# RECOVERING CONTROL FROM COMPACT CODES: RESIDUAL CONTROL IN DIFFUSION LANGUAGE MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

A compact condition can remain semantically informative while losing its influence on generation. We study this *representation–use gap* in an embedding-space diffusion language model. Under severe compression on TinyStories ($k$=16), direct conditioning loses control of one to three of six reliably encoded attributes across three seeds. Interventions that strengthen the denoiser's response to the same code recover control without increasing code dimension, implicating effective code–denoiser coupling as the bottleneck. Residual training provides a practical realization: it preserves all six TinyStories attributes and improves control on BeerAdvocate. Neutral-code guidance and Feynman–Kac particle steering improve both direct and residual models, while the residual model retains stronger control under matched controller settings. This shows that residual training can be combined with inference-time control to further strengthen controllability. The same residual-target intervention does not reliably improve an ELF flow model, establishing a generator-family boundary. These findings distinguish what a compact code represents from how effectively a generator uses it, and identify residual training as a targeted remedy for conditioning under-use.

## 1 INTRODUCTION

Controlling the semantic properties of generated text is an important goal in conditional language generation. Continuous representations have enabled fine-grained control in diffusion-based language models (Li et al., 2022), while compact conditioning interfaces offer a way to communicate desired properties through low-dimensional codes (Wu & Summa, 2026). Ideally, such a code should retain the information needed for generation while allowing the model to respond reliably to the specified conditions. This becomes particularly challenging when multiple semantic attributes must share a fixed-dimensional representation. Increasing code dimension may improve control, but it also weakens the motivation for a compact conditioning interface. An important question is therefore how much control can be achieved with the information already available in a compressed code.

When control deteriorates under compression, however, the underlying cause is ambiguous. The code may have lost task-relevant information (Tishby & Zaslavsky, 2015), or the generator may fail to use information that remains encoded. These failures can produce similar outputs but require different remedies: improving the representation or strengthening the generator's response to the existing code. Distinguishing them is therefore essential for understanding compressed conditional generation.

Existing evaluation approaches do not, by themselves, resolve this ambiguity. Output-level control metrics measure whether generated samples satisfy the requested attributes, but cannot identify the source of control failure. Conversely, probing can reveal whether attribute-related information remains recoverable from learned representations, but successful probing does not establish that the model actually uses this information (Hewitt & Liang, 2019; Belinkov, 2022). Diagnosing compressed conditioning therefore requires examining both the information retained by the code and the generator's effective response to it.

In this work, we investigate this distinction in a Diffusion-LM-style embedding-space text generator with a learned low-dimensional conditioning code. We focus on the regime in which compression weakens generation control even though attribute-related information remains measurable in the



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.

code. We refer to this mismatch as the *representation–use gap*. Our central question is whether control can be recovered by strengthening the generator's response to the existing conditioning signal, rather than increasing code dimension. To investigate this question, we compare direct and residual prediction targets, examining whether changing the denoising objective can improve the use of available conditioning information. We further introduce complementary interventions to test whether stronger effective code–denoiser coupling can recover control independently of residualization.

Our experiments reveal a representation–use gap under severe compression on TinyStories (Eldan & Li, 2023). Strengthening the denoiser's response to the compact code recovers control over attributes that remain detectable in the representation. Residual training provides a practical realization of this approach, with improvements extending to BeerAdvocate (McAuley et al., 2012) and a deeper denoiser. We further examine whether these improvements persist under inference-time control, including guidance and Feynman–Kac particle steering (Singhal et al., 2025). Both methods improve the direct model, but the residual model retains an advantage. Finally, experiments with an ELF conditional flow reveal a limitation: the same residual-target intervention does not consistently improve control in this alternative generator. Together, these findings highlight the importance of distinguishing representation from utilization and show that improving the use of existing conditioning information can be an alternative to increasing code capacity.

**Contributions.** Our main contributions are:

1. We experimentally demonstrate a *representation–use gap* in an embedding-space diffusion language model: compressed codes can retain semantic information while losing control over generation.
2. Through complementary interventions, we demonstrate that strengthening effective code–denoiser coupling can recover control at fixed code dimension, without requiring an increase in representational capacity.
3. We show that residual training improves control on TinyStories and BeerAdvocate. Its advantage persists when combined with neutral-code guidance and Feynman–Kac particle steering.

## 2 Related Work

**Embedding-space generation.** Diffusion-LM (Li et al., 2022) generates text by denoising continuous token embeddings and decoding them back to vocabulary items. ELF (Hu et al., 2026) uses continuous-time Flow Matching in contextual embedding space and a shared-weight denoiser/decoder, discretizing only at the final step. Latent text diffusion instead compresses the generated sequence itself: LD4LG (Lovelace et al., 2023) diffuses in the latent space of a pretrained encoder–decoder, COSMOS (Meshchaninov et al., 2025) learns a compressed latent space, and Cola DLM (Guo et al., 2026) models a global semantic prior over a text-VAE latent. Those works compress the generated object; we compress the *condition* and ask whether representational availability and generative use fail together under compression.

**Controllable generation and inference-time steering.** Representation-space steering modifies generation along semantic directions (Dathathri et al., 2020; Stolfo et al., 2025), including in diffusion language models (Shnaidman et al., 2025). Classifier-free guidance (Ho & Salimans, 2022) amplifies the difference between conditional and unconditional predictions, while FK steering (Singhal et al., 2025) uses Feynman–Kac interacting particle systems to reweight and resample diffusion trajectories according to a reward. Activation Transport (Rodríguez et al., 2025) provides a distinct activation-space intervention. RegDiff (Zhou et al., 2025) adds train-time attribute supervision to latent text diffusion, while SteeringDiffusion (Wu & Summa, 2026) learns a bottlenecked control interface for a frozen image-diffusion backbone. Conditioning Residuals (Xiang et al., 2025) feeds back intermediate features into diffusion conditioning embeddings; in contrast, our residual prediction target diagnoses under-use of an external compact code. We use downstream steering for a different purpose: to test whether inference-time control erases or preserves a host-level difference created during training.

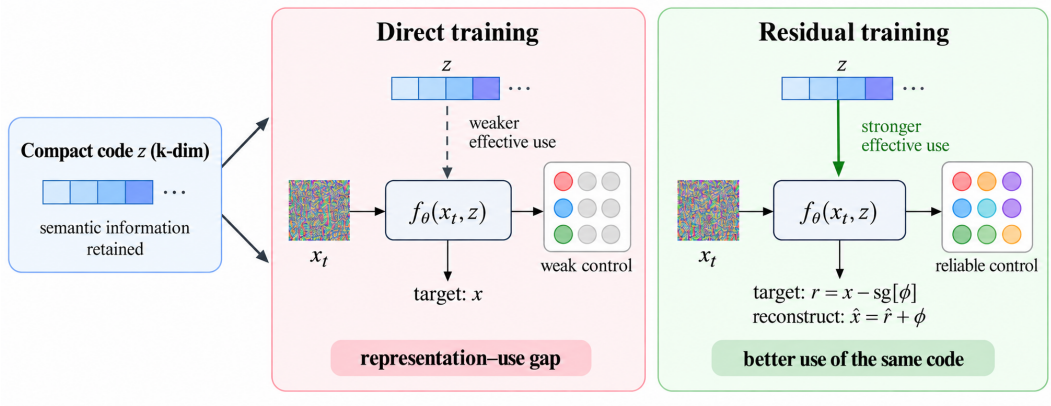



Figure 1: **Direct versus residual training.** Both share the same compact conditioning architecture. Direct predicts the clean sequence, while residual predicts its difference from the projected code and adds the code back after sampling, strengthening control without increasing code dimension.

**Compression and under-use.** Low-dimensional bottlenecks are widely used to encourage compact representations (Higgins et al., 2017), while superposition studies how many features can coexist in a limited representation (Elhage et al., 2022). Recent evidence suggests that diffusion models can leave large parts of a condition embedding unused (Pham et al., 2026). Our setting examines the complementary regime in which the conditioning code itself is aggressively compressed. Rather than measuring representation capacity or conditioning strength in isolation, we investigate whether semantic information remains recoverable from the code when the denoiser fails to use it effectively, and whether control can be restored without increasing code dimension.

## 3 Method

**Compact conditioning.** We study an embedding-space text diffusion model with a learned compact conditioning code $\boldsymbol{c} \in \mathbb{R}^k$. The generator follows Diffusion-LM (Li et al., 2022), operating on sequences of frozen T5-small token embeddings (Raffel et al., 2020), $\boldsymbol{x} \in \mathbb{R}^{L \times d}$. A frozen T5 encoder produces a mean-pooled representation of the conditioning text, which an MLP maps to a Gaussian bottleneck with a shared KL regularizer. A bias-free projection lifts the code to $\boldsymbol{\phi} = U\boldsymbol{c} \in \mathbb{R}^d$, which enters the denoiser through four learned prefix tokens, following the general principle of continuous prefix conditioning (Li & Liang, 2021). Unless stated otherwise, the code is provided only through these prefix tokens and is not added directly to the token embeddings. A full-rank reference uses the uncompressed conditioning representation.

We use a DDPM (Ho et al., 2020) with $x_0$ prediction and a Transformer denoiser, sampled using DDIM (Song et al., 2021). Across experiments, we keep the code dimension, model architecture, bottleneck regularization, and training budget fixed. We vary the prediction target, the conditioning interface, or the sampling-time response to the code to investigate different forms of effective *code–denoiser coupling*. Implementation details are provided in Appendix A. Figure 1 illustrates the main idea: direct and residual training use the same compact conditioning interface but differ in their prediction targets and effective use of the encoded information.

**Direct and residual prediction.** The direct model predicts the clean embedding sequence from $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$. Its objective is

$$\mathcal{L}_{\text{dir}} = \mathbb{E}\left[\|\hat{\boldsymbol{x}}_\theta(\boldsymbol{x}_t, \boldsymbol{\phi}, t) - \boldsymbol{x}\|^2\right] + \lambda_{\text{CE}}\mathcal{L}_{\text{CE}}(\hat{\boldsymbol{x}}_\theta) + \beta\mathcal{L}_{\text{KL}}. \tag{1}$$

Here, $\mathcal{L}_{\text{CE}}$ is the token-level rounding loss, computed from the predicted embeddings and the frozen vocabulary embedding matrix, and $\mathcal{L}_{\text{KL}}$ regularizes the compact code.

The residual model instead predicts $\boldsymbol{r} = \boldsymbol{x} - \mathrm{sg}[\boldsymbol{\phi}]$, where sg denotes stop-gradient and $\boldsymbol{\phi}$ is broadcast across sequence positions. Its objective is

$$\begin{aligned}\mathcal{L}_{\mathrm{res}} = \mathbb{E}\left[\|\hat{\boldsymbol{r}}_\theta(\boldsymbol{r}_t, \boldsymbol{\phi}, t) - \boldsymbol{r}\|^2\right] \\ + \lambda_{\mathrm{CE}}\mathcal{L}_{\mathrm{CE}}\left(\hat{\boldsymbol{r}}_\theta + \mathrm{sg}[\boldsymbol{\phi}]\right) + \beta\mathcal{L}_{\mathrm{KL}},\end{aligned} \tag{2}$$

where $\boldsymbol{r}_t$ is obtained by applying the same diffusion process to $\boldsymbol{r}$. During generation, the reverse process operates entirely in residual space, and the code is added back after sampling:

$$\hat{\boldsymbol{x}} = \hat{\boldsymbol{r}} + \boldsymbol{\phi}. \tag{3}$$

The stop-gradient prevents the residual target and additive reconstruction from directly updating the code encoder. The code is instead trained through the prefix pathway and bottleneck regularization. Direct and residual training therefore share the same code architecture and gradient pathways through the prefix interface, although their prediction objectives and learned codes may differ. Residualization also changes the noised variable and introduces an additive reconstruction path; its effect cannot be attributed to stronger denoiser response without additional controls.

**Diagnosing code–denoiser coupling.** We use three complementary interventions to investigate whether weak control reflects insufficient use of an informative code. All preserve the code dimension.

*Input-sum conditioning* retains the direct prediction target while additionally injecting $\boldsymbol{\phi}$ into every noisy input position, strengthening the conditioning interface without residualizing the target.

A *coupling sweep* interpolates between direct and residual prediction. For a residual strength $g$, the model predicts

$$\boldsymbol{r}^{(g)} = \boldsymbol{x} - g\,\mathrm{sg}[\boldsymbol{\phi}], \qquad \hat{\boldsymbol{x}} = \hat{\boldsymbol{r}}^{(g)} + g\boldsymbol{\phi}. \tag{4}$$

Here, $g = 0$ recovers direct prediction, while $g = 1$ recovers residual prediction.

*Sampling-time intervention* adjusts the response of a frozen direct model to the same code without retraining or modifying its representation. Together, these interventions examine whether control can be recovered by modifying the conditioning interface, prediction target, or sampling-time response rather than increasing code capacity.

**Inference-time control.** We examine whether the training-time advantage of residual conditioning persists under downstream control.

*Neutral-code guidance* uses a classifier-free-guidance-style interpolation (Ho & Salimans, 2022) between predictions under the target code $\boldsymbol{c}$ and a neutral code $\boldsymbol{c}_0$, defined as the mean code of the labelled validation set:

$$\hat{\boldsymbol{x}}_t^{(w)} = \hat{\boldsymbol{x}}_t(\boldsymbol{c}_0) + w\left[\hat{\boldsymbol{x}}_t(\boldsymbol{c}) - \hat{\boldsymbol{x}}_t(\boldsymbol{c}_0)\right]. \tag{5}$$

Here, $w = 1$ corresponds to ordinary conditioning. Unlike standard classifier-free guidance, this procedure uses a neutral code rather than a separately trained unconditional prediction. For the residual model, guidance is applied to the residual prediction, while the target code is added back once after sampling. Thus, the additive reconstruction term is not amplified.

*Feynman–Kac (FK) steering* (Singhal et al., 2025) resamples stochastic trajectories using attribute rewards computed by a held-out classifier on decoded predictions. We compare both models using the same particle budget and reward, with best-of-$K$ as a baseline. Since reward-based selection can directly improve the target attribute, we also report leakage and diversity.

*Classifier guidance* (Dhariwal & Nichol, 2021; Li et al., 2022) provides a code-independent alternative by injecting external classifier gradients, testing whether control improvements persist when the compact-code pathway is bypassed. Additional configurations are reported in Appendix D.

**Evaluation.** Our evaluation separates the information available in the compact code from its influence on generation.

To measure *output control*, we construct an orthogonalized difference-of-means direction for each attribute in code space and generate samples under positive and negative steering. The resulting standardized shift in a held-out attribute readout is normalized by the corresponding full-rank reference shift to obtain *control parity*. An attribute is counted as lost when its parity falls below $0.8$.

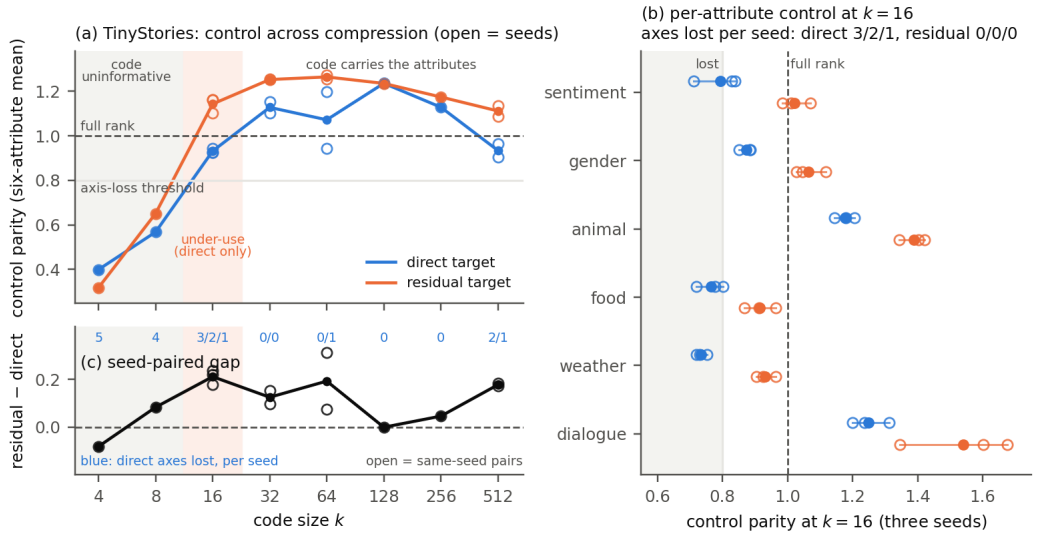



Figure 2: Compression separates representation from control on TinyStories (parity relative to the full-rank reference). (a) Six-attribute control parity across code sizes; open markers denote seeds and lines connect seed means. (b) Per-attribute parity at $k = 16$: direct loses one to three reliable axes across seeds, while residual retains all six. (c) Seed-paired residual–direct parity gap across code sizes. Residual training improves control at several code sizes, while attribute-level failure is concentrated at $k = 16$ among the compressed codes. Blue labels show lost direct-control axes.

To measure *representation capacity*, we use the Manifold Probe capacity ratio (Modell, 2026) and per-attribute logistic readouts of the compact codes. Comparing these measurements with output control allows us to identify cases in which semantic information remains detectable but is not effectively expressed in generation.

We additionally report normalized cross-attribute leakage, distinct-2, fixed-code resampling fidelity, and external generation-quality metrics, including GPT-2 perplexity and coherence. The primary TinyStories parity analysis covers six evaluated attributes; the remaining two are reported separately. Attribute definitions, steering procedures, selection criteria, implementation details, and preregistration records are provided in Appendices A–G.

## 4 Results

### 4.1 Compression creates a representation–use gap

Figure 2 shows that residual training improves control at several code sizes, but its gain is not monotonic in compression. The representation–use gap is particularly evident at $k = 16$: direct conditioning loses one to three of the six reliable attributes across three seeds, whereas residual training retains all six (Table 1). This distinction persists at a threshold of .9, where direct loses four attributes in every seed and residual loses at most one. Including all eight attributes preserves the ordering (Appendix A, Table 9).

The residual–direct gap is not unique to severe compression. Residual training also increases mean control at larger code sizes, although the gain varies across ranks and is small at $k = 128$. At $k = 512$, the models still use a learned, regularized bottleneck, whereas the full-rank reference conditions directly on the pooled representation. The direct model also loses attributes at this width, indicating that under-use is not exclusively a low-dimensional phenomenon (Appendix F). Among the compressed codes ($k \leq 256$), however, consistent attribute-level failure across seeds is concentrated at $k = 16$, motivating our analysis of this setting.

Crucially, this behavioral loss is not matched by a corresponding loss of representation. Across the rank sweep, direct and residual arms have nearly identical Manifold Probe capacity ratios; fixed-code

Table 1: Summary of control on the six reliably encoded TinyStories attributes. At $k$=16, direct conditioning loses one to three control axes across three seeds, whereas residual training retains all six. Mean parity averages the six attributes; full-rank conditioning is shown for reference. Per-attribute results are shown in Figure 2 and Appendix A.

| arm | mean parity | axes $< .8$ |
|---|---|---|
| full-rank (s1/s2) | 1.01/.99 | 0/0 |
| direct $k$=16 (s1/s2/s3) | .93/.93/.94 | 3/2/1 |
| residual $k$=16 (s1/s2/s3) | 1.16/1.10/1.16 | 0/0/0 |

Table 2: **Residual training restores sentiment control under compression.** TinyStories examples (seed 1) with sentiment steering at $\pm 3$ sd. Examples are selected for readability from the strongest sentiment quartile; bold indicates sentiment-lexicon words. Pole separation is averaged over 96 generations per pole.

| **Arm (seed 1)** | **Sentiment code $+3$ sd** | **Sentiment code $-3$ sd** | **Pole separation** |
|---|---|---|---|
| Full-rank | Once upon a time, a very little cat named Brian and **loved** to explore and collect animals. One day, Brian and Sunny went, two dream together. | Once upon a time, there was a mama in the kitchen. She ran to the pig. The piged and had lots of food on it. He saw the **bad** lady and wanted to eat. | **21.2** |
| Direct $k$=16 | Once upon a time there was a daddy, named Paul. Paul **loved** plan new material Sunday. He lived, just very not **smile** every day long. | Once upon a time, there was a little girl. She had something to eat, she **loved** it one day she decided for a twist. She was very **angry** atchy. | **15.3** |
| Residual $k$=16 | Once upon a time there was a familyboard that said named Timmy. Three children were popular **kind** little ones and their parents were very enjoyed. | One day, a little girl was **scared** and she was very **angry**. She had a carrots and a sauce. She said and ate her soup. | **20.7** |

fidelity is essentially unchanged; and a logistic probe reads the controllable attributes equally well from the two codes (mean ROC-AUC .89 for both targets across three seeds, compared with .90 for the full-rank code; Appendix A). Thus, at $k$=16, semantic information remains measurable in the compact code even as the direct denoiser translates it less effectively into output control. We refer to this separation as the *representation–use gap*.

The matched residual arm shows that this loss of control is not an inevitable consequence of reducing the code to $k$=16: the same compact dimensionality can support reliable control under a different training target. The comparison alone, however, does not explain *why* residual training avoids the representation–use gap. We therefore next diagnose the direct-model failure by manipulating how strongly the denoiser uses the same compact code.

Table 2 gives a qualitative view of the same effect: the compressed code still supports sentiment control, but the direct model responds more weakly, whereas residual training restores nearly the full-rank pole separation.

### 4.2 Diagnosing the gap: weak code–denoiser coupling

The representation–use gap suggests that the compact code still carries a usable control signal, but the direct denoiser does not use it strongly enough. We test this explanation with two interventions that change condition use while keeping the code dimension fixed.

First, *input-sum* keeps the direct prediction target but adds $U\boldsymbol{c}$ to every denoising position. This restores all six reliable TinyStories axes, showing that stronger access to the same compact code is sufficient to recover control. The resulting response is substantially amplified, however, so we use input-sum as a mechanism check rather than as our final training method.

Second, we vary the residual strength continuously. The model predicts $\boldsymbol{x} - g\,\mathrm{sg}[U\boldsymbol{c}]$ and reconstructs $\hat{\boldsymbol{x}} = \hat{\boldsymbol{r}} + gU\boldsymbol{c}$, with $g$=0 corresponding to direct training and $g$=1 to residual training. Mean parity stays near .93 up to $g = .5$, increases to 1.02 at $g = .75$ and 1.13 at $g = 1$, and reaches 1.25 at $g = 1.5$ (Table 3). Control therefore recovers as the residual contribution becomes stronger, while larger $g$ eventually moves into an amplified regime.

Table 3: Continuous interpolation from direct training ($g$=0) to residual training ($g$=1) on TinyStories $k$=16. Control remains weak for small $g$, begins to recover around $g = .75$, and all six reliable axes are retained at $g = 1$. Larger $g$ further amplifies the response.

| $g$ | 0 | .25 | .5 | .75 | 1 | 1.5 |
|---|---|---|---|---|---|---|
| mean parity | .93 | .93 | .92 | 1.02 | 1.13 | 1.25 |
| axes $<$ .8 (s1/s2) | 3/2 | 3/1 | 4/1 | 1/0 | 0/0 | 0/0 |

Table 4: Cross-corpus replication on BeerAdvocate at $k$=64. Residual training preserves substantially stronger eight-axis control across all three seeds. Full leakage and capacity measurements, together with $k$=16 and $k$=128 results, are reported in Appendix F.

| arm | mean parity (s1/s2/s3) | axes retained |
|---|---|---|
| direct $k$=64 | .58/.57/.62 | 1/1/2 |
| residual $k$=64 | 1.00/.90/.93 | 6/6/6 |

Additional diagnostics support the same interpretation. The direct model still responds when the code is steered more strongly, but with lower gain than the residual model. Moreover, removing the residual model's additive reconstruction path leaves most prefix-mediated control intact, while the additive path alone carries little control. The improvement is therefore not explained by simply adding $U\boldsymbol{c}$ back at decoding.

Together, these results are consistent with weak effective code–denoiser coupling as the source of the direct model's failure: the information remains in the code, but its effect on generation is too small. Residual training ($g$=1) provides a simple operating point that restores all reliable control axes without the stronger amplification seen at larger coupling strengths, motivating the training-time repair studied next.

### 4.3 The repair replicates on BeerAdvocate

We next ask whether the repair is specific to TinyStories semantics. On BeerAdvocate, the same ordering appears at $k$=64 (Table 4). Across three seeds, direct conditioning reaches mean eight-axis parity .58/.57/.62, whereas residual training reaches 1.00/.90/.93. Residual training retains six of eight axes in every seed, compared with only one or two for direct conditioning. The effect is not uniform across all compression levels. At $k$=16, neither target controls the BeerAdvocate attributes reliably (direct .61/.65, residual .47/.59), whereas at $k$=128 the residual advantage persists (.83 versus .53). Full leakage and capacity measurements are reported in Appendix F. Thus, BeerAdvocate provides an independent cross-corpus replication of the residual repair, while also showing that the repair requires a regime in which the compact interface still supports usable control.

### 4.4 A stronger host persists under downstream control

Having established residual training as a training-time repair, we examine whether its advantage persists under inference-time control. Both neutral-code guidance and FK steering improve direct and residual models while approximately preserving their control gap at matched controller strengths (Figure 3). Across guidance weights $w \in \{1, 2, 3\}$, best-of-8, and FK strengths $\lambda_{\mathrm{FK}} \in \{8, 32\}$, the gap remains between $+.18$ and $+.24$ across both seeds. FK also reduces leakage for both models with little change in distinct-2.

We further test whether stronger inference-time guidance can substitute for residual training through a preregistered equal-control comparison. We select direct+CFG weights to match unguided residual parity ($w^* = 1.75$ on TinyStories and $w^* = 3$ on BeerAdvocate) and evaluate both models on independent batches (Table 5). At comparable control, residual training achieves lower leakage (by .026 and .046), higher distinct-2 (by .015 and .040), and requires one rather than two denoiser evaluations per step. Direct+CFG achieves lower perplexity and higher coherence. These results reveal distinct trade-offs between training-time coupling and inference-time guidance. Full protocols and preregistered outcomes are reported in Appendix D.

Table 5: Equal-control comparison on independent fresh batches (two-model-seed means). Guidance weights $w^*$ were selected on separate matching batches so that direct+CFG matches residual parity. NFE is the number of denoiser evaluations per sampling step. Lower leakage and PPL, and higher distinct-2 and coherence, are preferable.

| Method | $w$ | parity | leakage | distinct-2 | PPL | coherence | NFE |
|---|---|---|---|---|---|---|---|
| *TinyStories*, $k = 16$ | | | | | | | |
| Residual | 1 | 1.14 | .174 | .647 | 145 | .347 | 1 |
| Direct+CFG | 1.75 | 1.15 | .200 | .632 | 140 | .359 | 2 |
| *BeerAdvocate*, $k = 64$ | | | | | | | |
| Residual | 1 | .97 | .392 | .804 | 401 | .289 | 1 |
| Direct+CFG | 3 | 1.00 | .438 | .764 | 327 | .308 | 2 |

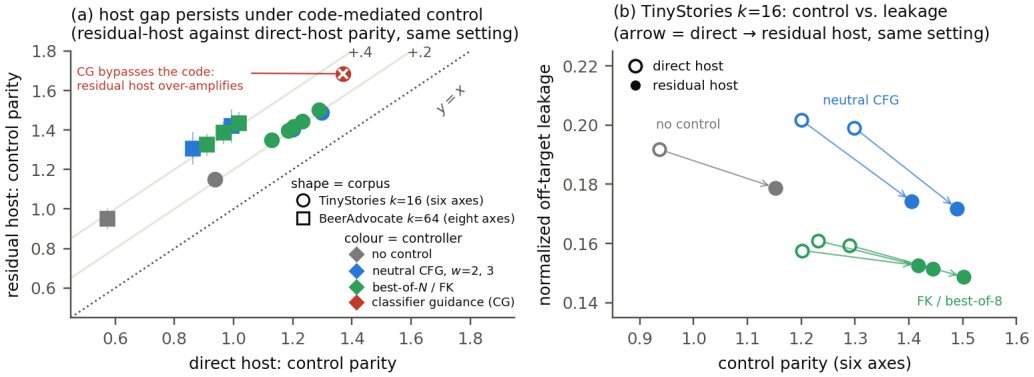



Figure 3: Residual training under inference-time control. (a) Residual-host versus direct-host control parity at matched settings on TinyStories $k$=16 (circles, six axes) and BeerAdvocate $k$=64 (squares, eight axes), including no control, neutral-code guidance ($w$=2, 3), and best-of-$N$/FK steering ($K$=4, 8). Code-mediated controllers improve both hosts while approximately preserving the residual–direct gap; classifier guidance (CG), which injects an external probe gradient, is the exception and over-amplifies the residual host. (b) Control parity versus normalized leakage on TinyStories $k$=16. Each arrow connects the direct and residual hosts under the same code-mediated controller setting. Residual hosts achieve higher control at comparable or lower leakage, while best-of-8/FK further reduces leakage through selection.

Together, these results show that residual training remains complementary to code-mediated inference-time control across both corpora. By contrast, classifier guidance bypasses the compact-code pathway and over-amplifies the residual host, reducing diversity at matched scale. Further results are reported in Appendix D.

Unlike code-mediated controllers, classifier guidance injects external attribute gradients, restoring direct control but over-amplifying the residual host and reducing diversity at matched scale. Overall, code-mediated controllers preserve the residual–direct gap at matched settings, while equal-control comparisons reveal trade-offs in leakage, diversity, generation quality, and denoiser evaluations. Further results are reported in Appendix D.

### 4.5 Generator-family boundary

Finally, we test whether the residual repair transfers to the ELF conditional flow. It does not: at $k$=16, residual-minus-direct control is $-0.18, -0.06, +0.06$ across three seeds, while at $k$=64 direct conditioning is already near parity. This contrasts with the stable residual advantage in our embedding-space DDPM. The boundary is not one of representational pressure. The Manifold Probe capacity ratio of the code stack is nearly the same in the two families at every rank ($.008/.100/.388$ for the ELF flow and $.008/.106/.375$–$.425$ for the DDPM at full rank, $k$=64, and $k$=16, the last a two-seed range), so compression places the same measured demand on the code in both generators.

Table 6: Downstream control on BeerAdvocate (seed 1/seed 2). Neutral guidance, best-of-8, and FK all improve the direct host, but the residual–direct advantage remains substantial at matched settings. Residual also has equal or lower leakage with little change in diversity. The no-control distinct-2 values are taken from the $w$=1 guidance cell, which uses the same sampler without guidance.

| BeerAdvocate $k$=64 | direct | residual | residual−direct |
|---|---|---|---|
| no control | .58/.57 | 1.00/.90 | $+.\mathbf{42}/+.\mathbf{33}$ |
| neutral guidance, $w = 3$ | 1.01/.98 | 1.51/1.34 | $+.\mathbf{50}/+.\mathbf{36}$ |
| $K = 1$ | .57/.59 | 1.06/.93 | $+.\mathbf{49}/+.\mathbf{34}$ |
| best-of-8 | .90/.91 | 1.38/1.27 | $+.\mathbf{48}/+.\mathbf{35}$ |
| FK($K$=8), $\lambda_{\mathrm{FK}} = 32$ | 1.01/1.02 | 1.49/1.38 | $+.\mathbf{48}/+.\mathbf{36}$ |

| configuration | nleak (D/R) | distinct-2 (D/R) |
|---|---|---|
| no control | .43/.44 / .40/.39 | .86/.86 / .85/.86 |
| neutral guidance, $w = 3$ | .43/.45 / .40/.40 | .83/.82 / .80/.81 |
| $K = 1$ | .44/.45 / .39/.39 | .85/.85 / .85/.84 |
| best-of-8 | .35/.36 / .35/.34 | .85/.85 / .84/.84 |
| FK($K$=8), $\lambda_{\mathrm{FK}} = 32$ | .34/.34 / .34/.33 | .85/.85 / .84/.84 |

The ELF null therefore does not imply that residual training is generally ineffective. Rather, it limits the repair to the failure mode diagnosed here: an informative compact code that the DDPM denoiser under-uses. Compression creates the same representational pressure in both generators; how that pressure appears in generation is generator-dependent.

## 5 Conclusion

Compression does not necessarily remove semantic information before it removes control. In our embedding-space DDPM, the compact code remains informative at settings where direct conditioning weakens sharply, revealing a representation–use gap. Input-sum conditioning and a continuous coupling-strength sweep both recover the lost control without increasing code dimension, indicating that the failure is associated with weak effective coupling between the code and the denoiser rather than a lack of representational capacity.

Residual training provides a simple training-time repair for this failure mode. The standard residual objective ($g = 1$) recovers all six reliable TinyStories attributes, remains effective with a deeper denoiser, and replicates on BeerAdvocate. Its advantage persists under neutral-code guidance and FK steering at matched controller strengths. When direct+CFG is tuned to achieve comparable control, residual training retains lower leakage, higher distinct-2, and requires one rather than two denoiser evaluations per sampling step, while guided direct conditioning achieves lower perplexity and higher coherence. Thus, residual training and inference-time guidance offer distinct control–quality–compute trade-offs rather than being interchangeable.

This repair is not universal. Classifier guidance, which injects an external gradient outside the compact-code pathway, does not preserve the same host-level relation, and the residual objective gives no consistent advantage in the ELF flow family. Together with the broader ELF analysis, this suggests that similar compression pressure can be expressed differently by different generators. The practical lesson is therefore to separate representation from use: when a compact code still carries the desired information but the generator responds to it too weakly, strengthening code–denoiser coupling through residual training can restore semantic control without increasing the conditioning dimension.

## Reproducibility Statement

All principal model comparisons use matched training budgets and preregistered decision rules. Appendices A–G report model architectures, training schedules, evaluation protocols, dataset details, additional seeds, decoder controls, and preregistration records, including amendments and unsuccessful predictions.

## AI USE STATEMENT

Generative AI tools were used to assist with language editing, literature search and discovery, research ideation and experimental implementation, and drafting and organizing sections of the manuscript. The authors independently reviewed and verified the cited literature, research methodology, experimental results, analyses, and final manuscript, and take full responsibility for the scientific content of this work.

## REFERENCES

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL https://mlanthology.org/iclr/2020/dathathri2020iclr-plug/.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8780–8794, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

Ronen Eldan and Yuanzhi Li. TinyStories: How small can language models be and still speak coherent english?, 2023. URL https://arxiv.org/abs/2305.07759.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/.

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.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. $\beta$-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL https://mlanthology.org/iclr/2017/higgins2017iclr-beta/.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

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.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 4582–4597. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.353.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/1be5bc25d50895ee656b8c2d9eb89d6a-Abstract.html.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b2a2bd5d5051ff6af52e1ef60aefd255-Abstract-Conference.html.

Julian J. McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *2012 IEEE 12th International Conference on Data Mining*, pp. 1020–1025. IEEE, 2012. doi: 10.1109/ICDM.2012.110. URL https://doi.org/10.1109/ICDM.2012.110.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry P. Vetrov. Cosmos: Compressed and smooth latent space for text diffusion modeling. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1506930fb75c82246e4d8648a66e4b27-Abstract-Conference.html.

Alexander Modell. Probing for representation manifolds in superposition. *arXiv preprint arXiv:2605.18537*, 2026.

Trung X. Pham, Kang Zhang, Ji Woo Hong, and Chang D. Yoo. A hidden semantic bottleneck in conditional embeddings of diffusion transformers. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1ec299a5229034141e58aeded0d0b9de-Abstract-Conference.html.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Pau Rodríguez, Arno Blaas, Michal Klein, Luca Zappella, Nicholas Apostoloff, Marco Cuturi, and Xavier Suau. Controlling language and diffusion models by transporting activations. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/df4f6e43446b1ee29c5a33d32c279f83-Abstract-Conference.html.

Adi Shnaidman, Erin Feiglin, Osher Yaari, Efrat Mentel, Amit Levi, and Raz Lapid. Activation steering for masked diffusion language models. *arXiv preprint arXiv:2512.24143*, 2025.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 55810–55827. PMLR, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.02502.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. Improving instruction-following in language models through activation steering. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/8c3262a4c965ba9888f120d4f9e13478-Abstract-Conference.html.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2015. doi: 10.1109/ITW.2015.7133169.

Fangzheng Wu and Brian Summa. SteeringDiffusion: A bottlenecked activation control interface for diffusion models, 2026. URL https://arxiv.org/abs/2605.01653.

Weilai Xiang, Hongyu Yang, Di Huang, and Yunhong Wang. Conditioning residuals for diffusion models via representation feedback, 2025. URL https://arxiv.org/abs/2505.10999. Current title in arXiv v4, revised July 2026.

Fan Zhou, Chang Tian, and Tim Van de Cruys. Controllable stylistic text generation with train-time attribute-regularized diffusion, 2025. URL https://arxiv.org/abs/2510.06386.

# A Experimental Details

## A.1 Generator and Training Configuration

**Generator architecture.** We use a Diffusion-LM-style DDPM (Li et al., 2022; Ho et al., 2020) over frozen T5-small token embeddings (Raffel et al., 2020). Each embedding sequence has length $L = 256$ and dimension $d = 512$. The denoiser is a 12-block ELF-B Transformer with hidden size 768 and approximately 104 million parameters. The diffusion process uses $T = 1000$ discrete steps, a cosine noise schedule, and $x_0$ prediction. Generation uses 60 DDIM steps (Song et al., 2021), followed by nearest-neighbor decoding into the frozen vocabulary embedding space.

**Compact conditioning.** A frozen T5 encoder produces a mean-pooled representation of the conditioning text. An MLP maps this representation to a $k$-dimensional Gaussian bottleneck, regularized by a KL penalty with weight $\beta = 10^{-3}$. A bias-free matrix $U \in \mathbb{R}^{d\times k}$ lifts the compact code $\boldsymbol{c}$ to $\boldsymbol{\phi} = U\boldsymbol{c}$. Four learned prefix slot embeddings receive a linear projection of $\boldsymbol{\phi}$ and are prepended to the denoiser input. All token positions can attend to these prefix tokens. The direct model receives the code exclusively through this prefix interface; $\boldsymbol{\phi}$ is not added directly to the token embeddings. The residual model uses the same prefix interface. The full-rank reference conditions directly on the uncompressed mean-pooled representation. It therefore differs from a learned $k = 512$ code, which retains the Gaussian bottleneck and its KL regularization.

**Optimizer and hardware.** All arms use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and decoupled weight decay $0.01$. The learning rate is scaled linearly with batch size from a base rate of $10^{-3}$ per 256 sequences, giving a learning rate of $3.125 \times 10^{-4}$ at batch size 80. It is warmed up linearly from zero over the first 1,000 steps and then held constant, without decay. Gradients are clipped to a global norm of $1.0$. Training uses single-precision floating-point computation without mixed precision or dropout. An exponential moving average (EMA) of the model parameters, with decay $0.9999$, is updated after every optimizer step. Only EMA weights are used for evaluation. Each arm is trained on a single GPU with the full batch of 80 sequences; no data or model parallelism is used. Experiments use NVIDIA H200 NVL (141 GB) or NVIDIA RTX PRO 6000 Blackwell Server (96 GB) GPUs, with a single-device JAX/Flax/Optax training implementation. A standard ELF-B arm completes 60 epochs (37,500 steps) at approximately 5–8 steps per second, requiring about 1.3–2.1 hours. The deeper ELF-M backbone runs at approximately 2.6 steps per second, requiring about 4 hours. Selected checkpoints were also evaluated in a second environment. Cross-environment consistency and any differences in reported metrics are documented in the corresponding experimental results.

**Training arms (TinyStories).** Direct and residual models are trained at $k \in \{256, 128, 64, 32, 16\}$ with seed 42; $k \in \{16, 32, 64\}$ additionally use seed 43. The three $k = 16$ arms (direct, residual, and input-sum) use a third seed, 44, registered after the main results as a seed-robustness check. The seed-3 direct arm was trained and evaluated in a second environment against same-environment full-rank denominators; the two environments agree to the second decimal. Full-rank references use two seeds. The coupling sweep uses two seeds at each of $g \in \{0.25, 0.5, 0.75, 1.5\}$, in addition to the direct and residual endpoints. Learned-head direct and residual controls use $k = 16$. A separate backbone-transfer experiment repeats the central comparison with the deeper ELF-M denoiser. All standard arms use the same training budget and optimizer settings. No arm- or corpus-specific hyperparameter tuning is performed unless recorded as a preregistered amendment.

### A.2 Residual Diffusion and Diagnostic Interventions

**Residual diffusion.** For the direct model, the clean embedding sequence $\boldsymbol{x}$ is perturbed according to

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I). \tag{6}$$

The residual target is defined as

$$\boldsymbol{r} = \boldsymbol{x} - \mathrm{sg}[\boldsymbol{\phi}], \tag{7}$$

where sg denotes stop-gradient and $\boldsymbol{\phi}$ is broadcast across all sequence positions.

The same forward diffusion process is applied to the residual:

$$\boldsymbol{r}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{r} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}. \tag{8}$$

During generation, the initial Gaussian sample and all DDIM updates operate in residual space. The final embedding sequence is reconstructed by adding the code once after sampling:

$$\hat{\boldsymbol{x}} = \hat{\boldsymbol{r}} + \boldsymbol{\phi}. \tag{9}$$

**Rounding loss and gradient pathways.** Let $E$ denote the frozen vocabulary embedding matrix. The rounding loss is the token-level cross-entropy of the true tokens under the softmax of the predicted embeddings' dot products with $E$. We use $\lambda_{\mathrm{CE}} = 1$ for both direct and residual training. For residual training, reconstruction inside the rounding loss uses $\hat{\boldsymbol{r}}_\theta + \mathrm{sg}[\boldsymbol{\phi}]$. The residual target and its noisy input also depend on $\mathrm{sg}[\boldsymbol{\phi}]$. Consequently, neither residual target construction nor additive reconstruction provides a direct gradient to the code encoder or lifting matrix $U$. The denoising objective updates the code parameters through the prefix-conditioning pathway, while the KL term additionally regularizes the bottleneck encoder. Direct and residual training share these gradient pathways, although their prediction objectives and resulting learned codes may differ.

**Diagnostic interventions.** Input-sum conditioning retains the direct prediction target while adding $\boldsymbol{\phi}$ to every noisy input position, in addition to the existing prefix interface.

The coupling sweep interpolates between direct and residual prediction:

$$\boldsymbol{r}^{(g)} = \boldsymbol{x} - g\,\mathrm{sg}[\boldsymbol{\phi}], \qquad \hat{\boldsymbol{x}} = \hat{\boldsymbol{r}}^{(g)} + g\boldsymbol{\phi}. \tag{10}$$

The corresponding forward process noises $\boldsymbol{r}^{(g)}$ rather than $\boldsymbol{x}$. Here, $g = 0$ recovers direct prediction, while $g = 1$ recovers residual prediction.

Changing $g$ jointly modifies the prediction target, noised variable, and additive reconstruction strength.

The sampling-time intervention uses neutral-code guidance on a frozen direct model, without retraining or changing the learned code. Its implementation and complete controller comparisons are reported in Appendix D.

**Learned decoder control.** We additionally evaluate a trainable contextual decoder with 23.3 million parameters. It consists of a dense projection, two ELF-style attention blocks, normalization, and a nonlinear projection, followed by vocabulary unembedding.

The decoder is trained jointly in both direct and residual arms. The additive-only pathway remains inert under this decoder, while the direct–residual control gap persists.

### A.3 Corpora and Evaluation Protocol

**Corpora.** TinyStories (Eldan & Li, 2023) contains 50,000 training stories and 2,000 validation examples.

BeerAdvocate (McAuley et al., 2012) contains 50,000 reviews of 20–200 T5 tokens with complete aspect ratings, streamed with a fixed shuffle seed, and 2,000 validation examples.

Both corpora use the same generator configuration except for dataset-specific attributes and readouts.

The BeerAdvocate control battery uses the gold-rating axes overall ($\geq 4$), appearance ($\geq 4$), and ABV ($\geq 7$), together with dark style and the lexicon axes hoppy, malty, fruit, and spice.

Aroma, palate, and taste are excluded because they correlate with overall above $0.5$, while roast is excluded because it correlates with dark style.

**Attribute steering.** For each attribute, we fit a difference-of-means direction in code space using 400 labelled stories. The direction is orthogonalized against the other attribute directions. We steer each code by $\pm 3$ standard deviations of the labelled stories' projection onto the corresponding direction. A held-out difference-of-means classifier in the frozen encoder space evaluates the generated samples under positive and negative steering.

**Control parity.** For each attribute, the difference between positive and negative steering responses is standardized by the attribute readout's standard deviation to obtain its output shift. Control parity divides this shift by the corresponding full-rank reference shift, averaged over two model seeds. An attribute is counted as *lost* when its parity is below $0.8$. We also report threshold sensitivity at $0.7$ and $0.9$.

**Eight-axis battery.** The eight TinyStories attributes are sentiment, gender, animal, length, food, weather, vehicle, and dialogue. The primary preservation count covers six attributes: sentiment, gender, animal, food, weather, and dialogue. Length is not controllable through the compact interface in any tested arm, and vehicle is marginal across compact arms. Both are reported separately rather than omitted. Table 8 provides per-attribute results for all eight attributes. Table 9 reports sensitivity to the preservation threshold and attribute set.

**Representation capacity.** We report the Manifold Probe capacity ratio (Modell, 2026) and per-attribute logistic readouts of the compact codes. The logistic readout uses five-fold evaluation on the exported codes of 1,500 validation stories, with labels from the paper's attribute lexicons. We report mean ROC-AUC over the six primary attributes, together with the weakest attribute, weather. These measurements quantify recoverable semantic information. They do not, by themselves, establish whether the generator uses that information.

**Leakage, fidelity, and quality.** Cross-attribute leakage is evaluated using the $8 \times 8$ steer-by-read matrix. Normalized leakage is the mean off-diagonal response divided by the mean diagonal response. We additionally report fixed-code resampling fidelity, distinct-2, GPT-2 perplexity, and semantic coherence. Complete generation-quality comparisons are reported in Appendix E.

**Evaluation robustness.** The 96-sample replicate resamples the 96 paired positive and negative generations per attribute 2,000 times. Parity intervals divide by the full-rank seed-mean point estimate. An independent general-domain RoBERTa sentiment classifier, which has not seen our lexicons, reproduces the $k = 16$ sentiment comparison: residual parity is $0.96/1.11$, compared with $0.75/0.86$ for direct conditioning.

### A.4 Additional TinyStories Measurements

| arm (per seed) | lost of 6, thr. .7 | lost of 6, thr. .8 | lost of 6, thr. .9 | lost of 8, thr. .8 |
|---|---|---|---|---|
| direct $k = 16$ (s1/s2/s3) | 0/0/0 | 3/2/1 | 4/4/4 | 5/4/3 |
| residual $k = 16$ (s1/s2/s3) | 0/0/0 | 0/0/0 | 1/0/0 | 2/2/2 |
| direct $k = 32$ (s1/s2) | 0/0 | 0/0 | 0/0 | 1/2 |
| residual $k = 32$ (s1/s2) | 0/0 | 0/0 | 0/0 | 1/1 |
| direct $k = 64$ (s1/s2) | 0/0 | 0/1 | 0/3 | 2/3 |
| residual $k = 64$ (s1/s2) | 0/0 | 0/0 | 0/0 | 1/1 |

Table 9: Sensitivity of the axis-loss count to the threshold and attribute set, using the same evaluation battery as Table 8. No arm loses an attribute at the $0.7$ threshold. At $0.8$ and $0.9$, the direct–residual ordering holds across all seeds at $k = 16$. The eight-attribute count additionally includes length, which no compact arm controls, and vehicle, which is marginal.

Table 7: Boundary measurements at $k = 16$ on TinyStories. Direct and residual training retain comparable measurable code information and fixed-code fidelity, despite their difference in output control. Capacity ratios are comparable across the rank sweep. The attribute readout is a five-fold logistic probe on 1,500 validation stories; ROC-AUC is averaged over the six primary attributes, with weather in parentheses. The three seeds of each target agree to within $0.005$ on the readout, and the full-rank code gives $0.90$ ($0.84$).

| $k = 16$ metric (TinyStories) | direct s1/s2 | residual s1/s2 | interpretation |
|---|---|---|---|
| capacity ratio | .375/.425 | .400/.400 | comparable demand |
| attribute readout AUC (mean of 6 / min) | .89/.89 (.79/.78) | .89/.89 (.78/.77) | comparable information |
| normalized leakage | .123/.175 | .108/.165 | no systematic reduction |
| quality margin | .010/.013 | .006/.001 | not improved; below the .02 gate |
| fixed-code fidelity | .875/.872 | .874/.874 | unchanged |

Table 8: Per-axis control parity on TinyStories. Length and vehicle are reported but excluded from the six-axis preservation count. At $k = 16$, direct conditioning loses one to three of the six axes across three seeds, whereas residual training loses none. At $k = 32$ and $k = 64$, seed-2 direct/residual mean parities are $1.10/1.26$ and $0.94/1.26$, respectively. All rows except the learned-head rows come from the same-environment re-evaluation of the saved checkpoints used for Figure 2 and Table 1. The original evaluation agrees to the second decimal on most axes. Its main difference is that direct $k = 16$ seed 1 measured food parity at $.88$ rather than $.78$, giving loss counts of $2/3/1$ instead of $3/2/1$. †Learned-head rows were evaluated only in the original environment.

*Panel A: sentiment, gender, animal, length, and food*

| arm | sent. | gender | animal | length | food |
|---|---|---|---|---|---|
| full s1/s2 | .98/1.02 | 1.02/.98 | 1.04/.96 | .91/1.09 | .98/1.02 |
| direct $k = 64$ s1 | 1.15 | 1.09 | 1.35 | -.76 | 1.04 |
| residual $k = 64$ s1 | 1.24 | 1.20 | 1.41 | -.63 | 1.09 |
| direct $k = 32$ s1 | 1.06 | 1.05 | 1.27 | -.52 | 1.03 |
| residual $k = 32$ s1 | 1.14 | 1.21 | 1.41 | -.21 | 1.06 |
| direct $k = 16$ s1 | .71 | .89 | 1.14 | -.32 | .78 |
| direct $k = 16$ s2 | .83 | .88 | 1.21 | -.17 | .72 |
| direct $k = 16$ s3 | .84 | .85 | 1.18 | -.35 | .80 |
| residual $k = 16$ s1 | 1.01 | 1.12 | 1.40 | -.11 | .87 |
| residual $k = 16$ s2 | 1.07 | 1.03 | 1.34 | .11 | .91 |
| residual $k = 16$ s3 | .98 | 1.05 | 1.42 | -.32 | .96 |
| direct + head† | .74 | .88 | 1.06 | .13 | .79 |
| residual + head† | 1.01 | 1.11 | 1.35 | .90 | .97 |
| input-sum s1/s2 | 1.46/1.28 | 2.06/2.06 | 1.98/2.24 | .03/.45 | 1.53/1.50 |
| input-sum s3 | 1.52 | 1.74 | 1.67 | .47 | 1.52 |

*Panel B: weather, vehicle, dialogue, and aggregate results*

| arm | weather | vehicle | dialogue | mean(8) | $n < .8$ of 6 |
|---|---|---|---|---|---|
| full s1/s2 | 1.01/.99 | .99/1.01 | 1.01/.99 | .99/1.01 | 0/0 |
| direct $k = 64$ s1 | .97 | .76 | 1.60 | .90 | 0 |
| residual $k = 64$ s1 | .97 | .83 | 1.73 | .98 | 0 |
| direct $k = 32$ s1 | 1.01 | .83 | 1.52 | .91 | 0 |
| residual $k = 32$ s1 | 1.03 | 1.03 | 1.65 | 1.04 | 0 |
| direct $k = 16$ s1 | .73 | .52 | 1.31 | .72 | 3 |
| direct $k = 16$ s2 | .72 | .55 | 1.20 | .74 | 2 |
| direct $k = 16$ s3 | .75 | .48 | 1.24 | .72 | 1 |
| residual $k = 16$ s1 | .90 | .74 | 1.68 | .95 | 0 |
| residual $k = 16$ s2 | .92 | .77 | 1.34 | .94 | 0 |
| residual $k = 16$ s3 | .96 | .77 | 1.60 | .93 | 0 |
| direct + head† | .73 | .51 | 1.34 | .77 | 3 |
| residual + head† | .84 | .69 | 1.73 | 1.08 | 0 |
| input-sum s1/s2 | 1.32/1.23 | 1.34/1.31 | 2.48/2.58 | 1.53/1.58 | 0/0 |
| input-sum s3 | 1.16 | 1.20 | 2.50 | 1.47 | 0 |

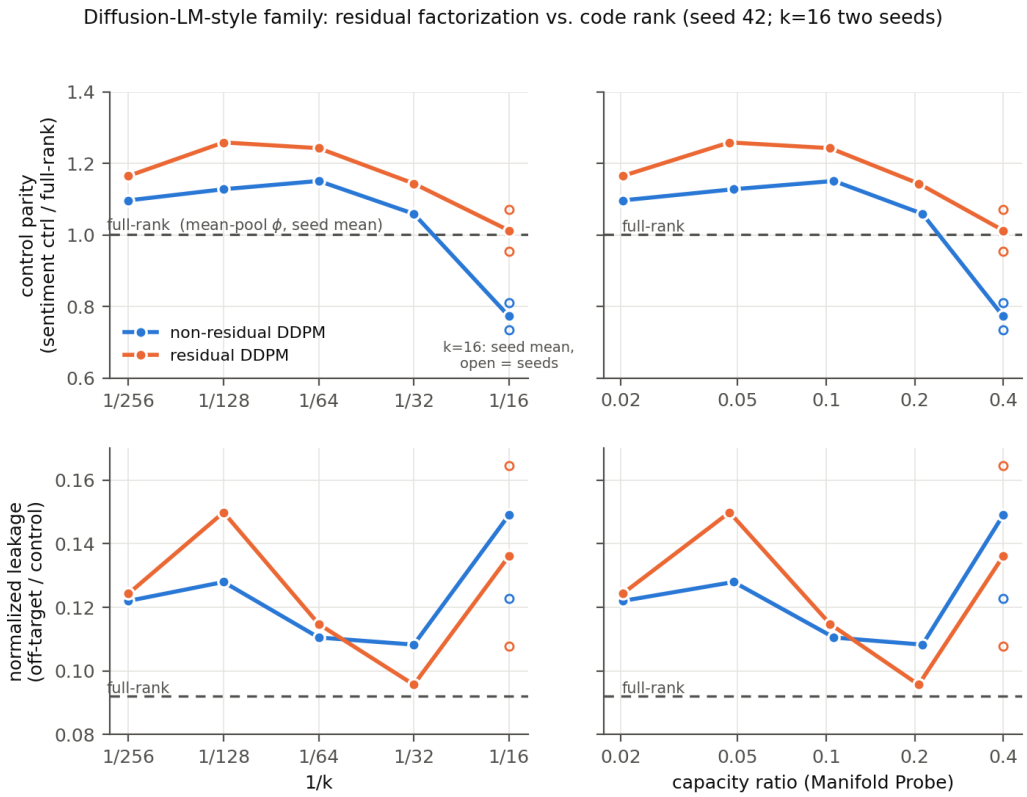



Figure 4: Sentiment control parity and normalized leakage across code sizes on TinyStories, plotted against $1/k$ and the Manifold Probe capacity ratio. Direct conditioning loses control at $k = 16$, while residual training remains near full-rank control. Leakage shows no corresponding separation, and both targets have comparable capacity ratios across code sizes.

## B Dose, Composition, and Informative Starting Codes

At fixed per-axis dose, retention of the single-axis shift decreases similarly across all arms, including full rank, as more attributes are steered together. With the total steering norm held fixed by scaling each axis as $1/\sqrt{m}$, full-rank response is nearly flat and compact-code arms lose only about 15–20% at $m$=6. Thus compositional load is not the phenomenon repaired by residual coupling. Input-sum, while stronger on single-axis control, shows larger off-target spillover and a larger quality cost under high compositional load.

The condition-start battery steers the codes of twelve held-out stories per pole stratum by $\pm 3$ sd on each axis with one noise draw shared by the unsteered, $+$ and $-$ conditions. Residual mean parity is $1.28/1.23$ versus $0.96/0.95$ for direct conditioning. Reinforcing an existing attribute is easier for every arm (reinforce above flip by 0.2–0.3 in the code arms), but when the edit opposes the story's current attribute, direct conditioning falls to $0.88$ parity while residual conditioning remains at $1.17$. Two registered companions were nulls: the realization ratio of unsteered generations from real codes is $\approx 0.8$ of the data separation under every target, full rank included; and last-third over first-third control is 1.0–1.1 in every arm, so the direct prefix, unlike an instruction in a prompt, is not position-limited. The natural-dose grid ($\alpha \in \{-1, -.5, 0, .5, 1\}$, 96 samples per point) gives gain per unit dose relative to full rank of $0.43/0.43$ (direct) and $0.72/0.70$ (residual) over the six axes; the gap at natural dose ($0.28$) is slightly larger than at standard dose ($0.21$), short of the registered $0.10$ margin.

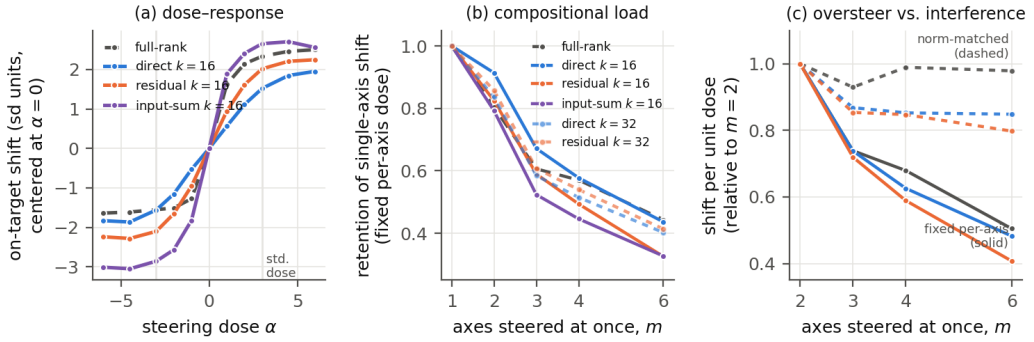



Figure 5: Dose response and compositional load. Direct $k$=16 conditioning has lower gain, residual coupling tracks full rank more closely, and input-sum enters an amplified regime. Degradation under simultaneous steering is largely shared with full rank and is not the phenomenon repaired by coupling.

## C Pathway and Trajectory Diagnostics

Table 10: Registered trajectory diagnostics on TinyStories. Residual trajectories are shorter after endpoint normalization, less variable across codes, and preserve larger conditional separation. On the coupling-sweep interpolation arms normalized path length falls $1.55 \to 1.50 \to 1.48$ and across-code sd $26 \to 20 \to 19$ from $g = .25$ to $.75$.

| | | $k$=256 | 128 | 64 | 32 | 16 (s1/s2) | 16 + head |
|---|---|---|---|---|---|---|---|
| $L/\|\text{end}\|$ | direct | 1.574 | 1.568 | 1.577 | 1.581 | 1.566/1.571 | 1.593 |
| | residual | 1.511 | 1.528 | 1.527 | 1.525 | 1.487/1.497 | 1.512 |
| Across-code sd($L$) | direct | 27.4 | 25.6 | 26.7 | 29.1 | 28.0/26.9 | 24.0 |
| | residual | 22.7 | 21.0 | 21.1 | 20.2 | 21.1/20.2 | 19.3 |
| $D$ in $\hat{\boldsymbol{x}}$ space | direct | 3.24 | 3.24 | 3.44 | 3.37 | 2.86/2.72 | 2.88 |
| | residual | 3.53 | 3.55 | 3.84 | 4.42 | 4.19/4.15 | 4.88 |
| $\cos(\Delta\hat{\boldsymbol{r}}, \Delta U\boldsymbol{c})$ | residual | -.96 | -.94 | -.95 | -.96 | -.98/-.98 | -.98 |

The residual target mechanically induces an approximately antiparallel conditional shift between $\hat{\boldsymbol{r}}$ and $U\boldsymbol{c}$, which explains why the decode-time addition alone is inert. The relevant quantity is the remaining shift in $\hat{\boldsymbol{x}}$ after this cancellation, which stays strong for the residual model and shrinks for the direct $k$=16 model. For input-sum, severing reveals a different anatomy: both the prefix and the per-position addition independently carry substantial endpoint control (prefix-only $\Delta = 0.88$, addition-only $\Delta = 1.00$). The two coupling mechanisms therefore share a behavioral endpoint but not an identical internal route.

## D Inference-Time Control and External Baselines

**Neutral-code guidance and FK steering.** Neutral-code guidance amplifies the difference between steered and neutral-code predictions using the same conditioned network. The direct-host sweep recovers all six reliable axes, while the residual–direct gap remains approximately constant at matched weights through $w = 3$; stronger weights eventually over-amplify both hosts. FK steering (Singhal et al., 2025) provides a selection-based alternative: we use $K = 8$ particles, stochastic DDIM ($\eta = 1$), decoded-text classifier rewards, and resampling every six steps. The $K = 1$ baseline agrees with unguided generation within .05 (FK-P0). At $\lambda_{\text{FK}} \in \{8, 32\}$, FK increases parity by about .3 for both hosts and preserves a $+.19$–$+.23$ residual–direct gap, with lower residual leakage and similar distinct-2 (FK-P2–P4). Neutral-code-only FK does not rescue control under this particle budget: direct and residual parity remain $.38/.39$ and $.39/.40$, respectively, with all six axes below .8 (FK-P1, missed).

**Equal-control comparison.** To test whether stronger training-time coupling is equivalent to inference-time amplification, we independently select CFG weights that approximately match residual-host parity at $w = 1$, then evaluate on a separate fresh batch (sampling seed 2024, 96 generations per pole, two model seeds). The matching batch gives TinyStories parity $1.15$ for residual and $1.10$, $1.15$, and $1.20$ for direct+CFG at $w = 1.5$, $1.75$, and $2$; we select $w = 1.75$. On BeerAdvocate, residual reaches .97, whereas direct+CFG reaches .86 and .99 at $w = 2$ and 3; we select $w = 3$.

Table 5 in the main text reports the fresh-batch results: EC-P1 (leakage), EC-P2 (distinct-2), EC-P4 (compute), and EC-P5 (out-of-sample match) hit, and EC-P3 (fluency) missed on both corpora. Because the fresh batch (96 generations per pole, sampling seed 2024) is a different draw from the 96-sample replicate behind Table 11, the same host's leakage differs slightly between the two tables (residual at $w$=1: .174 there versus $.18/.18$ here); within each table all cells share one batch.

**Protocol controls and amendments.** The same additive reconstruction term with CFG amplification leaves residual parity unchanged to two decimals. Best-of-8 explains about $+.24$ of FK's $+.33$ parity gain over $K = 1$; intermediate resampling explains about $+.09$. We adopted stochastic DDIM because duplicated particles under deterministic sampling do not diverge; decoded-text rewards replaced a weak diffused-space reward. Additional descriptive checks used $K = 4$, GPT-2 perplexity, and an independent RoBERTa sentiment classifier. Perplexity did not exceed the corresponding $K = 1$ value, but the sentiment readout saturated at one pole and was noisy at the other (24 samples per pole). A separately trained, half-budget unconditional branch for two-model CFG also failed to rescue the direct host, saturating at $3.3/3.6$ standard deviations.

**Classifier guidance and other baselines.** Classifier guidance backpropagates a decontaminated probe through the denoiser with unit-RMS normalization. At $s^* = .5$, it restores all six reliable direct-host axes (parity $1.39/1.34$), without the predicted leakage–diversity cost. No residual-host scale meets the preregistered minimum per-axis distinct-2 criterion. Registered fallback scales $s = 2/4$ over-amplify residual control (parity $1.67/1.70$) and increase leakage, whereas a post-hoc matched-scale comparison at $s = .5$ gives parity $1.72/1.65$, leakage $.163/.183$, and distinct-2 $.65/.68$. The matched-scale conclusion is over-amplification and lower diversity, not necessarily greater leakage; minimum per-axis distinct-2 differs from the aggregate distinct-2 displayed in Table 11.

BeerAdvocate $k = 64$ uses the same controllers and 48 samples per pole. All four hosts were evaluated in a common second environment because of checkpoint incompatibility; the direct hosts also agree with their first- environment evaluations to within .03 parity. Exploratory Linear-AcT (Rodríguez et al., 2025) at $\lambda_{\mathrm{AcT}} = 1$ does not rescue neutral-code control, but augments both hosts when combined with code steering while retaining the residual advantage.

Table 11: Inference-time control on TinyStories. Neutral-code guidance, best-of-$K$, and FK preserve the residual–direct gap at matched strengths. Classifier guidance restores direct-host control but over-amplifies the residual host; registered residual results use fallback scales and the matched-scale results are post-hoc. Linear-AcT comparisons are exploratory; dashes indicate unreported diversity measurements.

| route (TinyStories $k$=16) | parity(6) s1/s2 | norm. leakage s1/s2 | distinct-2 s1/s2 |
|---|---|---|---|
| direct, no guidance | $.94/.93$ | $.19/.19$ | $.80/.81$ |
| residual, no guidance | $1.18/1.12$ | $.18/.18$ | $.78/.78$ |
| direct + neutral CFG, $w = 2$ | $1.21/1.19$ | $.202/.201$ | $.76/.77$ |
| residual + neutral CFG, $w = 2$ | $1.44/1.37$ | $.176/.172$ | $.73/.74$ |
| direct + neutral CFG, $w = 3$ | $1.32/1.27$ | $.193/.205$ | $.74/.74$ |
| residual + neutral CFG, $w = 3$ | $1.53/1.45$ | $.175/.168$ | $.70/.72$ |
| direct, FK $K = 1$ | $.97/.96$ | $.193/.195$ | $.79/.80$ |
| residual, FK $K = 1$ | $1.20/1.15$ | $.184/.172$ | $.77/.77$ |
| direct, best-of-4 | $1.13/1.13$ | $.156/.172$ | $.79/.80$ |
| residual, best-of-4 | $1.38/1.32$ | $.152/.148$ | $.76/.77$ |
| direct + FK($K$=4), $\lambda_{\mathrm{FK}} = 32$ | $1.19/1.18$ | $.161/.166$ | $.79/.80$ |
| residual + FK($K$=4), $\lambda_{\mathrm{FK}} = 32$ | $1.42/1.37$ | $.156/.145$ | $.76/.77$ |
| direct, best-of-8 | $1.21/1.19$ | $.156/.160$ | $.79/.80$ |
| residual, best-of-8 | $1.44/1.39$ | $.160/.145$ | $.76/.77$ |

*Inference-time control (continued)*

| route (TinyStories $k$=16) | parity(6) s1/s2 | norm. leakage s1/s2 | distinct-2 s1/s2 |
|---|---|---|---|
| direct + FK, $\lambda_{\mathrm{FK}} = 8$ | $1.24/1.22$ | $.159/.161$ | $.79/.79$ |
| residual + FK, $\lambda_{\mathrm{FK}} = 8$ | $1.47/1.42$ | $.157/.145$ | $.76/.77$ |
| direct + FK, $\lambda_{\mathrm{FK}} = 32$ | $1.30/1.29$ | $.155/.162$ | $.78/.79$ |
| residual + FK, $\lambda_{\mathrm{FK}} = 32$ | $1.53/1.48$ | $.153/.140$ | $.76/.77$ |
| direct + FK, neutral code only ($\lambda^*_{\mathrm{FK}}$) | $.38/.39$ | $.282/.290$ | $.82/.83$ |
| residual + FK, neutral code only ($\lambda^*_{\mathrm{FK}}$) | $.39/.40$ | $.287/.289$ | $.83/.83$ |
| direct + classifier guidance, $s^* = .5$ | $1.39/1.34$ | $.187/.201$ | $.75/.72$ |
| residual + classifier guidance, registered $s = 2/4$ | $1.67/1.70$ | $.227/.265$ | $.62/.64$ |
| residual + classifier guidance, matched $s = .5$ | $1.72/1.65$ | $.163/.183$ | $.65/.68$ |
| classifier guidance, full-rank hosts, $s^* = .5$ | $1.54/1.61$ | $.219/.247$ | $.61/.57$ |
| classifier guidance, unconditional host, $s^* = 16$ | 1.84 | .326 | .35 |
| activation steering, full-rank hosts | .15 | .273 | — |
| direct + AcT ($\lambda_{\mathrm{AcT}} = 1$), combined | $1.16/1.16$ | $.171/.192$ | — |
| residual + AcT ($\lambda_{\mathrm{AcT}} = 1$), combined | $1.40/1.29$ | $.150/.151$ | — |
| direct + AcT ($\lambda_{\mathrm{AcT}} = 1$), neutral code | $.31/.34$ | $.356/.339$ | — |
| residual + AcT ($\lambda_{\mathrm{AcT}} = 1$), neutral code | $.32/.27$ | $.372/.346$ | — |

**FK sentiment strength sweep.** We test four FK strengths. At $\lambda_{\mathrm{FK}} \in \{2, 8, 32, 128\}$, direct seed 1 reaches sentiment shifts $3.43, 3.81, 3.93, 3.94$ (minimum distinct-2 .67–.68), remaining below the registered $4.02$ target. Direct seed 2 reaches $4.19/.70$ at $\lambda_{\mathrm{FK}} = 8$; both residual seeds reach the target at $\lambda_{\mathrm{FK}} = 2$ ($4.54/.65$ and $5.00/.64$) and then saturate. Neutral-code-only FK stays below roughly $1.2$ shift.

## E External Generation-Quality Checks

The family-specific within-minus-across margin is sensitive to whether conditioning codes remain distinguishable, but it is not a standard language-quality metric. We therefore score the 96-sample TinyStories batteries with GPT-2 perplexity (lower is better), MiniLM sentence-to-sentence semantic coherence, and distinct-2. Table 12 reports raw seed means at the unsteered neutral code and at the standard $\pm 3$ dose averaged over the six controllable axes. Under the frozen protocol, direct and residual $k$=16 satisfy the registered no-collapse comparison: steering lowers perplexity and raises coherence for both. Input-sum obtains stronger control but incurs the clearest diversity cost. The full-rank companion and the neutral-code perplexity band miss their registered thresholds. In particular, neutral-code PPL remains elevated for the compressed arms, so these checks rule out a residual-specific collapse but do not establish uniformly high absolute generation quality under severe compression.

Table 12: External generation-quality metrics on TinyStories (two-seed means). The registered no-collapse criterion (perplexity ratio $\leq 1.15$, coherence drop $\leq .03$) holds for the direct and residual compressed codes. The registered neutral-code band (code-arm PPL within 15% of full rank) does not hold.

| arm (seed means) | $\mathrm{PPL}_0$ | $\mathrm{PPL}_{\pm 3}$ | ratio | $\mathrm{coh.}_0$ | $\mathrm{coh.}_{\pm 3}$ | distinct-2 $_{0 \to \pm 3}$ |
|---|---|---|---|---|---|---|
| full-rank | 110 | 138 | 1.25 | .255 | .333 | $.63 \to .60$ |
| direct $k$=16 | 181 | 164 | .90 | .266 | .318 | $.70 \to .66$ |
| residual $k$=16 | 162 | 145 | .90 | .264 | .349 | $.70 \to .63$ |
| input-sum $k$=16 | 148 | 171 | 1.16 | .283 | .440 | $.68 \to .54$ |

## F Additional Scope Checks

**BeerAdvocate replication.** The BeerAdvocate comparison provides the cross-corpus replication used in the main text. At $k$=64, direct conditioning reaches parity $.58/.57$ across the two full-metric seeds, while residual training reaches $1.00/.90$; a third registered seed reproduces the ordering (.62

versus .93). The residual model also has equal or lower normalized leakage across these runs. At $k$=128, the same ordering persists (.53 direct versus .83 residual). These results establish that the coupling advantage is not specific to TinyStories semantics.

Table 13: BeerAdvocate cross-corpus replication. Residual training restores substantially more control than direct conditioning at $k$=64 and preserves the ordering at $k$=128, without higher normalized leakage.

| BeerAdvocate | parity(8) s1/s2 | $n$<.8 | leakage | capacity |
|---|---|---|---|---|
| full-rank | 1.02/.98 | 0/0 | .217/.218 | .008 |
| direct $k$=64 | .58/.57 | 7/7 | .431/.437 | .069/.066 |
| residual $k$=64 | 1.00/.90 | 2/2 | .397/.392 | .062/.069 |
| direct $k$=128 | .53 | 7 | .478 | .034 |
| residual $k$=128 | .83 | 4 | .409 | .031 |

**Learned full-width code ($k$=512).** The $k$=512 arms in Figure 2 keep the whole code stack of the compressed arms, including the Gaussian bottleneck and its KL penalty, at the width of the pooled representation; the full-rank reference bypasses that stack and conditions on the pooled representation directly. The two are therefore different conditions, and the direct target behaves differently under them. Direct $k$=512 reaches six-attribute parity $.90/.96$ and loses food and weather on seed 1 and weather on seed 2, the same attributes that go first at $k$=16; residual $k$=512 reaches $1.09/1.14$ and loses none. Normalized leakage is $.181/.177$ (direct) and $.173/.190$ (residual) against $.141/.132$ for the reference, and the residual target's lower normalized path length replicates at this width as at every compressed rank. The dip is thus not a compression effect: what the direct target under-uses is the learned, regularized code, whatever its width, and widening the code to $d$ does not remove the difference. We did not pursue the mechanism further.

**ELF flow family.** Applying the same residual-target ablation to ELF's conditional flow does not reproduce the DDPM effect. At $k$=16, residual-minus-direct control is $-0.18, -0.06, +0.06$ across three seeds, and at $k$=64 direct conditioning is already near parity. At full rank ($k$=$d$=512), control is equal while the residual target leaks more ($0.55/0.51$ versus $0.42/0.40$). We therefore treat the DDPM result as a generator-family-specific repair for condition under-use, not as a universal advantage of residualization.

## G Preregistration Summary

For the experiments reported here, confirmatory arms and decision rules were frozen before their corresponding runs; amendments were recorded before the affected batteries were read.

**Supported predictions.** Supported findings cover compressed-code control (R-P2, A4-A), coupling-sweep monotonicity and over-coupling (C-P1, C-P2), condition-start, flip, natural-dose, and interval effects (CS-P1, CS-P2, FD-P1, CI-P1), seed-2 ordering at $k = 32, 64$ (KS2-P2), external-classifier reproduction (X-P1), BeerAdvocate $k = 64$ replication (BR-P1, BR-P3), third-seed replications (S-P1, S-P3), and deeper-backbone robustness (M-P1, M-P2). Inference-time confirmations include direct-host rescue by classifier and neutral-code guidance (GR-P1, GR-P3), residual–guidance interactions (RC-P1, RC-P2, RC-P4), FK baseline and complementarity (FK-P0, FK-P2–FK-P4), BeerAdvocate control (BC-P0–BC-P5), and the external-quality comparison (Q-P1).

The independently matched equal-control test reaches approximately matched parity on fresh batches for both corpora (EC-P5). Residual training has lower normalized leakage (EC-P1), higher distinct-2 (EC-P2), and one rather than two denoiser evaluations per sampling step (EC-P4); see Table 5.

**Missed predictions and boundaries.** The decode-time additive path carried little control (R-P4, H-P1), direct conditioning responded at doubled dose (A5 D-P2), and residual training did not consistently reduce leakage in the original same-strength comparisons (R-P3, F-P2). Contrary to the initial prediction, input-sum restored parity (A4-C), motivating the coupling interpretation. Additional misses concerned realization and positional persistence (CS-P3, CS-P4), a wider natural-dose gap

(FD-P2), seed robustness (KS2-P1, S-P2), BeerAdvocate $k = 16$ recovery and the proposed collapse point (B-P1, B-P2, BR-P2), and ELF-family transfer (G-P1, G-P2, RW-P1, RW-P5). Inference-time misses include guidance cost (GR-P2), same-regime classifier guidance (CR-P1–CR-P3), the residual–guidance diversity companion (RC-P3), neutral-code FK (FK-P1), and the separately trained two-model classifier-free construction.

The equal-control generation-quality prediction (EC-P3) was not supported: direct+CFG has lower perplexity and higher coherence than residual training on both corpora, despite residual's leakage, diversity, and compute advantages.

**Amendments and exploratory analyses.** Recorded amendments include stochastic FK sampling, decoded-text rewards, best-of-$N$ comparators, $K = 4$ experiments, and the post-hoc matched-scale classifier-guidance comparison. Combined Linear-AcT $2 \times 2$ analyses in Appendix D were exploratory rather than preregistered. These amendments are documented in the preregistration ledger.