedes RoPE, while V bypasses both operations. Residual branches are gated by the 256-dimensional noise embedding (dashed arrows). A calibrated logit shortcut bypasses the Transformer and untied vocabulary head. Only its explicit codebook factor is detached; $C = s\widetilde{E}$ remains trainable through noisy-state construction. **(b)** A no-gradient call to the same denoiser supplies the posterior mean and token-wise entropy gate w (Equation 13) for training self-conditioning. Inference instead passes the unweighted mean to the next step. Branch dropout and shortcut warmup are omitted for clarity.

Table 7: Architectural principles and their concrete realization. The distinctions define testable design hypotheses, not established performance guarantees.

Interface	Design principle	Instantiation
Noise-dependent computation	Control feature scale while retaining explicit noise-dependent modulation.	Backbone and Q/K RMSNorm; wider conditioning.
Vocabulary prediction	Control forward evidence weighting and direct codebook gradients separately.	Shortcut detachment; bounded learned calibration.
Self-conditioning	Distinguish the clean-state estimate from the auxiliary signal used in training.	Confidence-weighted bootstrap inputs; unchanged inference estimate.