Optimizing Importance Sampling Methods for Rare Output Estimation in Language Models

Amanda Cao[1, 2*], **Ivan Betancourt**[1, 3*], **Manish Rangan**[1, 4], **Yuqi Sun**[5]

[1]Algoverse AI Research Program
[2]Yale University, Department of Statistics & Data Science, New Haven, USA
[3]Amherst College, Department of Computer Science, Amherst, USA
[4]Georgia Institute of Technology, Department of Computer Science, Atlanta, USA
[5]Mindoverflow Co.

Abstract

Large language models (LLMs) have the ability to produce rare but potentially catastrophic outputs, especially in deployment settings where models may experience distribution shift. Understanding a model's worst-case performance is necessary to ensuring the safety and reliability of artificial intelligence systems. However, because rare outputs have true probabilities as low as 10^{-9}, standard sampling methods are computationally unfeasible. To reliably quantify rare output probabilities, it is essential to develop algorithms that are both computationally efficient and scalable to the agentic setting. Previous works have introduced Metropolis-Hastings Important Sampling (MHIS), an algorithm that samples from a Boltzmann-weighted distribution to better explore rare-event regions in the input space. In the following paper, we introduce **Delayed-Acceptance Metropolis–Hastings Importance Sampling (DA-MHIS)**, an algorithm that optimizes MHIS for efficiency by filtering out poor-quality proposals. We also mention **Multiple-Try Metropolis Hastings Importance Sampling** as an additional variant, which attempts to improve serialized runtime. It is particularly important to develop efficient sampling algorithms to work toward a longer-term goal of eliciting and understanding rare behaviors in large-scale, agentic settings. In order to validate our methods, we compare the performance of our algorithms against the original MHIS as a baseline, and find that DA-MHIS reduces runtime by 16.4% on the GELU 1-layer model. Our Github repository and code will be shared upon publication.

1. Introduction

Modern machine learning systems are typically optimized to perform well on average across a training distribution. However, even when the overall accuracy or loss metrics are strong, these models can still produce highly undesirable outputs on extremely rare inputs. Such low-probability behaviors can become far more likely under distribution shift, especially when inputs are chosen adversarially, as in the case of large language model jailbreaks. Accurately estimating the probabilities of these rare but potentially catastrophic behaviors is therefore a prerequisite for achieving robust and trustworthy AI systems.

The task of *low probability estimation*, i.e. quantifying the chance that a formally specified undesirable behavior occurs under a given input distribution, poses a unique challenge. When the true probability of the target behavior is exceedingly small (e.g., 10^{-9} or lower), naive random sampling is essentially useless: under realistic compute budgets, one is unlikely to observe a single positive instance. This limitation mirrors the fundamental problem of safety evaluation for frontier models: if catastrophic behaviors are rare, then detecting and mitigating them requires specialized methods that can efficiently explore regions of the input space where these behaviors are more likely to occur.

Recent work by Wu and Hilton (2025) introduced several importance-sampling and activation-extrapolation techniques that dramatically improved the accuracy of probability estimation for rare events in small language models. In particular, their Metropolis–Hastings Importance Sampling (MHIS) algorithm provided a principled and unbiased estimator by drawing samples from a Boltzmann-weighted distribution that up-weights inputs associated with higher target logits. However, MHIS and related approaches remain computationally intensive, as each proposed move requires a full forward and backward pass through the model to evaluate the acceptance ratio. This cost severely limits their scalability to larger architectures or broader token distributions.

In this work, we pursue the same goal of reliable low-probability estimation, but focus on improving its *efficiency*. We introduce the **Delayed-Acceptance Metropolis–Hastings Importance Sampling (DA-MHIS)** algorithm inspired by Banterle et al. (2015), which maintains the unbiasedness and theoretical guarantees of MHIS while substantially reducing its computational overhead. DA-MHIS accomplishes this by introducing a lightweight, gradient-based surrogate test that screens proposals before expensive model evaluations are performed. This two-stage mechanism allows the estimator to concentrate computation on high-value proposals, yielding faster convergence without sacrificing accuracy. We also introduce **Multiple-Try Metropolis Hastings (MT-MHIS)**, which proposes multiple candidates for a random walk step at each iteration, instead of proposing only one next-candidate. By selecting the best candidate among multiple proposed candidates, MT-MHIS seeks to decrease runtime by reducing the steps required for

*These authors contributed equally.

Copyright © 2026, Trustworthy Agentic AI Workshop@ Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

convergence to an up-weighted target distribution.

Beyond the direct gains in runtime, efficient estimators can meaningfully expand the scope of rare-event analysis. They make it feasible to evaluate larger models, longer contexts, or more complex output properties within a fixed compute budget. Ultimately, improving the speed of estimation is not merely a matter of engineering efficiency: it is essential for the practical detection and mitigation of rare failures in large-scale machine learning systems.

Problem Setting Here, we'd like to formally define the problem of low probability estimation. For consistency, we retain the same notation as Wu and Hilton (2025).

Given vocabulary $\mathcal{V}$, let $M : \mathcal{V}^* \to \mathcal{V}$ be a predictive language model that outputs the next token based on a sequence of previous tokens over the vocabulary. We take a distribution $\mathcal{D}$ over the sequence of previous tokens $\mathcal{V}^*$. In low probability estimation, we want to evaluate the probability of predicting a target token, $t \in \mathcal{V}$. Hence, given input x, the problem can be written as:

$$\underset{x\sim\mathcal{D}}{P}[M(x) = t] \tag{1}$$

We can also express this equation using the following notation. Let $M_i(x)$ be the logit that the model predicts for token $i \in \mathcal{V}$. We can write the probability that the model predicts target token t as:

$$\underset{x\sim\mathcal{D}}{P}[M_x(t) > M_i(x), \forall i \neq t] \tag{2}$$

In Wu and Hilton's methodology, distribution $\mathcal{D}$ is constrained to be only distributions with independent tokens.

2. Related Works

Understanding and mitigating undesirable model outputs has arisen as an emergent field within deep learning. Existing methods include adversarial training (Zhao et al. 2024), which is the process of systematically integrating adversarial examples into the training data. Adversarial training has seen success across variety of deep learning architectures, including diffusion models, Convolutional Neural Networks (CNNs), Graph Neural Networks (GNNs), Recurrent Neural Networks (RNNs), and LLMs (Bai et al. 2021). In the context of language models, recent work has explored areas including adversarial pre-training for improving generalization and robustness in natural language processing (NLP) tasks (Liu et al. 2020), exploring alternate generation strategies to elicit harmful behavior (Huang et al. 2023), and conducting attacks in a LLM's continuous embedding space rather than performing discrete attacks over training iterations (Xhonneux et al. 2024).

The purpose of such techniques is to reduce the impact of distribution shift (Wu and Hilton 2025), which refers to the phenomenon by which models exhibit unintended behaviors after ingesting inputs which are outside the high-density regions of their training distribution. While adversarial training attempts to address the distribution shift problem through incorporating small, targeted perturbations, a related challenge lies in attempting to understand and quantify a model's rare behaviors. This body of work is known as *low-probability estimation*.

Recent work by Webb et al. (2019) has explored the low-probability estimation space by developing a framework which measures neural network robustness based on the probability that a particular property is violated under a given input distribution. This work addresses the low-probability estimation problem in the context of computer vision. In the region of LLMs, work conducted by Phuong et al. (2024) and Højmark et al. (2024) evaluate the capabilities of frontier models and LLM-based agents. In particular, Højmark et. al use Monte-Carlo based methods in order to estimate the probabilities of an agent successfully completing a given task.

However, in this paper, we build on the methodology developed by Wu and Hilton (2025), specifically their *importance sampling* methods for estimating low-probability language model outputs. Importance sampling offers a remedy to naive sampling, in which low-probability events take far too many samples to observe consistently. In contrast, importance sampling allows for a more frequent elicitation of low-probability outputs because it requires defining a new input distribution. This input distribution narrows the input space such that it up-weights regions in which the low-probability outputs occur more frequently. After sampling from this up-weighted distribution, the weights of the samples must be readjusted in order to ensure that this sampling process is an unbiased estimator for the true probability. We can formalize the importance sampling process in accordance with Wu and Hilton's notation.

First, we take $p(x)$ to be the probability mass function (PMF) of token distribution $\mathcal{D}$. Then, we also take $q(x)$ to be the PMF of another distribution. The concept of importance sampling can be expressed as such:

$$\begin{aligned}\underset{x\sim p}{P}[M(x) = t] &= \mathbb{E}_{x\sim p}[\mathbb{1}[M(x) = t]] \\ &= \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}\mathbb{1}[M(x) = t]\right]\end{aligned}$$

Here, the importance sampling equation states that the probability of the model predicting rare event t can be treated as an indicator variable, taking the expected value over original distribution $p(x)$. However, once we create a surrogate distribution, $q(x)$, that up-weights the desired areas, we must re-weight the probability of achieving the rare event by $\frac{p(x)}{q(x)}$ so that the remaining probability is an unbiased estimator of the original distribution. The idea is that our surrogate distribution $q(x)$ has a smaller variance, such that fewer samples are required to observe the rare event t.

Wu and Hilton previously propose two importance sampling methods: Independent Token Gradient Importance Sampling (ITGIS) and Metropolis-Hastings Importance Sampling (MHIS).

Independent Token Gradient Importance Sampling The ITGIS method takes a Boltzmann posterior $q(x)$ with a given prior $p(x)$. However, as its name suggests, this method

treats each of the input tokens as independent. Each token's probability is changed based on its average linear contribution to the logit of rare token t, and a score function is calculated based on . For example, given a string of input tokens $\mathbf{x} = (x_1, x_2, ...x_k)$, we could factor the input distribution into the product of the individual input tokens x_i, as in $p(\mathbf{x}) = p_1(x_1)p_2(x_2)...p_k(x_k)$. Up-weighted distribution q can be similarly parameterized as $q(\mathbf{x}) = q_1(x_1)...q_k(x_k)$. We calculate a score function s_i based on the gradient calculated from each input sequence, hence distribution $q(\mathbf{x})$ can be expressed with the following equation:

$$q_i(x_i) \alpha p_i(x_i) \cdot exp\left(\frac{s_i(x_i)}{T}\right) \tag{3}$$

where

$$s_i(x_i) = \mathbb{E}_{\mathbf{x'} \sim q}[\nabla_{\mathbf{x'}} M_t(\mathbf{x'})]_{i,x_i} \tag{4}$$

Here, the parameter T is the temperature of the model. For consistency, we follow Wu and Hilton and sample with temperature 0 to obtain deterministic results. The gradient $\nabla_{\mathbf{x}'} M_t(\mathbf{x}')_{i,x_i}$ at the one-hot vector $\mathbf{x}'$ can be interpreted as the change in logit of t given that the i-th token of $\mathbf{x}'$ were replaced by x_i. The result of the ITGIS algorithm is an importance sampling estimate taken from our up-weighted distribution.

However, in our work, we do not focus on ITGIS given its constraint that the input tokens are treated as independent. The next method, Metropolis-Hastings Importance Sampling (MHIS) will serve as the baseline to our experiments.

Metropolis-Hastings Importance Sampling In contrast to ITGIS, the Metropolis-Hastings Importance Sampling (MHIS) algorithm is more robust to dependencies between input tokens. Because ITGIS only takes into account the linear contribution of each token to the target logit, it may fail to consider cases where the model relies on inter-token interactions that affect the target logit. Hence, the score function must depend on the entire input, as it cannot be factored into independent token interactions. The score function can then be taken to be the target logit, which is written as

$$q(\mathbf{x}) \; \alpha \; p(\mathbf{x}) \cdot exp\left(\frac{M_t(\mathbf{x})}{T}\right)$$

First, the Metropolis-Hastings algorithm is used to generate a random walk in the input space which converges to a stationary distribution of q. Each next step of the random walk is generated by a proposal distribution, $\phi(\mathbf{x}'|\mathbf{x})$.

The proposal distribution is inspired by the Greedy Coordinate Gradient (GCG) method (Zou et al. 2023), an gradient-based algorithm which optimizes an adversarial suffix for creating jailbreak prompts. Similarly, our optimization methods are also inspired by methods for optimizing GCG (Zou et al. 2023). In this original paper, the Greedy Coordinate Gradient algorithm combines both greedy and gradient-based methods for optimizing over discrete tokens. The authors calculate the gradients with respect to the one-hot token indicators to identify a subset of candidates for replacement. This gradient is an approximation of the effect of replacing the ith token in the adversarial suffix prompt. Then, in order to identify the single-token replacement candidates, the top-k values with the largest *negative* gradient are selected. A candidate token set is calculated for all identified tokens, and the token replacement is selected such that it maximally decreases the loss.

Wu and Hilton adapt the GCG algorithm into a Metropolis-Hastings proposal function through the following steps: first, randomly sample a token position i to replace, then, take the gradient of the score function $s(\mathbf{x})$ as defined above with respect to $\mathbf{x}_i$, treating $\mathbf{x}_i$ as a one-hot vector. We can denote this gradient as $\mathbf{g}$. Finally, a replacement token x_i' from a distribution proportional to the gradient. The distribution from which a replacement token is sampled can be given by

$$p_i(x_i) \; \cdot \; exp(\frac{\mathbf{g}_{x_i}}{T})$$

After the replacement token is inserted into the correct position, the result can be expressed as a new input which contains the replaced token $\mathbf{x}' = (x_1, ..., x_i', ..., x_k)$. The overall result of the MHIS algorithm is similar to ITGIS, i.e. the average importance sampling estimate taken after a burn-in period. The burn-in period is an initial phase where the first n samples are not counted in order for the algorithm to have enough samples to fully converge towards its stationary distribution. In their work, Wu and Hilton utilize a burn-in period of 2^{10} batches.

In this paper, we explore methods of optimizing the MHIS algorithm for efficiency. In the following section, we will introduce two optimization methods: Delayed-Acceptance Metropolis-Hastings (DA-MHIS) and Multiple Try Metropolis-Hastings (MT-MHIS). In particular, the Multiple-Try Metropolis-Hastings algorithm applies the work of Doig and Wang (2025), which outlines the necessity for multi-candidate proposals and reviews several implementations of the Multiple-Try Metropolis as a generalized algorithm. In brief, instead of proposing one token for the next step of the random walk, Multiple-Try Metropolis suggests altering the proposal distribution such that it proposes several candidates.

3. Methodology

The following section provides a detailed description of the two optimization algorithms to improve upon the efficiency of the original Metropolis-Hastings Importance Sampling algorithm.

Delayed-Acceptance Metropolis–Hastings Importance Sampling (DA-MHIS)

The Metropolis–Hastings Importance Sampling (MHIS) algorithm samples from a Boltzmann-weighted distribution

$$q(x) \propto p(x) \exp\big(M_t(x)/T\big),$$

where $p(x)$ is the probability mass function (PMF) of the input distribution, $M_t(x)$ is the scalar logit assigned by the

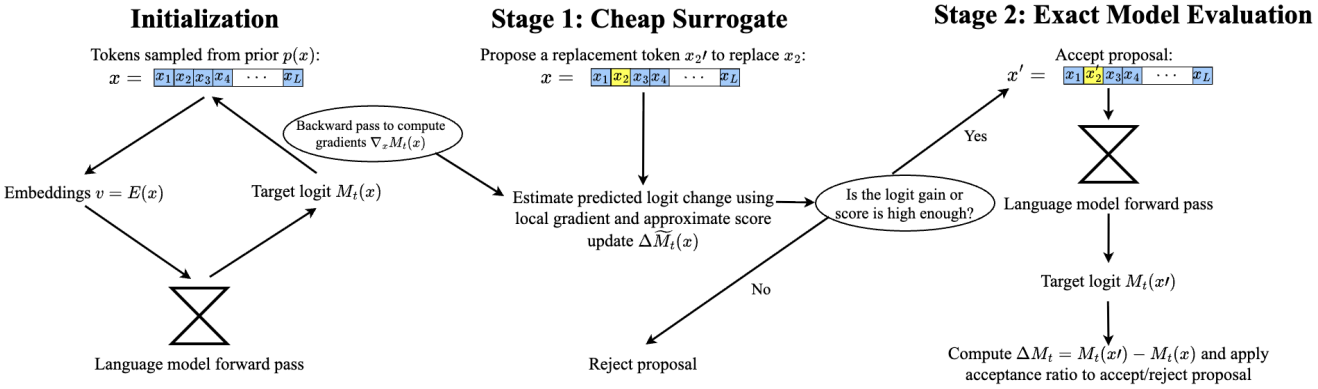

Figure 1: DA-MHIS Flow Diagram. The yellow token marks the proposed replacement. Stage 1 uses the local gradient to predict its effect on the target logit and quickly rejects weak proposals. Only promising ones move to Stage 2, where the model is re-evaluated exactly before acceptance or rejection.

model to the target token t, and $T > 0$ is a temperature hyperparameter controlling exploration. Although MHIS provides unbiased importance-sampling estimates, each proposed move requires a full forward and backward pass through the model, making the method computationally expensive for large networks or long sequences.

Notation. Let $x = (x_1, \ldots, x_L)$ denote the current sequence of discrete input tokens of length L, and let $x' = (x_1, \ldots, x'_i, \ldots, x_L)$ be a proposed sequence that differs from x at exactly one position i. Each token x_i belongs to the vocabulary V of size $|V|$, and has an associated embedding vector $v_i = E(x_i) \in \mathbb{R}^d$, where $E(\cdot)$ is the model's embedding lookup matrix of dimension $|V|\times d$. The full embedded input is therefore $v = E(x) \in \mathbb{R}^{L\times d}$, and the embedding of the proposed sequence is $v' = E(x') \in \mathbb{R}^{L\times d}$. The model's forward computation produces the target logit $M_t(x)$ and a corresponding gradient

$$g = \nabla_v M_t(x) \in \mathbb{R}^{L\times|V|},$$

where $g_{i,j}$ measures the sensitivity of the target logit to replacing the token at position i with vocabulary item j.

Stage 1: Linearized Surrogate Screening. To avoid the cost of evaluating $M_t(x')$ for every proposed sequence, DA-MHIS introduces a cheap surrogate test based on the local linearization of $M_t(x)$ around the current state. For a proposed token replacement at position i, the approximate change in the target logit is estimated as

$$\tilde{\Delta}M_t = g_{i,x'_i} - g_{i,x_i},$$

where g_{i,x_i} and g_{i,x'_i} denote the gradient components corresponding to the current and proposed tokens, respectively. This term quantifies the predicted increase in the target logit if the model were to replace x_i with x'_i. Combining this surrogate with the prior ratio $\log p(x') - \log p(x)$, the Stage 1 log acceptance ratio is defined as

$$\log r_1 = \big[\log p(x') - \log p(x)\big] + \frac{\tilde{\Delta}M_t}{T}.$$

The proposal passes the surrogate screen if either $\log r_1 \geq 0$ or $\log u_1 \leq \log r_1$, where $u_1 \sim \mathrm{Uniform}(0,1)$. Proposals that fail this criterion are immediately rejected without running a forward pass at x'.

Stage 2: Exact Correction. For proposals that pass the surrogate stage, the algorithm performs a full forward and backward pass to evaluate the exact change in the target logit

$$\Delta M_t = M_t(x') - M_t(x),$$

and computes the reverse-proposal probability $\phi(x \mid x')$ and forward-proposal probability $\phi(x' \mid x)$,

each defined by the same gradient-based Boltzmann sampler used in MHIS. The corrected log acceptance ratio is then

$$\log r_2 = \big[\log p(x') - \log p(x)\big] + \frac{\Delta M_t}{T} + \log\frac{\phi(x \mid x')}{\phi(x' \mid x)} - \log r_1.$$

A new uniform random variable $u_2 \sim \mathrm{Uniform}(0,1)$ is drawn, and the proposal is accepted if $\log r_2 \geq 0$ or $\log u_2 \leq \log r_2$. If accepted, the Markov chain transitions to x'; otherwise it remains at x.

Properties and Efficiency. By construction, the two-stage procedure preserves detailed balance with respect to the target distribution $q(x)$, ensuring that DA-MHIS yields an unbiased importance-sampling estimator identical in expectation to MHIS. However, because the majority of proposals are filtered out during Stage 1, the algorithm requires far fewer full forward passes. The gradient g computed at the current state can be reused across many proposals, amortizing its cost across multiple surrogate tests. See Figure 1 for a visual representation of the algorithm.

Multiple-Try Metropolis Hastings Importance Sampling (MT-MHIS)

Wu & Hilton's original MHIS algorithm is limited by utilizing only one proposed token change at each iteration. If the proposal distribution $\phi(x'|x)$ suggests an unsatisfactory

next step toward the target distribution, many iterations may be required by the Markov Chain to converge to a consistent target distribution. To address this, we introduce a *Multiple–Try Metropolis* variant of MHIS (MT-MHIS), which evaluates many candidates in parallel and selects the most appropriate candidate for proposal. Although the number of calculations in MT-MHIS versus MHIS is not reduced at each step, the MT-MHIS method attempts to reduce the number of steps required for convergence. MT-MHIS also takes advantage of parallel GPU computation, thereby improving practical efficiency.

In MTM–MHIS, given a current input x, the algorithm draws a set of K candidate proposals $\{x'_1, x'_2, \ldots, x'_k\}$ from the proposal distribution $\phi(x'|x)$. Each candidate is assigned a weight proportional to its Boltzmann-weighted likelihood under the target distribution:

$$w_j = p(x'_j) \exp\left(\frac{M_t(x'_j)}{T}\right).$$

One candidate x' is then selected with probability proportional to its weight, $w_j / \sum_{i=1}^{K} w_i$, and is chosen as the tentative move. For the preservation of detailed balance, MT-MHIS draws a reverse set of $K-1$ proposals $\{x_1, x_2, \ldots, x_{k-1}\}$ from $\phi(x|x')$ and includes the current state x in this reverse pool. The acceptance probability for transitioning from x to x' is given by:

$$\alpha(x, x') = \min\left(1, \frac{\sum_{j=1}^{k} w_j(x'_j)\phi(x'_j|x)}{\sum_{j=1}^{k} w_j(x_j)\phi(x_j|x')}\right).$$

This acceptance rule ensures detailed balance is maintained as per the distribution of MHIS, $q(x) \propto p(x)\exp(M_t(x)/T)$, ensuring the estimator is unbiased.

4. Results

Method	Runtime (sec)	MSE
MHIS	13559	5.71×10^{-11}
MT-MHIS (n=2)	13669	1.02×10^{-09}
MT-MHIS (n=4)	13608	3.41×10^{-08}
MT-MHIS (n=8)	13727	1.17×10^{-10}
DA-MHIS	**11170**	$\mathbf{7.91 \times 10^{-12}}$

Table 1: Runtime and accuracy of baseline MHIS vs our optimization methods on GELU-1l

To evaluate the effectiveness of the proposed algorithms, we tested DA-MHIS and MT-MHIS against the original MHIS algorithm. Initial experiments were conducted on a GELU 1-layer model for proof of concept. To ensure a representative sample, we tested on $n = 350$ ground truths. Our experiments utilized a compute budget of 2^{16} model calls, and we maintained the burn-in period of 2^{10} batches from Wu and Hilton's methodology. Model temperature was also set to 0 to maintain deterministic results. Experiments were performed on an NVIDIA RTX 4090 24GB GPU. To evaluate accuracy, the mean squared error (MSE) was calculated against ground-truth probabilities. The resulting runtime and accuracy is reported in Table 1.

We find that on the GELU 1-layer model reduces runtime by 16.4% as compared to the baseline. Hence, we repeated our experiments on incrementally larger models: GELU 2-layer, GELU 4-layer, and GPT-2. The results from these experiments can be seen in Figure 2. Figure 2 also demonstrates that DA-MHIS maintained comparable accuracy as compared to original MHIS on the GELU 1-layer and GELU 2-layer models. However, when scaling up to GPT-2, the performance of both methods deviated significantly from the ground truth. We speculate that the GPT-2 estimates skew low because in the 10^{-9} regime and with a fixed 2^{16}-call budget, effective sample sizes are tiny and the chain rarely visits the rare-event region. Self-normalized MHIS yields zero or near-zero in most runs, pulling estimates below the ground truth. Small proposal–target mismatch and chain correlation further reduce hit rates; any log-prob precision/clipping can accentuate the downward skew. With more compute or a better-matched proposal, the estimates rise toward the ground truth.

To test MT-MHIS, we repeated initial experiments on a GELU 1-layer model, varying the number of candidates proposed by MT-MHIS for each experiment. Henceforth, the number of candidates will be referred to as n. MT-MHIS was tested on a sweep of $n = [1, 2, 4, 8]$, and the results can be observed in Figure 3. We can verify the faithfulness of our algorithm by observing the results on MT-MHIS where $n = 1$. With one proposed candidate, the accuracy is on par with the original MHIS. However, we observe that as the number of proposals per iteration increases, MT-MHIS estimates experience an upward drift. This can be attributed to the phenomenon where increasing n increases the probability of selecting over-weighted proposals, leading to a small positive bias in accepted samples.

Additionally, as indicated in Table 1, MT-MHIS did not offer significant runtime improvements at any n.

5. Discussion

Our results highlight two optimization methods for improving the efficiency of the MHIS algorithm. It is important to introduce highly-efficient, scalable algorithms so that rare behavior elicitation can be extended to large-scale LLM-based agents with a more flexible range of capabilities.

Delayed-Acceptance Metropolis Hastings Importance Sampling (DA-MHIS) introduces a two-stage screening process which immediately discards unsatisfactory proposals. This method has demonstrated significant runtime improvements without sacrificing accuracy.

On the other hand, Multiple-Try Metropolis Hastings Importance Sampling (MT-MHIS) relies on selecting the "best" proposed next token from multiple candidates at each step. However, the theoretical parallel runtime advantages of MT-MHIS are only actualized when there is an under-utilization of GPU resources. When GPU capacity is under-utilized, batching is able to amortize the cost of evaluating multiple proposals. However, in the case where the hardware capacity is fully utilized, extra candidates proposed by MT-MHIS are queued rather than executed in parallel.

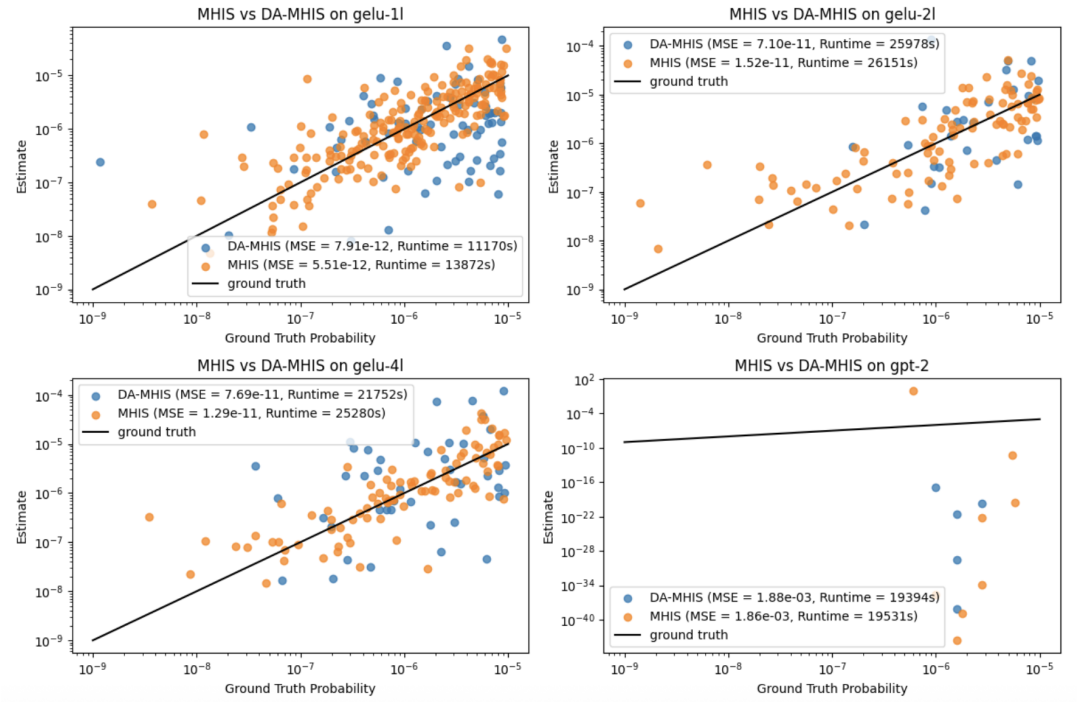

Figure 2: Performance of MHIS versus DA-MHIS. We test both methods on the gelu-1l, gelu-2l, gelu-4l, and gpt-2 models, plotting the estimates against the ground truth diagonal—estimates closer to the main diagonal are more accurate. The plot also includes the mean squared error across 350 trials, as well as the runtime across the various models.

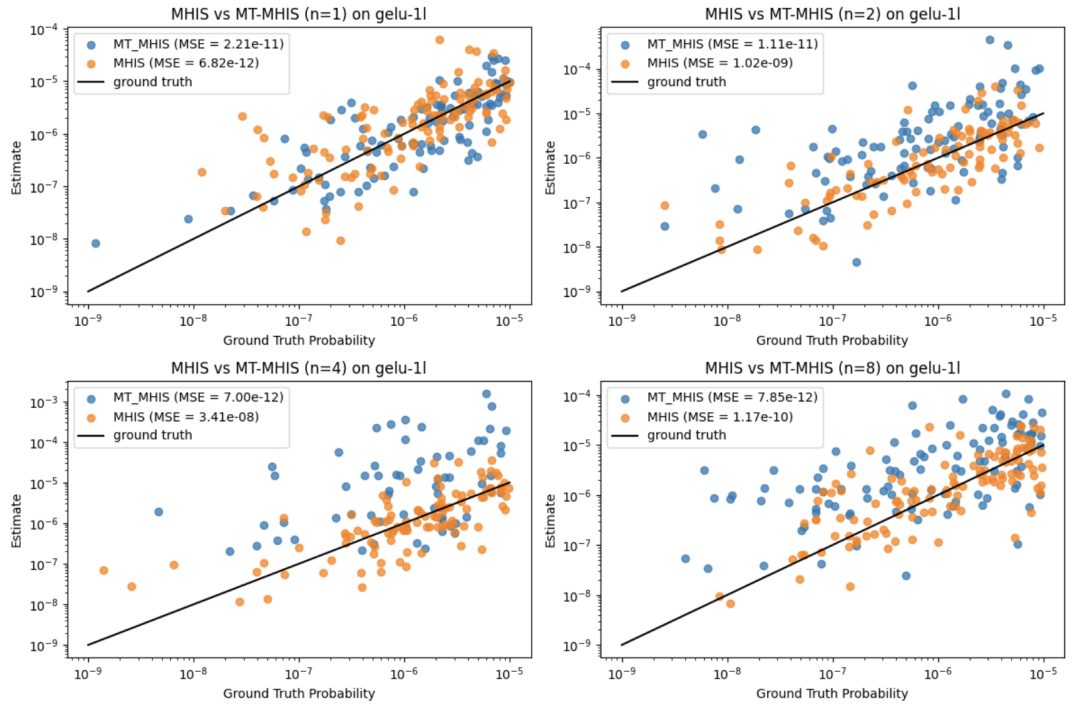

Figure 3: Performance of MHIS versus MT-MHIS. Here, we test MHIS versus MT-MHIS on the gelu-1l model, varying the number of candidates generated by MT-MHIS. We utilize a sweep of $n = [1, 2, 4, 8]$ candidates.

Upon examining the Streaming Multiprocessor (SM) utilization of our original experiments, we find that the original MHIS exhibits on average 98% SM utilization, revealing that the GPU is already saturated to nearly full capabilities. The SM utilization from MT-MHIS attained similar levels, hence runtime improvements were not observed. In order to examine this further, we repeated our experiments on a 48GB GPU. In these experiments, we continued to observe no clear runtime improvement exhibited by MT-MHIS as well as similar average rates of SM utilization. Our experiments are constrained by hardware access, but MT-MHIS may exhibit potential for large, multi-GPU setups.

Hence, we identify DA-MHIS as a promising method for optimizing the efficiency of the Metropolis Hastings Importance Sampling algorithm. DA-MHIS delivers substantial improvements upon runtime without sacrificing low-probability estimation accuracy.

We would be excited about further research on scaling such optimization methods to larger models, through methods such as examining the gradient descent processes to determine whether current efficiency gains scale consistently with model size.

6. Acknowledgements

We are grateful to Jacob Hilton at the Alignment Research Center and Gabriel Wu at OpenAI for their feedback on our work, and Kevin Zhu and the organizers of the Algoverse AI Research Program for the computing and mentorship resources.

References

Bai, T.; Luo, J.; Zhao, J.; Wen, B.; and Wang, Q. 2021. Recent Advances in Adversarial Training for Adversarial Robustness. arXiv:2102.01356.

Banterle, M.; Grazian, C.; Lee, A.; and Robert, C. P. 2015. Accelerating Metropolis-Hastings algorithms by Delayed Acceptance. arXiv:1503.00996.

Doig, R.; and Wang, L. 2025. A unified framework for multiple-try Metropolis algorithms. arXiv:2503.11583.

Huang, Y.; Gupta, S.; Xia, M.; Li, K.; and Chen, D. 2023. Catastrophic Jailbreak of Open-source LLMs via Exploiting Generation. arXiv:2310.06987.

Højmark, A.; Pimpale, G.; Panickssery, A.; Hobbhahn, M.; and Scheurer, J. 2024. Analyzing Probabilistic Methods for Evaluating Agent Capabilities. arXiv:2409.16125.

Liu, X.; Cheng, H.; He, P.; Chen, W.; Wang, Y.; Poon, H.; and Gao, J. 2020. Adversarial Training for Large Neural Language Models. arXiv:2004.08994.

Phuong, M.; Aitchison, M.; Catt, E.; Cogan, S.; Kaskasoli, A.; Krakovna, V.; Lindner, D.; Rahtz, M.; Assael, Y.; Hodkinson, S.; Howard, H.; Lieberum, T.; Kumar, R.; Raad, M. A.; Webson, A.; Ho, L.; Lin, S.; Farquhar, S.; Hutter, M.; Deletang, G.; Ruoss, A.; El-Sayed, S.; Brown, S.; Dragan, A.; Shah, R.; Dafoe, A.; and Shevlane, T. 2024. Evaluating Frontier Models for Dangerous Capabilities. arXiv:2403.13793.

Webb, S.; Rainforth, T.; Teh, Y. W.; and Kumar, M. P. 2019. A Statistical Approach to Assessing Neural Network Robustness. arXiv:1811.07209.

Wu, G.; and Hilton, J. 2025. Estimating the Probabilities of Rare Outputs in Language Models. https://arxiv.org/abs/2410.13211. arXiv:2410.13211.

Xhonneux, S.; Sordoni, A.; Günnemann, S.; Gidel, G.; and Schwinn, L. 2024. Efficient Adversarial Training in LLMs with Continuous Attacks. arXiv:2405.15589.

Zhao, M.; Zhang, L.; Ye, J.; Lu, H.; Yin, B.; and Wang, X. 2024. Adversarial Training: A Survey. arXiv:2410.15042.

Zou, A.; Wang, Z.; Carlini, N.; Nasr, M.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. arXiv:2307.15043.