Enough Coin Flips Can Make LLMs Act Bayesian

Anonymous ACL submission

Abstract

Large language models (LLMs) exhibit the ability to generalize given few-shot examples in their input prompt, an emergent capability known as in-context learning (ICL). ICL allows models to generalize without explicit weight updates. Despite its empirical success, the underlying mechanism of ICL remains opaque. It is unclear whether LLMs perform structured reasoning akin to Bayesian inference or rely solely on pattern matching. In this work, we investigate the Bayesian nature of ICL in a controlled setting by analyzing LLMs' ability to model biased coin flips. Our findings reveal several key insights: (1) LLMs often possess biased priors, leading to initial divergence in zero-shot settings, (2) in-context evidence outweighs explicit bias instructions provided in a prompt, (3) when updating beliefs, they broadly adhere to Bayesian posterior updates, with deviations stemming from miscalibrated priors rather than incorrect updates, and (4) attention magnitude has little impact on Bayesian inference.

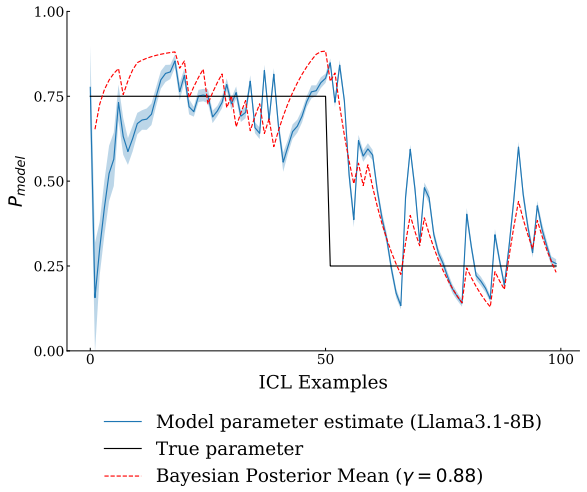

Figure 1: When we ask large language models (LLMs) to model sequences with in-context learning (ICL), how do they adapt their posterior probabilities given the provided examples? This figure explores how model probabilities change as we add new ICL examples in a biased coin-flipping experiment. The X-axis represents steps in the trajectory, while the Y-axis shows the predicted parameter of a Bernoulli distribution. Our results reveal that, while LLMs often have poorly calibrated priors, their updated parameter estimates broadly align with Bayesian behavior.

1 Introduction

Large language models (LLMs) designed for next-token prediction have gained significant popularity, largely because of their ability to generalize beyond language prediction and perform a wide range of novel tasks without requiring explicit weight updates (Brown et al., 2020). This emergent ability, referred to as *In-Context Learning (ICL)*, remains the backbone of modern test-time prompting techniques, including chain-of-thought prompting (Wei et al., 2022) and prompt chaining (Wu et al., 2022).

Despite significant empirical success, the underlying mechanisms of ICL are still unknown. Many aspects of how models adapt their own predictive distributions given a context and to what extent in-context learning aligns with principles of statistical inference remain unclear. In particular, it is unclear whether models are merely memorizing patterns from their training data or if they are performing structured reasoning akin to Bayesian inference.

A prominent explanation for ICL's behavior is based on Bayesian learning. Prior works have suggested that models can approximate Bayesian learning in certain scenarios by updating an implicit prior distribution over latent structures when provided with contextual information (Xie et al., 2021; Hahn and Goyal, 2023; Akyürek et al., 2022; Zhang et al., 2023; Panwar et al., 2023). However, these works explore the mechanism in environments where the true posterior distribution is unknown (such as question-answering or language modeling), or in restricted theoretical settings with known

posterior distributions but with constraints on model architecture or data type. Thus, the true degree to which pre-trained LLMs explicitly follow Bayesian update rules *at test time*, and whether their behavior aligns with normative probabilistic reasoning, remains an intriguing open question.

In this work, we strip away the complexity induced by traditional approaches to investigating ICL in LLMs and explore a simple, yet non-trivial, stochastic phenomenon: their ability to model the outcomes of biased coin flips. This simple setting provides a controlled environment where we can analyze whether pre-trained LLMs implicitly construct and update priors similarly to a Bayesian manner when presented with sequential observations of stochastic events. By examining how models estimate coin biases and incorporate new evidence, we can precisely characterize their convergence to Bayesian behavior, and explore several effects, including attention, scale, and the impact of instruction tuning, without inducing significant distributional complexity.

In this work we find several results, including (1) that language models often have biased priors for stochastic phenomena and that such priors lead to significant initial divergence in their ability to perform zero-shot modeling; (2) that LLMs often disregard explicit evidence in constructing their prior distributions, and are more responsive to in-context evidence-based updates; (3) that LLMs follow Bayesian update rules when considering new evidence, with much of the divergence between true and expected posteriors derived from the prior update; and (4) the magnitude of attention seems to have minimal influence on how evidence is incorporated into the posterior update process. Taken together, these results imply LLMs are capable of, and do, implicitly perform, Bayesian modeling in simple cases, and that in more complex environments, poor priors, rather than failures of updates due to in-context learning, may cause reduced performance.

2 Background & Related Work

Representing probabilities in language models. As LLMs have proliferated across a wide set of applications, many have examined whether LLMs can properly represent the concept of probability. Much of this examination has been through the lens of model calibration and alignment. Zhu and Griffiths (2024) show that LLMs are biased judges of probability much in the same fashion as human probability judgments. Gu et al. (2024) asks whether LLMs can play dice and finds that while LLMs know what probability is, they struggle to accurately sample from distributions. They attempt to solve this through tool use, but find that this is not a guaranteed solution to the problem. Meister et al. (2024) evaluates how well LLMs can align to human groups' distributions over a diverse set of opinions. They find that LLMs are good at describing biased distributions but are incapable of simulating these distributions.

In this work, we explore the ability of LLMs to simulate biased probability distributions and explore the mechanism of in-context learning as a natural method by which LLMs can align their priors to requested distributions.

In-context learning. Brown et al. (2020) introduces in-context learning (ICL) as a mechanism for few-shot generalization in language models. Although ICL usage has surged, users rarely employ it as a method to align models with target distributions. Further, issues with models' sensitivity to the positioning of tokens in their prompts have complicated the effective use of ICL as an alignment technique. Lu et al. (2022) demonstrates that the positioning of information within an ICL prompt affects model performance and devises a permutation-based approach to overcoming this bias. Liu et al. (2023) extends this analysis to highlight a persistent "lost-in-the-middle" effect, in which information in the middle of a prompt is down-weighted.

Our work shows that in-context rollouts of a probability distribution correlate well with the mean of a Bayesian posterior, and we further show that LLMs have a time-variant discount factor over the ICL prompt.

Bayesian updating in language models. Many authors have explored the mechanism by which ICL emerges in language models. Xie et al. (2021) finds that ICL can be viewed as a language model implicitly performing Bayesian inference—i.e., ICL emerges via modeling long-range coherence during pretraining. Jiang (2023) shows that emergent capabilities of LLMs, such as ICL, are Bayesian inference on the sparse joint distribution of languages. Wang et al. (2024) react to the ordering sensitivity of ICL prompts and pose ICL as a natural side effect of LLMs functioning as latent variable

models. Finally, Zhang et al. (2023) posit that ICL is an implicit form of Bayesian model averaging.

In this work, we confirm the ordering sensitivity of ICL prompts and further show empirically that ICL has several implicit Bayesian modeling behaviors, however demonstrate that it is unlikely that attention magnitude is a key component of the formalization.

3 Preliminaries

Bayesian systems: General Bayesian systems are expected to update their beliefs in a manner consistent with Bayes' rule. Given some evidence, D, a prior distribution $p(\theta)$ and a likelihood $p(D|\theta)$, the posterior distribution is obtained via:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{1}$$

where $p(D)$ is the marginal likelihood (or evidence) ensuring the posterior is properly normalized. While prior work (Falck et al., 2024) has explored additional assumptions (such as exchangeability), here we aim to explore the fundamental update process in a restricted environment.

Modeling coin-flips as Bayesian processes: In our setup, we model a biased coin by treating the probability of obtaining heads, denoted by θ, as a random variable with a binomial distribution. Suppose we perform n independent coin flips and observe k heads and $n-k$ tails. The likelihood of the observed data is given by:

$$p(D|\theta) = \theta^k(1-\theta)^{n-k} \tag{2}$$

A common choice for the prior distribution of θ is the Beta distribution due to its conjugacy with the binomial likelihood:

$$p(\theta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)} \tag{3}$$

where $B(\alpha,\beta)$ is the Beta function. By applying Bayes' theorem, the posterior distribution is thus proportional to the product of the likelihood and the prior:

$$p(\theta|D) \propto p(D|\theta)p(\theta) \tag{4}$$

$$\propto \theta^k(1-\theta)^{n-k} \cdot \theta^{\alpha-1}(1-\theta)^{\beta-1} \tag{5}$$

$$= \theta^{\alpha+k-1}(1-\theta)^{\beta+n-k-1} \tag{6}$$

And the posterior distribution for θ is also a Beta distribution:

$$\theta|D \sim \text{Beta}(\alpha+k, \beta+n-k). \tag{7}$$

It is often useful to consider the case where we have no strong prior beliefs about the coin's bias, leading us to adopt a uniform prior for θ. The uniform prior over the interval $[0,1]$ is a special case of the Beta distribution with parameters $\alpha=1$ and $\beta=1$, i.e., $p(\theta) = \text{Beta}(\theta;1,1) = 1$. When using the uniform prior, the posterior distribution becomes:

$$p(\theta|D) \propto \theta^k(1-\theta)^{n-k}, \tag{8}$$

This Bayesian framework allows us to update our beliefs about the coin's bias as more coin-flip data is collected, providing both a point estimate and a measure of uncertainty for θ.

Experimental design: We focus on open-source language models and extract stochastic representations directly from the underlying learned model distributions. Consider a sequence of tokens

$$x = \{x_1, x_2, ..., x_n\} \tag{9}$$

drawn from a vocabulary V (with $|V|$ elements). A large next-token prediction-based language model, $\mathcal{M}$, approximates a probability distribution over the next token:

$$p_{\mathcal{M}}(x_{i+1} \,|\, x_{1:i}) \tag{10}$$

where $x_{1:i} = \{x_1, x_2, ..., x_i\}$.

To evaluate stochastic processes, we define a fixed set of possible outcomes $\Omega = \{o_1, o_2, ..., o_k\}$, where each outcome $o \in \Omega$ is a sequence of tokens corresponding to a specific string value (e.g., when modeling a coin flip, the outcomes "heads" and "tails" might correspond to token sequences `[_heads]` and `[_tails]`, respectively). For each outcome o, we compute the probability given a prompt—analogous to updating our beliefs in a Bayesian framework—as follows:

$$p_{\mathcal{M}}(o \,|\, \text{prompt}) = \prod_{i=1}^{|o|} p_{\mathcal{M}}(o_i \,|\, o_{1:i-1}, \text{prompt}) \tag{11}$$

where $|o|$ denotes the number of tokens in o and $o_{1:i-1}$ represents the subsequence of tokens preceding the ith token in o.

Because these outcomes are a subset of all possible token sequences that $\mathcal{M}$ could generate, we renormalize the distribution over the support Ω. We denote the renormalized model distribution as $\hat{p}_{\mathcal{M}}(o)$ for $o \in \Omega$ (see subsection C.2 for further details on the renormalization process).

In our experiments, we measure the total variation distance (TVD) between the true posterior distribution $p^*(o)$ and the normalized model distribution $\hat{p}_{\mathcal{M}}(o)$ over the support Ω:

$$\delta(p^*, \hat{p}_{\mathcal{M}}) = \frac{1}{2} \sum_{o \in \Omega} |p^*(o) - \hat{p}_{\mathcal{M}}(o)| \qquad (12)$$

This distance metric quantifies the discrepancy between the two distributions—zero indicating perfect alignment and higher values indicating greater divergence.

We would like to clearly state that we are not claiming that LLMs themselves are explicitly Bayesian, rather, we ask the question: *do model predictive distributions have Bayesian behavior?* In this paper we treat models themselves as point-wise estimators of distributional parameters (in our case, we use them to estimate the parameters of a binomial distribution), and ask if those point-wise estimates align with reasonable Bayesian frameworks.

We evaluate several models, including Gemma-2 (Team et al., 2024), Phi-2/Phi-3.5 (mini) (Abdin et al., 2024), Llama-3.1 (8B) (Dubey et al., 2024), Mistral 7B (Jiang et al., 2023), and OLMoE (7B) (Muennighoff et al., 2024), along with their instruction-tuned variants. For scaling experiments, we leverage the Pythia Scaling Suite (Biderman et al., 2023) For more details regarding these models, please refer to Appendix D.

4 Understanding the LLM Prior

Due to data-intensive pre-training, language models inherently encode a prior over θ (the likelihood of heads in the coin-flip). We are interested in understanding these priors and understanding how to update the priors via explicit prompting.

To extract a prior over heads and tails, we query the models for a coin flip through 50 different prompt variants (e.g. "`I flipped a coin and it landed on`"), and compute the normalized logit value ascribed to heads (discussed in detail in Appendix C). As shown in Figure 2, all language models evaluated begin with fundamental priors

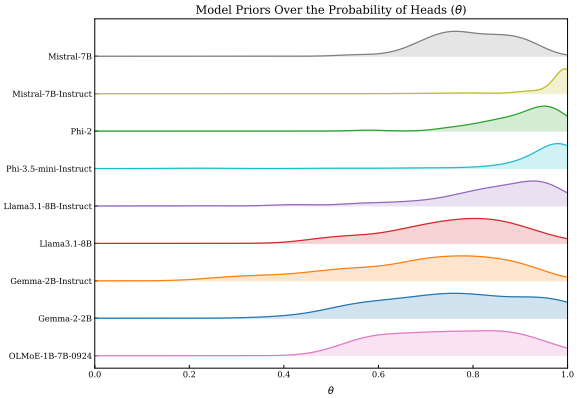

Figure 2: **Model priors:** All language models evaluated present a bias towards heads.

for θ that are heads-biased, and in some cases, significantly so. This observation is reflected in the tokenization structure itself; in some cases, models do not see sufficient data to assign a full token to `[_tails]` and instead encode this in a pair of tokens (which we handle when computing probability, see Appendix C). Thus, models begin divergent from an unbiased estimate of coin priors.

Effect of explicit biasing via prompting. Next, we explore if we can encourage models to update their priors by providing an explicit value for θ in the prompt. We define a set of biasing statements, i.e. describing unfair coins, of the form "`When I flip coins, they land on heads X% of the time.`", and run a set of trials, evaluating the TVD between models' probabilities over outcomes and the expected distribution for the biased θ.

Results from this experiment are presented in Figure 3. Given an explicit bias in the input prompt, non-instruct LLMs fail to converge to the expected biased distribution with their token probabilities following their originally computed prior—generally showing a tendency to ascribe $\approx$ 60%-80% probability to heads, independent of explicit context. Instruct models performed slightly better, though they still exhibited a bias toward heads. Additionally, instruct models showed improved performance at the extremes of bias values, with TVD values dropping for 0% and 100% heads biases (matching observations from Zhao et al. (2021)).

Effect of model size on priors. Scaling the language model size has shown effectiveness in many tasks. Therefore, we explore whether scaling also boosts performance on modeling expected biased distribution. We use Pythia Scaling Suite (Biderman

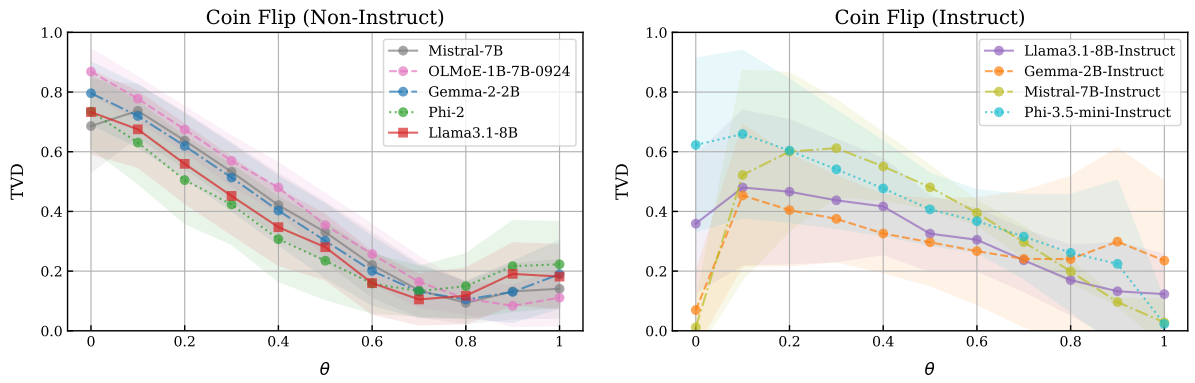

Figure 3: **Biased coins:** Plots of mean total variation distance (TVD, ↓) against bias (θ) for non-instruct (left) and instruct (right) models when aggregated across prompts (N=50) for the biased coin flip experiment. Shaded areas show one standard deviation. While non-instruct models both (1) ignore biasing instructions in the prompts and (2) almost always generate a biased distribution ($\approx 70\%$ heads), instruct-based models pay better attention to biasing information, and perform significantly better when modeling extreme bias (always generating heads/tails).

et al., 2023) that covers model size ranging from 70M to 12B and test on different biased θ. Results from this experiment are presented in Figure 4. For a given bias, scaling the model size does not substantially change the language models' priors or improve the performance of modeling expected distributions. However, the relative ordering among different biases does shift as the model size increases.

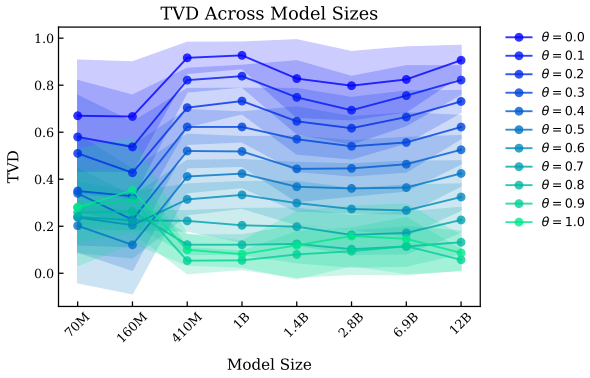

Figure 4: **Biased coins and parameter scaling:** Mean total variation distance (TVD, ↓) vs. model size for different bias percentages. We use the models from the Pythia Scaling Suite. As the size of the model increases, the performance does not change for a certain bias. The relative ordering among different biases does shift as the model size increases

5 Does In-Context Learning Improve Parameter Estimates?

Next, we are interested in understanding if and how LLMs incorporate in-context evidence into their posteriors. Specifically, rather than explicitly describing the underlying distribution as before, we implicitly specify it by providing the LLM with a sequence of samples from that distribution in its prompt (e.g., "`I flipped a coin and it landed on heads, then on tails, then on tails, then on tails, then on...`" for a coin biased toward tails). We then assess the expected distribution of the coin flip outcomes under each model after presenting these ICL prompts.

Figure 5, shows results from the coin flip experiment on Llama-3.1-8B and Llama-3.1-8B-Instruct (see Appendix E for results from other models). We find that models converge to the expected distribution as more evidence is provided via in-context learning.

5.1 Effect of model scale

We investigate if larger models are better able to incorporate in-context-based evidence. *Chinchilla*-scaling Hoffmann et al. (2022) would suggest that larger models would also have more powerful emergent behaviors such as ICL.

In Figure 6, we show the results of running the ICL experiments on the Pythia Suite for $\theta = 0.20$ (See subsection E.2 for all settings of θ). Although ICL performance generally improves as the number of examples grows, we find that model scale has negligible impact on order dynamics, with models performing comparably across scales. Surprisingly, however, larger models appear worse at incorporating model updates on the whole, with most TVD values higher for the 12B model compared to their respective smaller models.

5.2 Do models perform pure Bayesian updates?

To explore if models actually perform Bayesian updates during a single trial, we look directly at

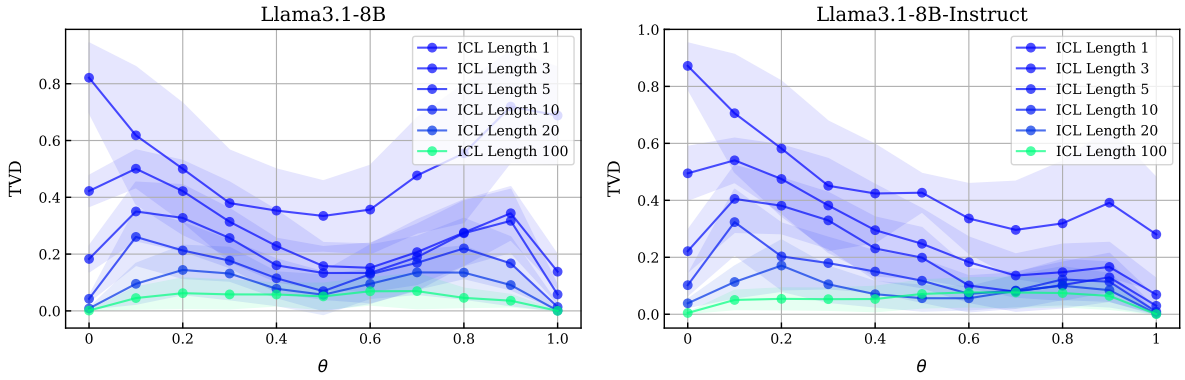

Figure 5: **Biased coins and ICL:** Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths for Llama3.1-8B model (left) and Llama3.1-8B-Instruct (right). As the number of in-context samples increases, the performance of the models at modeling the stochastic process improves as well. Notably, adding as few as 3 in-context examples significantly improves performance, but even adding 100 in-context examples does not fully allow the model to capture the biased distribution. For other models, see Appendix E.

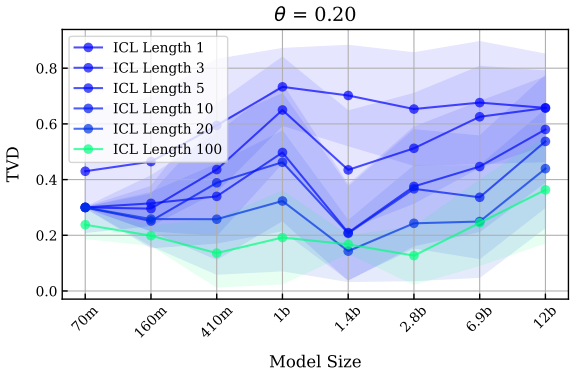

Figure 6: **ICL and parameter scaling:** Mean total variation distance (TVD, ↓) vs. model size across the Pythia Scaling Suite family with a biasing statement for $\theta = 0.20$. Model size does not have a clear impact on the benefits from ICL.

several "online" ICL trajectories. To generate these trajectories, instead of drawing trajectories entirely from a single distribution, we instead model a generative process containing 100 steps, where the first 50 samples are drawn $\sim Bernoulli(\theta_1)$ and the second 50 samples are drawn $\sim Bernoulli(\theta_2)$, where $\theta_1 = 0.75$ and $\theta_2 = 0.25$. This trajectory, shown in Figure 1 (the black line), gives a moving target which evolves over time for the model to approximate. In this dynamic environment, we then explore how well the LLM's pointwise estimates are modeled by a Bayesian update process.

To define this Bayesian update process, we first note that classical Bayesian filtering updates a Beta prior $\text{Beta}(\alpha, \beta)$ with each observation, treating all data equally. Given a prior and a binomial likelihood, the posterior is also Beta-distributed:

$$p(\theta|D) = \text{Beta}(\alpha+k, \beta+n-k), \qquad (13)$$

where k is the number of heads observed in n coin flips.

In dynamic environments, on the other hand, recent data may be more relevant. To model this, we can introduce an exponential decay factor γ, modifying the updates to:

$$\alpha \leftarrow \gamma\alpha + I(H), \quad \beta \leftarrow \gamma\beta + I(T) \qquad (14)$$

where $I(H)$ and $I(T)$ indicate the latest result. This ensures older observations gradually contribute less, allowing the model to adapt. The posterior mean remains:

$$\mathbb{E}[p] = \frac{\alpha}{\alpha+\beta} \qquad (15)$$

This decay ensures older data contributes less, allowing adaptation to shifts in θ. For $\gamma = 1.0$, this remains the classical Bayesian filtering update.

Returning to our environment, Figure 7 shows a single example roll-out of both classical and the gamma-modified Bayesian filter, along with the associated model probabilities. We can see that while the general shape of the trajectory fits the model behavior, pure Bayesian filtering (i.e. $\gamma = 1.0$) alone does not explain the behavior of the model. Instead, using a $\gamma < 1$, implying a shortened time horizon, fits the behavior almost perfectly in some cases, empirically suggesting that models are performing local Bayesian updates with a slight discount factor.

Extending this idea, we leverage L-BFGS-B Zhu et al. (1997) to fit a γ value to each model, with the results shown in Table 1. We can see in this table that the value of γ is notably different for each model, suggesting that models have architecture-specific

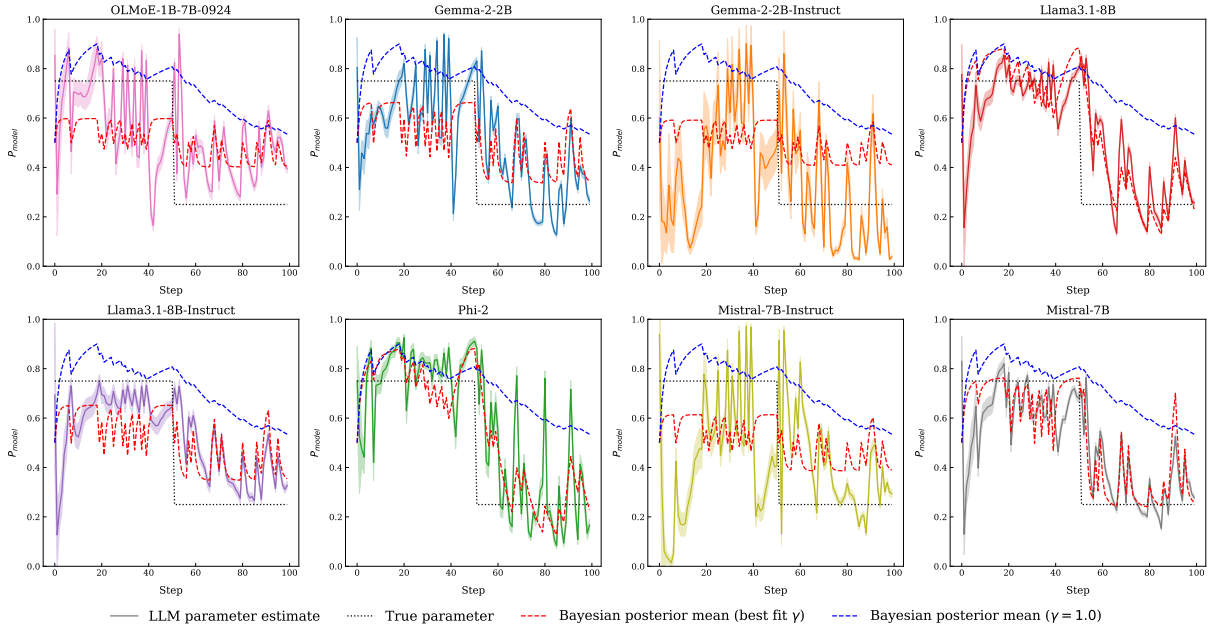

Figure 7: **Posterior evolution during Bayesian filtering:** The figure shows a single rollout of classical Bayesian filtering alongside model predictive probabilities in a 100-sample coin flip ICL task. While the overall shape of the model's predictions aligns with Bayesian updates, the direct application of standard Bayesian filtering ($\gamma = 1.0$) does not fully explain the observed behavior. Instead, the empirical fit suggests that models implicitly apply a localized Bayesian update with a shorter time horizon, aligning better with a slightly discounted filtering process.

Table 1: Bayesian filtering best fit γ value.

Model	**Best-Fit γ**
OLMoE-1B-7B-0924	0.3268
Gemma-2-2B	0.4910
Gemma-2-2B-Instruct	0.3087
Llama3.1-8B	0.8807
Llama3.1-8B-Instruct	0.4655
Phi-2	0.8781
Mistral-7B	0.6903
Mistral-7B-Instruct	0.9107

time-horizon behavior. Interestingly, instruction-tuned models generally have much lower γ values than their non-instruction-tuned counterparts. This implies that these models may be more local when performing ICL and are more willing to switch behaviors when prompted with new ICL evidence.

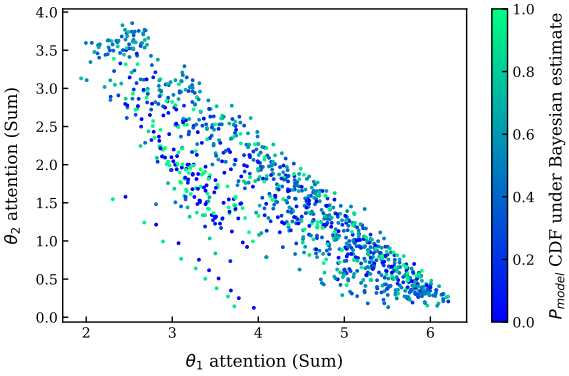

Figure 8: Relationship between total attention and model point-estimate extremity under the Bayesian posterior ($\gamma = 1.0$). Overall, the extremity of the model point estimate under the Bayesian model appears uncorrelated with the attention.

5.3 Does attention impact updates?

Some prior work, such as Zhang et al. (2023), suggests that attention helps to weight the Bayesian update. In this section, we aim to leverage our simplified setup to empirically understand the impact that attention has on the convergence behavior of the model. We use the same setup as subsection 5.2 but instead draw the center K samples $\sim Binom(K,\theta_1)$ and the outer $M = 100 - K$ samples $\sim Binom(M,\theta_2)$.

Figure 8 plots the relationship between total attention and model point-estimate extremity under the Bayesian posterior ($\gamma = 1.0$) (i.e. the value of the CDF of the true posterior at the model point estimate). We can see that the amount of attention paid to any segment is generally uncorrelated with the overall quality of the point estimate (θ_1 : $(R = 0.02, p = 0.48)$, θ_2 : $(R = -0.03, p = 0.36)$), suggesting that the total magnitude of the attention paid to each segment does not dramatically impact model quality.

In addition, the fraction of attention has a similar lack of correlation, as shown in Figure 9, which

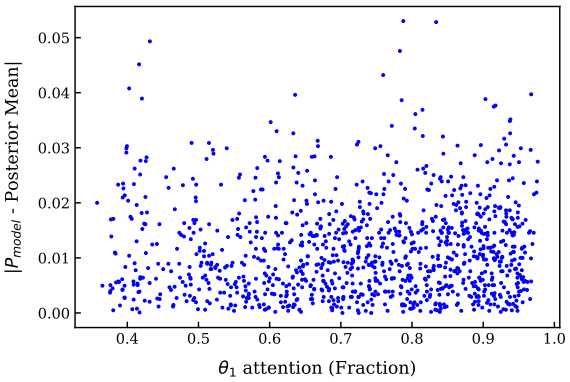

Figure 9: Fraction of attention assigned to samples from θ_1 versus the deviation between the model-predicted distribution and the true posterior mean for LLaMA-3.1-8B. The findings suggest that the relative attention paid to in-context examples does not directly predict the model's update performance.

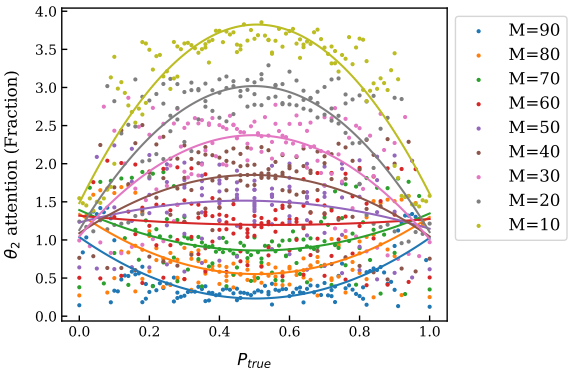

Figure 10: The fraction of attention on samples from θ_2 vs. the true posterior distribution of the mixture for different values of M for LLama-3.1-8B. Lines represent the degree-2 line of best fit. When M is low, the model primarily attends to θ_2 when it aligns with θ_1. As M increases, the model pays more attention to θ_2 when it significantly influences the final distribution.

suggests that paying any special attention (in terms of magnitude) to any particular ICL example is uncorrelated with downstream performance during model updates.

Interestingly, the an important indicator of attention is the (non-estimated) true parameter value. We can see in Figure 10 that when M is low (i.e. few samples are drawn from θ_2, the model only pays attention to θ_2 when it matches the θ_1 distribution. When M is high, the model pays attention more to samples from θ_2 when θ_2 is more likely to bias the distribution. These observations support a nuanced view of model attention: models pay relatively more attention to data which is more likely to lead to changes in the final distribution, but higher/lower attention is somewhat uncorrelated with final model quality.

6 Discussion & Conclusion

In this work, we investigate the ability of large language models (LLMs) to model simple stochastic processes, specifically coin flips, through in-context learning (ICL). By stripping away complexities inherent in previous ICL studies, we provide a controlled setting to analyze how LLMs implicitly construct and update priors. Our findings reveal that while LLMs often begin with priors that reflect head-biased coins, they approximate Bayesian updates when incorporating new evidence, suggesting that limitations in stochastic modeling stem primarily from flawed priors rather than failures in in-context adaptation.

Correctly modeling stochastic behavior in LLMs has far-reaching implications. Recent work has explored language models as "world models," assuming they can accurately simulate probability distributions and stochastic processes. This paradigm has gained traction across diverse fields, including robotics simulations (Dagan et al., 2023; Song et al., 2024; Zhao et al., 2024), human behavior modeling (Aher et al., 2023; Park et al., 2023; Moon et al., 2024; Axtell and Farmer, 2022; Argyle et al., 2023; Loyall, 1997), and scientific reasoning (Shojaee et al., 2024), among others (Ge et al., 2024; Yang et al., 2024; Nottingham et al., 2023; Xie et al., 2024). However, here, we demonstrate that, out of the box, LLMs fail to correctly simulate even simple stochastic processes such as coin flips. Instead, their alignment with true probabilistic reasoning only emerges as they incorporate increasing amounts of in-context evidence. We further show that as the underlying distributions dynamically transition, the model's predictions do so in a correspondingly Bayesian way, albeit with notable time-discounting effects in how evidence is weighted.

Overall, our work highlights both the limitations and emergent strengths of LLMs in probabilistic modeling. While their initial priors are poorly calibrated, ICL enables them to approximate Bayesian reasoning, providing a pathway toward improving their ability to simulate stochastic environments. By grounding these behaviors in simple and explainable settings, we take a step toward refining the LLM-as-world-model framework, ensuring more reliable and interpretable performance in complex domains.

7 Limitations

While this paper provides insight into how LLMs approximate Bayesian inference in stochastic modeling, our approach has certain limitations that highlight both methodological constraints and fundamental challenges in treating LLMs as Bayesian reasoners.

One key limitation is that our evaluation method captures only a restricted slice of the full posterior distribution. In Bayesian inference, the posterior should account for the entire probability space, but our approach only evaluates the model's explicit token probabilities for a predefined set of completions. For example, if the expected response is "The coin came up 'heads'", the model might alternatively generate "The coin landed on the edge of heads" or "The coin was slightly tilted toward heads". While we verify that these are low-probability outcomes in our experiments, they still represent probability mass that is not incorporated into our evaluation. If LLMs allocate significant probability to such alternatives, our benchmark may misrepresent their ability to perform Bayesian updates accurately.

Furthermore, while our experiments assess LLM performance in simple Bayesian updating tasks, they do not fully capture the complexities of real-world probabilistic reasoning. Bayesian inference in natural settings often requires reasoning over continuous distributions, hierarchical priors, or distributions with long tails. Our analysis focuses on discrete, categorical predictions, which may not generalize well to more complex probabilistic environments where likelihoods are less structured or where prior distributions must be inferred over high-dimensional latent spaces.

Another methodological limitation arises in evaluating closed-source models. Since our approach relies on extracting logits to approximate posterior distributions, it cannot be directly applied to black-box models such as GPT-4 or Claude, where such internals are inaccessible. While an alternative approach using sampling via API calls could approximate the posterior, this method is costly and susceptible to distortions from API-side interventions such as caching, response smoothing, or temperature adjustments. These factors could introduce artifacts that obscure the model's true Bayesian reasoning capabilities.

Beyond these methodological constraints, there are deeper concerns about the limitations of LLMs as Bayesian agents. A fundamental challenge in Bayesian modeling is the specification of a well-calibrated prior. Our findings suggest that LLMs often exhibit poorly calibrated priors when performing in-context learning, which can lead to systematic misestimation in early predictions. While the models do update their beliefs in a manner consistent with Bayesian inference, an inaccurate prior can cause significant initial divergence from the true posterior. This misalignment is particularly concerning in high-stakes applications such as financial forecasting, scientific modeling, and decision-making systems, where incorrect priors can propagate errors through downstream reasoning.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Gati V Aher, Rosa I Arriaga, and Adam Tauman Kalai. 2023. Using large language models to simulate multiple humans and replicate human subject studies. In *International Conference on Machine Learning*, pages 337–371. PMLR.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2022. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*.

Lisa P Argyle, Ethan C Busby, Nancy Fulda, Joshua R Gubler, Christopher Rytting, and David Wingate. 2023. Out of one, many: Using language models to simulate human samples. *Political Analysis*, 31(3):337–351.

Robert L Axtell and J Doyne Farmer. 2022. Agent-based modeling in economics and finance: Past, present, and future. *Journal of Economic Literature*, pages 1–101.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric

Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Gautier Dagan, Frank Keller, and Alex Lascarides. 2023. Dynamic planning with a llm. *ArXiv preprint*, abs/2308.06391.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Fabian Falck, Ziyu Wang, and Chris Holmes. 2024. Is in-context learning in large language models bayesian? a martingale perspective. *arXiv preprint arXiv:2406.00793*.

Zhiqi Ge, Hongzhe Huang, Mingze Zhou, Juncheng Li, Guoming Wang, Siliang Tang, and Yueting Zhuang. 2024. Worldgpt: Empowering llm as multimodal world model. *ArXiv preprint*, abs/2404.18202.

Jia Gu, Liang Pang, Huawei Shen, and Xueqi Cheng. 2024. Do LLMs Play Dice? Exploring Probability Distribution Sampling in Large Language Models for Behavioral Simulation. *Preprint*, arXiv:2404.09043.

Michael Hahn and Navin Goyal. 2023. A theory of emergent in-context learning as implicit structure induction. *arXiv preprint arXiv:2303.07971*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. *Preprint*, arXiv:2203.15556.

Aspen K Hopkins, Alex Renda, and Michael Carbin. 2023. Can llms generate random numbers? evaluating llm sampling in controlled domains. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Hui Jiang. 2023. A Latent Space Theory for Emergent Abilities in Large Language Models. *Preprint*, arXiv:2304.09960.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. *Preprint*, arXiv:2307.03172.

Toni JB Liu, Nicolas Boullé, Raphaël Sarfati, and Christopher J Earls. 2024. Llms learn governing principles of dynamical systems, revealing an in-context neural scaling law. *ArXiv preprint*, abs/2402.00795.

Aaron Bryan Loyall. 1997. Believable agents: building interactive personalities.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Nicole Meister, Carlos Guestrin, and Tatsunori Hashimoto. 2024. Benchmarking Distributional Alignment of Large Language Models. *Preprint*, arXiv:2411.05403.

Suhong Moon, Marwa Abdulhai, Minwoo Kang, Joseph Suh, Widyadewi Soedarmadji, Eran Kohen Behar, and David M Chan. 2024. Virtual personas for language models via an anthology of backstories. *ArXiv preprint*, abs/2407.06576.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. 2024. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. 2023. Do embodied agents dream of pixelated sheep: Embodied decision making using language guided world modelling. In *International Conference on Machine Learning*, pages 26311–26325. PMLR.

Madhur Panwar, Kabir Ahuja, and Navin Goyal. 2023. In-context learning through the bayesian prism. *arXiv preprint arXiv:2306.04891*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy. 2024. Llm-sr: Scientific equation discovery via programming with large language models. *ArXiv preprint*, abs/2404.18400.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for llm agents. *ArXiv preprint*, abs/2403.02502.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Katherine Van Koevering and Jon Kleinberg. 2024. How random is random? evaluating the randomness and humaness of llms' coin flips. *ArXiv preprint*, abs/2406.00092.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. 2024. Large Language Models Are Latent Variable Models: Explaining and Finding Good Demonstrations for In-Context Learning. *Preprint*, arXiv:2301.11916.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *Proceedings of the 2022 CHI conference on human factors in computing systems*, pages 1–22.

Junlin Xie, Zhihong Chen, Ruifei Zhang, Xiang Wan, and Guanbin Li. 2024. Large multimodal agents: A survey. *ArXiv preprint*, abs/2402.15116.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Yijun Yang, Tianyi Zhou, Kanxue Li, Dapeng Tao, Lusong Li, Li Shen, Xiaodong He, Jing Jiang, and Yuhui Shi. 2024. Embodied multi-modal agent trained by an llm from a parallel textworld. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26275–26285.

Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. 2023. What and how does in-context learning learn? bayesian model averaging, parameterization, and generalization. *arXiv preprint arXiv:2305.19420*.

Guosheng Zhao, Xiaofeng Wang, Zheng Zhu, Xinze Chen, Guan Huang, Xiaoyi Bao, and Xingang Wang. 2024. Drivedreamer-2: Llm-enhanced world models for diverse driving video generation. *ArXiv preprint*, abs/2403.06845.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12697–12706. PMLR.

Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560.

Jian-Qiao Zhu and Thomas L. Griffiths. 2024. Incoherent Probability Judgments in Large Language Models. *arXiv*.

Appendix

The appendix consists of the following further discussion:

- Appendix A discusses the data used and created in this paper, and the licenses and usage.
- Appendix B discusses the use of artificial intelligence in the creation of this manuscript.
- Appendix C explains the methodologies including distribution normalization and comparisons with prior work.
- Appendix D details the models used in this study, their specifications, and training sources.
- Appendix E presents additional prior results for the coin flipping experiments.
- Appendix F explores similar results to section 4 and section 5 but with dice rolling (as opposed to coin flips).

A Data Usage

This paper relies on several model artifacts including:

- Gemma-2 (Team et al., 2024) released under the Gemma license.
- Llama3.1 (Dubey et al., 2024) released under the Llama 3 Community License Agreement.
- Phi-3.5 and Phi-3 (Abdin et al., 2024) released under the MIT license.
- Mistral 7B (Jiang et al., 2023) released under the Apache 2.0 license.
- Olmo 7B (Muennighoff et al., 2024) released under the Apache 2.0 license.
- Pythia Scaling Suite (Biderman et al., 2023) released under the Apache 2.0 license.

Our usage of the models is consistent with the above license terms. Our code for computing the analyses in this paper will be released under the MIT license.

B Use of Artificial Intelligence

This paper includes contributions generated with the assistance of AI tools. Specifically, AI assistants including ChatGPT were used for sentence/paragraph-level editing of the content, the creation of LaTeX tables and figures from raw data sources, and as a coding assistant through GitHub Copilot. All intellectual and creative decisions, including the final content and conclusions, remain the responsibility of the authors. The use of AI in this process was supervised to ensure accuracy and alignment with the intended research outcomes.

C Methods

C.1 Preliminaries

We focus on open-source language models, and extract stochastic representations directly from the underlying learned model distributions. For a sequence of tokens, $x = \{x_1, x_2, ...x_n\}$ in a vocabulary V (of size $|V|$), a large next-token prediction-based language model, $\mathcal{M}$, approximates a probability distribution over the next token: $P_{\mathcal{M}}(x_{i+1}|x_i,...,x_1)$.

To evaluate stochastic processes, for each process we define a fixed set of possible "outcomes" that a sample from the process can take. Formally, each outcome $o \in \Omega = \{o_1 ... o_k\}$ is a sequence of tokens corresponding to a string value (for example, when flipping an coin, the outcomes are "heads" and "tails", corresponding to token sequences `[_heads]` and `[_t,ails]`). For each outcome, we then aim to compute $P_{\mathcal{M}}(o|\text{prompt})$, where the prompt is a sequence of tokens that both (1) describes the process and (2) asks for a sample. While several works estimate this probability by sampling (Hopkins et al., 2023; Van Koevering and Kleinberg, 2024), we found that sampling was often unreliable, and thus, we extract this distribution directly from the language model as:

$$P_{\mathcal{M}}(o|\text{prompt}) = \prod_{i=1}^{k} P_{\mathcal{M}}(o_i|o_{i-1},...,o_1,\text{prompt}) \tag{C.1}$$

Note here that for multi-token sequences, we compute the probability conditioned on picking the correct token, and we assume that there is only one unique generator for the sequence o. Because these outcomes are a subset of all of the potential token sequences generated by the LLM, we re-normalize the distribution over the support of the options.

See subsection C.2 for more details about the re-normalization process.

In this paper, we primarily measure the total variation distance (TVD) between the true distribution $P^*(o)$ and the normalized model distribution $\hat{P}_{\mathcal{M}}(o)$ over the support Ω:

$$\delta(P^*, \hat{P}_{\mathcal{M}}) = \frac{1}{2} \sum_{\omega \in \Omega} \left| P^*(\omega) - \hat{P}_{\mathcal{M}}(\omega) \right| \quad \text{(C.2)}$$

The TVD is an intuitive distance measure, which arises as the optimal transport cost between the distributions given a unit cost function. When the TVD is high, the distributions are quite different, and when it is zero, the distributions are identical.

In this paper, we explore the performance of several models including Gemma-2 (Team et al., 2024), Phi-2/Phi-3.5 (mini) (Abdin et al., 2024), Llama-3.1 (8B) (Dubey et al., 2024), Mistral 7B (Jiang et al., 2023) and OLMoE (7B) (Muennighoff et al., 2024) along with their instruction-tuned variants. For more details on the models, see Appendix D.

C.2 Distribution Normalization

Because the set of outcomes Ω is only a small part of the possible sequences that the LLM can generate, it is often necessary to re-normalize the probability distribution against the support Ω, instead of the full vocabulary space V. There are many options that could be picked for re-normalization. In our experiments, we choose to use a linear re-normalization:

$$\hat{P}_{\mathcal{M}}(o) = \frac{P_{\mathcal{M}}(o|\text{prompt})}{\sum_{\omega \in \Omega} P_{\mathcal{M}}(\omega|\text{prompt})} \quad \text{(C.3)}$$

This is in contrast to prior work (Liu et al., 2024), who normalize using a softmax distribution:

$$\hat{P}_{\mathcal{M}}(o) = \frac{exp(P_{\mathcal{M}}(o|\text{prompt}))}{\sum_{\omega \in \Omega} exp(P_{\mathcal{M}}(\omega|\text{prompt}))} \quad \text{(C.4)}$$

Unfortunately, in the limit of small probabilities, for $p_i, 1 < i < |\Omega|$, as $p_i \to 0$:

$$\lim_{p_i \to 0, p_j \to 0} \frac{e^{p_i}}{\sum_j e^{p_j}} = \frac{1}{\sum_j e^{p_j}} \approx \frac{1}{|\Omega|} \quad \text{(C.5)}$$

This can significantly impact the computation of downstream measures. Normalizing linearly avoids this issue, but can sometimes cause numeric instability.

C.3 Instruct Models Chat Templates

In order to make instruction-tuned models compatible with our formulation for extracting token probabilities, we employ chat templates in the following manner. First, we construct the chat as follows:

```
chat = {'user': instruct_prompt
, 'assistant': input_prompt}
```

Here `instruct_prompt` explicitly provides a directive to the LLM such as `"Please toss a coin and tell me whether it landed on heads or tails."`. The `input_prompt` field contains the portion to be completed by the LLM spoken in the first person, such as `"I tossed a coin and it landed on"`. Next, in order to properly extract token probabilities pertaining to the support of the random process at the end of the input sequence, we modify models' chat templates to omit the EOT tokens.

C.4 Prompt Normalization

The specific prompt itself may have an impact on the outcomes of the experiment, given the myriad of ways that a coin flip can be specified. To help alleviate this variance, in all experiments in the paper we normalize across a large set of 50 potential result prompts, shown in Figure C.1, and 5 instruct prompts, shown in Figure C.2.

D Models

google/gemma-2-2b : Gemma-2 (Team et al., 2024) is a 2-billion parameter model from Google DeepMind, built on a decoder-only Transformer architecture. It uses a model dimension of 2304, 26 layers, 8 heads with Grouped-Query Attention (GQA), and a feedforward dimension of 18,432. The model features 256,128 vocabulary entries using a SentencePiece tokenizer optimized for multiple languages. Trained on 2 trillion tokens from sources like web documents, code, and scientific articles, it incorporates local sliding window attention (4096 tokens) and global attention (8192 tokens). It also uses knowledge distillation from larger models, which enables it to achieve strong results with more efficient training.

google/gemma-2-2b-it: Gemma-2-IT (Team et al., 2024) is the instruction-tuned version of Gemma-2, featuring the same architecture as Gemma-2-2B. The model has been fine-tuned on supervised datasets and further improved using

```
I just flipped a coin, and it came up
The result of my coin flip is
After flipping the coin, it landed on
I flipped a coin, and the outcome was
The coin flip resulted in
After tossing the coin, it ended up on
I tossed a coin, and it fell on
The coin I flipped landed on
The coin I tossed resulted in
Following the coin flip, it showed
The coin fell on
The flip of the coin resulted in
I flipped the coin, and it settled on
The result after flipping the coin is
The outcome of my coin flip is
I tossed the coin, and the outcome is
The result of my coin toss is
I flipped the coin, and it came up
The coin came to rest on
After flipping, the coin showed
The toss of the coin revealed
I flipped the coin, and it turned up
The coin toss ended with
After tossing the coin, it showed
The coin flipped over to
After flipping, the coin settled on
My coin toss resulted in
The outcome
     of my coin flip turned out to be
I flipped
    the coin, and its final position was
The coin fell, showing
I tossed the coin, and it landed showing
Following the toss, the coin showed
The flip resulted in the coin landing on
The coin toss revealed
The outcome of the coin landing is
After tossing, the coin landed on
I flipped the coin and saw it land on
After the flip, the coin showed
The result of tossing the coin was
When I flipped the coin, it landed on
The coin
     showed this side after the flip:
The flip of the coin ended with
After tossing, the coin fell to show
The result of my toss came out as
The toss of the coin came to rest on
The coin after the flip landed on
I flipped the coin, and it ended on
The result
     of the coin toss ended up being
I flipped a coin, and its final side was
The coin flip showed the result:

```

Figure C.1: A list of possible prompts describing a coin flip result.

RLHF (Reinforcement Learning from Human Feedback) for better instruction-following capabilities. It uses the same 256,128-entry vocabulary and was trained on similar data sources. Gemma-2-IT includes additional tuning to enhance safety and reduce hallucinations.

```
Please complete this sentence: I just
     flipped a coin, and it landed on
Finish this sentence
    : The result of my coin flip is
Complete the sentence: After
     flipping the coin, it landed on
Fill in the rest: I
     flipped a coin, and the outcome was
Complete the
     phrase: The coin flip resulted in

```

Figure C.2: A list of possible instruct prompts describing a coin flip result.

meta-llama/llama-3.1-8B: Llama-3 (Dubey et al., 2024) is a foundation model developed by Meta, built with an 8 billion parameter dense Transformer architecture. The model has 32 layers, a model dimension of 4096, a feedforward dimension of 14,336, and 32 attention heads. It supports multilingual tasks, coding, and reasoning with a context window of 8K tokens. Llama-3 was pre-trained on a dataset of 15 trillion tokens, spanning a variety of sources such as web documents, code, and multilingual texts, with a vocabulary size of 128,000 tokens using a tokenizer optimized for multilingual use.

meta-llama/llama-3.1-8B-Instruct: Llama-3-Instruct (Dubey et al., 2024) is the instruction-tuned variant of Llama-3, also comprising 8 billion parameters, 32 layers, 4096 model dimensions, and a feedforward dimension of 14,336. This version is fine-tuned to follow human instructions better, leveraging supervised fine-tuning and Direct Preference Optimization (DPO). It is designed for tasks requiring precise instruction following, including coding, reasoning, and complex dialogue, while supporting tools like code generation and multilingual text processing. It also includes additional tuning to enhance safety and reduce hallucinations.

microsoft/phi-3.5-mini-instruct: Phi-3 (Abdin et al., 2024) is a 3.8-billion parameter Transformer model designed by Microsoft, optimized for both small-scale deployment and high-performance tasks. The model has 32 layers, 3072 hidden dimensions, 32 attention heads, and a default context length of 4K tokens, extendable to 128K using LongRope. It was trained on 3.3 trillion tokens, with a dataset comprising heavily filtered publicly available web data and synthetic data. Its instruction-following capability is enhanced through supervised fine-tuning and Reinforcement Learning from Human Feedback (RLHF)

microsoft/phi-2: Phi-2 (Abdin et al., 2024) is a 2.7-billion parameter model, part of Microsoft's Phi series, designed for efficient performance in smaller-scale models. Like Phi-3, it uses a transformer-based decoder architecture with Grouped-Query Attention (GQA) and a vocabulary size of 320641 tokens and is trained on a mixture of filtered web data and LLM-generated synthetic data.

mistalai/Mistral-7B: Mistral-7B (Jiang et al., 2023) is a 7-billion parameter model developed by Mistral AI, built with a Transformer architecture optimized for efficiency and performance. The model has 32 layers, a model dimension of 4096, a feedforward dimension of 14,336, and 32 attention heads. Mistral-7B uses Grouped-Query Attention (GQA) and Sliding Window Attention (SWA) to handle sequences up to 8192 tokens.

mistralai/Mistral-7B-Instruct: Mistral-7B-Instruct (Jiang et al., 2023) is the instruction-tuned variant of Mistral-7B, featuring the same architecture with 7 billion parameters, 32 layers, 4096 model dimensions, and a feedforward dimension of 14,336.

allenai/OLMoE-1B-7B: OLMoE-1B-7B (Muennighoff et al., 2024) is a Mixture-of-Experts LLM with 1B active and 7B total parameters developed by Allen AI, designed for open access and transparency. The model consists of 32 layers, a model dimension of 4096, a feedforward dimension of 11,008 (due to its SwiGLU activation), and 32 attention heads. The vocabulary size is 50,280 tokens, based on a modified BPE tokenizer that includes special tokens for anonymizing personally identifiable information (PII). OLMo-7B was trained on Dolma, which comprises 2.46 trillion tokens from diverse sources like Common Crawl, GitHub, Wikipedia, and scientific papers.

allenai/OLMoE-1B-7B-Instruct: OLMoE-1B-7B-Instruct (Muennighoff et al., 2024) is a Mixture-of-Experts LLM with 1B active and 7B total parameters that has been adapted via SFT and DPO from OLMoE-1B-7B. Like OLMoE-1B-7B, it features 32 layers, a model dimension of 4096, and 32 attention heads, with a feedforward dimension of 11,008. This variant was fine-tuned using a mixture of human-annotated and distilled instruction data, optimized further using Direct Preference Optimization (DPO) for better alignment with human preferences.

Pythia Scaling Suite: Pythia (Biderman et al., 2023) is a suite of 16 publicly available autoregressive language models, spanning parameter sizes from 70M to 12B, designed to facilitate scientific research into the dynamics of training and scaling in large language models. Each model in the suite was trained on the Pile dataset in a controlled, consistent manner, ensuring identical data ordering and architecture across scales. The suite includes models trained on both the original Pile dataset and a deduplicated version to allow comparative studies of data redundancy effects. Pythia's intermediate checkpointing—offering 154 checkpoints per model—enables detailed longitudinal studies of model behavior over training.

E Additional Results

In this section, we present additional results for the coin flip experiments in section 4 and section 5.

In Figure E.2, Figure E.3, Figure E.4, Figure E.5, and Figure E.6, we present the Mean total variation distance (TVD, ↓) against bias percentage for several ICL (In-Context Learning) example lengths across different models. These plots help analyze how well each model handles bias in a coin flip prediction task as the ICL context varies. The lower the TVD score, the better the model performs in generating unbiased predictions.

E.1 Longer Convergence Chains

In addition to a roll-out of length 100, we also looked at a roll-out of length 200, with the trajectory given in Figure E.1. We can see that in general, the convergence pattern matches the 100 sample case.

E.2 ICL Scaling Results

Here we present all the results from the ICL scaling experiments in Section 5.1.

F Rolling Dice

To explore the applicability of our results beyond coin flips, we also experiment with a similar simple distribution, rolling dice. We then ask the LLM to complete the prompt "`I rolled a die and it landed on`" over the choices of one through six. For biased variants, we provided explicit biasing statements within prompts to the model such as: "When I flip coins, they land on heads X% of the time,"

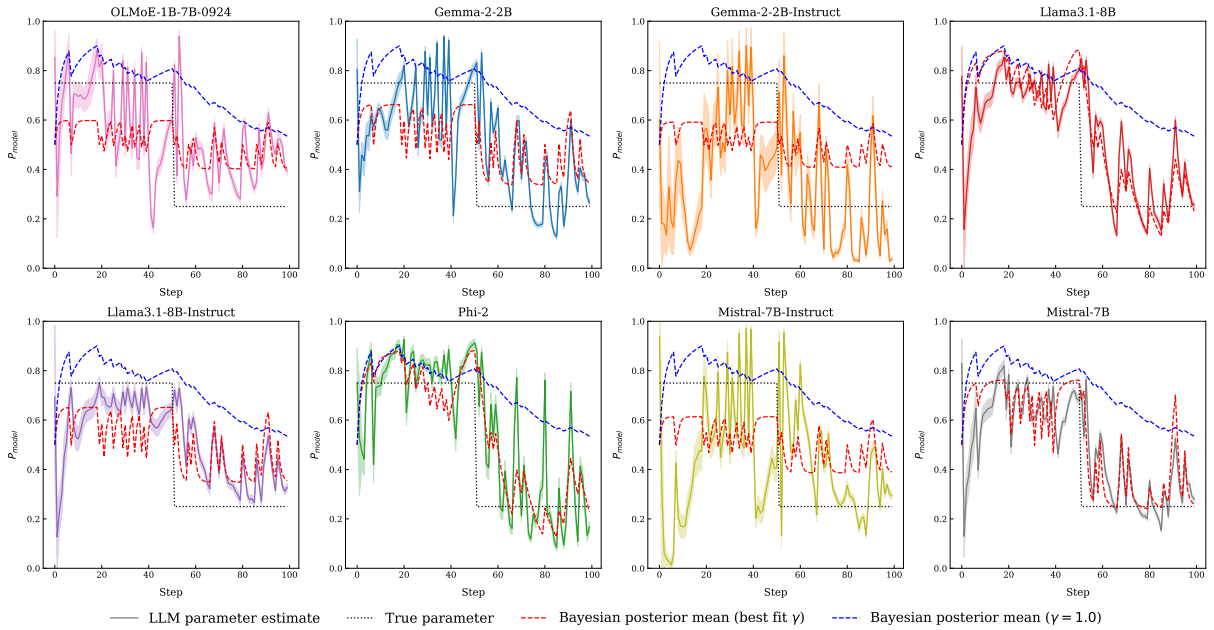

Figure E.1: **Posterior evolution during Bayesian filtering:** The figure shows a single rollout of classical Bayesian filtering alongside model predictive probabilities in a 200-sample coin flip ICL task.

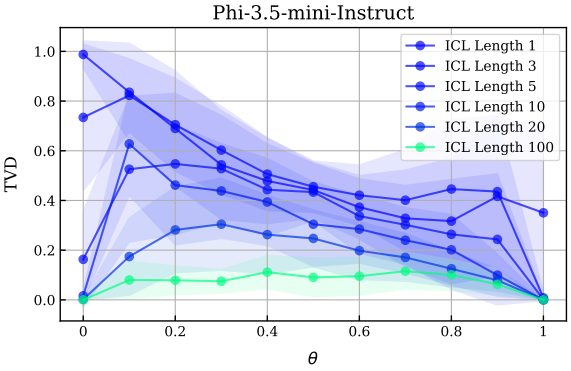

Figure E.2: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the coin flipping task for the Phi-3.5-mini-instruct model.

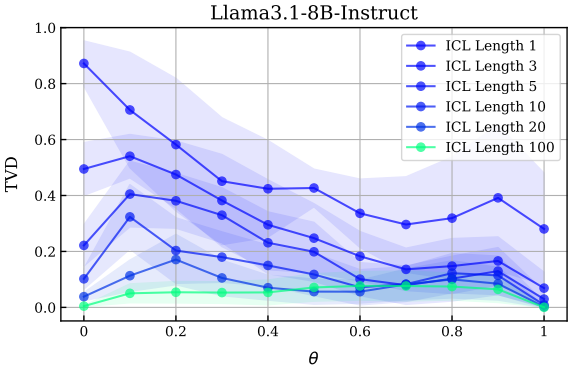

Figure E.3: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the coin flipping task for the Llama-3.1-8B-Instruct model.

where X is a percentage between 0% and 100%, or "When I roll dice, they land on N X% of the time."

The results are shown in Figure F.8. For each bias percentage, we averaged results across the six die faces and 50 prompt variants, totaling 300 trials per bias percentage. Non-instruct models generally performed better than their instruct counterparts, and best around a 50%-60% bias, struggling more with higher biases. Instruct model performance was more varied, with some models showing little change in behavior and others improving as the bias value increased.

Results on die-rolling for in-context learning are shown below. While both instruction finetuned and non-instruction-finetuned variants benefit from increasing numbers of examples, the non-instruction-finetuned variants benefit more and generally exhibit better performance.

In Figure F.3, Figure F.4, Figure F.5, Figure F.6, and Figure F.7, we present ICL plots measuring TVD for a variety of model variants on the simple dice rolling experiment. These results correlate well with the results observed in section 4, the coin flip experiments.

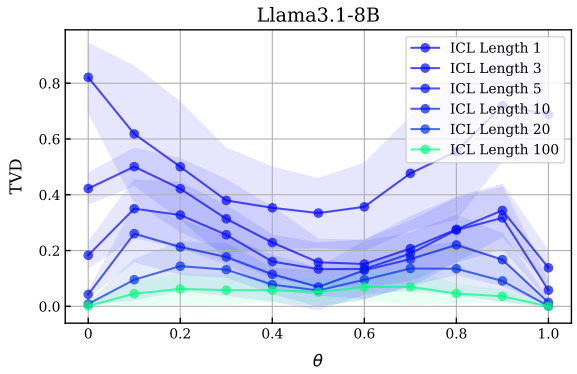

Figure E.4: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the coin flipping task for the Llama-3.1-8B model.

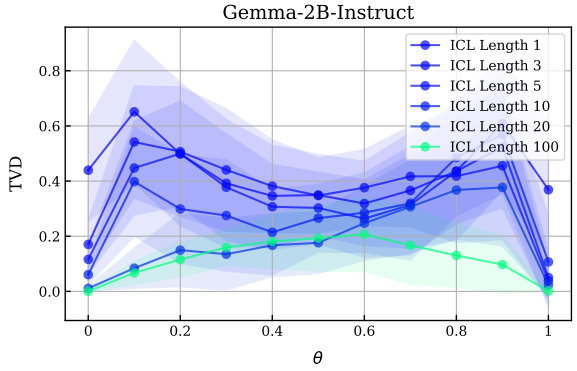

Figure E.5: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the coin flipping task for the Gemma-2-2B-IT model.

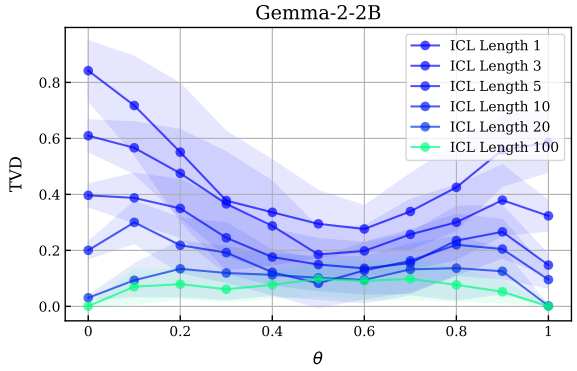

Figure E.6: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the coin flipping task for the Gemma-2-2B model.

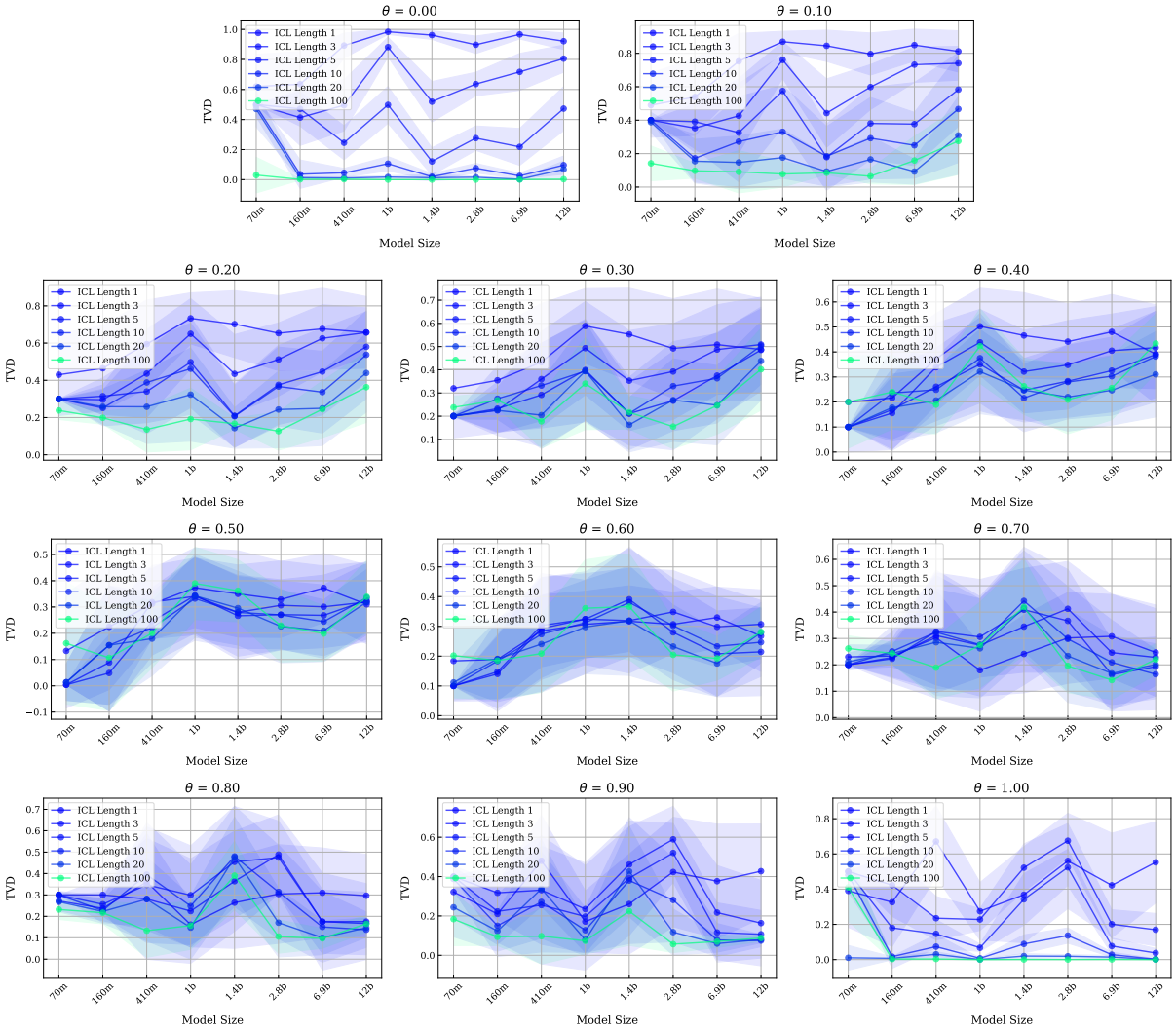

Figure E.7: **ICL and parameter scaling:** Mean total variation distance (TVD, ↓) vs. model size across the Pythia Scaling Suite family with a biasing statement for all values of θ. Model size does not have a clear impact on the benefits from ICL.

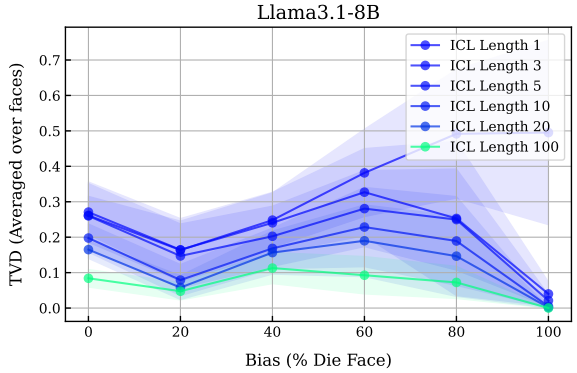

Figure F.1: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Llama3.1-8B model.

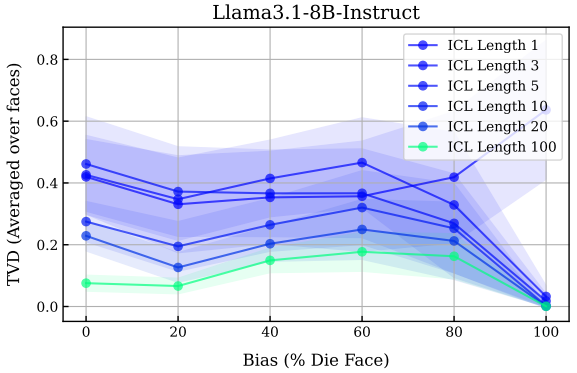

Figure F.2: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Llama3.1-8B-Instruct model.

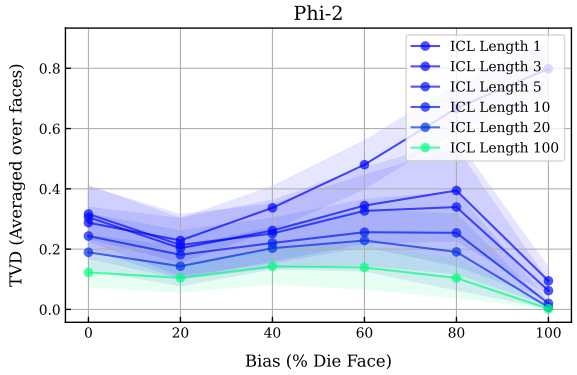

Figure F.3: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Microsoft Phi-2 model.

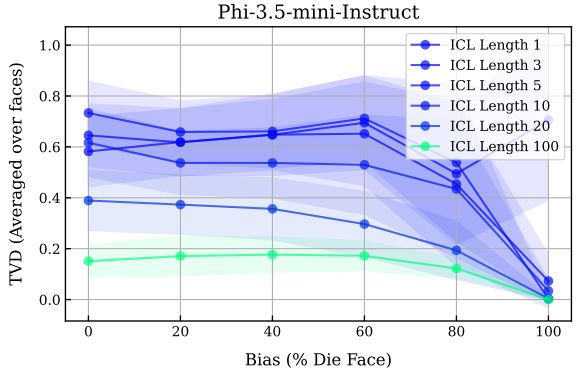

Figure F.4: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Microsoft Phi-3.5-mini-instruct model.

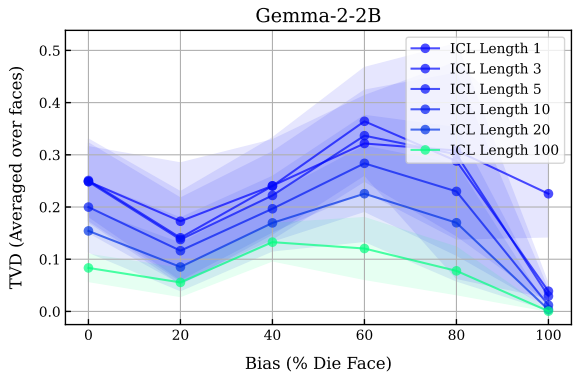

Figure F.5: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Google Gemma-2-2B model.

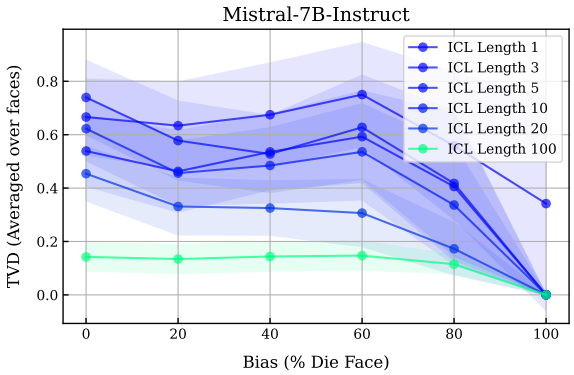

Figure F.6: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Mistral-7B-Instruct model.

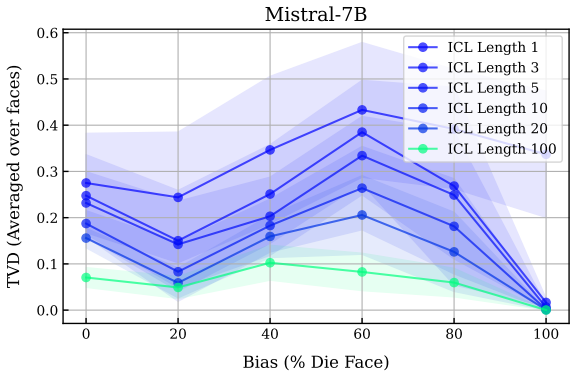

Figure F.7: Mean total variation distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Mistral-7B model.

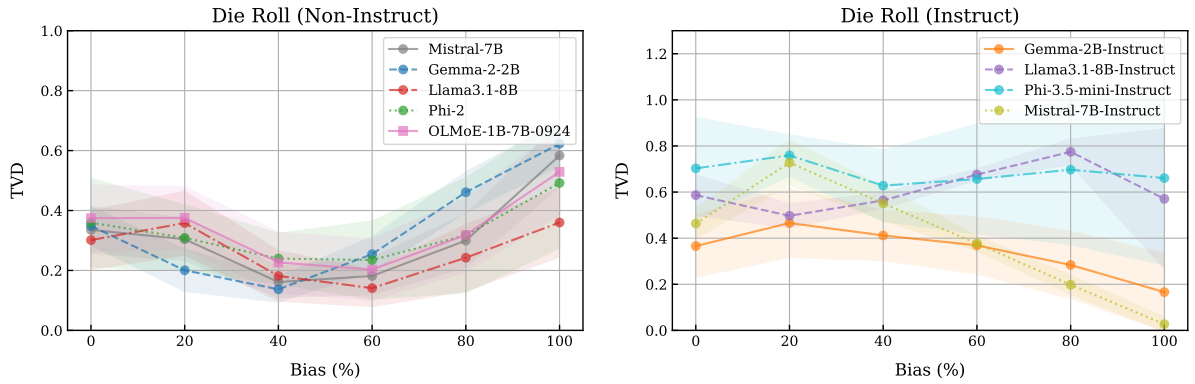

Figure F.8: **Biased die rolls:** Plots of mean total variation distance (TVD, ↓) against bias percentage for non-instruct (left) and instruct (right) models when aggregated across prompts (N=50) for the biased die rolling experiment.