 ISSN: **3107-6513**

Concept-Level Activation Steering for Self-Correcting Mathematical Reasoning in Large Language Models: A Theoretical and Empirical Study

Karan Gupta[1], Krishna Dwivedi[2], Tanishk Kulshrestha[3], Dr. S.K Sharma

{1, 2, 3}CS-Data Science, ITM GOI, Gwalior, India {4} Associate

Professor, HOD Department of Mechanical Engineering, ITM GOI, Gwalior, India

[1]singhalkaran9770@gmail.com, [2]krishnadwivedi3578@gmail.com , [3]kulsh.tanishk23@gmail.com

***Abstract*—ConceptSteer injects concept-specific correction vectors directly into the residual stream of a Large Language Model (LLM) to self-correct multi-step mathematical reasoning without external tools. The core theoretical contribution formalizes reasoning errors as deviations from linear concept subspaces in the activation space of transformer layers. Under mild assumptions, the optimal steering direction—minimizing the forward KL divergence between the corrected and target correct distributions—is the difference of expected activations of correct and incorrect steps. A sparse autoencoder refinement isolates monosemantic concept features, and a lightweight probe triggers steering only when an error is detected. Experiments on GSM8K, MATH, and the newly introduced ConceptMath benchmark show that ConceptSteer improves final-answer accuracy by +8.2 points over chain-of-thought and cuts step-level errors by 31%, with under 0.5% added computation per step. ConceptSteer connects mechanistic interpretability with controllable generation for real-time, self-contained error correction.**

Keywords—large language models; mechanistic interpretability; activation steering; mathematical reasoning; self-correction

I. INTRODUCTION

Large language models (LLMs) still struggle with multi-step reasoning tasks, particularly in mathematics [1], [2]. Even when prompted to produce chain-of-thought (CoT) reasoning [3], models frequently generate intermediate steps that violate basic algebraic rules, commit arithmetic errors, or produce logically inconsistent statements.

Existing mitigation strategies fall into two families. External augmentation relies on symbolic solvers or neural verifiers that post-process or rerank completions [1], [4]. Self-refine methods [5] prompt the model to review its own output, but recent work shows they often fail without external feedback [6]. Both families treat the LLM as a black box and carry high computational cost.

Mechanistic interpretability has shown that high-level concepts are encoded as linear directions in the residual stream of transformers [7]-[9]. Adding or subtracting such directions steers generation—for example, suppressing harmful outputs [10] or shifting sentiment.

We combine these observations to build ConceptSteer, a self-contained method that detects an upcoming error by probing the model's own internal state and then steers the residual stream along a concept-specific correction vector, all within a single autoregressive pass. Our contributions are:

- **Theoretical formalization**: We model reasoning as a trajectory through concept subspaces and derive the optimal error-correction vector as a contrastive mean difference, with convergence guarantees.
- **Sparse concept refinement**: Using sparse autoencoders (SAEs), we decompose the residual stream into monosemantic concept features, yielding concept-specific steering vectors.
- **Probe-guided intervention**: A linear error detector activates steering only when needed, preserving fluency and keeping overhead low.
- **Empirical validation**: On three benchmarks, ConceptSteer outperforms CoT and self-refine, coming within 0.7 points of an 8-sample verifier at one-fifth the cost.

II. RELATED WORK

Interpretability and linear representations. The linear representation hypothesis holds that transformers encode concepts as linear subspaces of the residual stream [7], [11]. Sparse autoencoders [8] extract a disentangled feature basis where each dimension often corresponds to a human-interpretable concept. We use this structure to define error directions specific to mathematical operations.

Activation engineering. Activation addition [9] and related work [12] modify the forward pass to elicit desired behaviors. We apply this class of techniques to self-correction of multi-step reasoning—a setting that also requires a probe to identify when to intervene.

Self-correction in LLMs. Self-refine [5] and self-verification [4] attempt to fix errors but depend on extra prompting or sampling. ConceptSteer uses internal representations alone, with no external validation loop.

III. Problem Formulation and Theoretical Foundations

Consider a transformer language model p_theta that maps a prefix sequence x_<t to a distribution over the next token x_t. During multi-step reasoning, the model generates a chain of steps s = (s_1, s_2, ..., s_T), where each step s_t is a short textual segment (e.g., "Let x = 5"). We assume access to a ground-truth decomposition into reasoning steps and, for each step, a binary label y_t indicating correctness.

A. Concept Subspaces and Error Signals

Let h_(l)_t denote the residual stream activation at layer l immediately after processing step s_t (typically the activation at the last token of the step). For each mathematical concept c (e.g., equation balancing, substitution), we hypothesize a low-dimensional linear subspace V_c that encodes correct application of that concept. A step that misapplies concept c will have its activation shifted away from V_c.

Define the concept-conditional mean for correct steps as mu+_c = E[h | y=1, concept=c] and for incorrect steps as mu-_c = E[h | y=0, concept=c]. The error direction is then:

v_c = mu+_c - mu-_c (1)

We show below that this direction is optimal under a KL-divergence objective.

B. Optimality of the Contrastive Direction

Suppose that after step t, the model uses h_t to sample the next step, and an error has occurred so that h_t is drawn from the erroneous distribution Q. We seek a small additive perturbation delta so that the resulting token distribution approximates the correct distribution P. The objective is:

min_delta E_{h~Q} [D_KL(P(. | h) || P(. | h + delta))] (2)

Assuming the log-probability of the next step is locally linear in the residual stream [9], the logit shift is logit(h + delta) ~ logit(h) + W_U*delta, where W_U is the unembedding matrix, and the KL divergence reduces to a quadratic form. Minimizing under a norm constraint selects the Fisher linear discriminant direction.

Theorem 1. Let correct and incorrect activations follow multivariate Gaussians with shared covariance Sigma: h|y=1 ~ N(mu+, Sigma) and h|y=0 ~ N(mu-, Sigma). The direction that maximizes the log-odds of correctness is v = Sigma^{-1}(mu+ - mu-). For isotropic Sigma = sigma^2 * I, this reduces to v proportional to mu+ - mu-. The direction of maximum influence is Sigma^{-1}(mu+ - mu-), which for Sigma = sigma^2*I is proportional to mu+ - mu-.

C. Sparse Refinement via Autoencoders

The raw contrastive vector may conflate several concepts due to polysemanticity. To extract a cleaner, monosemantic direction, we train a sparse autoencoder (SAE) [8] on residual stream activations from mathematical reasoning traces. The SAE encoder computes f = ReLU(W_e*h + b_e) and the decoder reconstructs h-hat = W_d*f + b_d with sparsity penalty lambda*||f||_1. After training, we identify the latent index i_c whose activation correlates most strongly with concept c (via mutual information). The refined concept vector v^SAE_c = W_d[:,i_c] represents a single SAE feature, improving concept specificity.

D. Error Detection Probe

We train a linear probe g_phi: R^d -> [0,1] via logistic regression to predict P(y_{t+1} = 0 | h_(l)_t) on a balanced set of correct and incorrect step activations. A step is flagged as erroneous when g_phi(h_(l)_t) > tau, with tau chosen to maximize F1 on a held-out validation set. Layer choice matters: intermediate layers that balance local and global context give the highest detection accuracy [7], as shown in the appendix.

IV. ConceptSteer Methodology

Algorithm 1 summarizes the inference procedure.

A. Generic and Concept-Specific Steering

We maintain a generic error direction v_err = E[h|y=0] - E[h|y=1] aggregated over all concepts. When the probe fires but concept confidence is low, we apply -alpha*v_err, shifting the activation toward the correct-reasoning region. When the concept is confidently identified as c, we instead add +gamma*v^SAE_c, reinforcing correct application of that concept.

B. Coefficient Selection

Steering strengths alpha and gamma are tuned on a held-out validation set. We maximize final-answer accuracy while keeping text perplexity within 5% of the unsteered model, ensuring fluency. A linear grid search over [0.2, 2.0] is sufficient in practice.

Algorithm 1 ConceptSteer Inference

Require: Pretrained LLM, concept vectors {vc}, probe gϕ, layer L, coefficients α, γ, threshold τ

1: Generate step st via autoregressive decoding.

2: Extract residual stream activation ht at layer L after the last token of st.

3: if gϕ(ht) > τ then

4: Predict current concept cˆ with a lightweight classifier (or fall back to the generic error vector).

5: h˜ t ← ht − αverr (or +γvcˆ).

6: Replace ht with h˜ t in the residual stream.

7**: end if**

8: Continue generation from the modified activation.

9**: return** final answer

C. Computational Overhead

The probe is a single linear layer; the correction is a single vector addition. Total cost is under 0.5% of a forward pass. No additional completions are generated, unlike verifier-based approaches.

V. Experimental Setup

A. Datasets

- **GSM8K** [1]: 8.5K grade-school math word problems; 7.5K train / 1K test.
- **MATH** [2]: 12.5K competition-level problems from AMC and AIME; 7.5K train / 5K test.
- **ConceptMath**: 5,000 algebraic word problems (3-7 steps each), each step annotated with one of 20 algebraic concepts and verified by SymPy. See Appendix A for generation details.

B. Models and Implementation

We use Llama-3-8B (32 layers, hidden size 4096) as the backbone. SAE training follows [8] with expansion factor 4, lambda=0.001, on 2 million math-related activations. The probe is logistic regression trained with SGD; the concept classifier is a 2-layer MLP. All experiments run on 4xA100-80GB GPUs.

C. Baselines

- **Vanilla CoT:** 8-shot chain-of-thought prompting.
- **Self-Refine** [5]: one round of self-review.
- **Majority@5**: majority vote over 5 independent CoT completions.
- **Verifier@8**: process-reward model from [4] scoring 8 sampled chains.

VI. Results

A. Overall Accuracy

Table I reports final-answer accuracy. ConceptSteer outperforms both CoT and self-refine on all three benchmarks and comes within 0.7 points of the 8-sample verifier while generating only one completion.

TABLE I. **TABLE I. FINAL ANSWER ACCURACY (%)**

Method	GSM8K	MATH	ConceptMath
Vanilla CoT	74.6	42.1	68.3
Self-Refine	78.1	45.4	72.5
Majority@5	80.3	49.2	74.8
Verifier@8	83.5	54.3	79.1
ConceptSteer	**82.8**	**52.8**	**79.4**

B. Step-Level Accuracy

Table II shows step-level exact-match accuracy. ConceptSteer reduces erroneous intermediate steps by 31% relative to vanilla CoT.

TABLE II. **TABLE II. STEP-LEVEL ACCURACY (%)**

Method	GSM8K	MATH	ConceptMath
Vanilla CoT	81.2	60.3	73.1
Self-Refine	84.7	64.1	77.9
ConceptSteer	**89.4**	**71.8**	**84.5**

C. Ablation Study

Table III breaks down each module's contribution on GSM8K. Replacing concept-specific vectors with the generic error direction drops accuracy by 2.6 points. Removing the probe and steering after every step hurts performance, as unnecessary interventions inject noise into the residual stream. Layer choice is critical: layer 22 was optimal.

TABLE III. **TABLE III. ABLATION ON GSM8K (ACCURACY %)**

Ablation	Acc.
Full ConceptSteer	82.8
Use only generic error vector	80.2
Remove SAE refinement (raw contrastive)	81.5
No probe (always steer)	78.3
Steer at layer 1	75.4
Steer at all layers	76.8

D. Interpretability Analysis

To verify that the concept vectors capture genuine mathematical structure, we measure the lift in probability of generating a step that correctly applies a target concept when its vector is added to a neutral prompt. Table IV shows lifts of 29-42%, confirming semantic specificity.

TABLE IV. **TABLE IV. CONCEPT STEERING LIFT (%) OVER UNSTEERED BASELINE**

Concept	Lift (%)
Equation balancing	42.3
Substitution	37.8
Order of operations	33.2
Unit conversion	29.5
Distributive property	40.1

VII. Discussion

Generalization. We focused on mathematics, but the framework applies to code reasoning and logical puzzles with appropriate concept labels. Preliminary results on a logic grid dataset support this.

Concept coverage. ConceptSteer requires a predefined concept set; unknown error types may be missed. Future work could pursue unsupervised discovery of novel error directions, for instance via anomaly detection in the residual stream.

Probe reliability. Probe recall is approximately 72%; some errors go undetected. A recurrent state tracker or richer probe architecture could close this gap.

Theoretical extensions. The Gaussian assumption in Theorem 1 is strong but plausible locally around each step representation. Relaxing it to exponential family distributions via a Bregman divergence framework is one direction for future work.

VIII. Conclusion

ConceptSteer corrects multi-step reasoning errors in LLMs at inference time, without external tools or additional sampling. Steering directions derived from linear discriminant analysis and sharpened by sparse autoencoders give consistent gains—+8.2 accuracy points over CoT and 31% fewer step errors—at under 0.5% overhead per step. The result is a self-contained, interpretable correction mechanism that requires no re-prompting or extra inference passes.

Acknowledgment

The authors are grateful to the Department of Computer Science and Engineering and the Department of Mechanical Engineering at the Institute of Technology and Management, Gwalior, for providing the computational resources and academic support that made this work possible.

REFERENCES

[1] [1] K. Cobbe et al., "Training verifiers to solve math word problems," arXiv preprint arXiv:2110.14168, 2021.

[2] [2] D. Hendrycks et al., "Measuring mathematical problem solving with the MATH dataset," in NeurIPS, 2021.

[3] [3] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in NeurIPS, 2022.

[4] [4] H. Lightman et al., "Let's verify step by step," arXiv preprint arXiv:2305.20050, 2024.

[5] [5] A. Madaan et al., "Self-refine: Iterative refinement with self-feedback," in NeurIPS, 2023.

[6] [6] J. Huang et al., "Large language models cannot self-correct reasoning yet," arXiv preprint arXiv:2310.01798, 2024.

[7] [7] N. Elhage et al., "Toy models of superposition," arXiv preprint arXiv:2209.10652, 2022.

[8] [8] T. Bricken et al., "Towards monosemanticity: Decomposing language models with dictionary learning," Anthropic, 2023.

[9] [9] A. M. Turner et al., "Activation addition: Steering language models without optimization," arXiv preprint arXiv:2308.10248, 2023.

[10] [10] A. Arditi et al., "Refusal in language models is mediated by a single direction," arXiv preprint arXiv:2406.11717, 2024.

[11] [11] K. Meng et al., "Locating and editing factual associations in GPT," in NeurIPS, 2022.

[12] [12] K. Li et al., "Inference-time intervention: Eliciting truthful answers from a language model," arXiv preprint arXiv:2306.03341, 2023.