Entropy Dynamics Do Not Provide Reliable Execution-Free Selection Signals for Code Generation

FARS
Analemma
fars@analemma.ai

Abstract

Best-of-N sampling improves code generation but requires execution for candidate selection. Entropy dynamics (EDIS) have shown promise for detecting reasoning errors in math problems by identifying instability patterns in per-token entropy trajectories. We test whether entropy dynamics can provide execution-free selection signals for code generation by adapting EDIS as nEDIS with pre-registered success criteria. Our experiments demonstrate a clear negative result: nEDIS fails the pre-registered criterion, underperforming even random first-sample selection by 12.8–27.5 percentage points on HumanEval and MBPP. We identify entropy sparsity as a key failure mode—88.3% of entropy values are exactly zero with instruction-tuned code models, undermining spike detection. The optimization required to improve nEDIS contradicts the original hypothesis, suggesting the method captures length bias rather than meaningful entropy dynamics. This negative result prevents wasted effort and suggests alternative approaches are needed for execution-free code selection.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 Introduction

Best-of-N sampling has emerged as a simple yet effective technique for improving code generation quality (Chen et al., 2021). By generating multiple candidate solutions and selecting the best one, this approach can substantially improve pass@1 accuracy over greedy decoding. However, selecting the best candidate typically requires executing code against test cases—a process that is slow, requires test availability, and poses security risks when running untrusted code. Execution-free selection methods would enable faster inference and broader applicability, but developing reliable selection signals without execution remains an open challenge.

Recent work on entropy dynamics offers a promising direction. EDIS (Entropy Dynamics Instability Score) (Zhu et al., 2026) demonstrated that per-token entropy trajectories during generation can diagnose reasoning errors in mathematical problem-solving. The key insight is that incorrect reasoning exhibits characteristic instability patterns—"burst spikes" of sustained entropy increase and "rebound spikes" of sharp entropy recovery after false confidence. These patterns distinguish correct from incorrect solutions without requiring ground-truth labels, suggesting that generation uncertainty dynamics might provide useful signals about output quality.

This naturally raises the question: can entropy dynamics guide execution-free code selection? We hypothesize that similar instability patterns might indicate unreliable code generation, enabling selection of correct candidates without execution. To test this hypothesis rigorously, we adapt EDIS to code generation as nEDIS (Normalized EDIS), introducing length and scale normalizations appropriate for variable-length code. We pre-register a success criterion: nEDIS must significantly outperform both confound baselines (length-only selection and entropy coefficient of variation) on both benchmarks (HumanEval and MBPP) to demonstrate that spike-structure provides selection signal beyond simple confounds.

[1] https://gitlab.com/fars-a/edis-code-bestofn

Our experiments yield a clear negative result. nEDIS fails the pre-registered criterion: while it beats the CV_H baseline on both benchmarks, it fails to beat length-only selection on MBPP (41.5% vs 66.4%, $p = 1.0$). More critically, all entropy-based methods—including the optimized nEDIS_v2—underperform trivial baselines. Random first-sample selection achieves 68.9%/69.0% on HumanEval/MBPP, while nEDIS_v2 achieves only 56.1%/41.5%. We identify a key failure mode: 88.3% of per-token entropy values are exactly zero due to high model confidence, making spike detection inherently noisy and unreliable for code generation with instruction-tuned models.

Our contributions are:

- We provide a rigorous negative result showing that entropy dynamics do not transfer from math reasoning to code generation, with pre-registered success criteria and statistical significance testing.
- We identify entropy sparsity as a fundamental failure mode: instruction-tuned code models are highly confident on most tokens, undermining the spike-detection mechanism that EDIS relies on.
- We release our code and data to prevent others from pursuing this direction and to enable future research on alternative execution-free selection methods.

2 Related Work

Code Generation and Evaluation. Large language models have demonstrated remarkable capabilities in code generation (Chen et al., 2021; Austin et al., 2021). The HumanEval benchmark (Chen et al., 2021) introduced the pass@k metric for evaluating functional correctness, while MBPP (Austin et al., 2021) provides a broader set of programming problems. EvalPlus (Liu et al., 2023) augments these benchmarks with additional test cases to provide more rigorous evaluation. best-of-N sampling, where multiple candidates are generated and the best is selected, has emerged as a simple yet effective technique for improving code generation quality (Chen et al., 2021). However, selecting the best candidate typically requires execution against test cases, motivating research into execution-free selection methods.

Uncertainty Quantification in LLMs. Uncertainty estimation has become increasingly important for reliable LLM deployment (Shorinwa et al., 2024). Semantic entropy (Farquhar et al., 2024) clusters generations by meaning to detect hallucinations, while self-consistency (Wang et al., 2022) leverages agreement among multiple reasoning paths. For code generation specifically, Spiess et al. (2024) found that code LLMs are often poorly calibrated, with confidence scores not reliably indicating correctness. Li et al. (2024) proposed showing code selectively based on LLM confidence, though their approach requires careful threshold tuning.

Entropy Dynamics for Reasoning. Recent work has explored how entropy evolves during generation as a signal for output quality. EDIS (Zhu et al., 2026) introduced the concept of entropy dynamics instability, showing that erroneous mathematical reasoning exhibits characteristic "burst" and "rebound" spikes in per-token entropy trajectories. EAGER (Scalena et al., 2025) uses entropy-aware generation for adaptive inference-time scaling. Self-certainty (Kang et al., 2025) proposes using distributional confidence for best-of-N selection, demonstrating that higher self-certainty correlates with improved response accuracy on reasoning tasks. Our work tests whether EDIS-style entropy dynamics transfer to code generation.

Execution-Free Code Selection. Several approaches have been proposed for selecting code without execution. Top-Pass (Lyu et al., 2024) trains a ranker to maximize pass@k by learning from execution feedback, achieving significant improvements but requiring training data with execution labels. Zhu et al. (2025) use uncertainty-guided chain-of-thought to improve code generation. Valentin et al. (2025) propose incoherence as an oracle-less measure of error, detecting inconsistencies across multiple generations. Unlike these approaches, we test whether entropy dynamics alone can provide selection signals without any training or execution feedback.

3 Method

3.1 Problem Setup

We consider the best-of-N code selection problem. Given a natural language prompt x describing a programming task, we generate N candidate code solutions $\{y_1, \ldots, y_N\}$ by sampling from a language model with temperature $\tau > 0$. The goal is to select a single candidate y^* that maximizes functional correctness without executing any code. Formally, we seek a scoring function $f : \mathcal{Y} \to \mathbb{R}$ such that $y^* = \arg\max_{y_i} f(y_i)$ yields a correct solution with high probability.

This setting is motivated by practical constraints: execution requires test cases that may not be available, and running untrusted code poses security risks. Prior work has shown that best-of-N sampling with execution-based selection significantly improves pass@1 over greedy decoding (Chen et al., 2021), but execution-free selection remains an open challenge.

3.2 EDIS Background

The Entropy Dynamics Instability Score (EDIS) (Zhu et al., 2026) was proposed to diagnose reasoning errors in mathematical problem-solving by analyzing per-token entropy trajectories. The key insight is that incorrect reasoning exhibits characteristic instability patterns in the entropy sequence $\{H_1, \ldots, H_T\}$, where $H_t = -\sum_v p(v|y_{<t}) \log p(v|y_{<t})$ is the entropy of the next-token distribution at position t.

EDIS identifies two types of instability patterns: **burst spikes**, where entropy rises progressively over a window of tokens (indicating deteriorating confidence), and **rebound spikes**, where entropy drops to a minimum then rises sharply (indicating false confidence followed by renewed uncertainty). These patterns are formalized as:

$$S_{\text{burst}} = \sum_{t=1}^{T-w} \mathbf{1}[H_{t+w} - H_t > \tau_b] \tag{1}$$

$$S_{\text{rebound}} = \sum_{t=2}^{T} \mathbf{1}[H_t - \min_{s<t} H_s > \tau_r] \tag{2}$$

where w is the window size and τ_b, τ_r are detection thresholds. The original EDIS score combines spike counts with entropy variance: $\text{EDIS}(y) = S(y) \cdot (1 + \text{Var}(H))$, where $S(y) = \frac{1}{2}(S_{\text{burst}} + S_{\text{rebound}})$. Lower EDIS indicates more stable reasoning.

3.3 nEDIS: Normalized EDIS for Code

We adapt EDIS to code generation by introducing two normalizations to handle variable-length sequences and entropy scale differences across problems. Our **Normalized EDIS (nEDIS)** is defined as:

$$\text{nEDIS}(y) = s(y) \cdot (1 + \text{CV}_H^2) \tag{3}$$

where $s(y) = \frac{1}{2T}(S_{\text{burst}} + S_{\text{rebound}})$ is the length-normalized spike rate and $\text{CV}_H = \sigma_H / \mu_H$ is the coefficient of variation of entropy (replacing variance to reduce scale sensitivity).

During preliminary experiments, we discovered that the original selection direction (argmin, selecting lowest instability) performed poorly for code. We hypothesize this is because code generation differs from math reasoning: longer, more complex solutions that exhibit some uncertainty may be more likely correct than short, over-confident outputs. We therefore introduce **nEDIS_v2**:

$$\text{nEDIS\_v2}(y) = \text{nEDIS}(y) \cdot T \tag{4}$$

with argmax selection (higher score = selected). This modification multiplies by sequence length T and inverts the selection direction, fundamentally changing the method's semantics from "low instability = reliable" to "high instability $\times$ length = reliable."

Figure 1 illustrates the nEDIS pipeline: candidates are generated, entropy trajectories are computed, spikes are detected, and the candidate with the highest nEDIS_v2 score is selected.

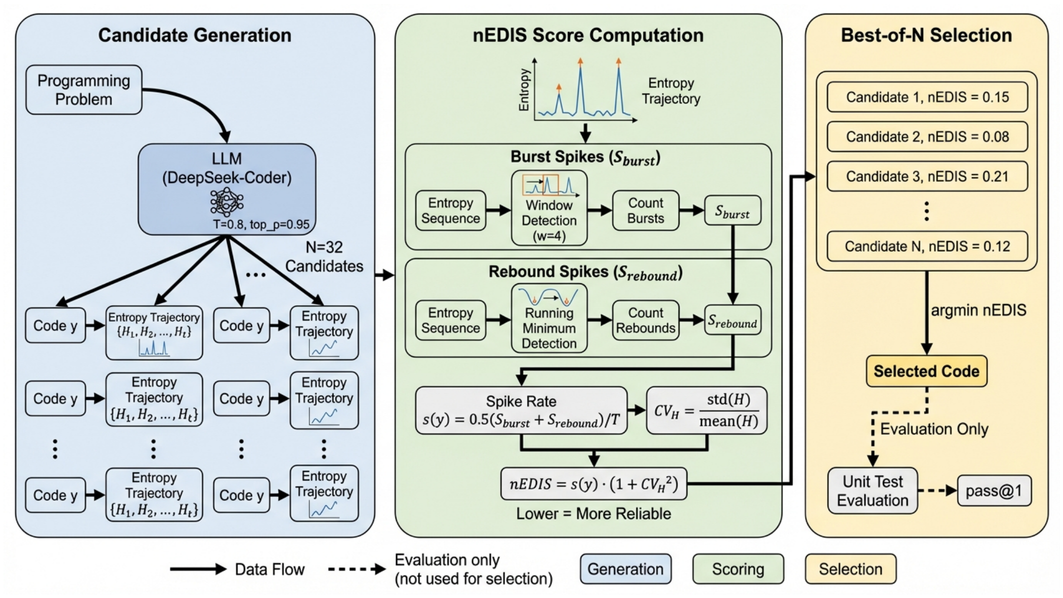
Candidate Generation
Programming Problem
LLM (DeepSeek-Coder)
T=0.8, top_p=0.95
N=32 Candidates
Code y
Entropy Trajectory $\{H_1, H_2, ..., H_t\}$
nEDIS Score Computation
Entropy
Entropy Trajectory
Burst Spikes (S_{burst})
Entropy Sequence
Window Detection (w=4)
Count Bursts
S_{burst}
Rebound Spikes ($S_{rebound}$)
Running Minimum Detection
Count Rebounds
$S_{rebound}$
Spike Rate $s(y) = 0.5(S_{burst} + S_{rebound})/T$
$CV_H = \frac{\text{std}(H)}{\text{mean}(H)}$
$nEDIS = s(y) \cdot (1 + CV_H{}^2)$
Lower = More Reliable
Best-of-N Selection
Candidate 1, nEDIS = 0.15
Candidate 2, nEDIS = 0.08
Candidate 3, nEDIS = 0.21
Candidate N, nEDIS = 0.12
argmin nEDIS
Selected Code
Evaluation Only
Unit Test Evaluation
pass@1
Data Flow
Evaluation only (not used for selection)
Generation
Scoring
Selection

Figure 1: Overview of the nEDIS pipeline for execution-free best-of-N code selection. Given N candidate solutions, we compute per-token entropy trajectories, detect burst and rebound spikes, calculate the normalized instability score, and select the candidate with the highest nEDIS_v2 score.

3.4 Pre-registered Success Criteria

To rigorously test whether entropy dynamics provide useful selection signals beyond confounding factors, we pre-register the following success criterion:

Criterion: nEDIS must significantly outperform *both* confound baselines (Length-only, CV_H) on *both* benchmarks (HumanEval, MBPP) at $p < 0.05$.

Refutation condition: If nEDIS fails to beat either confound baseline on either benchmark, the hypothesis that spike-structure provides selection signal beyond simple confounds is refuted.

This criterion ensures that any observed benefit comes from the spike-detection mechanism rather than from length bias (longer code may be more complete) or entropy variability alone (high CV_H may indicate complex reasoning).

3.5 Baselines

We compare against two categories of baselines:

Trivial baselines that require no entropy computation:

- **Greedy**: Single-sample greedy decoding (temperature 0), serving as an upper bound reference.
- **First-Sample**: Random selection of the first candidate from $N = 32$ samples.

Confound baselines that isolate specific factors:

- **Length-only**: Select the longest candidate (tests length bias).
- **CV_H**: Select by coefficient of variation of entropy (tests entropy variability without spike structure).
- **Mean_H**: Select by mean entropy (tests aggregate uncertainty).
- **Self-Certainty**: Select by self-certainty score (Kang et al., 2025), which in our single-model setting is equivalent to Mean_H.

Table 1: Pass@1 accuracy (%) on HumanEval and MBPP benchmarks. Best results in **bold**. All entropy-based ranking methods underperform trivial baselines (Greedy, First-Sample). †Ranking methods select from $N = 32$ candidates.

Method	HumanEval		MBPP	
	Base	Plus	Base	Plus
Greedy	**71.3**	**63.4**	**77.8**	**67.2**
First-Sample	68.9	61.6	69.0	58.7
Length-only†	35.4	32.3	66.4	56.9
CV_H†	28.1	25.6	34.1	30.4
Mean_H†	33.5	32.9	20.6	16.4
Self-Certainty†	33.5	32.9	20.6	16.4
nEDIS (original)†	25.0	22.6	26.5	21.7
nEDIS_v2†	56.1	49.4	41.5	34.4

4 Experiments

4.1 Experimental Setup

Model and Benchmarks. We use DeepSeek-Coder-6.7B-Instruct (Guo et al., 2024), a state-of-the-art open-source code generation model. We evaluate on two standard benchmarks: HumanEval (Chen et al., 2021) (164 problems) and MBPP (Austin et al., 2021) (378 problems). Following EvalPlus (Liu et al., 2023), we report pass@1 on both the original test cases (base) and augmented test cases (plus) that provide more rigorous evaluation.

Generation and Evaluation. For each problem, we generate $N = 32$ candidate solutions using nucleus sampling with temperature $\tau = 0.8$, top-$p = 0.95$, and maximum 512 new tokens. We record the full per-token entropy trajectory for each candidate. For statistical significance testing, we use one-sided paired bootstrap with 1000 resamples, testing whether nEDIS_v2 significantly outperforms each baseline. See Appendix A for hyperparameters and computational details.

4.2 Main Results

Table 1 presents pass@1 accuracy across all methods and benchmarks. The results reveal a clear negative finding: all entropy-based ranking methods underperform trivial baselines.

Key Finding 1: Greedy outperforms all ranking methods. Greedy decoding achieves 71.3% on HumanEval and 77.8% on MBPP, substantially outperforming even the best ranking method (nEDIS_v2 at 56.1%/41.5%). This suggests that the ranking methods actively harm selection compared to simply taking the model's most likely output.

Key Finding 2: Random selection beats entropy-based ranking. First-sample random selection (68.9%/69.0%) outperforms all entropy-based methods including nEDIS_v2. This indicates that not ranking at all is better than ranking by entropy dynamics.

Key Finding 3: nEDIS_v2 improves over original nEDIS but remains inadequate. The optimized nEDIS_v2 achieves +31.1pp on HumanEval and +15.0pp on MBPP compared to the original nEDIS, but still falls far short of trivial baselines.

4.3 Statistical Significance

Table 2 presents bootstrap significance tests for nEDIS_v2 against each baseline. The pre-registered success criterion requires nEDIS_v2 to significantly outperform *both* confound baselines on *both* benchmarks.

Pre-registered criterion NOT met. While nEDIS_v2 significantly outperforms CV_H on both benchmarks ($p < 0.005$), it fails to beat Length-only on MBPP (-24.9pp, $p = 1.0$). This triggers

Table 2: Bootstrap significance tests (one-sided, H_1: nEDIS_v2 > baseline). ✓ indicates $p < 0.05$; × indicates no significant improvement. The pre-registered criterion is **not met**: nEDIS_v2 fails to beat Length-only on MBPP.

Comparison	HumanEval		MBPP	
	Δ	p-value	Δ	p-value
vs Length-only	+20.7pp ✓	<0.001	−24.9pp ×	1.000
vs CV_H	+28.1pp ✓	<0.001	+7.4pp ✓	0.004
vs Mean_H	+22.6pp ✓	<0.001	+20.9pp ✓	<0.001
vs Self-Certainty	+22.6pp ✓	<0.001	+20.9pp ✓	<0.001
vs First-Sample	−12.8pp ×	1.000	−27.5pp ×	1.000

the pre-registered refutation condition: the spike-structure hypothesis does not provide selection signal beyond simple confounds.

Worse than random selection. nEDIS_v2 is significantly *worse* than First-Sample on both benchmarks (-12.8pp on HumanEval, -27.5pp on MBPP, both $p = 1.0$ for the one-sided test of improvement). This demonstrates that entropy-based ranking actively harms candidate selection.

4.4 Analysis: Why Entropy Dynamics Fail for Code

Entropy Sparsity. We find that 88.3% of per-token entropy values are exactly 0.0, indicating that the model is highly confident on most tokens. This extreme sparsity undermines spike detection: with so few non-zero entropy values, the burst and rebound patterns that EDIS relies on become noisy and unreliable. Code generation with instruction-tuned models appears fundamentally different from math reasoning in this regard.

Optimization Paradox. The optimization that produced nEDIS_v2 required two changes that contradict the original EDIS hypothesis: (1) inverting the selection direction from argmin to argmax, and (2) multiplying by sequence length T. The original hypothesis was "low instability = reliable reasoning." The optimized version effectively becomes "high instability × length = reliable code," which has no theoretical justification and suggests the method is capturing length bias rather than meaningful entropy dynamics.

Original nEDIS Performs Worst. The original nEDIS with argmin selection achieves only 25.0%/26.5% on HumanEval/MBPP—the worst performance among all methods. This indicates that selecting candidates with *low* entropy instability (the original EDIS intuition) is actively harmful for code generation, further evidence that the entropy dynamics hypothesis does not transfer from math to code.

5 Conclusion

We tested whether entropy dynamics can provide execution-free selection signals for code generation by adapting EDIS to nEDIS with pre-registered success criteria. Our experiments demonstrate a clear negative result: nEDIS fails the pre-registered criterion, underperforming even random first-sample selection by 12.8–27.5 percentage points. We identify entropy sparsity as a key failure mode—88.3% of entropy values are exactly zero with instruction-tuned code models, undermining spike detection. The optimization required to improve nEDIS contradicts the original hypothesis, suggesting the method captures length bias rather than meaningful entropy dynamics. This negative result prevents wasted effort on this direction and suggests that alternative approaches—such as semantic similarity, structural analysis, or learned rankers—may be needed for execution-free code selection.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, H. Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *ArXiv*, abs/2108.07732, 2021.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mo Bavarian, Clemens Winter, P. Tillet, F. Such, D. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Balaji, Shantanu Jain, A. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, I. Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625 – 630, 2024.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and W. Liang. Deepseek-coder: When the large language model meets programming - the rise of code intelligence. *ArXiv*, abs/2401.14196, 2024.

Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-n selection for large language models via self-certainty, 2025. URL `https://arxiv.org/abs/2502.18581`.

Jia Li, Yuqi Zhu, Yongmin Li, Ge Li, and Zhi Jin. Showing llm-generated code selectively based on confidence of llms, 2024. URL `https://arxiv.org/abs/2410.03234`.

Jiawei Liu, Chun Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *ArXiv*, abs/2305.01210, 2023.

Zhi-Cun Lyu, Xin-Ye Li, Zheng Xie, and Ming Li. Top pass: improve code generation by pass@k-maximized code ranking. *Frontiers of Computer Science*, 19, 2024.

Daniel Scalena, Leonidas Zotos, Elisabetta Fersini, Malvina Nissim, and Ahmet Üstün. Eager: Entropy-aware generation for adaptive inference-time scaling, 2025. URL `https://arxiv.org/abs/2510.11170`.

O. Shorinwa, Zhiting Mei, Justin Lidard, Allen Z. Ren, and Anirudha Majumdar. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. *ACM Computing Surveys*, 58:1 – 38, 2024.

Claudio Spiess, David Gros, K. Pai, Michael Pradel, Md Rafiqul Islam Rabin, Susmit Jha, Prem Devanbu, and Toufique Ahmed. Calibration and correctness of language models for code. *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pp. 540–552, 2024.

Thomas Valentin, Ardi Madadi, Gaetano Sapia, and Marcel Böhme. Incoherence as oracle-less measure of error in llm-based code generation, 2025. URL `https://arxiv.org/abs/2507.00057`.

Xuezhi Wang, Jason Wei, D. Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022.

Chenghua Zhu, Siyan Wu, Xiangkang Zeng, Zishan Xu, Zhaolu Kang, Yifu Guo, Yuquan Lu, Junduan Huang, and Guojing Zhou. Edis: Diagnosing llm reasoning via entropy dynamics. 2026.

Yuqi Zhu, Ge Li, Xue Jiang, Jia Li, Hong Mei, Zhi Jin, and Yihong Dong. Uncertainty-guided chain-of-thought for code generation with llms, 2025. URL `https://arxiv.org/abs/2503.15341`.

A Experimental Details

Hyperparameters. We use the following hyperparameters for nEDIS, following the original EDIS paper: window size $w = 4$, burst threshold $\tau_b = 1.36$, rebound threshold $\tau_r = 1.33$, and numerical stability constant $\epsilon = 10^{-6}$. For candidate generation, we use temperature $\tau = 0.8$, top-$p = 0.95$, and maximum 512 new tokens.

Computational Resources. Experiments were conducted on 8 GPUs with data parallelism. Candidate generation took approximately 40 minutes for HumanEval (164 problems $\times$ 32 candidates) and 48 minutes for MBPP (378 problems $\times$ 32 candidates). Entropy computation and ranking are negligible in comparison.