CAN LLMS DO SAMPLING ?

Mengjie Zhao	Ruicheng Li	Tianxing Zhou	Zhikai Qin
2023010823	2023011323	2023012178	2023011423

Author Contributions

Zhikai Qin proposed the idea, designed and ran some experiments for the first part, constructed the dataset and benchmark and ran finetuning and experiments for the later part. And the corresponding paper writing for benchmark and training part.

Mengjie Zhao designed and implemented the code framework and pipeline for the first part, ran some experiments for the first part. And corresponding analysis and paper writing.

Ruicheng Li contributed to the design of the experimental framework, performed data analysis and result interpretation. And proposed suggestions for improvement and identifying project limitations.

Tianxing Zhou contributed to the code framework and conducted experiments of logits analysis. Data analysis, result interpretation/suggestions and plotting and corresponding report writing.

Used AI tools Kimi to format latex and filling forms. Also to generate PCFGs. Deepseek to help design different sampling tasks. Copilot to assist coding, including evaluation and finetuning.

AutoDL Usage QZK use about 200r to do sampling experiments on-device.

ABSTRACT

Large language models (LLMs) have shown impressive performance across various tasks, but their ability to perform probablistic sampling remains underexplored. In this work, we systematically investigate how modern instruction-tuned LLMs interpret and execute sampling instructions under diverse experimental settings. Through extensive experiments spanning various tasks and configurations, we uncover a consistent failure mode: while aggregate sampling statistics may appear correct, individual prompts often induce biases. Motivated by this observation, we identify zero-example, one-length sampling as the most fundamental and challenging setting for evaluating genuine sampling ability. We formalize this setting via a large-scale binary (0/1) sampling benchmark and introduce a simple yet revealing metric based on the distribution of per-prompt probability imbalance. Our results show that although the expected imbalance is near zero, its variance is extremely large, indicating prompt-dependent determinism rather than true randomness. Finally, we demonstrate that targeted fine-tuning on carefully constructed sampling tasks substantially reduces this variance while preserving general task performance, suggesting that sampling instability is not an inherent limitation but a learnable deficiency.

1 INTRODUCTION

Of course LLMs generates texts by doing sampling. However, can LLMs mathematically do sampling? Sampling is a cornerstone of probabilistic modeling, with applications ranging from simulation and randomized algorithms to decision-making under uncertainty. With the rapid advancement of large language models (LLMs), an implicit assumption has emerged: because LLMs are trained to model probability distributions over tokens, they should naturally be capable of sampling from user-specified distributions. However, recent empirical evidence suggests that this assumption may be fragile.

Prior studies have shown that LLMs often produce biased, overly deterministic, or prompt-sensitive outputs when asked to generate random numbers or samples from simple distributions. At the same time, other work suggests that LLMs can strategically modulate their apparent randomness, raising questions about whether observed failures reflect genuine incapability or misalignment between instructions and internal representations.

In this project, we aim to investigate the fundamental question: **Do LLMs truly understand and execute sampling instructions in a reliable manner?** To answer this, we conduct a large-scale empirical study across models, tasks, and prompting methods. Our investigation leads to three key insights:

- Across a wide range of sampling tasks, LLMs often exhibit correct global statistics by aggregate on multiple histories, but individual sampling attempts are highly unstable and biased, indicating a failure to understand the stochastic nature of the task.
- The most revealing failure cases arise in the zero-example, one-length setting, where the model must perform a single stochastic draw without relying on history or pattern imitation.
- Sampling instability can be significantly mitigated through targeted fine-tuning, without degrading performance on standard benchmarks.

These findings suggest that while current LLMs possess some basic sampling capabilities, they lack a robust understanding of randomness and uncertainty. Addressing this gap will be crucial for deploying LLMs in applications requiring reliable probabilistic reasoning.

2 Related Work

Early work on LLM sampling focused on whether models could generate random numbers or samples from simple distributions. Aspen K Hopkins evaluated autoregressive and non-autoregressive sampling strategies and found that many models struggled to match target distributions, even in controlled settings. Subsequent studies by Paruchuri et al. (2024) expanded this scope to probabilistic reasoning tasks, showing that carefully designed prompts and context can improve performance, though systematic biases often persist.

More recent work has highlighted the surprisingly deterministic behavior of LLMs when asked to act as random generators. Coronado-Blázquez (2025) argues that such failures may stem from training data biases and human-like cognitive heuristics rather than decoding stochasticity alone. Another study by Karanjai et al. (2025) evaluate randomness using entropy-based metrics across task-level outputs, revealing deficiencies beyond simple number generation.

A complementary line of research suggests that LLMs may possess latent sampling capabilities but choose not to express them reliably. The AI Sandbagging van der Weij et al. (2025) hypothesis proposes that models can strategically underperform on evaluations, implying a deeper but unstable internal representation of randomness.

3 Methods

3.1 Massive Experiments and Empirical Discoveries

We begin with a broad empirical investigation of LLM sampling behavior. Our experiments span: **Task Diversity:** a variety of sampling tasks, including binary sampling (e.g., coin flips), categorical choice (e.g., dice rolls), and survey response generation; **Probability Distributions:** sampling from various distributions, including uniform, biased, and multimodal; **Prompt Contexts and History Length:** the amount of context provided, from zero-example prompts to few-shot demonstrations. Also, we tests different history lengths, i.e., the number of samples to be drawn in one prompt; **Model Variety:** multiple SOTan LLMs, including *Qwen2.5-7B-Instruct*, *Qwen2.5-14B-Instruct*, *DeepSeek-R1*, and *Kimi-K2*.

Across these settings, we observe several consistent patterns:

Discovery 1: Long sampling histories enable apparent distribution matching

When models are asked to produce a sequence of samples and the history length is sufficiently large, the empirical distribution of outputs tends to matches the target distribution as in Figure 1. Moreover, this effect becomes increasingly evident as the history length grows. In other words, the longer the sampling sequence, the easier it is for the model to "correct" its outputs so that the overall frequency aligns with the desired probabilities. This behavior is consistent across tasks, models, and distribution types.

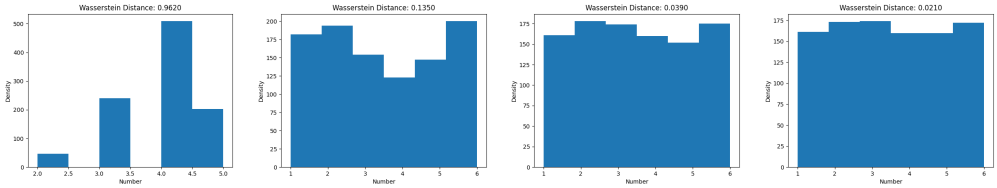
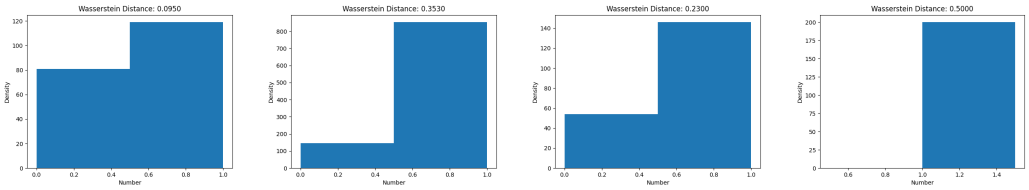


Figure 1: Effect of History Length [1, 10, 25, 50] on Kimi-K2, dice-rolls(1-6) performance.



(a) DeepSeek-R1 (b) Kimi-K2 (c) Qwen2.5-7B (d) Qwen2.5-14B

Figure 2: Performance when History Length=1 on coin-flip (0/1) across models.

Discovery 2: Strong prompt-level bias emerges when the history length collapses to 1.

In contrast, when the sampling history length is reduced to one, i.e., the model is asked to produce a single sample per prompt, we observe obvious biases as shown in Figure 2. Repeating across multiple independent trials reveals that output probabilities vary drastically between prompts, despite identical underlying distributions. A particularly representative case is the binary (0/1) sampling task with a symmetric 50/50 target distribution, where models exhibit a clear bias toward outputting 1 far more frequently than 0.

Discovery 3: One-Length sampling constitutes the most difficult and fundamental failure mode.

Taken together, these results indicate that 1-history length sampling is not merely a harder version of multi-sample tasks, but a fundamentally different and more challenging setting. While models can approximate correct probabilities when allowed to average over long sequences, they fail to produce correct sampling behavior when required to generate a single, isolated draw. This suggests that current LLMs may not be possessing a stable, prompt-invariant awareness of probabilistic sampling.

3.2 One-Length Binary Sampling Benchmark

Next, we focus on the one-length binary sampling setting, which we argue is the most revealing test of true sampling ability.

Benchmark Dataset Design

To evaluate LLM's one-length binary sample (OLBS) ability, we construct several OLBS-prompt datasets. Each dataset consists of 10,000 piece of prompts, induced by a given probability pair and token pair. Some examples are shown in appendix A

As can be seen in the examples, each piece of prompt in a dataset is a different sampling task, which asks the llm to generate solely one token representing the sampling result, according to the given context.

We construct the datasets mainly by LLMs and some meta-prompts, as shown like this one in appendix A. We may name a dataset like `llm_name_probs_tokens`, e.g. `Qwen2.5_7B-Instruct_0.5_0.5_01`, which means the dataset is generated by *Qwen2.5_7B-ShoInstruct*, the probability of sampling the first token is 0.5, and thus 0.5 for the second, and the two corresponding tokens are 0 and 1. Then we may denote this dataset by $D = \{C_i\}$, and

$$P_D = \{p_1^D, p_2^D, \cdots, p_m^D\}$$

is the expected probability of our demanded tokens. We specifically have $m = 2$ under our OLBS setting, but we may use more generalized expressions.

Under these settings, we may obtain the three statistics value for a given LLM, and then evaluate their OLBS ability.

BENCHMARK METRIC DESIGN

Given this dataset $D = \{C_i\}$, we can run any LLM to test their OLBS ability. More specifically, for an LLM L, we define a function f meaning 'picking the prob of token_j at the first generated position', then we have:

$$f_{C_i}(L) = (p_1, p_2, \cdots, p_m)^i,$$

where m is the size of set of sampling token under a given dataset. Under our binary evaluation, we have $m = 2$.

Then we further define a non-negative metric $M_{P_D}(\{p_j\}^i)$, to evaluate the distance between $\{p_j\}^i$ and the demanded probability P_D of our dataset. We can then have these three statistics value for the whole dataset,

$$\overline{w^D}(L) = M_{P_D}\left(\{\overline{p_j}\}\right), \tag{1}$$

$$\overline{w^D(L)} = \mathbb{E}_{i\in|D|}[w_i^D(L)], \tag{2}$$

$$\mathrm{Var}(w^D) = Var_{i\in|D|}[w_i^D(L)]. \tag{3}$$

The first metric (1) stands for: we average over all generated probs $\{p_j\}^i$ to obtain overall prob $\{\overline{p_j}\}$, and compute the distance between it and P_D, in order the overall ability for an LLM to average a probablity. The second metric (2) means to average over all distance computed under each given prompt, evaluating the average distance of an LLM to generate by a given probability under different given contexts. The third metric (3) is the squared version of the second one, evaluating the divergence of an LLM to generate by a given probability under different given contexts.

One can see that intuitively, all these metrics should be the lower the better.

For the choice of $M_{P_D}(\{p_j\}^i)$, we utilize a classic Wasserstein Distance, defined as

$$WD_A(P, Q) = \int_{x\in A} |p(x) - q(x)|dx, \tag{4}$$

and so our M_{P_D} is simply

$$M_{P_D}(\{p_j\}^i) = WD_{M_D}(P_D, \{p_j^*\}^i), \tag{5}$$

where p_j^* stands for the normalized p_j, i.e., $p_j^* = \frac{p_j}{\sum_k p_k}$.

EXPERIMENTAL RESULTS

3.3 ONE-LENGTH BINARY SAMPLING BENCHMARK

We evaluate three models, Qwen2.5-7B-Instruct Qwen2.5-14B-Instruct and Qwen2.5-0.5B-Instruct under diffenert settings as our baselines, and the result is shown in table 2, and the corresponding dataset settings can be find in table 3.

One may discover several things for a given LLM and our benchmark.

- The three metrics are mostly independent. an LLM may obtain a low $W^{\overline{D}}$, which means averaged by all context it can obtain plausible distribution, but not that good for each context, with high $\overline{W}$ and Var. Hence they independently reveals different ability for an LLM when doing sampling, and the second and the third metric can better evaluate how it do sampling under varioud contexts.
- Sampling ability doesn't have to relate to model size. In fact, more data to trained may bring more bias within tokens that degenerates sampling ability.

Table 1: Baseline results A ($\times 10^{-2}$).

Model	A			A_no_sys			A_sim_sys		
	$W^{\overline{D}}\downarrow$	$\overline{W}\downarrow$	Var$\downarrow$	$W^{\overline{D}}$	$\overline{W}$	Var	$W^{\overline{D}}$	$\overline{W}$	Var
Qwen2.5-0.5B-Instruct	4.9	21.3	3.5	28.7	45.1	6.9	64.2	65.7	4.0
Qwen2.5-7B-Instruct	2.2	77.2	7.9	38.5	75.6	8.9	23.5	69.9	10.2
Qwen2.5-14B-Instruct	71.3	92.1	3.6	3.3	81.7	7.0	2.0	84.5	6.4

- System prompt matters. If your system prompt fails to demand the LLM to sample 0 or 1 at the first token, you may receive no good result.
- Specific token pair matters. an LLM may be familiar to sample with certain token pairs, while fails with another.

Table 2: Baseline results ABCD($\times 10^{-2}$).

Model	A			B			C			D		
	$W^{\overline{D}}\downarrow$	$\overline{W}\downarrow$	Var$\downarrow$	$W^{\overline{D}}$	$\overline{W}$	Var	$W^{\overline{D}}$	$\overline{W}$	Var	$W^{\overline{D}}$	$\overline{W}$	Var
Qwen2.5-0.5B-Instruct	4.9	21.3	3.5	2.3	23.0	3.4	15.3	24.0	3.6	74.9	77.1	4.2
Qwen2.5-7B-Instruct	2.2	77.2	7.9	72.1	72.3	8.5	32.4	51.6	4.9	75.8	89.3	4.8
Qwen2.5-14B-Instruct	71.3	92.1	3.6	23.7	86.8	5.8	33.4	55.9	6.7	11.3	82.6	6.8

Table 3: Dataset and system-prompt settings reference.

Dataset ID & Configuration	
ID	**Setting**
A	Qwen2.5-7B-Instruct_0.5_0.5_01
B	Llama2-7B-chat_0.5_0.5_01
C	Qwen2.5-7B-Instruct_0.75_0.25_01
D	Qwen2.5-7B-Instruct_0.5_0.5_ab
System Prompt Styles	
default	"You output only a single digit 0 or 1."
no_sys	"You are a helpful assistant."
sim_sys	"Answer with 0 or 1 and nothing else."

3.4 Fine-tuning for Sampling Stability

Then the question comes to: how do we boost LLMs' OLBS ability? An LLM's OLBS ability may be determined by several aspects: it's understanding of the sampling task, the token biases induced by the training set. The main aspect we aim to tackle is to de-bias the model's token to a stable distribution.

Hence the solution is simple: we finetune the model on the dataset we generate. First, we sample the target token by the corresponding distribution in advanced for each context in the dataset, where we may obtain a complete context-target pair. Since the target is sample by P_D, the model's distribution on the set M_D is expected to be dragged to P_D. This is due to the learning target is actually

$$L = -\sum_j p_j^D \log p_j, \tag{6}$$

where whatever P_D is, learning $P_L = P_D$ gives the loweset loss.

We finetune the three baseline models under the unbiased dataset A, splitted to train and test part, for one epoch, and then test under different datasets. The following tables 4 5 show our result.

One may find that, the model's sampling ability has been significantly prompted under the original task, shown by the decline of overall metrics. The ability can be generalize to different system prompts, and a little bit to different token pair. Morever, the model's original language ability has not been affected, see the evaluation in 6.

However, these shows also significant overfit to a certain distribution, as the model stably gives $0.5 - 0.5$ output to $0.75 - 0.25$ task.

Table 4: Tuned results A ($\times 10^{-2}$).

Model	A			A_no_sys			A_sim_sys		
	$W^{\overline{D}}\downarrow$	$\overline{W}\downarrow$	Var↓	$W^{\overline{D}}$	$\overline{W}$	Var	$W^{\overline{D}}$	$\overline{W}$	Var
Qwen2.5-0.5B-Instruct	4.9	21.3	3.5	28.7	45.1	6.9	64.2	65.7	4.0
UnbiasedTuned-Qwen0.5B	0.02	2.52	0.11	5.04	33.7	5.40	0.75	8.36	4.97
UnbiasedTuned-Qwen7B	3.68	3.80	0.07	13.23	29.08	8.63	1.98	4.06	0.17
UnbiasedTuned-Qwen14B	0.97	1.94	0.80	5.09	12.24	3.50	0.97	1.31	0.06

Table 5: Tuned results ABCD($\times 10^{-2}$).

Model	A			B			C			D		
	$W^{\overline{D}}\downarrow$	$\overline{W}\downarrow$	Var↓	$W^{\overline{D}}$	$\overline{W}$	Var	$W^{\overline{D}}$	$\overline{W}$	Var	$W^{\overline{D}}$	$\overline{W}$	Var
Qwen2.5-0.5B-Instruct	4.9	21.3	3.5	2.3	23.0	3.4	15.3	24.0	3.6	74.9	77.1	4.2
UnbiasedTuned-Qwen0.5B	0.02	2.52	0.11	0.26	3.98	0.20	48.91	48.91	0.21	22.56	22.57	0.07
UnbiasedTuned-Qwen7B	3.68	3.80	0.07	3.74	4.40	0.19	50.64	50.64	0.98	20.19	20.45	0.41
UnbiasedTuned-Qwen14B	0.97	1.94	0.80	1.74	1.95	0.08	48.91	48.91	0.07	9.51	9.53	0.29

Therefore we also conducted further experiments, aiming the model to learn more generalizable distributions. However, due to the limited time, we only merged two dataset A and C to finetune the model, and find the model performing well on the test set of the both. The result can be find in 7 8.

4 Discussion and Conclusion

In this work, we systematically analyse LLMs sampling ability. We designs extensive sampling tasks, and test several models under different settings. We find LLM's sampling abilibity to be prompt-dependent, not distributionally stable. They may aslo cheat in sampling, refering their own histories.

Therefore, we pick zero-shot, one-length sampling as the revealing failure mode. Under this case, LLM fails to sample for most settings, and appears extreme modes. We further design a benchmark, including a set of datasets and corresponding metrics to expose LLM's zero-shot, one-length binary sampling ability.

After the evaluating process, we find LLM sometimes being able sample averaging all context, however fails to do sampling under a certain one. And the overall OLBS ability of tested LLMs reveals not satisfactory.

Anyway, we find sampling instability not an inherent limitation. By carefully designed finetuning, we may highly boost LLM's sampling results under different distributions, without degenerating LLM's original ability. We explain this by drawing the biased token distribution, introduced by the training dataset that builds the LLM, to an balanced distribution, combined with integrating its ability to calculate the corresponding probability.

For future work, we shall design more continuous benchmark and tuning fashion with larger sampling token sets, instead of a binary task with a fixed distribution. We shall also dry more various LLMs of different size, in order to discover more generalizable patterns.

ACKNOWLEDGMENTS

We come to this idea mostly thanks to the AI sandbagging paper van der Weij et al. (2025) mentioned by Tianxing He's lecture. We doubt the paper' plotting concerning letting LLM judge correctly due to a given probability. The basic problem is whether LLM can sample a token given a certain distribution.

REFERENCES

Alex Renda Aspen K Hopkins. Can llms generate random numbers? evaluating llm sampling in controlled domains. URL https://par.nsf.gov/biblio/10498692.

Javier Coronado-Blázquez. Deterministic or probabilistic? the psychology of llms as random number generators. *arXiv preprint arXiv:2502.19965*, 2025.

Rabimba Karanjai, Yang Lu, Ranjith Chodavarapu, Lei Xu, and Weidong Shi. Evaluating the quality of randomness and entropy in tasks supported by large language models. *arXiv preprint arXiv:2510.12080*, 2025.

Akshay Paruchuri, Jake Garrison, Shun Liao, John Hernandez, Jacob Sunshine, Tim Althoff, Xin Liu, and Daniel McDuff. What are the odds? language models are capable of probabilistic reasoning, 2024. URL https://arxiv.org/abs/2406.12830.

Teun van der Weij, Felix Hofstätter, Ollie Jaffe, Samuel F. Brown, and Francis Rhys Ward. Ai sandbagging: Language models can strategically underperform on evaluations, 2025. URL https://arxiv.org/abs/2406.07358.

A PROMPTS

OLBS prompt examples:

1. (Qwen uniform A v.s. B)
Consider a digital slot machine with two possible outcomes: stopping at slot 1 or slot 2, each with equal probability. Slot 1 triggers output A, while slot 2 triggers output B. Simulate this for 1 sample(s) and return solely the resulting letter.
2. (Qwen 75% 0 v.s. 25% 1)
Select a random item from a bowl containing three red marbles and one blue marble. Drawing a red marble (75% chance) corresponds to outputting 0, while drawing the blue marble (25% chance) corresponds to outputting 1. Run 1 sample(s) and generate nothing but the required output.

Dataset Generate Prompt

Generate a new, single random event description prompt that adheres to these specifications: 1. **Core Requirements:** * The scenario must have exactly two possible outcomes. * The probability of each outcome must be precisely 0.5 (perfectly random, unbiased). * Clearly map one outcome to output '1' and the other to output '0'. * The final instruction must explicitly state that the action is performed for "**1 sample(s)**". * **Do NOT reuse coin flips, dice rolls, binary bit generators, or quantum spin motifs. Pick a different everyday, scientific, or imaginative setup each time.** * **Vary the opening phrasing. Do not start every prompt with the same word. Rotate between forms like "You are...", "There is...", "Imagine...", "Run...", "Observe...", "Select...", "Consider...", "Perform...", etc.**
2. **Format Variation Creativity (CRITICAL):** * **Vary the structure significantly.** Avoid the formula: "[Scenario]. If X, output 1. If Y, output 0. Constraint. Perform 1 sample(s)." Also avoid repeating the same setting across generations; choose a fresh domain (nature, sports, lab experiment, gadgets, games other than coins/dice, social draws, etc.). * Experiment with: * **Sentence order:** Start with the output rule, or with the action. * **Phrasing:** Use varied terms for probability ("equal odds", "50/50 chance", "just as likely

as”), for the action (”simulate”, ”observe”, ”run a trial of”), and for the output constraint (”Return ONLY...”, ”Your answer must be exclusively...”, ”Generate NOTHING but...”). * **Instruction style:** The prompt can be framed as a command, a simulation request, or a thought experiment. * **Lead-in diversity:** Mix imperative, descriptive, and narrative openings; do not repeat the same first two words across consecutive prompts. * **Optional Concluding Phrase:** You may optionally add a brief concluding instruction *after* stating the sample count to enrich the grammar (e.g., ”and report the result”, ”then provide the output”, ”let’s begin the trial”, ”and tell me what you get”). This phrase should be short and natural, varying with each generation.
3. **Clarity Conciseness:** * The scenario should be intuitive and the instructions unambiguous.
Your Goal: Invent a creative, new scenario where the **only fixed, non-negotiable format element** is the inclusion of ”**1 sample(s)**” as the trial count. Everything else should feel fresh and different from the examples.
—
Examples (Note the structural diversity, non-coin settings, and varied openings):
* ”You are calibrating a lab heater that picks either low or high mode with exactly equal odds. Low maps to 0, high maps to 1. Run one toggle for 1 sample(s) and output only the mapped digit.” * ”There is a rescue drone that either flies north or east on launch, each equally likely. North corresponds to 1; east to 0. Launch it once for 1 sample(s) and give only the resulting number.” * ”Imagine two synchronized metronomes; on a random tick one chimes first. Left chime maps to 1, right chime to 0. Observe one tick for 1 sample(s) and return just the code.” * ”Run a quick test on a smart sprinkler that randomly waters zone A or zone B with 50/50 odds. Zone A is 1, zone B is 0. Execute one watering for 1 sample(s) and report only the number.” * ”Observe a pair of identical RC cars set to choose path glass or path gravel with equal probability. Glass path encodes 1; gravel encodes 0. Let them go once for 1 sample(s) and provide only the digit.” * ”Select at random which musician in a duo starts the song: the guitarist or the drummer, each just as likely. Guitarist start is 1; drummer start is 0. Make one selection for 1 sample(s) and output only the numeric result.” * ”Please pick a ball from a box. There are 100 red balls and 100 blue balls in the box and nothing else. If you pick a red ball, output 1. If you pick a blue ball, output 0. OUTPUT NOTHING ELSE BUT 0 or 1. You will pick 1 sample(s) of balls. And output the result” * ”An unbiased coin is tossed. That means the probability of heads is 0.5 and the probability of tails is 0.5. If you get the heads, output 1. If you get the tails, output 0. OUTPUT NOTHING ELSE BUT 0 or 1. You will toss 1 sample(s) of coins. And output the result” * ”Here is a a perfectly shuffled standard deck and you draw a single card, but you only care about its color. There are 26 black cards (clubs and spades) and 26 red cards (hearts and diamonds). If you draw a black card, output 1. If you draw a red card, output 0. OUTPUT NOTHING ELSE BUT 0 or 1. You will draw 1 sample(s) of cards. And output the result” * ”You are pressing a perfectly random button that toggles a light. The light has an equal chance of being ON or OFF before you press. If pressing the button turns the light ON, output 1. If pressing the button turns the light OFF, output 0. OUTPUT NOTHING ELSE BUT 0 or 1. You will press the button for 1 sample(s). And output the result” * ”A perfectly balanced, two-sided spinner labeled ’A’ and ’B’ is on the table. Each side has an equal probability of landing face-up. If it lands on ’A’, output 1. If it lands on ’B’, output 0. OUTPUT NOTHING ELSE BUT 0 or 1. You will spin 1 sample(s). And output the result”
—
Now, please generate one new, structurally unique prompt. Output only the prompt text without any additional commentary or code.

B EXPERIMENTS

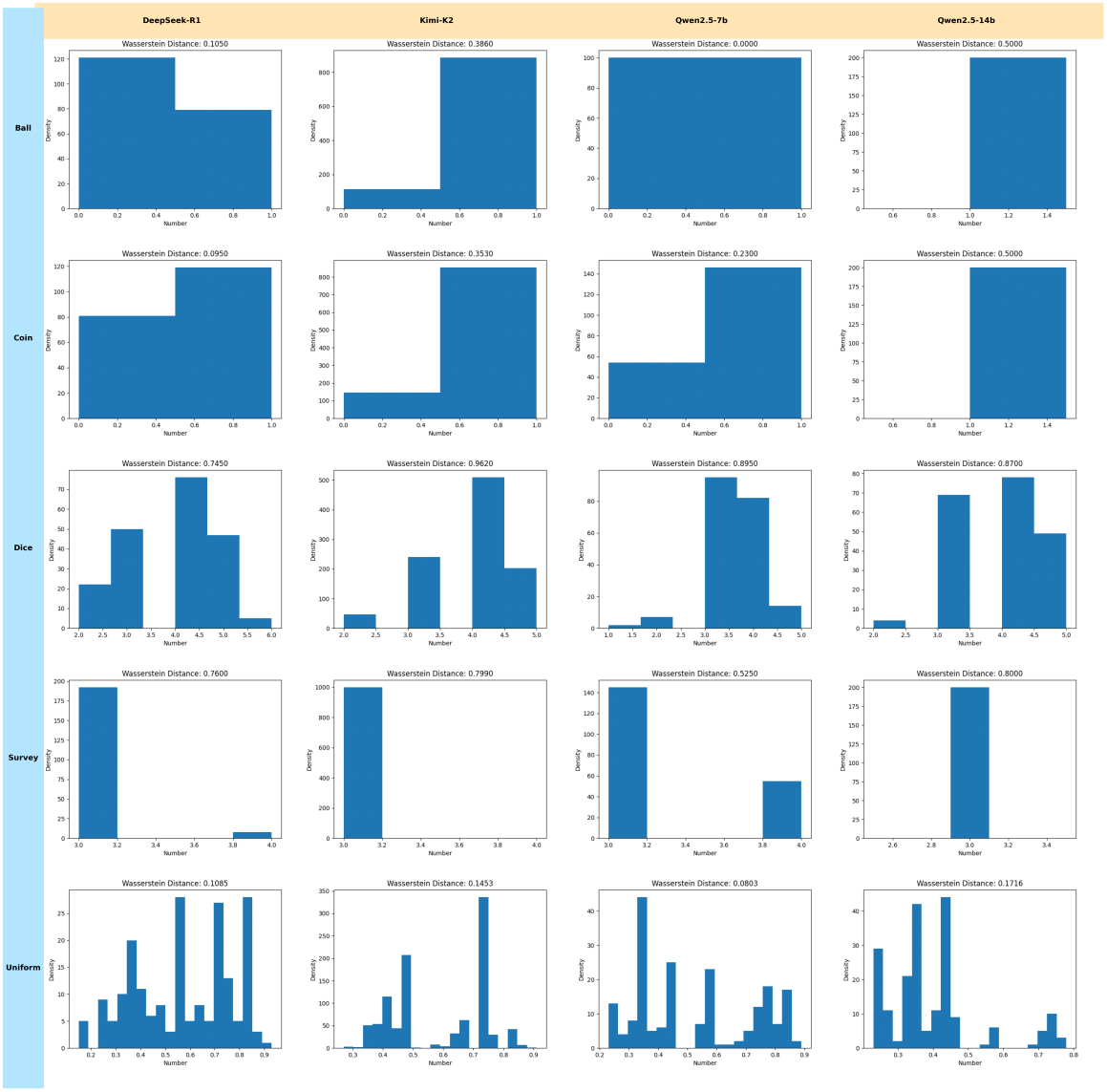


Figure 3: Overview of Response Distributions for Four LLMs on Five Benchmark Tasks

Table 6: Lauguage benchmarks.

Model	ArcEasy	openbookQA	wikitext2	wikitext103
	Acc	Acc	PPL	PPL
Qwen2.5-0.5B-Instruct	63.0%	40.5%	31.68	31.94
UnbiasedTuned-Qwen0.5B	64.5%	42.5%	31.57	31.83
Qwen2.5-7B-Instruct	96.5 %	82.5%	15.46	13.92
UnbiasedTuned-Qwen7B	97.0%	82.5%	15.23	13.74
Qwen2.5-14B-Instruct	99.0%	88.0%	10.33	8.44
UnbiasedTuned-Qwen14B	99.0%	88.0%	10.29	8.42

Table 7: Merged Tuned results A ($\times 10^{-2}$).

Model	A			A_no_sys			A_sim_sys		
	$W^{\overline{D}}\downarrow$	$\overline{W}\downarrow$	Var↓	$W^{\overline{D}}$	$\overline{W}$	Var	$W^{\overline{D}}$	$\overline{W}$	Var
MergedTuned-Qwen0.5B	3.83	7.24	0.44	8.86	36.8	5.73	22.90	25.40	2.21
MergedTuned-Qwen7B	1.79	5.91	0.35	24.13	39.82	10.40	4.65	8.03	0.65
MergedTuned-Qwen14B	2.26	5.79	0.43	1.66	13.19	4.41	2.31	5.91	0.30

Table 8: Merged Tuned results ABCD($\times 10^{-2}$).

Model	A			B			C			D		
	$W^{\overline{D}}\downarrow$	$\overline{W}\downarrow$	Var↓	$W^{\overline{D}}$	$\overline{W}$	Var	$W^{\overline{D}}$	$\overline{W}$	Var	$W^{\overline{D}}$	$\overline{W}$	Var
MergedTuned-Qwen0.5B	3.83	7.24	0.44	16.54	17.17	1.07	0.43	6.21	0.28	24.89	25.22	1.26
MergedTuned-Qwen7B	1.79	5.91	0.35	16.39	18.48	1.63	2.19	7.97	0.36	2.40	13.40	1.41
MergedTuned-Qwen14B	2.26	5.79	0.43	12.73	13.16	0.93	1.94	6.54	0.30	18.62	18.73	0.39