Beyond Steering Vector: Flow-Based Activation Steering for Inference-Time Intervention

Zehao Jin [*1] **Ruixuan Deng** [*1] **Junran Wang** [*1] **Xinjie Shen** [1] **Chao Zhang** [1]

Abstract

Activation steering has emerged as a promising alternative for controlling language-model behavior at inference time by modifying intermediate representations while keeping model parameters frozen. However, large-scale evaluations such as AxBench show that existing steering methods are often outperformed by simple in-context prompting and generalize poorly to unseen concepts. We hypothesize that these limitations arise from unvalidated simplifying assumptions shared across prior methods, which typically restrict steering interventions to fixed, single-step, position-invariant transforms. We propose **FLAS** (**Fl**ow-based **A**ctivation **S**teering), which learns a general, concept-conditioned velocity field $v_\theta(h, t, c)$ that transports unsteered activations to steered ones without relying on these assumptions. On AxBench, FLAS is the first learned method to consistently outperform prompting, reaching held-out harmonic means of 1.015 on Gemma-2-2B-IT and 1.113 on Gemma-2-9B-IT without per-concept tuning. Analysis of the learned flow shows curved, multi-step, token-varying trajectories, which suggests that previous hypotheses on activation space geometry might be incomplete. Our demo and code is available at https://flas-ai.github.io.

1. Introduction

Large language models have demonstrated strong capabilities across diverse tasks (Brown et al., 2020; Grattafiori et al., 2024; Team et al., 2024), yet reliably controlling their behavior to align with human preferences remains a persistent challenge (Anwar et al., 2024). Existing control mechanisms such as prompting and fine-tuning face limitations in robustness, cost, and side effects (Anwar et al., 2024; Hu et al., 2021; Kotha et al., 2024; Luo et al., 2025). Activation steering has emerged as a complementary alternative that offers lightweight, interpretable control across a growing range of behaviors (Park et al., 2025; Boxo et al., 2025; Lee et al., 2025; Frising & Balcells, 2026; Yang et al., 2025b) by modifying intermediate representations at inference time while leaving model parameters frozen (Giulianelli et al., 2021; Turner et al., 2024; Zou et al., 2025).

Despite these successes, AxBench (Wu et al., 2025), a benchmark that evaluates thousands of natural-language steering concepts, reveals a consistent limitation of existing steering methods. In particular, simple in-context prompting outperforms the tested steering methods, and increasing the scalar steering strength improves concept incorporation while monotonically degrading instruction following and fluency. The requirement for concept-specific strength tuning on a development set (Wu et al., 2025) limits the real-world application of previous steering methods.

We hypothesize that this performance gap stems from simplifying assumptions that most activation-steering approaches adopt at design time without rigorous validation. While most recent methods have relaxed the Linear Representation Hypothesis (Park et al., 2024; Panickssery et al., 2024; Templeton et al., 2024) by introducing adaptive transforms (Wu et al., 2024; Rodriguez et al., 2024; You et al., 2026; Raval et al., 2026; Oozeer et al., 2026; Sun et al., 2025), other assumptions persist widely (Table 1), typically restricting interventions to single-step, position-invariant transforms trained per concept on contrastive data. These assumptions define, for each method, a prescribed operator family that constrains both what information the intervention may use and how it may act on the activation. Individual methods relax one or more of these constraints while retaining the rest. Recent flow- and ODE-based formulations (Wang et al., 2025; Li et al., 2026; Zhao et al., 2026) loosen the single-step constraint by allowing multi-step, state-dependent trajectories, yet they retain the dependence on contrastive data and per-concept training. These restrictions shape how in-

[*]Equal contribution [1]Georgia Institute of Technology, Atlanta, GA, USA. Correspondence to: Zehao Jin <zjin350@gatech.edu>, Ruixuan Deng <rdeng62@gatech.edu>, Junran Wang <jwang3668@gatech.edu>, Xinjie Shen <xinjie@gatech.edu>, Chao Zhang <chaozhang@gatech.edu>.

Accepted to the ICML 2026 Workshop on Mechanistic Interpretability, Seoul, South Korea, 2026. Non-archival. Copyright 2026 by the author(s).

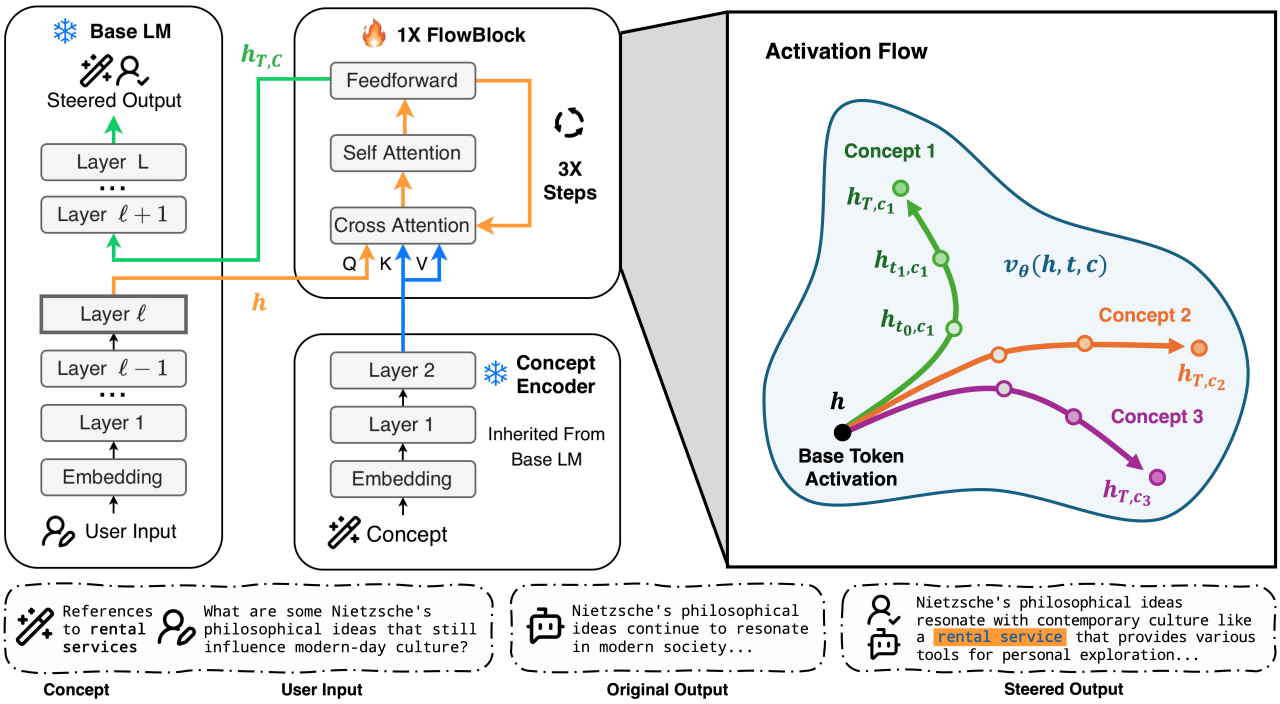

Figure 1. **FLAS model architecture overview.** The velocity field $v_\theta(h, t, c)$ transports activations at layer ℓ of a frozen base LM. A frozen concept encoder ϕ produces concept representations consumed by a single FlowBlock via cross-attention. The flow is integrated by N-step Euler, shared between training and inference, yielding a steered activation $h' = \varphi_T(h)$. The entire base language model (base LM) is frozen; only the FlowBlock parameters are trained.

terventions behave in practice and can limit the attainable trade-off between concept incorporation and instruction following.

To address these restrictions, we propose to learn a more expressive steering operator directly from data by introducing **FLAS** (**Fl**ow-based **A**ctivation **S**teering). FLAS replaces a fixed one-step intervention with a learned, time-conditioned velocity field $v_\theta(h, t, c)$ that transports an unsteered activation h to a steered activation $h' = \varphi_T(h)$ through N steps of Euler integration, conditioned on a natural-language concept description c. Because the velocity depends on the current activation state, the resulting intervention adapts as the activation evolves and, when integrated over multiple steps, can follow curved trajectories through activation space. Evaluating the velocity independently at each token position further allows the displacement to vary across a sequence. The method trains on positive examples under a standard language-modeling objective, without requiring contrastive pairs, and employs the flow time T as a continuous integration horizon that decouples intervention strength from direction.

Our contributions are as follows.

1. We propose **FLAS** (**Fl**ow-based **A**ctivation **S**teering), a concept-conditioned velocity field integrated by N-step Euler that enables adaptive, multi-step, position-sensitive steering trained on positive examples alone. The flow formulation recovers many single-step methods as special cases for $N = 1$ and fixed T.

2. FLAS is the first learned steering method to consistently outperform prompting on AxBench (Wu et al., 2025), achieving held-out HMean 1.015/1.113 (Gemma-2-2B/9B-IT) vs. prompting 0.762/1.091 and HyperSteer (Sun et al., 2025) 0.608/0.934, with $< 1/26$ the parameters. Performance remains stable across $T \in [0.5, 4.0]$ without per-concept tuning, and generalizes to held-out concepts with monotonic scaling at 16k without clear saturation.

3. The learned velocity field serves as an analysis probe of activation space, revealing curved, position-dependent, multi-step structure. Our method provides empirical evidence that effective steering requires nonlinear and position-sensitive interventions, suggesting that previous hypotheses on activation space geometry might be incomplete.

2. Related Work

Table 1 situates FLAS in the landscape of activation-steering methods along five structural axes.

Method	Intervention	Training data	Adaptive	Multi-step	Per-token	Zero-shot
DiffMean / CAA (Panickssery et al., 2024)	static vector	pos+neg	✗	✗	✗	✗
SAE features (Templeton et al., 2024)	feature clamp	N/A⋆	✗	✗	✗	✗
ReFT-r1 (Wu et al., 2024)	learned affine	**pos only**	✓	✗	✗	✗
Spherical Steer. (You et al., 2026)	geodesic rotation	pos+neg	✓	✗	✗	✗
Curveball Steer. (Raval et al., 2026)	kernel curve	pos+neg	✓	✗	✗	✗
AcT (Rodriguez et al., 2024)	OT affine	pos+neg	✓	✗	✓	✗
TruthFlow (Wang et al., 2025)	flow correction	pos+neg	✓	✓	✗	✗
K-Steering (Oozeer et al., 2026)	classifier gradient	pos+neg	✓	✓	✓	✗
FlowSteer (Li et al., 2026)	flow transport	pos+neg	✓	✓	✓	✗
ODESteer (Zhao et al., 2026)	barrier ODE	pos+neg	✓	✓	✓	✗
HyperSteer (Sun et al., 2025)	conditional vector	**pos only**	✓	✗	✗	✓
FLAS (ours)	**flow transport**	**pos only**	✓	✓	✓	✓

Table 1. **Structural comparison of activation-steering methods. Adaptive**: depends on current h. **Multi-step**: iterative integration. **Per-token**: uses inter-position context. **Zero-shot**: no per-concept retraining. **Training data**: "pos only" = concept-aligned responses only, "pos+neg" = additionally requires paired negatives. ⋆Relies on a pretrained sparse autoencoder for feature extraction.

Linear Activation Steering. Activation Addition (Turner et al., 2024) and CAA (Panickssery et al., 2024) each extract or optimize a fixed steering direction and add a scaled copy at a chosen layer. Recent work computes the displacement through learned mechanisms such as low-rank interventions (Wu et al., 2024) and cross-attention hypernetworks (Sun et al., 2025), but still produce a single displacement at inference time, and none have been reported to consistently surpass prompting on AxBench.

Concurrent Nonlinear and Flow-Based Steering. Activation Transport (Rodriguez et al., 2024), Curveball Steering (Raval et al., 2026), Spherical Steering (You et al., 2026), and Householder Pseudo-Rotation (Pham & Nguyen, 2024) introduce nonlinear single-step interventions ranging from affine maps to norm-preserving rotations, requiring paired source-target data. K-Steering (Oozeer et al., 2026), TruthFlow (Wang et al., 2025), FlowSteer (Li et al., 2026), and ODESteer (Zhao et al., 2026) adopt multi-step continuous-dynamics formulations, but each targets a single attribute and requires task-specific paired data. FLAS combines a concept-conditioned velocity field with zero-shot generalization via end-to-end LM-loss training on positive data only.

Flow Matching and Representation Geometry. Our velocity-field parameterization draws on flow matching (Lipman et al., 2023; Tong et al., 2024; Lipman et al., 2024) and its extensions to manifolds (Ben-Hamu et al., 2022) and latent spaces (Dao et al., 2023). Where flow matching transports noise to data, FLAS transports unsteered activations to steered ones under a downstream language-modeling objective rather than a flow-matching regression target. The manifold view of LLM representations (Modell et al., 2025; Valeriani et al., 2023; Mabrok, 2026; Facco et al., 2017; Yusupov et al., 2025) treats hidden states as lying on low-dimensional submanifolds, and our trajectory analyses in Sections 6.1 and 6.2 give a concrete picture of how a learned intervention traces on such a submanifold.

3. Method

3.1. Flow-Based Steering

Fix a pretrained language model with L layers and hidden width d. At a chosen layer ℓ, the forward pass produces activations $h \in \mathbb{R}^{n\times d}$, where n is the sequence length. Given a natural-language concept description c (e.g., a short phrase specifying the target behavior), we want to replace h with a steered version h' so that subsequent layers generate text exhibiting the concept while preserving instruction following and fluency.

We realize the map from h to h' as a learned, concept-conditioned flow. Let $\{\varphi_t\}_{t\in[0,T]}$ be a family of maps $\varphi_t : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ generated by a velocity field v_θ, defined by the ODE

$$\frac{d}{dt}\varphi_t(h) = v_\theta\big(\varphi_t(h),\, t,\, c\big), \tag{1}$$

with initial condition $\varphi_0(h) = h$. The steered activation is obtained by integrating the velocity field from 0 to T:

$$h' = \varphi_T(h) = h + \int_0^T v_\theta\big(\varphi_t(h),\, t,\, c\big)\, dt. \tag{2}$$

In practice, we approximate this integral using an N-step forward Euler method:

$$h_{k+1} = h_k + \frac{T}{N}\, v_\theta\left(h_k,\, \frac{kT}{N},\, c\right), \tag{3}$$

for $k = 0, \ldots, N-1$, with $h_0 = h$. The resulting h_N serves as a numerical approximation to $h' = \varphi_T(h)$ and is passed to layer $\ell + 1$ in place of h.

Three properties of v_θ together distinguish this formulation from previous steering methods. First, the map φ_t depends on the initial state h, so the flow adapts to different activations. Second, the time-dependent velocity field can prescribe different directions at each step along the integration path, producing curved trajectories. Third, v_θ is computed per token position, thus the steering trajectory varies per token.

Taken together, these properties make v_θ sufficiently expressive that the integral in Eq. 2 can in principle realize a multi-step transport from h to h'. As a consequence, our formulation naturally subsumes prior steering approaches as restricted instances of the velocity field. The standard additive formulation $h' = h + \alpha\delta(c)$ is recovered as the special case $v_\theta(h, t, c) = \delta(c)$ with $T = \alpha$.

3.2. FlowBlock Architecture and Forward Process

We instantiate v_θ with a transformer-style block, which we call a FlowBlock (Figure 1). To encode the concept description c into a sequence of vectors that the FlowBlock can attend to, we apply a concept encoder ϕ. By default ϕ reuses the token embedding and first few transformer layers of the base model, so that $\phi(c)$ inherits the early-layer features of the base model.

At step k, the FlowBlock takes the current activation h_k, the encoded concept $\phi(c)$, and the current time $t_k = kT/N$ as input. We first inject the time signal through a sinusoidal embedding,

$$\tilde{h}_k = h_k + \mathrm{TimeEmbed}(t_k). \tag{4}$$

Since c is a sequence of arbitrary length, the FlowBlock attends to it through cross-attention,

$$u_k = \mathrm{CrossAttn}\big(Q = \tilde{h}_k,\ K = \phi(c),\ V = \phi(c)\big), \tag{5}$$

whose keys and values are cached once and reused across N integration steps and decoding positions. A causal self-attention layer and a feedforward network then produce the per-step displacement,

$$\Delta h_k = \mathrm{Feedforward}\big(\mathrm{SelfAttn}(u_k)\big). \tag{6}$$

Iterating this procedure N times yields h_N. In practice, each component is wrapped with a residual connection and a learnable per-channel gate, and the update at each step is scaled by the Euler step size T/N. Full implementation details are included in Appendix B.

3.3. Training

To control the steering strength at inference, we use the flow time T as a scalar parameter. Under the Euler method (Eq. 3) with fixed N, increasing T scales the per-step updates and pushes activations further along their concept-specific trajectories.

For T to provide continuous control, the velocity field v_θ must remain valid across varying horizons. Unlike prior flow-based methods with a fixed training-time strength (Lipman et al., 2023), we enable training-free extrapolation at inference by exposing the model to a range of integration horizons during training. Like classifier-free guidance (Ho & Salimans, 2022), our approach enables dynamic strength control at inference, achieved by simply scaling the integration time of the learned flow.

We implement this by randomizing the integration horizon during training. At each training step we sample $T \sim \mathrm{Uniform}[T_{\min}, T_{\max}]$, run N Euler steps using Eq. 3, inject the resulting h_N at layer ℓ, and supervise with language-modeling cross-entropy on the output tokens,

$$\mathcal{L}_{\mathrm{LM}} = -\sum_{(x,y,c)\in\mathcal{D}} \sum_i \log p\big(y_i \mid y_{<i}, x;\, h_N\big), \tag{7}$$

where $\mathcal{D}$ is the training dataset, with each triple consisting of an input prompt x, a concept c to steer toward, and the desired output y that reflects steering toward c.

Since velocities for different concepts should point in distinct directions, we add a diversity penalty on the mean-pooled final-step velocities within each minibatch,

$$\begin{aligned} \mathcal{L}_{\mathrm{div}} &= \frac{1}{|\{(i,j) : c_i \neq c_j\}|} \sum_{i,j:\, c_i \neq c_j} \cos\big(\bar{v}_i,\ \bar{v}_j\big), \\ \bar{v}_i &= \frac{1}{P}\sum_{p=1}^{P} v_i^{(p)}, \end{aligned} \tag{8}$$

where p indexes token positions, $v_i^{(p)} = v_\theta(h_{N-1}^{(p)}, t_{N-1}, c_i)$ is the final-step velocity at position p for sample i. The total loss is $\mathcal{L}_{\mathrm{LM}} + \lambda\mathcal{L}_{\mathrm{div}}$ with $\lambda = 0.1$. Ablations in Sec. 5 confirm that this diversity penalty is important for steering quality, and especially for extrapolation along T (detailed discussion in Appendix D).

4. Experiments

Training Data and Base Model. We follow the protocol of AxBench (Wu et al., 2025) and train on Concept16k. Base models are Gemma-2-2B-IT and Gemma-2-9B-IT (Team et al., 2024), with steering at layer 20. We use a single FlowBlock (97.6M trainable parameters on 2B, 255M on 9B), with $N = 3$ Euler steps and $T \sim \mathrm{Uniform}[0.5, 2.0]$. The concept encoder is frozen and reuses the base model's token embedding and first two layers. Training details are included in Appendix A.

Evaluation Protocol. We follow the AxBench evaluation pipeline. GPT-4o-mini (OpenAI et al., 2024) scores each generation on Concept incorporation (C), Instruction following (I), and Fluency (F), with $C, I, F \in \{0, 1, 2\}$. The primary metric is the harmonic mean of the three scores: $\mathrm{HMean} = 3/(1/C + 1/I + 1/F) \in [0, 2]$. Held-in evaluates on concepts seen during training but with previously unseen prompts. Held-out is strictly zero-shot, evaluating on concepts never seen during training paired with unseen prompts. Evaluation details are included in Appendix E.

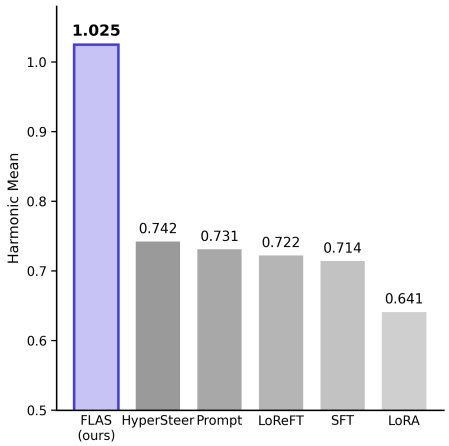

Figure 2. **Held-in results on Gemma-2-2B-IT, layer 20.** FLAS exceeds the in-context prompting baseline by $+0.294$ and HyperSteer by $+0.283$.

Method	Gemma-2-2B-IT		Gemma-2-9B-IT	
	Held-in	Held-out	Held-in	Held-out
Prompting	0.731	0.762	1.075	1.091
Fine-tuning				
LoReFT	0.722	—	0.777	—
SFT	0.714	—	—	—
LoRA	0.641	—	0.602	—
RePS	0.606	—	0.892	—
Activation Steering				
HyperSteer	0.742	0.608	1.091	0.934
ReFT-r1	0.509	—	0.630	—
AcT (All Layers)	0.187	—	0.161	—
DiffMean	0.178	—	0.322	—
SAE	0.151	—	0.191	—
AcT (Layer 20)	0.144	—	0.270	—
SAE-A	0.132	—	0.186	—
FLAS (ours)	**1.025**	**1.015**	**1.185**	**1.113**

Table 2. **Full steering results on AxBench.** Empty entries (—) indicate methods that do not support zero-shot steering. Baselines from AxBench (Wu et al., 2025) and HyperSteer (Sun et al., 2025). FLAS evaluated at fixed $T = 2$. The intervention happens at Layer 20 of both models.

4.1. Main Results

Table 2 and Figure 2 present the main results. All FLAS results are given using a single fixed flow time $T = 2$ with no per-concept tuning. On Gemma-2-2B-IT held-out evaluation, FLAS reaches a harmonic mean of 1.015, exceeding HyperSteer (0.608, $+0.407$) and in-context prompting (0.762, $+0.253$). On Gemma-2-9B-IT held-out evaluation, FLAS reaches the score of 1.113, above both in-context prompting (1.091, $+0.022$) and HyperSteer (0.934, $+0.179$). To illustrate the advantage of FLAS over in-context prompting, we provide case studies in Appendix I where FLAS succeeds while in-context prompting fails. Overall, FLAS incorporates concepts into outputs more naturally and flexibly, especially for complex concepts.

To further assess cross-model generalization, we additionally apply FLAS to Qwen3-4B-Instruct (Yang et al., 2025a) at layer 20 under the same training and evaluation pipeline, achieving a held-out harmonic mean of 0.960 (detailed in Appendix C). This demonstrates that FLAS generalizes across model families.

4.2. Concept Scaling

We investigate how FLAS performance scales with the number of training concepts. We train models on subsets of 9, 500, 1.9 k, 5.5 k, and the full 16 k concepts with identical hyperparameters, and evaluate on the same held-out concepts at $T = 2$. As shown in Figure 3, the held-out harmonic mean increases monotonically with the number of training concepts, surpassing the in-context prompting baseline between 1.9 k and 5.5 k concepts. The curve shows no sign of saturation at 16 k, suggesting further gains from larger concept pools.

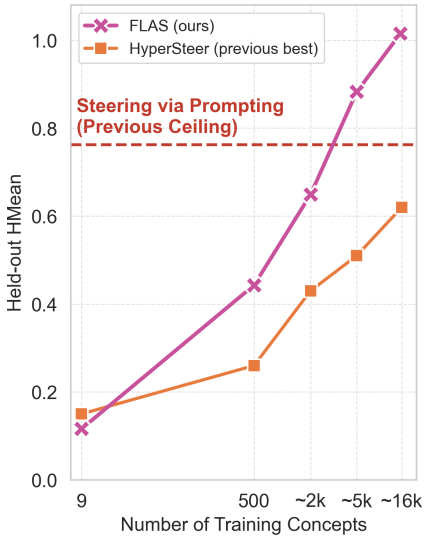

Figure 3. **Concept scaling.** Held-out harmonic mean versus the number of training concepts.

4.3. Flow Time Robustness

Activation steering typically involves a trade-off where increased concept incorporation degrades instruction following and fluency. Figure 4 contrasts FLAS with three baselines on Gemma-2-9B-IT: ReFT-r1, DiffMean, and AcT (Rodriguez et al., 2024) (reproduced at layer 20, see Appendix F). All three baselines collapse at higher strengths, while FLAS steadily improves concept score and maintains high instruction and fluency across the entire range.

This robustness is not an artifact of training data abundance. Figure 5 decomposes the score across $T \in [0.5, 4.0]$ for five concept pool sizes on Gemma-2-2B-IT, and the qualitative

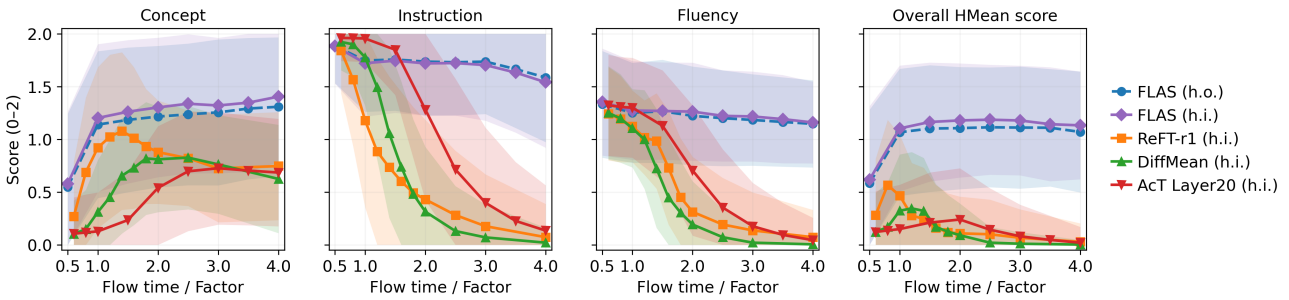

Figure 4. **Steering strength trade-off (Gemma-2-9B-IT).** Score decomposition across steering strengths for FLAS (held-out, h.o., in blue; held-in, h.i., in purple) and baselines (ReFT-r1, DiffMean, AcT). Shaded bands show ± 1 std, clipped to $[0, 2]$.

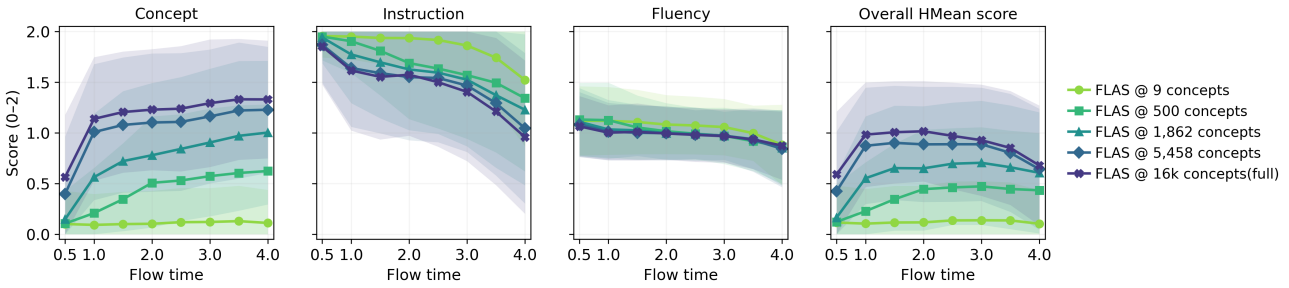

Figure 5. **Flow time across training-set sizes (Gemma-2-2B-IT held-out).** Score decomposition versus T for five concept scales. Shaded bands show ± 1 std, clipped to $[0, 2]$.

shape of the curves is preserved across scales. Increasing the training pool mainly raises concept score, while instruction and fluency remain roughly unchanged. In the data-scarce regime (500 or 1.9 k concepts), increasing T at inference time substantially boosts concept incorporation, suggesting that flow time can compensate for limited training data.

5. Ablations

We ablate the main design choices of FLAS on Concept16k held-out using Gemma-2-2B-IT at $T = 2$. The base configuration uses $B = 1$ FlowBlock, $N = 3$ Euler steps, with three phases enabled (cross-attention, self-attention, MLP), diversity loss, a frozen concept encoder, and weights initialized from the corresponding Gemma-2 layer. All scores are averaged over held-out concepts (10 prompts each). We report 95% bootstrap confidence intervals (10 000 resamples over concept-level means) and paired t-statistics against the base configuration.[1]

Model Architecture. Table 3 shows that the only ablation causing a large performance drop is disabling cross-attention ($t = -37.82$, $p < 0.001$), which removes the pathway for concept information to enter the activation stream. Disabling self-attention causes a moderate drop to 0.969 ($t = -2.19$, $p < 0.05$), indicating that inter-position coordination contributes. Removing the MLP causes a similar drop to 0.955 ($t = -3.05$, $p < 0.01$). The effect of adding FlowBlocks beyond $B = 1$ is statistically indistinguishable, confirming that the minimal single-block architecture is already sufficient for Concept16k dataset.

Training. We ablate the diversity loss and the warm-start initialization strategy during training. Removing the diversity loss degrades performance to 0.932 ($t = -4.41$, $p < 0.001$). We observe a severe degradation in held-out performance without the diversity loss, which we discuss in Appendix D. Replacing Gemma-2 weight warm-start with Xavier initialization drops performance to 0.968 ($t = -2.49$, $p < 0.05$), confirming that initializing from the base model aids optimization.

Intervention Layers. To verify our model's sensitivity to the choice of layer, we substitute layer 10 or layer 15 for layer 20. Results in Table 3 show that steering at layer 10 performs comparably to the base and layer 15 shows a moderate drop to 0.946. Both substantially outperform the prompting baseline at 0.762. This proves that FLAS is not sensitive to the choice of intervention layer.

Number of Integration Steps. Table 3 ablates the number of Euler steps. At $N = 1$ the flow reduces to a single adaptive displacement and performance drops significantly to 0.837 ($t = -9.56$, $p < 0.001$), but still exceeds prompting (0.762). Adding a second step recovers most of the remaining gap (0.970, $t = -2.59$, $p < 0.01$), and beyond $N = 3$ further steps yield no significant improvement. Three Euler steps are sufficient for the velocity field to capture the required curvature. We analyze this structure in Section 6.2.

[1] Significance: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

Configuration	HMean	95% CI	Paired t
Base ($B=1$, $N=3$)	1.015	[0.968, 1.060]	—
Architecture			
+1 FlowBlock ($B=2$)	1.009	[0.963, 1.051]	−0.34
+2 FlowBlocks ($B=3$)	0.996	[0.944, 1.044]	−1.06
Disable self-attention	0.969*	[0.922, 1.015]	−2.19
Disable MLP	0.955**	[0.905, 1.003]	−3.05
Disable cross-attention	0.109***	[0.078, 0.142]	−37.82
Training			
Xavier init	0.968*	[0.921, 1.012]	−2.49
Remove diversity loss	0.932***	[0.879, 0.982]	−4.41
Intervention layer			
Layer 10	1.044	[0.989, 1.096]	+1.22
Layer 15	0.946**	[0.884, 1.006]	−2.93
Integration steps (N)			
$N=1$	0.837***	[0.790, 0.884]	−9.56
$N=2$	0.970**	[0.928, 1.010]	−2.59
$N=4$	0.981	[0.936, 1.024]	−1.86
$N=5$	1.011	[0.962, 1.058]	−0.23
$N=10$	1.020	[0.974, 1.064]	+0.26

Table 3. **Ablations** (Concept16k held-out, $T=2$). HMean: harmonic mean of C/I/F. CI: 95% bootstrap over concept-level means. Paired t: versus base on the same held-out concepts.

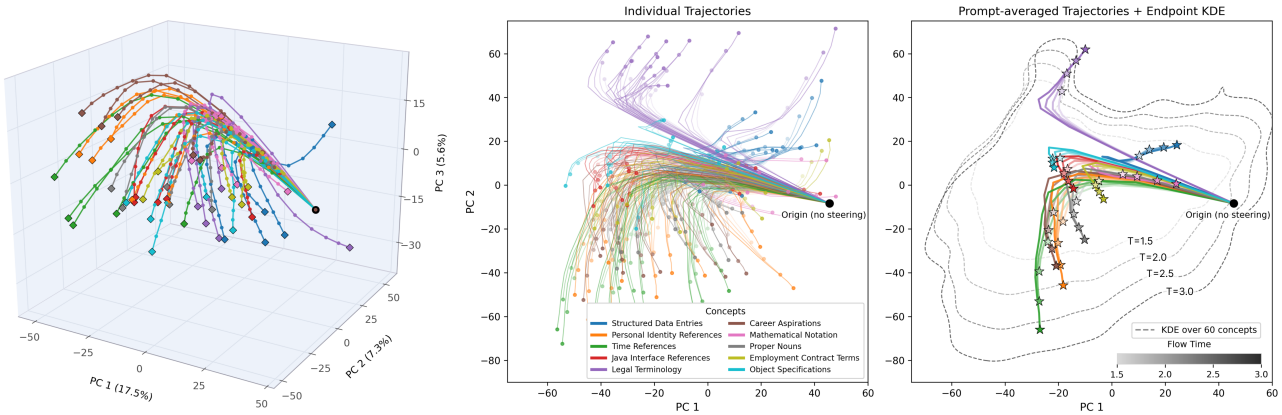

Figure 6. **Steering trajectories of the learned flow** ($N=10$). Color encodes concept identity and lightness encodes flow time T, with lighter tints corresponding to lower T. **Left:** 3D PCA projection of trajectories at $T=2$. **Middle:** per-concept, per-prompt 2D PCA trajectories at $T \in [1.5, 3.0]$. **Right:** prompt-averaged trajectories with dashed gray KDE contours showing the spread of 60 concepts at each T. Trajectories bend from a shared initial direction into concept-specific endpoint regions, and increasing T extends the displacement along each concept's direction.

6. The Geometry of Flow Steering

The velocity field of FLAS can be inspected to understand the steering trajectories. We use the $N=10$ model for the trajectory and per-step analyses, where the flow is exposed at high temporal resolution, and the $N=3$ model (our default configuration) for the per-token analysis. These three analyses show that effective activation steering requires curved, multi-step, token-varying interventions. Detailed settings of analysis experiments are included in Appendix G.

6.1. Steering Trajectories Are Curved

Figure 6 visualizes the flow trajectories projected onto the leading principal components of the displacement vectors across various concepts, prompts, and integration steps.

The trajectories are not straight lines. Every concept's path leaves the origin in a shared direction, executes a pronounced bend, and then enters a concept-specific region. Once the bend completes, T controls how far along the concept-specific direction the activation travels.

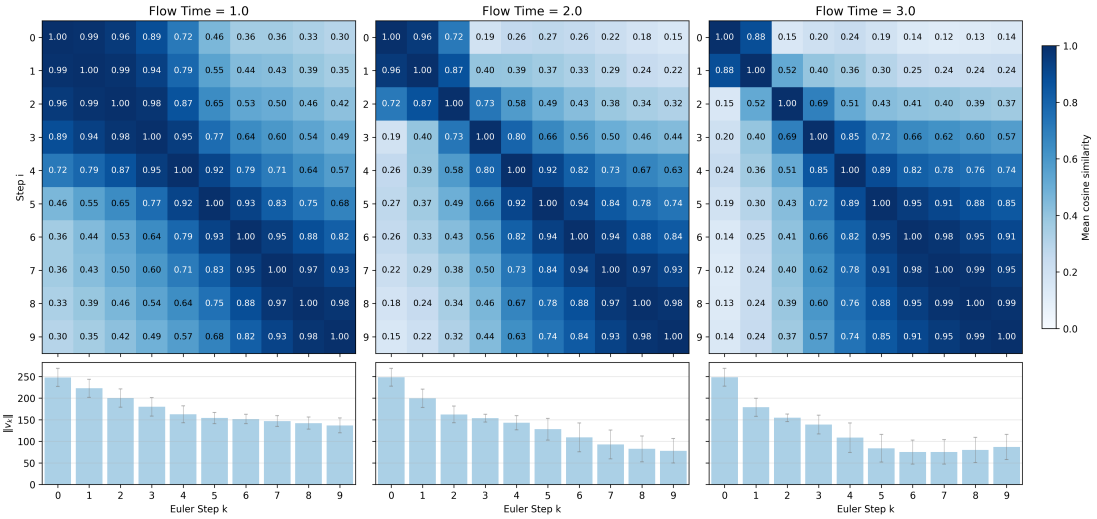

Figure 7. **Step-to-step velocity cosine and magnitude** ($N = 10$, $T \in \{1.0, 2.0, 3.0\}$). **Top:** 10×10 cosine matrix between Euler velocities. **Bottom:** mean $\|v\|$ per step.

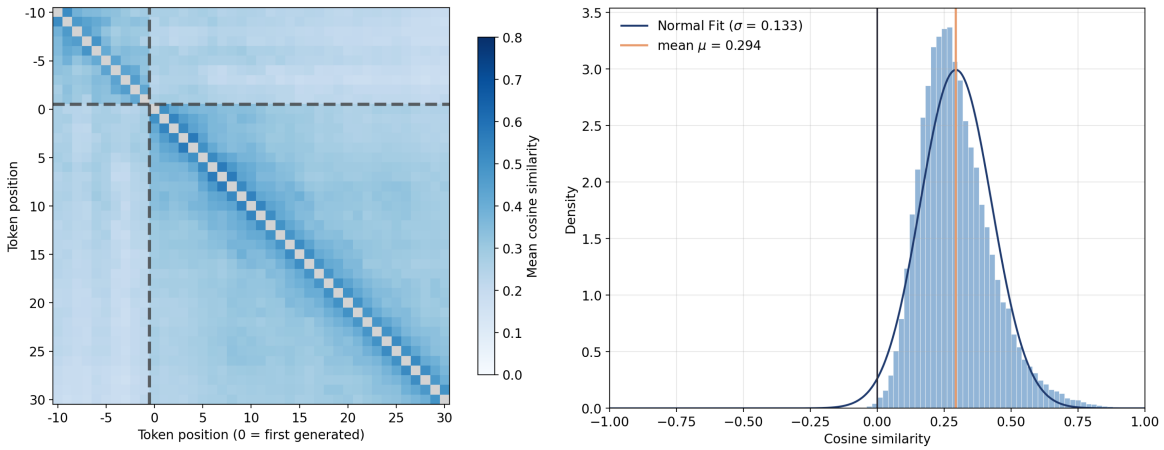

Figure 8. **Per-token displacement cosines** ($N = 3$, $T = 2$). **Left:** mean pairwise cosine of total displacements $h_N - h_0$ across token positions. **Right:** distribution of off-diagonal cosines ($\mu = 0.294$, $\sigma = 0.133$). Per-token steering is far from uniform.

6.2. The Learned Flow Requires Multiple Steps

Figure 7 quantifies the per-step structure of the learned flow. At larger flow times ($T = 2.0$ and $T = 3.0$), the late steps point in mutually consistent directions (cosine similarity > 0.7), while the early steps are markedly misaligned with these later directions (cosine similarity < 0.25). This separation between early and late step directions provides quantitative evidence that the bending observed in Figure 6 is a statistically robust phenomenon rather than an artifact of individual trajectories.

6.3. Per-Token Steering Is Non-Uniform

Most previous activation-steering methods apply the same displacement to every token position. FLAS evaluates the velocity field per position, and each token's total displacement is the sum of N Euler increments. Figure 8 shows the average pairwise cosine between per-token displacements is only 0.294 ± 0.133, far below the 1.0 that a position-invariant method produces. We observe that nearby tokens exhibit higher steering similarity, and that similarities within prompt tokens and within generated tokens are higher than across the two groups, revealing position-dependent structure.

7. Limitations and Future Work

Our evaluation focuses on AxBench because it provides large-scale natural-language concepts, allowing us to test FLAS on zero-shot extrapolation to unseen concepts. This scope gives a controlled evaluation of the main claim of FLAS, but it does not cover all uses of inference time intervention. Extending FLAS to broader concept collections is

an important direction for future work. The AxBench evaluation uses an automatic LM judge, which may introduce systematic biases. To assess the stability of the resulting comparisons, we report paired statistical tests across held out concepts and provide evaluation details in Appendix E.

FLAS introduces acceptable additional inference cost because it accepts arbitrary text concepts, which requires concept encoding and cross attention during steering. We quantify this overhead in Appendix H. Reducing latency is a future direction for deployment. The learned velocity field is also tied to a specific LM backbone, so a separate FlowBlock is trained for each base model. Our experiments intervene at a single layer, and future work can study cross layer composition and multi concept steering.

8. Conclusion

We presented **FLAS**, a flow-based activation-steering method that replaces the fixed, single-step interventions used by prior steering approaches with a learned, concept-conditioned velocity field integrated over multiple Euler steps. By relaxing the assumptions of position-invariance, single-step transport, and contrastive supervision, FLAS becomes the first learned steering method to consistently surpass in-context prompting on AxBench, achieving held-out harmonic means of 1.015 on Gemma-2-2B-IT and 1.113 on Gemma-2-9B-IT with a single fixed flow time and no per-concept tuning, while generalizing across model families.

Beyond benchmark performance, the learned velocity field can be inspected to understand steering trajectories. The trajectories we observe are curved, require multiple steps to resolve, and vary substantially across token positions. These properties suggest that the geometric assumptions underlying much of the prior steering literature are incomplete. We hope that treating activation interventions as flows rather than vectors opens a more faithful path toward controlling and understanding the internal computations of large language models.

References

Anwar, U., Saparov, A., Rando, J., Paleka, D., Turpin, M., et al. Foundational challenges in assuring alignment and safety of large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=oVTkOs8Pka. Survey Certification, Expert Certification.

Ben-Hamu, H., Cohen, S., Bose, J., Amos, B., Grover, A., et al. Matching normalizing flows and probability paths on manifolds, July 2022.

Boxo, G., Socha, R., Yoo, D., and Raval, S. Caught in the act: a mechanistic approach to detecting deception, 2025. URL https://arxiv.org/abs/2508.19505.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., et al. Language models are few-shot learners, July 2020.

Dao, Q., Phung, H., Nguyen, B., and Tran, A. Flow matching in latent space, July 2023.

Dubois, Y., Galambosi, B., Liang, P., and Hashimoto, T. B. Length-controlled alpacaeval: A simple way to debias automatic evaluators, March 2025.

Facco, E., d'Errico, M., Rodriguez, A., and Laio, A. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, 7 (1):12140, September 2017. ISSN 2045-2322. doi: 10.1038/s41598-017-11873-y.

Frising, M. and Balcells, D. Linear personality probing and steering in llms: A big five study, January 2026.

Giulianelli, M., Harding, J., Mohnert, F., Hupkes, D., and Zuidema, W. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information, 2021. URL https://arxiv.org/abs/1808.08079.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., et al. The llama 3 herd of models, November 2024.

Ho, J. and Salimans, T. Classifier-free diffusion guidance, July 2022.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., et al. Lora: Low-rank adaptation of large language models, October 2021.

Kotha, S., Springer, J. M., and Raghunathan, A. Understanding catastrophic forgetting in language models via implicit inference, April 2024.

Lee, B. W., Padhi, I., Ramamurthy, K. N., Miehling, E., Dognin, P., et al. Programming refusal with conditional activation steering, February 2025.

Li, Y., Bergner, B., Zhao, Y., Patil, V. P., Chen, B., et al. Steering large reasoning models towards concise reasoning via flow matching, February 2026.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling, February 2023.

Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., et al. Flow matching guide and code, December 2024.

Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., et al. An empirical study of catastrophic forgetting in large language models during continual fine-tuning, 2025. URL https://arxiv.org/abs/2308.08747.

Mabrok, M. A. Latent semantic manifolds in large language models, March 2026.

Modell, A., Rubin-Delanchy, P., and Whiteley, N. The origins of representation manifolds in large language models, May 2025.

Oozeer, N., Marks, L., Jain, S., Barez, F., and Abdullah, A. Beyond linear steering: Unified multi-attribute control for language models, April 2026.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., et al. Gpt-4 technical report, March 2024.

Panickssery, N., Gabrieli, N., Schulz, J., Tong, M., Hubinger, E., et al. Steering llama 2 via contrastive activation addition, July 2024.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models, July 2024.

Park, S., Du, X., Yeh, M.-H., Wang, H., and Li, Y. Steer llm latents for hallucination detection, 2025. URL https://arxiv.org/abs/2503.01917.

Pham, V.-C. and Nguyen, T. H. Householder pseudo-rotation: A novel approach to activation editing in llms with direction-magnitude perspective, December 2024.

Raval, S., Song, H. J., Wu, L., Harrasse, A., Phillips, J. M., et al. Curveball steering: The right direction to steer isn't always linear, March 2026.

Rodriguez, P., Blaas, A., Klein, M., Zappella, L., Apostoloff, N., et al. Controlling language and diffusion models by transporting activations, November 2024.

Sun, J., Baskaran, S., Wu, Z., Sklar, M., Potts, C., et al. Hypersteer: Activation steering at scale with hypernetworks, June 2025.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., et al. Gemma 2: Improving open language models at a practical size, October 2024.

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Tong, A., Fatras, K., Malkin, N., Huguet, G., Zhang, Y., et al. Improving and generalizing flow-based generative models with minibatch optimal transport, March 2024.

Turner, A. M., Thiergart, L., Leech, G., Udell, D., Vazquez, J. J., et al. Steering language models with activation engineering, October 2024.

Valeriani, L., Doimo, D., Cuturello, F., Laio, A., Ansuini, A., et al. The geometry of hidden representations of large transformer models, October 2023.

Wang, H., Cao, B., Cao, Y., and Chen, J. Truthflow: Truthful llm generation via representation flow correction, February 2025.

Wu, Z., Arora, A., Wang, Z., Geiger, A., Jurafsky, D., et al. Reft: Representation finetuning for language models, May 2024.

Wu, Z., Arora, A., Geiger, A., Wang, Z., Huang, J., et al. Axbench: Steering llms? even simple baselines outperform sparse autoencoders, March 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., et al. Qwen3 technical report. https://arxiv.org/abs/2505.09388v1, May 2025a.

Yang, S., Zhu, S., Liu, L., Hu, L., Li, M., et al. Exploring the personality traits of llms through latent features steering, 2025b. URL https://arxiv.org/abs/2410.10863.

You, Z., Deng, C., and Chen, H. Spherical steering: Geometry-aware activation rotation for language models, February 2026.

Yusupov, V., Maksimov, D., Alaeva, A., Vasileva, A., Antipina, A., et al. From internal representations to text quality: A geometric approach to llm evaluation, September 2025.

Zhao, H., Sun, H., Kong, J., Li, X., Wang, Q., et al. Odesteer: A unified ode-based steering framework for llm alignment, February 2026.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., et al. Representation engineering: A top-down approach to ai transparency, March 2025.

A. Training Details

Parameter	Value
Base model	Gemma-2-2B-IT / Gemma-2-9B-IT (frozen)
Steering layer ℓ	20
FlowBlock count B	1
FlowBlock trainable parameters	97.6M on 2B, 255M on 9B
Optimizer	AdamW with weight decay 0.01
Learning rate	5×10^{-5}
Gradient clipping	norm 1.0
Batch size	32 on 2B, 16 on 9B, gradient accumulation $\times 2$ on 9B
Max steps	80,000 (early-stopped on val LM loss)
Warmup steps	2,000
LR schedule	cosine with linear warmup
Flow time T at training	$\sim$ Uniform$[0.5, 2.0]$
Euler steps N	3
Diversity loss weight λ	0.1
Max sequence length	256
Max concept length	64
Validation frequency	every 500 steps
Checkpoint selection	lowest validation LM loss
Hardware	NVIDIA A100 & H100

Table 4. **Training hyperparameters (default setting).**

Data Format. Each training example is a triple of prompt, concept-target output, and concept text. The prompt is formatted with the Gemma chat template. Labels on prompt and padding positions are set to -100 so the LM loss covers only output tokens.

Causal Guarantees. Cross-attention uses the frozen concept encoder's output as keys and values, which depends only on the concept text and is independent of the generation. Self-attention uses a causal mask so the activation stream never attends to future positions. At inference, the concept representation is computed once and reused for every generated token.

B. Architecture Details

ConceptEncoder. Our model reuses the base LM's token embedding, first two decoder layers, and the final RMSNorm as our ConceptEncoder for natural-language concepts. All parameters are frozen during training and inferencing.

FlowBlock. The single FlowBlock applies three phases: cross-attention, causal self-attention, and gated MLP. Each phase starts with RMSNorm, applies its operation, passes through a second RMSNorm, and adds to the residual stream with a learnable per-channel gate initialized to 0.1. Cross-attention uses Gemma-2's grouped-query configuration with QK-normalization, logit soft-capping, and rotary embeddings.

Time Conditioning. Given a flow time t, we compute a sinusoidal embedding with 64 frequency pairs,

$$\tau(t)_k = \sin(t\,\omega_k), \quad \tau(t)_{64+k} = \cos(t\,\omega_k),$$
$$\omega_k = 10000^{-k/64}, \quad k = 0, \ldots, 63,$$

yielding $\tau(t) \in \mathbb{R}^{128}$. A two-layer MLP projects this to the model dimension,

$$e(t) = W_2\,\mathrm{SiLU}(W_1\tau(t) + b_1) + b_2,$$

with W_2 and b_2 zero-initialized so that $e(t) = 0$ at the start of training. The vector $e(t)$ is added to the activation h at the entry of each FlowBlock and broadcast across the sequence dimension.

Velocity Computation. Given h, c, and t, the time embedding is added to h. The FlowBlock then applies cross-attention (activations query concept representations), causal self-attention on the activation stream, and a gated feedforward pass. The velocity is $v_\theta(h_{\text{in}}, t, c) = h_{\text{out}} - h_{\text{in}}$.

Initialization Regime. The zero-initialized time-MLP output, the per-channel gates at 0.1, and the Gemma-2 weight initialization jointly ensure that the FlowBlock begins as a near-identity map.

C. FLAS on Qwen3

To check that FLAS transfers across architectures we re-run the minimal configuration on Qwen3-4B-Instruct-2507 (Yang et al., 2025a) as the frozen base. The training and evaluation pipeline are unchanged across backbones, and only the base LM and ConceptEncoder swap. The training and evaluation concepts of AxBench originally came from Gemma-2 SAEs. We do not re-extract concepts from Qwen3 SAEs, so the training and evaluation data are built from Gemma-2-2B feature directions.

Architectural Adaptations. FLAS inherits the base model's architecture, so porting to Qwen3 amounts to matching its design choices. We replace Gemma-2's RMSNorm with Qwen3's variant, switch the MLP from GeGLU with GELU-tanh to SwiGLU with SiLU, remove attention logit soft-capping, and drop the $\sqrt{d_{\text{model}}}$ embedding scaling in the ConceptEncoder. Qwen3 layers carry two RMSNorms rather than Gemma-2's four, so the pre-attention and pre-MLP norms are loaded from the source layer while the post-attention and post-MLP norms keep their default unit weights. The cross-attention inherits Qwen3-4B's GQA configuration with 32 query and 8 key-value heads, head dimension 128, hidden size 2560, RoPE base 5×10^6, full attention at every layer, and QK-normalization preserved.

Hyperparameters. We keep the minimal config of Section 4 except as listed in Table 5. Batch size is halved to fit a single A100-80GB and gradient accumulation restores the effective batch of 32, and the maximum step budget is reduced from $80{,}000$ to $60{,}000$. As with the Gemma runs, training is early-stopped on validation LM loss before

reaching this cap, and we report the best checkpoint. We keep the absolute layer index $\ell = 20$ for direct comparability, although this corresponds to roughly 77% depth on Gemma-2-2B (26 layers) versus 56% on Qwen3-4B (36 layers).

Parameter	Value
Base model	Qwen3-4B-Instruct-2507 (frozen)
Steering layer ℓ	20 of 36 (∼56% depth)
FlowBlock count B	1
FlowBlock trainable parameters	134M
ConceptEncoder	frozen, 590M
Batch size	16, grad. accum. ×2 (effective 32)
Max / warmup steps	60,000 / 1,000 (early-stopped on val LM loss)
Other hyperparameters	identical to Table 4

Table 5. **Hyperparameters changed for the Qwen3-4B port.**

Result. On the 100 held-out concepts FLAS reaches HMean 0.960 at $T = 2$, compared to 1.015 on Gemma-2-2B-IT under the same data and eval. Both substantially outperform the prompting baseline on Gemma-2-2B-IT at 0.762, suggesting fluent concept incorporation. Larger Qwen variants, Qwen-native concept supervision, and longer training are left to future work.

D. Diversity Loss

The diversity loss $\mathcal{L}_{\text{div}}$ defined in Eq. 8 penalizes cosine similarity between mean-pooled final-step velocities of different concepts within each minibatch. It prevents the velocity field from collapsing to a single concept-independent direction in the early stages of training, when the LM loss alone provides only a weak signal for distinguishing concepts. Removing it drops held-out HMean from 1.015 to 0.932 at $T = 2$ ($p < 0.001$, Table 3).

Figure 9 decomposes the score across $T \in [0.5, 4.0]$ on Gemma-2-2B-IT held-out concepts. Without $\mathcal{L}_{\text{div}}$ the concept score plateaus near 1.05 around $T \approx 1.5$ and then declines, while the full configuration climbs monotonically and reaches 1.33 at $T = 4$. At large flow times the LM-only variant also suffers a sharp collapse in all scores (especially, fluency score drops to around 0.2 at $T = 4$ versus 0.85 for the full configuration). This empirical analysis demonstrates that, under the default FLAS configuration, $\mathcal{L}_{\text{div}}$ yields substantial gains at large flow times, suggesting that explicitly penalizing inter-concept similarity enhances the model's ability to extrapolate concept intensity beyond the training regime.

E. Evaluation Protocol

Held-out concept selection. AxBench (Wu et al., 2025) defines a held-out evaluation protocol but does not publicly release the specific held-out concept list they use. Following their protocol, we exclude 500 concepts from the Concept16k training set prior to training using a deterministic random permutation. From these 500 held-out concepts we sample 100 at random for evaluation, and we similarly sample 100 held-in concepts from the remaining training pool. The same 100-concept splits are reused for every held-out and held-in number reported in this paper, which also allows the paired t-tests in Table 3 across ablation configurations. Both the 500-concept holdout and the 100-concept evaluation subsets are reproducible from our code release, and the exact concept-id files used for every result in this paper are shipped with the repository at `data/eval_c16k_ho100.json` and `data/eval_c16k_hi100.json`. For each concept we generate steered outputs on 10 AlpacaEval (Dubois et al., 2025) prompts with 256 max new tokens at temperature 1.0, yielding 1,000 generations per condition with no further subsampling. We validate below that this sample size provides a stable estimate of the full 500-concept population mean.

Sample-size stability. To verify that 100 concepts yield a stable estimate of the held-out mean, we evaluate the base configuration on the full 500-concept holdout at $T = 2$ (4,998 of 5,000 samples pass Azure's content filter, with 500 concepts retained). Table 6 partitions these 500 concepts into five disjoint subsets of 100 using different random seeds and reports the mean HMean of each subset. The five subset means span a range of only 0.030 and are statistically indistinguishable under one-way ANOVA ($F(4, 495) = 0.268$, $p = 0.90$). A 10,000-trial bootstrap that samples 100 concepts without replacement from the 500 confirms that any single draw falls within ± 0.038 of the population mean with 95% probability, yielding a bootstrap 95% interval of $[0.964, 1.041]$. All significant ablation effects in Table 3 exceed this sampling uncertainty, while the non-significant differences ($|\Delta| < 0.01$) fall well below the sampling SE and are correctly identified as null effects regardless of which 100 concepts are drawn.

Split	n	Mean	Std	SEM	95% CI
Seed 42	100	1.006	0.199	0.020	[0.967, 1.045]
Seed 1	100	1.023	0.238	0.024	[0.977, 1.070]
Seed 7	100	1.016	0.207	0.021	[0.975, 1.056]
Seed 100	100	1.010	0.241	0.024	[0.963, 1.057]
Seed 2024	100	0.994	0.184	0.018	[0.958, 1.030]
Full held-out	500	1.003	0.221	0.010	[0.984, 1.022]

Table 6. **Evaluation stability across 100-concept subsamples** (base configuration, Gemma-2-2B-IT, $T = 2$). Five disjoint random subsets of 100 concepts drawn from a 500-concept holdout. One-way ANOVA: $F(4, 495) = 0.268$, $p = 0.90$. All 10 pairwise Welch t-tests yield $p > 0.32$. Bootstrap 95% interval (10,000 draws of 100 without replacement): $[0.964, 1.041]$.

Judging. Each generation is scored by GPT-4o-mini (accessed via Azure OpenAI) on three axes: Concept incorpo-

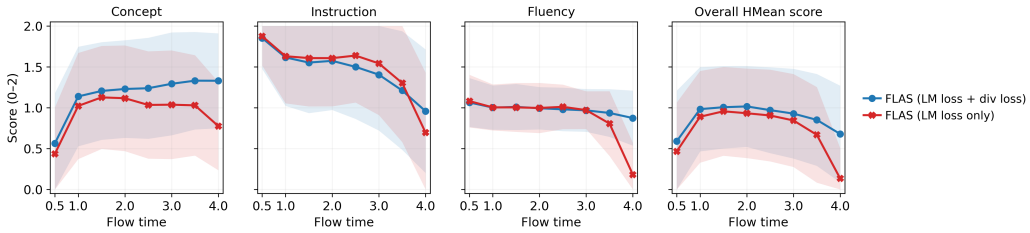

Figure 9. **Effect of the diversity loss on score decomposition versus flow time** (Gemma-2-2B-IT held-out). Removing $\mathcal{L}_{\text{div}}$ caps the concept score at moderate flow times and triggers a sharp collapse of instruction following and fluency at large T, while the full configuration maintains monotonic concept growth and graceful degradation across the full range. Shaded bands show ± 1 std.

ration (C), Instruction following (I), and Fluency (F), each on a 0–2 scale using the judge templates from AxBench (Wu et al., 2025). Azure OpenAI's content filter occasionally flags AxBench-style judge prompts as policy violations, causing a small fraction (<0.2%) of judge calls to fail. Because the failure rate is small and not correlated with score, these missing judgments do not affect the statistical conclusions.

Fixed Flow Time Versus Per-Concept Tuning. AxBench (Wu et al., 2025) and most prior methods report scores using a protocol that selects the best steering strength per concept on a development set. This per-concept optimization can mask sensitivity to the steering hyperparameter. All FLAS numbers use a single fixed flow time $T = 2$ with no per-concept tuning, which is a stronger evaluation setting. Baseline numbers for other methods are taken directly from AxBench (Wu et al., 2025) and HyperSteer (Sun et al., 2025) and use their respective evaluation protocols.

Variance Decomposition. Table 7 decomposes the total score variance into between-concept and within-concept components. For each run, σ_{conc} is the standard deviation across 100 concept-level means (each averaged over 10 prompts), and σ_{within} is the average of per-concept standard deviations. The sample-level standard deviation satisfies $\sigma_{\text{samp}} \approx \sqrt{\sigma^2_{\text{conc}} + \sigma^2_{\text{within}}}$. Across all runs with reasonable performance, $\sigma_{\text{within}} > \sigma_{\text{conc}}$, confirming that within-concept prompt-to-prompt variation exceeds between-concept variation and that concept-level aggregation (rather than sample-level) is the appropriate unit of analysis. The low σ_{within} for the no-cross-attention variant (0.205) and the 9-concept variant (0.223) reflects floor effects where most scores collapse near zero.

Table 7. **Variance decomposition at** $T = 2$ (Concept16k held-out, Gemma-2-2B-IT). σ_{samp}: std across $\sim$1000 samples (diagnostic only, overestimates due to within-concept correlation). σ_{conc}: std across 100 concept-level means. σ_{within}: mean of per-concept stds. SEM: $\sigma_{\text{conc}}/\sqrt{100}$, used for single-run uncertainty.

Run	σ_{samp}	σ_{conc}	σ_{within}	SEM
Base ($B=1$, $N=3$)	0.495	0.241	0.427	0.024
$N=1$	0.562	0.247	0.519	0.025
$N=2$	0.525	0.212	0.487	0.021
$N=4$	0.515	0.229	0.465	0.023
$N=5$	0.504	0.246	0.434	0.024
$N=10$	0.494	0.231	0.428	0.023
$B=3$	0.519	0.257	0.452	0.026
No MLP	0.528	0.252	0.467	0.025
No div loss	0.548	0.266	0.481	0.027
No self-attn	0.519	0.242	0.453	0.024
No cross-attn	0.344	0.164	0.205	0.016
$B=2$	0.508	0.229	0.450	0.023
Xavier init	0.526	0.233	0.476	0.023
Layer 10	0.554	0.281	0.478	0.028
Layer 15	0.588	0.316	0.493	0.032
9 concepts	0.362	0.166	0.223	0.017
500 concepts	0.586	0.364	0.430	0.036
1862 concepts	0.613	0.389	0.457	0.039
5458 concepts	0.568	0.305	0.479	0.031

F. AcT Baseline Reproduction

We reproduce Linear-AcT (Rodriguez et al., 2024) as a per-concept activation-steering baseline. For each concept, AcT fits a per-dimension affine map $f(h) = w \odot h + b$ between source (concept-absent) and target (concept-present) activation distributions, then steers via $h' = h + \lambda(f(h) - h)$ where λ is the intervention strength. Each concept is fit independently with no cross-concept generalization. We use 72 positive and 72 negative pairs from AxBench's training data, mean-pool over assistant-response tokens, and fit (w, b) in closed form via 1-D optimal transport followed by per-dimension linear regression, matching the official `ml-act` reference. We report two variants in Table 2: **AcT (Layer 20)** hooks only the AxBench reference layer, while **AcT (All Layers)** hooks every transformer block. Each (concept, prompt) pair is evaluated across 11 strengths $\lambda \in \{0.2, 0.4, 0.6, 0.8, 1.0, 1.5, 2.0, \ldots, 3.5, 4.0\}$ using 10 AlpacaEval prompts, with the best λ selected on a 5-prompt dev split.

On Gemma-2-2B-IT, all-layer AcT improves over single-layer (0.187 vs. 0.144), but on Gemma-2-9B-IT the same

Method	Prefill (ms)	Prefill slowdown	Gen (ms)	Gen slowdown	Steerer params
Gemma-2-2B-IT					
Base	35.0	1.00×	34.1	1.00×	—
DiffMean	35.9	∼1.00×	34.5	∼1.00×	—
SAE	36.5	∼1.00×	34.0	∼1.00×	—
HyperSteer	124.1	**3.54**×	34.8	∼1.00×	2.62B
FLAS $N=3$	55.1	**1.57**×	51.8	1.52×	97.6M
Gemma-2-9B-IT					
Base	57.0	1.00×	57.2	1.00×	—
DiffMean	59.8	∼1.00×	57.0	∼1.00×	—
SAE	59.6	∼1.00×	57.5	∼1.00×	—
HyperSteer	182.3	**3.20**×	57.9	∼1.00×	9.17B
FLAS $N=3$	93.6	**1.64**×	79.5	1.39×	255M

Table 8. **Inference latency** on a single A100 (batch size 1, 128 tokens, mean of 10 runs). Steerer params count the trainable FlowBlock only, with the frozen ConceptEncoder excluded. HyperSteer pays instead at prefill (3.2–3.5×) to run the concept through its 2.6–9.2B hypernetwork. FLAS has a lighter prefill (1.6×) but adds 1.4–1.5× per-token generation cost from the $N=3$ FlowBlock evaluations.

Method	Zero-shot?	Per-concept setup	Steerer params
Prompting	No	GPT-4o-mini API call†	0
DiffMean	No	Contrast-pair collection	d
ReFT-r1	No	Fine-tune (∼666 TFLOPs)‡	low-rank
HyperSteer	Yes	1 hypernetwork forward (22–34 layers)	2.6 / 9.2B
FLAS ($N=3$)	Yes	1 encoder forward (2 layer of base LM)	97.6 / 255M

Table 9. **Cost structure comparison.** FLAS steerer params = FlowBlock + frozen ConceptEncoder. †AxBench prompting uses GPT-4o-mini to generate optimized per-concept steering prompts (Wu et al., 2025). ‡Per-concept ReFT cost from Sun et al. (2025).

setup degrades performance (0.161 vs. 0.270). We report both variants to make this sensitivity explicit. The CIF tradeoff plot in Figure 4 shows the AcT (Layer 20) curve on Gemma-2-9B-IT.

G. Analysis Details

Trajectory Analysis for Section 6.1. Computed on the Concept16k $N=10$ checkpoint. For each (concept, prompt, flow time) triple, the base LM greedy-generates 40 continuation tokens from the steered model, and the trained flow is integrated from $t=0$ to $t=T$ using 10 Euler sub-steps, yielding 11 activation states (the initial state plus one per sub-step). Each state is mean-pooled across the 40 generated-token positions to produce a single d-dimensional vector, and the step-0 vector is subtracted to form a displacement trajectory in hidden space. PCA is fitted on the full pool of displacement vectors from 60 concepts (10 drawn as colored trajectories in the figure and are the same as AxBench Concept10, 50 used only for PCA fitting and KDE computation), 10 AlpacaEval prompts per concept, and 8 flow times $T \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$. The 2D panels display four flow times $T \in \{1.5, 2.0, 2.5, 3.0\}$ for the 10 explicit concepts, with color encoding concept identity and lightness encoding T. The dashed KDE contours in the right panel are computed over 60 concepts (600 endpoints per flow time). The 3D panel uses the top three principal components from the same PCA basis, restricted to $T=2$ and 5 prompts per concept for legibility.

Step-Cosine Analysis for Section 6.2. Computed on the Concept16k $N=10$ checkpoint at $T \in \{1.0, 2.0, 3.0\}$. For each concept-prompt pair we run steered generation and capture the ten per-step velocities $v_0, \ldots, v_9$ at each of the first 40 tokens. The 10×10 cosine matrix is averaged over $10\times10\times40 = 4000$ samples per flow time.

Per-Token Analysis for Section 6.3. Computed on the Concept16k $N = 3$ main checkpoint at $T = 2$. For each of 100 held-out concept-prompt pairs we sum the $N = 3$ per-step Euler increments at each token position to obtain the total displacement $h_N - h_0$ per position, then compute pairwise cosines between positions and aggregate on a prompt-relative index in which position 0 is the first generated token and negative indices are the last prompt-content tokens.

H. Computational Cost

Activation-steering methods distribute computational cost unevenly across three phases: one-time training, per-concept setup when switching to a new concept at deployment, and per-token overhead during generation. Methods that appear lightweight at generation time often carry substantial cost in earlier phases.

Inference Overhead. Table 8 compares inference latency across methods on Gemma-2-2B-IT and Gemma-2-9B-IT (single A100, batch size 1, 128 generated tokens, mean of 10 runs). Static-vector methods (DiffMean, SAE) add negligible overhead in both prefill and generation. HyperSteer

and FLAS, the two zero-shot methods, present complementary cost profiles. HyperSteer's 22/34-layer hypernetwork (22 for 2B, 34 for 9B) has a large prefill overhead ($3.54\times$ on 2B and $3.20\times$ on 9B), but adds no per-token generation cost because the steering vector is computed once and applied as a single addition. FLAS uses a single FlowBlock and has a lighter prefill and smaller memory footprint, but adds per-token generation latency because we have to compute steering on each new token.

The per-token generation overhead is the **principal and acceptable** computational cost of FLAS. It arises because the FlowBlock must be evaluated at each Euler step for each generated token, whereas static-displacement methods apply a pre-computed vector. This cost buys the state-dependent, multi-step, per-token expressivity that drives the quality gains in Table 2. The overhead ratio decreases on larger models (from $1.52\times$ on 2B to $1.39\times$ on 9B) because the base-model forward pass dominates the total cost. **Note that the current implementation has not been optimized for inference speed.** A single FlowBlock is architecturally equivalent to one additional transformer layer, and with standard optimizations (fused kernels, KV-cache reuse across Euler steps) we expect the per-token overhead to decrease to roughly 25–30% on 2B and 18–22% on 9B.

Cost Structure Across Methods. Table 9 summarizes the deployment cost profile. Static-vector methods achieve near-zero per-token cost but require per-concept offline computation that does not generalize: DiffMean needs contrast-pair activations, SAE steering needs feature selection, and ReFT-r1 needs per-concept fine-tuning at $\sim$666 TFLOPs per concept (Sun et al., 2025). HyperSteer and FLAS both enable zero-shot steering, but HyperSteer's hypernetwork is a modified copy of the full base model with cross-attention in every decoder block: 22 layers and $\sim$2.6B parameters on Gemma-2-2B-IT, 34 layers and $\sim$9.2B parameters on Gemma-2-9B (Sun et al., 2025). FLAS uses a single FlowBlock (97.6M on 2B, 255M on 9B) plus a frozen 2-layer ConceptEncoder, with only the FlowBlock parameters trained, roughly $1/27$ the trainable parameter count of HyperSteer on 2B.

In-context prompting appears cost-free but involves hidden setup cost. AxBench's prompting baseline calls GPT-4o-mini to synthesize an optimized steering prompt for each concept, using a meta-prompt that instructs the external model to craft task-specific instructions and optionally generate in-context examples (Wu et al., 2025). This introduces a per-concept API cost and a dependency on a more capable model, neither of which is reflected in per-token latency measurements.

I. Case Study: FLAS vs. In-Context Prompting

We present qualitative examples comparing three conditions: (1) the **Base** model (Gemma-2-2B-IT, unsteered), (2) **FLAS** (our method, $T = 2$, $N = 3$), and (3) **In-Context Prompting** (the AxBench prompting baseline, where GPT-4o-mini synthesizes a steering prompt prepended to the user instruction). Each example shows the target concept, the user instruction, the GPT-4o-mini-generated steering prompt, and model outputs truncated to 128 tokens (generated with max 256 new tokens at temperature 1.0). Scores are reported as C / I / F (Concept incorporation / Instruction following / Fluency, each 0–2) with the harmonic mean (HM). In the FLAS outputs, highlighted text highlights concept-relevant phrases. Emojis present in the original model outputs have been removed for typesetting.

Case 1: Numerical Data Formatting × Everyday Instruction

Target Concept: *numbers and their related formatting in data representation*

User Instruction: How do I take care of a wooden table?

In-Context Steering Prompt (GPT-4o-mini): When responding to questions or prompts, please incorporate content related to numbers and their formatting in data representation. This includes discussing concepts such as numerical precision, data types (integers, floats), formatting styles (currency, percentages), and ways to visually represent data (tables, charts). Even if the response seems unrelated to numbers, make sure to reference these concepts in some capacity. For example, if asked about a historical event, you could mention the significance of dates and their formatting in timelines, or if discussing a literary theme, reference how statistics can support arguments.

Base Model C=0 I=2 F=1 HM=0.00

Taking care of a wooden table is simple but maintaining its beauty and longevity requires a bit of effort. Here's a comprehensive guide: **1. Regular Cleaning:** * **Dust frequently:** Wipe down the surface with a soft, dry cloth or microfiber duster daily to prevent dust buildup. * **Avoid abrasive cleaners:** Do not use harsh abrasive cleaners, harsh chemicals, or scouring powders as they can damage the finish of your table. **2. Deeper Cleaning:** * **Mild soap and water:** Use warm, soapy water and a gentle sponge to wipe down the entire surface. For stubborn spills or stains, add a small…

FLAS (Ours) C=2 I=2 F=1 HM=1.50

To ensure longevity, consider the basic units of wooden care as numbers: 1 for oil application, 2 for cleaning the surface, 3 for polishing to protect against scratches. Each of these can be formatted into specific actions: 1. Apply thin layers of oil every 3-6 months; 2. Use warm water with a microfiber cloth for daily spills avoid abrasive cleaners. 3. Apply a finish with a polish pencil or rag every 8-12 weeks for further protection. Additionally, ensure proper humidity levels, often quantified on a relative humidity scale of 30-50%, to preserve the integrity of the material. Regular monitoring…

In-Context Prompting C=0 I=2 F=1 HM=0.00

To take care of your wooden table, you want to prevent grime and scratches from forming while making sure your beautiful wood ages gracefully. Just like a number's precise digits can enhance its meaning, you should focus on keeping the table clean across all its parts. * **Regular Cleaning:** Dust is the enemy of beautiful wood. A simple damp cloth is usually enough to wipe away daily dust or fingerprints. For stubborn stains, use a mild, wood-safe cleaner, and always test it on a hidden part first. * **Protect Your Investment:** Think of wood as a long-term asset (like a…

Analysis. FLAS naturally embeds numerical formatting throughout the response, using numbered steps, percentage ranges, time intervals, and data visualization references as integral parts of the care instructions. The prompting baseline attempts superficial analogies ("like a number's precise digits", "like a dataset") that the judge does not recognize as genuine concept incorporation.

Case 2: Numerical References × Science Explanation

Target Concept: *numerical references or identifiers*

User Instruction: Why can't you see the stars in the city, but you can in the countryside?

In-Context Steering Prompt (GPT-4o-mini): Please respond to the following inquiries while incorporating numerical references or identifiers in your answers. Ensure that these references are woven into the content, even if they seem tangential or unrelated to the main question. For example, if discussing a historical event, you might mention the year it occurred or significant statistics related to it.

Now, proceed with your response.

Base Model C=0 I=2 F=1 HM=0.00

You can see the stars much more clearly in the countryside than in the city! Here's why: **It's all about Light Pollution:** * **The city's light:** Cities are bright, filled with artificial light from streetlamps, buildings, car headlights, and billboards. This strong, bright light washes out the faint stars from the night sky. The light acts like a barrier, blocking your view. * **The countryside's darkness:** In the countryside, the light pollution is minimal to non-existent. The lack of artificial light allows light to travel freely skyward and prevents it from blocking our vision of the stars. **The Science behind…

FLAS (Ours) C=1 I=2 F=1 HM=1.20

The reason you cant see stars in the citys night sky often lies in the data of urban light pollution; it's mostly about the decimal reduction of clarity, as recorded in urban science reports. Counting lights conducted in major urban areas illustrates a 90% reduction in visibility, noted in articles from celestial astronomy advocates. The National Oceanic and Atmospheric Administration (NOAA) explains that 181,000 artificial lights illuminate the sky at night, emphasizing how $1,200 in damages to the environment cost the city in energy annually through miscalculations. In comparisons, rural areas might exhibit 75 stars, highlighted by studies,…

In-Context Prompting C=0 I=2 F=1 HM=0.00

The reason you can see the stars more clearly in the countryside than in a city is primarily due to **light pollution**. Light pollution is any type of artificial light that is too bright or directed in a way that scatters in the atmosphere. Here at the **International Astronomical Union (IAU)**, we formally recognize the impact of light pollution as a significant contributor to the decline in how easily we can see stars. This phenomenon of light pollution has a significant impact on astronomy: research and observation, as well as our overall understanding of the universe. Because of the extensive…

Analysis. FLAS weaves concrete numerical identifiers (90%, 181,000 lights, 75 stars) throughout the explanation while preserving the scientific content about light pollution. The prompting baseline produces a standard explanation without incorporating numerical references, failing to satisfy the concept requirement.

Case 3: Time Indicators × Business Proposal

Target Concept: *phrases indicating time indicators or duration*

User Instruction: "I am trying to win over a new client for my writing services and skinny brown dog media to as as a ghost writer for their book Unbreakable Confidence. Can you help me write a persuasive proposal that highlights the benefits and value of having a editor/publisher"

In-Context Steering Prompt (GPT-4o-mini): When generating responses, please ensure to include time indicators or duration phrases, such as "in the past," "for a long time," "recently," "over the years," "shortly," or "in the future." Even if the response doesn't directly relate to time, incorporate these phrases to enrich the content. For example, if asked about a book, you might say, "Recently, I read a fascinating book that explores themes of time and change over the years." Or, in response to a question about a historical event, you could say, "This event took place in the past but continues to influence our society today.

Base Model C=0 I=2 F=1 HM=0.00

Unbreakable Confidence: Fueling Your Book's Power Through Partnership **Dear [Client Name],** My name is [Your Name] and I'm a freelance writer and founder of Skinny Brown Dog Media. It's thrilling to connect with you about "Unbreakable Confidence," your upcoming project, and how I can help transform it into a truly impactful book that resonates with your readers. I know the journey to publishing a book is challenging finding the right voice, crafting engaging stories, and navigating the world of publication itself. That's where Skinny Brown Dog Media steps in. **Here's what we bring to the table:** **1. Crafting…

FLAS (Ours) C=1 I=2 F=1 HM=1.20

Subject: Achieving Unbreakable Confidence Through Powerful Writing Dear [Client's Name], Over the years, Skinny Brown Dog Media has consistently helped individuals like you achieve their literary goals. We believe that Unbreakable Confidence has the potential to resonate deeply and inspire countless readers. As you navigate the journey of self-publishing, entrusting the right elements can make all the difference. Our team's experience in editorial services can propel your book into the forefront of your target audience, ensuring it stands out long after the initial moments of interest. It is within this timeframe that we offer our expertise to guide you through…

In-Context Prompting C=0 I=2 F=1 HM=0.00

Here's a persuasive proposal template, incorporating time indicators to provide a more impactful feel: **Subject: Unbreakable Confidence: Empowering Your Story Through Expert Editing and Publishing** Dear [Client Name], It's exciting to be considering the next chapter for "Unbreakable Confidence"a book that promises to inspire and empower readers. Many authors have a vivid story that deserves to reach a wide audience. But bringing your vision to life goes beyond just the writing. As a boutique publishing and editing company, Skinny Brown Dog Media understands the unique demands of self-publishing and the invaluable power of professional assistance. For years, we've helped authors…

Analysis. FLAS distributes temporal phrases naturally across the proposal ("over the years," "several phases," "throughout the weeks," "the final stages"). These function as credibility and process markers rather than forced insertions. The prompting baseline produces a proposal with almost no temporal language despite explicit instructions.

Case 4: Punctuation/Formatting × Lifestyle Tips

Target Concept: *punctuation and formatting markers in the text*

User Instruction: You can easily make your holidays more eco-friendly by following these simple steps:

In-Context Steering Prompt (GPT-4o-mini): When responding, please incorporate references to punctuation and formatting markers, emphasizing their role in shaping the structure and clarity of text. Even if the topic seems unrelated, make sure to include a mention of how punctuation or formatting affects communication. For example, if discussing a topic like storytelling, you might say, "Effective storytelling relies on punctuation to convey emotions, such as using ellipses (...) to suggest a pause or uncertainty." Remember to creatively integrate this concept throughout your answers.

Base Model C=0 I=2 F=2 HM=0.00

Green Up Your Holidays: Simple Steps for an Eco-Friendly Season Holidays are meant to be joyful and relaxing, but they often come with a hefty environmental impact. We feel you, but fret not! Luckily, adopting a few eco-conscious choices can help make your celebrations greener. **Before the Holiday Rush:** * **Rethink Your Gifts:** * **Experiences over Stuff:** Consider giving experiences (concerts, museum tickets, cooking classes) instead of physical gifts. * **Support Local Businesses:** Opt for handmade or locally sourced gifts. This reduces transportation impact and supports your community. * **Regift or Repurpose:** If you have to gift, check for…

FLAS (Ours) C=1 I=2 F=1 HM=1.20

Make your holidays a *season* of joy! Use **commas** to separate actions, and highlight *bolded* steps. **Holidays** shouldnt harm the planet. Check out the list: 1. **Prioritize locally sourced gifts** ; consider supporting small businesses. This reduces **emissions**! Also, avoiding mass-produced items, crafts, or secondhand options can lessen waste. 2. **Reduce excess** - a few *exclamation marks* emphasize this! Think twice before buying *shower gels* simple *scents* suffice. Remember, the fewer *packets* of snacks, the better. 3. **Plan your trip** wisely: choose **eco-friendly accommodations**! Reading *labels*, both on **groceries** and **hotels**, can reveal **sustainable* *practices*. Remember to…

In-Context Prompting C=0 I=2 F=1 HM=0.00

You can easily make your holidays more eco-friendly by following these simple steps: 1. **Plan ahead:** Before you embark on your festive escapades, **conduct your research.** Explore sustainable transportation options, such as train travel or even cycling to minimize your carbon footprint. 2. **Pack light:** Carry only the essentials. **Think minimal!** Reducing the amount of luggage you bring to your destination **streamlines both travel logistics and your personal footprint.** 3. **Choose eco-friendly accommodation:** Seek out hotels or vacation rentals that prioritize sustainability. **Seek certifications!** Look for labels signifying environmentally conscious practices. They might utilize renewable…

Analysis. FLAS explicitly names punctuation marks (commas, periods, exclamation marks, dashes) within the eco-tips, calling attention to them as formatting devices while still delivering the instructed content. The prompting baseline produces standard eco-tips with no meta-references to punctuation or formatting.

Case 5: Numerical References × Philosophy

Target Concept: *numerical references or identifiers*

User Instruction: What are some Nietzsche's philosophical ideas that still influence modern-day culture?

In-Context Steering Prompt (GPT-4o-mini): Please respond to the following inquiries while incorporating numerical references or identifiers in your answers. Ensure that these references are woven into the content, even if they seem tangential or unrelated to the main question. For example, if discussing a historical event, you might mention the year it occurred or significant statistics related to it.

Now, proceed with your response.

Base Model C=0 I=2 F=1 HM=0.00

Nietzsche, with his provocative and often complex teachings, has left an undeniable mark on contemporary culture. Here are some of his philosophical ideas that still resonate today: **1. The Will to Power:** This is perhaps Nietzsche's most iconic and influential concept. **He didn't believe it's a literal power over others**, but rather the inherent human drive to self-affirmation and striving. * **Influence:** This drives entrepreneurialism, ambition in the workplace, and the pursuit of personal meaning. It's seen in how art, literature, and science offer ways to expand our own power and understanding. **2. Beyond Good and Evil:** Nietzsche challenged traditional…

FLAS (Ours) C=1 I=2 F=1 HM=1.20

Friedrich Nietzsche's philosophical concepts can be numerically expressed through the High-Impact-Recall-Source (HIRS) framework, where his ideas resonate with cultural identifiers such as Superman, Dionysianism, and Uber-Man. The 'S' score signifies strength, portraying the essential authenticity that defines much modern thought. The 'A' score might denote approachability, as his ideas often are cited in dialogues or biographical accounts. A comparative assessment, like "A10B5," illustrates the rating system's numerical alignment. For instance, Nietzsche's concept of the 'Übermensch' (connotes a superior being or person) influences modern narratives by portraying self-actualization and existential identity. Consider figures such as Tim Ferris, who emphasizes goal-setting and…

In-Context Prompting C=0 I=2 F=1 HM=0.00

Friedrich Nietzsche, a towering figure in 19th-century German philosophy, continues to resonate deeply in the contemporary cultural landscape. Though his work can appear complex and challenging, his ideas on morality, value, perspectivism, and the concept of "Eternal Recurrence" have profoundly influenced modern thought and culture in numerous ways. One of his core ideas, the concept of **will to power**, has become a central theme in several modern literary works and films. This concept, central to Nietzsche's concept of the master-slave dialectic, suggests that…

Analysis. FLAS invents a numerical framework (HIRS scores, "A10B5" identifiers) that satisfies the concept incorporation criterion by embedding numerical references throughout the philosophical discussion. The prompting baseline produces a standard overview without systematic numerical identifiers.

Case 6: Communication Tips × Programming Q&A

Target Concept: *words and phrases related to guidance and tips for communication and social interactions*

User Instruction: Answer the following question.

How do I compare strings in Java?
I've been using the == operator in my program to compare all my strings so far. However, I ran into a bug, changed one of them into .equals() instead, and it fixed the bug.
Is == bad? When should it and should it not be used? What's the difference?

In-Context Steering Prompt (GPT-4o-mini): You are a communication coach. When responding to questions, incorporate words and phrases related to guidance and tips for effective communication and social interactions, regardless of the topic. For example, if asked about a specific situation like a job interview, you might say, "In addition to preparing your answers, remember to maintain eye contact and listen actively to engage effectively." Always weave in practical communication advice, even if it seems tangential to the main question.

Base Model C=0 I=2 F=2 HM=0.00

You're right to be cautious about using just 'equals()' for string comparison in Java. While '==' can sometimes be convenient, 'equals()' is the ideal way to compare strings in most cases. Let's break down how they work and why. **Understanding '==' (Equals Operator)** * **Purpose:** The '==' operator in Java checks for *reference equality*. This means it essentially asks: "Does this object reference point to the *same* memory location as that object?" * **How it works:** * Two string variables that point to the same object in memory will be considered equal…

FLAS (Ours) C=1 I=2 F=1 HM=1.20

Using the == operator to compare strings in Java can sometimes feel like navigating a maze with poorly marked paths. The == operator acts as a guide, but misunderstandings may lead you astray, much like a conversation where intentions are overlooked. When bridging personal opinions with technical requirements, embracing a dialogue about what makes the most sense akin to articulating your thoughts clearly is essential. Consider utilizing the method .equals() when your strings do not merely reference the same character sequence; instead, you need clarity and precision. With .equals(), communicate with intentionality, ensuring your thoughts resonate accurately. Just. . .

In-Context Prompting C=0 I=2 F=1 HM=0.00

You touched upon a crucial concept in programming that can often trip up even seasoned developers! Let's dive into how to handle string comparisons in Java, and understand the difference between '==' and '.equals()' Let's break it down: * **The '==' Operator:** This operator simply checks for **object identity**. When you use '==', Java examines both strings for the same memory locations ('objects'). This means that while '=="name"' checks if the strings refer to the exact same object in memory, any string changes will cause the '==' operator to return *False*. * **The '.equals()' Method:** This method specifically checks if. . .

Analysis. FLAS transforms the Java explanation into an extended communication metaphor, weaving in guidance language ("navigating," "guide," "dialogue," "communicate with intentionality") while still conveying the technical content. The prompting baseline, assigned the role of "communication coach," reverts to a standard technical explanation without communication-related vocabulary.