Introducing Background Temperature to Characterise Hidden Randomness in Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Even when decoding with temperature $T = 0$, large language models (LLMs) can produce divergent outputs for identical inputs. Recent work by Thinking Machines Lab highlights implementation-level sources of nondeterminism, including batch-size variation, kernel non-invariance, and floating-point non-associativity. In this work, we formalize this behavior by introducing the notion of *background temperature* T_{bg}, the effective temperature induced by an implementation-dependent perturbation process observed even when nominal $T = 0$. We provide clean definitions, show how T_{bg} relates to a stochastic perturbation governed by the inference environment I, and propose an empirical protocol to estimate T_{bg} via the equivalent temperature $T_n(I)$ of an ideal reference system. We conclude with a set of pilot experiments run on a representative pool from the major LLM providers that demonstrate the idea and outline implications for reproducibility, evaluation, and deployment.

1 Introduction

A common assumption in LLM deployment is that setting the decoding temperature to $T = 0$ (greedy decoding) ensures determinism. However, empirical evidence shows output variability persists under nominally deterministic settings. The recent work in He & Lab (2025) argues that nondeterminism in LLM inference often arises from practical systems issues such as varying batch sizes and the lack of batch-invariant kernels, along with floating-point non-associativity and reduction-order effects. This paper proposes a rigorous framing of such effects via the notion of a *background temperature*.

Contributions. (i) A concise formal model that addresses the phenomenon of nondeterminism as a stochastic effect on the output probability; (ii) a formal definition of background temperature T_{bg}; (iii) the outline of a practical protocol to estimate T_{bg}; (iv) a set of pilot studies illustrating the concept.

2 Related Work

The recent work by Thinking Machines Lab provides a systems-first analysis of LLM nondeterminism, emphasizing batch-size variation and batch-invariant kernels for inference; they also explain how floating-point non-associativity and reduction ordering contribute to variability. He & Lab (2025).

In addition to this work, several recent studies have quantified non-determinism in large language model outputs even under settings intended to be deterministic (e.g. temperature $T = 0$, fixed seeds). For example:

- Atil et al. (2025) systematically evaluate multiple LLMs configured under deterministic settings across zero-shot and few-shot tasks. They observe large accuracy variations (up to 15%) across runs with the same input, and show that even the string outputs are often not identical.
- Song et al. (2024) explore how evaluation practices often ignore variability arising from different decoding configurations (greedy vs sampling). They show that even for greedy decoding, evaluation metrics vary, and that alignment methods can help reduce sampling variance.

- Ouyang et al. (2025) analyze code generation benchmarks and show that many coding tasks produce different code outputs across repeated prompt invocations, even when using $T = 0$. This confirms that deterministic temperature settings do not guarantee output consistency.

These works align closely with observations from Thinking Machines Lab's blog He & Lab (2025) about system-level implementation factors (batch size, kernel non-invariance, floating point non-associativity, etc.) causing output variation even under nominally deterministic decoding.

While prior work largely documents the existence and magnitude of non-determinism, there remains a gap in formalizing this behavior in terms of an equivalent temperature transformation functional and in proposing standard protocols to measure the effective background randomness. Our work addresses this by introducing the notion of an equivalent temperature $T_n(I)$ and its expectation T_{bg}. In the next sections, we transition from formal definitions to a concrete empirical protocol aimed at estimating an *equivalent temperature* $T_n(I)$ induced by implementation noise, and ultimately the background temperature.

To give a concrete description of what an overall measurement protocol for T_{bg} would look like, we first describe general criteria for selecting prompts and datasets that are sensitive to small perturbations in model behaviour (including general, task-oriented, and adversarial/synthetic prompts). We then introduce the actual measurement protocol, made up of reference runs under known nonzero temperature settings to calibrate output variability. Following this, based on a suite of quantitative metrics - such as exact-match frequency, first-divergence token index, edit-distance or string similarity, distributional divergence (e.g. JS or KL) over next-token / top-k probabilistic outputs, and entropy/confidence measures - we finally outline a fitting procedure to infer $T_n(I)$ by minimizing divergence between outputs under noisy $T = 0$ runs and reference nonzero-T runs, and describe how to aggregate over I to compute T_{bg} with statistical confidence.

3 Preliminaries and Notation

Let D denote the token vocabulary with size $|D|$. At generation step i, the model produces logits $z \in \mathbb{R}^{|D|}$ and associated probabilities $P(t) \in [0, 1]$ that the i-th token in the sequence is the $t - th$ token in D, such that $\sum_{t=1}^{|D|} P(t) = 1$ via softmax:

$$P(t) = P(\tau^t | \tau_{<i}) = \frac{\exp(z_t)}{\sum_{s \in D} \exp(z_s)} \quad \text{for } t = 1, \ldots, |D|, \tag{1}$$

where τ^t denotes the t-th token in D and $P(\tau^t | \tau_{<i})$ is the probability of generating τ^t given the sequence of tokens generated up to the i-th token. At $T = 0$, the conventional model is greedy decoding by argmax:

$$\tau_i = \arg\max_{\tau \in D} P(\tau \mid \tau_{<i}), \tag{2}$$

where τ_i denotes the i-th token produced by the model. Decoding at temperature $T > 0$ is equivalent to do the same operation but with modified logits $\hat{z} \in \mathbb{R}^{|D|}$:

$$\hat{P}_T(\tau_i | \tau_{<i}) = \frac{\exp(\hat{z}_i)}{\sum_{s \in D} \exp(\hat{z}_s)}. \tag{3}$$

Then the i-th token is distributed as some Categorical random variable depending on the probability distribution above, i.e.

$$\tau_i \sim Categorical(\hat{P}(\tau \mid \tau_{<i})). \tag{4}$$

Logits are modified through the randomization effects that are included in the decoding process by the specific LLM implementation. In standard autoregressive language models, the decoding temperature parameter modifies the randomness of next-token selection by adjusting the probability distribution derived from logits. Typically, one scales or transforms the raw (pre-softmax) logits via a temperature parameter and then passes them through softmax to obtain the final distribution for sampling or greedy selection. In general, but as a sufficient assumption for the sake of this work, lower temperatures concentrate probability mass on the

most likely tokens, making output more deterministic, while higher temperatures flatten the distribution and increase variability.

Equivalently, this can be seen as the result of the application of an opportune temperature transformation functional F_T:

$$F_T : \mathbb{R}^{|D|} \to \mathbb{R}^{|D|}, \qquad \hat{P} = F_T(P), \tag{5}$$

with the ideal identity limit $F_0(P) = P$. Many implementations use temperature T so that the model effectively computes something like $F_T(P)$, a functional transformation of the original token probability vector P, where $T = 0$ corresponds (ideally) to purely greedy decoding, and $T > 0$ allows stochastic sampling.

4 Modelling Intrinsic Nondeterminism at $T = 0$

As noted by authors in He & Lab (2025), real systems exhibit implementation-dependent perturbations even under $T = 0$. Let $I \in \mathcal{I}$ denote the *inference environment* (batch size and composition, concurrency/load, hardware/backends, kernel choices, numeric precision, reduction ordering, etc.) and F'_T the temperature transformation functional of the real system. We model a perturbation ϵ_I, mapping probability distribution over the set D to probability distribution over the same set, that alters the effective distribution as:

$$F'_0(P) = \epsilon_I(F_0(P)) \approx \epsilon_I(P). \tag{6}$$

While ϵ_I may differ only slightly from $F_0(P)$, in regions where multiple tokens have similar probability mass, even slight changes can flip the argmax in equation 2 and thus the emitted token sequence.

5 Equivalent Temperature and Background Temperature

We posit that the perturbation in equation 6 behaves *as if* decoding were performed by an inference environment - free (ideal) system at a nonzero equivalent temperature $T_n(I)$:

$$F'_0(P) \approx \epsilon_I(P) \approx F_{T_n(I)}(P). \tag{7}$$

This motivates the following definition.

Definition (Background temperature). The *background temperature* of an LLM implementation is the expected equivalent temperature induced by the inference environment under nominal $T = 0$:

$$T_{\text{bg}} := \mathbb{E}_{I \in \mathcal{I}}[T_n(I)]\,. \tag{8}$$

Intuitively, T_{bg} captures the *implicit* randomness in a deployment stack even when the user selects $T = 0$.

6 Estimating $T_n(I)$ and T_{bg} Empirically

The problem with the definition given in equation 8 is that the inference environment-free (ideal) system may be not at hand. In fact, the key challenge in estimating $T_n(I)$ is that it requires comparing to a perfect, deterministic reference - which may not exist in practice. To make $T_n(I)$ calibration feasible without an unattainable ideal, one can first identify a quasi-ideal environment: for example, by using inference pipelines with batch-invariant kernels (in normalization, matrix multiplication, attention), fixed numeric precision, minimal or single-request concurrency, and deterministic configuration flags. Thinking Machines Lab demonstrates that replacing standard kernels with batch-invariant ones drastically reduces output divergence under zero temperature He & Lab (2025) Similarly, Shanmugavelu et al. (2024) show that floating-point non-associativity and asynchronous parallel reductions are major sources of run-to-run variability, and that enforcing deterministic alternatives significantly stabilizes inference and scientific computing pipelines. Based on this evidence, one can anchor measurement of $T_n(I)$ relative to such quasi-ideal baselines, or employ multiple such baselines (differing in hardware, kernel implementation, or precision) to absorb uncertainty.

Further, measuring various output statistical distributions (rather than only output strings) allows matching of environments I to baselines via statistical divergence metrics, reducing sensitivity to rare argmax flips. Reporting T_n together with such baseline variances yields operationally meaningful estimates even in the absence of a perfect oracle.

Another practical way to assess the background temperature of an online model (e.g. ChatGPT) is to use a local installation of another model (e.g. Llama) as a benchmark reference. The local model must be configured to be as deterministic and stable as possible—fixed precision, consistent batch sizes, kernel implementations that do not alter behavior when batch shape changes, deterministic reduction orders, disabled non-deterministic/autotuned operations. This reference becomes a baseline environment that approximates "ideal behavior". Then, by comparing output distributions from the online model versus those from the stable local model, one can compute how far the online model's behavior diverges, for example via measures like Jensen-Shannon divergence or KL divergence. By finding what temperature setting of the local model would make its distribution match the diverged distribution of the online model, it is possible to infer an equivalent temperature for the online model in that environment. Repeated across many prompts and local configurations, this yields an estimate of the online model's background temperature, together with uncertainty bounds. This method avoids relying on an unattainable perfect system, by using the best stable reference you can build.

With these considerations in mind, we can outline a practical protocol to estimate $T_n(I)$ and T_{bg} which is pictorially described in Figure 1.

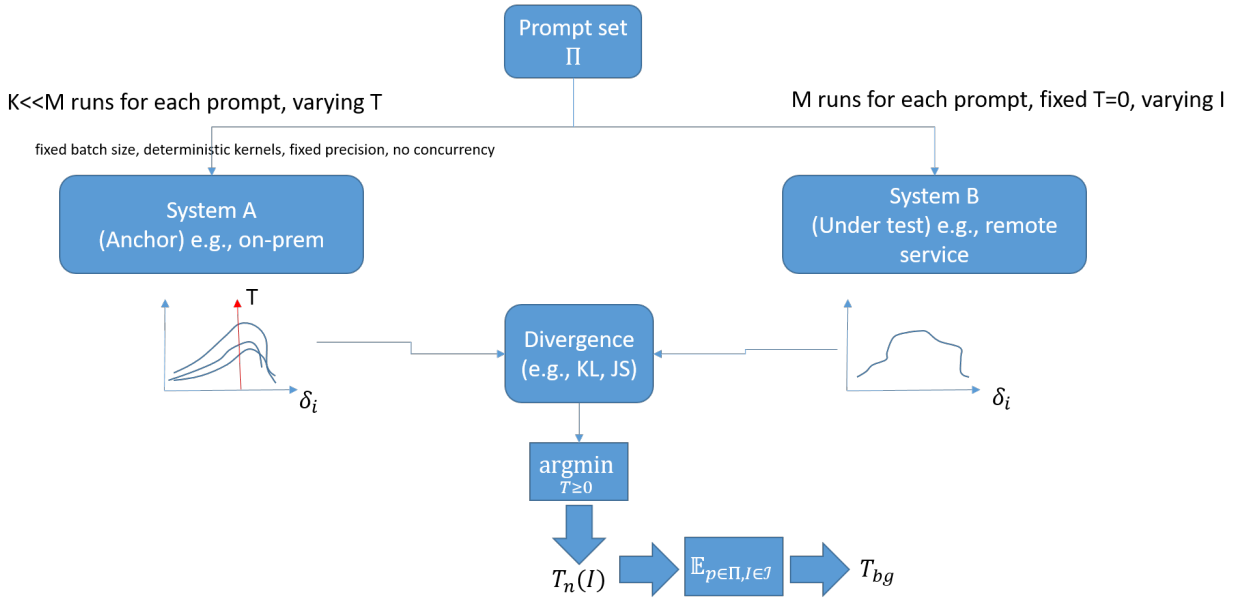


Figure 1: Measuring protocol.

6.1 Prompt Sets and Datasets

The first element of the protocol is constituted by a relevant prompt set Π, an element of the theoretical set of all the possible combinations of prompts $\mathcal{P}$. To explore the full range of behavior of the system under test, the suggestion is to use a diverse evaluation suite, e.g.:

- **General generation** prompts (short/long, common/rare vocab).
- **Task benchmarks**: QA (e.g., SQuADPrice & Cote (2025)/TriviaQAJoshi et al. (2017)), summarization, close, and short-format classification. Code-generation prompts if applicable.
- **Edge/adversarial** prompts: long contexts, rare tokens, near-ties among top-k token probabilities.

- **Synthetic** prompts engineered to create finely balanced next-token choices.

6.2 Controlling the Inference Environment I

Run repeated inference (e.g., $M \geq 50$ per prompt) at $T = 0$ while varying I along axes known to influence nondeterminism:

- **Batch structure**: batch size, e.g. $\in \{1, 2, 4, 8, 16, 32, \dots\}$; co-batching with other prompts vs. serial.
- **Concurrency/load**: single request vs. many simultaneous requests.
- **Hardware/backends**: GPU types, CPU vs. GPU, precision (fp16/bf16/fp32), kernel implementations (batch-invariant vs. standard).
- **Numerics**: reduction order, deterministic flags in frameworks, fused vs. unfused kernels.

For remote systems, for which it may be impossible or impractical to govern the inference environment, one can assume that prolonged and repeated operation is a good way to sample the inference environment statistical distribution.

6.3 Reference Runs at Known Temperatures

Under a *stable* environment I_{stable}, e.g. a local anchor system, run the same prompts e.g. at a grid of $T \in \{0, \dots, 1, \dots\}$ to build a mapping between T and output-variability statistics. As noted earlier, this stable environment can either be a specific configuration of the system under test or another anchor used as reference. Given that the anchor configuration is supposed to be stable for what concerns the inference environment, a lower number K of runs for each prompt in the prompt set should suffice.

6.4 Variability Metrics

The key element of the protocol is the set of metrics used to obtain the association between the sought-for background temperature parameter for the system under test and the reference measurements on the anchor system. Since T_{bg} is thought of as a generic high level account of the system's nondeterminism, metrics should be content-agnostic. Furthermore, since different systems are trained independently, it is practically certain that the same prompt would produce different outputs even under strict deterministic configurations. For example, for each prompt, and across the M (or K) runs of Figure 1, compute process parameters like e.g.:

- **Exact-match rate**: fraction of runs producing identical strings for the same prompt.
- **First-divergence index**: position of first token mismatch across pairs of runs.
- **Edit distance** first order and second order statistics between different outputs.
- **Distributional divergence**: e.g., symmetrized KL or JS divergence between empirical next-token distributions (top-k) across runs.
- **Entropy** of next-token distributions.

Then, for each variability metric computed across the runs, construct a multidimensional distribution f that captures the values of the variability metrics for the system considered. In particular, we denote by $f_T(I_{\text{stable}})$ and $g(I)$ respectively the distribution of the variability metrics for the reference system when the temperature is T and for the system under test set at temperature 0. Note that these distributions depend on multiple factors, including the specific LLMs used; for notational simplicity, we omit these dependencies.

6.5 Estimators for T_n and T_{bg}

As explained in Section 6 the ideal reference system does not exist. However, it is possible to estimate T_n using some reference model running in an environment I_{stable} as stable as possible. In particular, for a reference LLM ℓ, it is possible to compute an estimator $\hat{T}_n^\ell = \hat{T}_n^\ell(I, \Pi)$ of T_n, for each I in the set of environments considered $\tilde{\mathcal{I}} \subseteq \mathcal{I}$ and each Π in the set of all the collections of prompts considered $\tilde{\mathcal{P}} \subseteq \mathcal{P}$, as

$$\hat{T}_n^\ell = \arg\min_{T \geq 0} \mathcal{D}\Big(f_T(I_{\text{stable}}), g(I)\Big), \tag{9}$$

where $\mathcal{D}$ is a chosen divergence (e.g., JS or KL divergence, or a weighted combination) applied to the variability distributions $g(I)$ and $f_T(I_{\text{stable}})$, corresponding respectively to the system under test and to the reference system based on ℓ (see Section 6.4). Therefore, it is possible to compute an estimate $\hat{T}_{\text{bg}} = \hat{T}_{\text{bg}}(\ell)$ of T_{bg}, for each $\hat{T}_n$, as

$$\hat{T}_{\text{bg}}(\ell) = \frac{1}{|\tilde{\mathcal{I}}|} \frac{1}{|\tilde{\mathcal{P}}|} \sum_{I \in \tilde{\mathcal{I}}} \sum_{\Pi \in \tilde{\mathcal{P}}} \hat{T}_n^\ell(I, \Pi), \tag{10}$$

where $|\tilde{\mathcal{I}}|$ and $|\tilde{\mathcal{P}}|$ denote, respectively, the number of all the I and Π considered. To further improve robustness, we repeat the same process across a set of different reference LLMs $\mathcal{L}$ and take the average[1]

$$\overline{T}_{\text{bg}} = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \hat{T}_{\text{bg}}(\ell), \tag{11}$$

where $|\mathcal{L}|$ denotes the number of different ℓ (LLMs) used. Theoretically, as the set of reference LLMs $\mathcal{L}$, prompts, environments, and variability metrics grows, we can expect $\overline{T}_{\text{bg}}$ to converge to the true T_{bg}, as defined in equation 8.

6.6 Engineering to Reduce T_{bg}

Once for a certain system the background temperature T_{bg} is available, several mechanisms can be put in place to mitigate its effect. For example, empirical and systems work suggests several interventions:

- **Batch-invariant kernels** for core ops (matmul, attention, RMSNorm) to prevent batch-shape–dependent numerics He & Lab (2025).
- **Deterministic reductions** and stable accumulation orders where feasible Shanmugavelu et al. (2024).
- **Consistent pipelines**: fix kernel configs across shapes; avoid opportunistic algorithm switching that alters reduction paths Ravi et al. (2025).
- **Deterministic flags** in frameworks and careful precision selection Shanmugavelu et al. (2024).
- **Operational controls**: cap concurrency or shape buckets to reduce co-batching variability He & Lab (2025).

Ablation studies can further determine what intervention is impacting the most on the background temperature. This transforms the outlined protocol into an iterative practice aimed at controlling the nondeterministic characteristics of the system in use, as opposed to a mere observation of an empirical phenomenon.

7 Pilot Experiments

In this section, we present some experiments to validate the theory presented in this work. In particular, in Section 7.1, we present a simple pipeline for estimating the background temperature for a given model. After that, we present additional experiments that could clarify and add elements to analyze the background temperature.

[1]Beyond the average estimate, the availability of multiple reference LLMs and configurations also allows the computation of higher-order moments and confidence intervals, providing a more precise characterization of the uncertainty associated with this kind of estimate.

7.1 Basic pipeline for estimating T_{bg}

Here we perform a pilot experiment to estimate T_{bg} for the OpenAI model *gpt-4.1-nano* accessed via the Microsoft Azure AI services with temperature $T = 0$ (i.e., considering it as System B in Figure 1). Note that, being the model used a via third part service, we can not control the inference environment I but only the temperature. The prompt set Π used is composed of the first 200 questions of the dataset *truthful_qa* [2] (see Lin et al. (2022)).

The reference LLM ℓ, playing the role of System A in Figure 1, is Hugging-Face LLM *SmolLM3-3B*[3] (see Bakouch et al. (2025)). As outlined in previous sections, we selected representative temperature values Θ sampled in increments of 0.01 from 0 to 0.2, in increments of 0.05 from 0.2 to 0.5 and in increments of 0.1 from 0.5 to 1, i.e.

$$\Theta = \{0, 0.01, \ldots, 0.19, 0.2, 0.25, \ldots, 0.45, 0.5, 0.6, \ldots, 0.9, 1\}.$$

For each $T \in \Theta$, we generated $K = 32$ responses, limited to 32 tokens, with the reference LLM for each of the 200 prompts in Π. As variability metric (see Section 6.4), we used the exact-match fraction, i.e. for each temperature considered and each prompt in Π, we computed the maximum fraction of identical answers among the 32 generated. In this way, for each $T \in \Theta$ we obtained 200 values in the interval $[1/32, 1]$, which constitute the discrete distribution f_T of the exact-match fraction for that temperature in the answers given by the reference LLM.

In Figure 2, these distributions are graphically represented, showing how the density estimate shifts from a delta concentrated at 1 when the temperature is 0 - indicating that all answers are identical - to a distribution with most of its mass near 0, indicating that the answers tend to be unique.

After computing the reference distributions f_T for $T \in \Theta$ of the chosen variability measure, we computed the same for the model for which we want to estimate T_{bg}, i.e., *gpt-4.1-nano*, accessed via the Microsoft Azure AI services. To do this, we prompted the model 100 times for each of the 200 prompts in Π, but this time setting the temperature at $T = 0$ and limiting the answers to 32 tokens, as done for the reference system. Then, analogously to the procedure for the reference system, for each prompt in Π we computed the maximum fraction of identical answers provided by *gpt-4.1-nano*. These 200 values, in $[1/100, 1]$, form the discrete distribution g (see Figure 3) that we need to compare with the reference distributions computed in system A (see equation 9).

In order to compare the discrete distributions of observations, f_T for $T \in \Theta$ and g, we chose to use the Kolmogorov–Smirnov (K-S) distance, which is equal to 0 for identical distributions and 1 for completely different ones. The computed values of K-S distance are reported in Figure 4.

From the values in Table 4 (b), we can conclude that the estimator of T_{bg} found in this experiment is $\hat{T}_{\text{bg}}(\ell) = 0.05$ (which, in this simple case, coincides with $\overline{T}_{\text{bg}}$), as this is the case where f_T is closest to g, considering only the reference distributions computed from $T \in \Theta$. Figure 5 shows the two matching histograms. Ideally, this experiment should be repeated using a wider range of T values - especially lower ones - more prompts, fewer token limits, and different variability metrics (see Sections 6.4 and 6.5). However, the purpose of this pilot experiment was simply to demonstrate the full procedure to estimate T_{bg}.

7.2 Extending the reference model set $\mathcal{L}$

One of the possibilities for making the estimate of the T_{bg} more robust is to add reference models, i.e. extend the set $\mathcal{L}$ introduced in Section 6.5. In particular, we used the LLM *Llama-3.2-3B-Instruct*[4] and made it answer 32 times to the same 200 prompts (the same set Π used in Section 7.1), limiting the answers to 32 tokens, analogously to what we did for *SmolLm3-3B* in Section 7.1, for each $T \in \tilde{\Theta} = \Theta \cup \{1.05, 1.1, \ldots, 1.45, 1.5\}$. So, hereinafter the set of reference LLM $\mathcal{L} = \{\text{smoll}, \text{llama}\}$, where smoll and llama stand, respectively, for *SmolLm3-3B* and *Llama-3.2-3B-Instruct*. Then we computed for llama the

[2] `https://huggingface.co/datasets/truthfulqa/truthful_qa`
[3] `https://huggingface.co/HuggingFaceTB/SmolLM3-3B`
[4] `https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct`

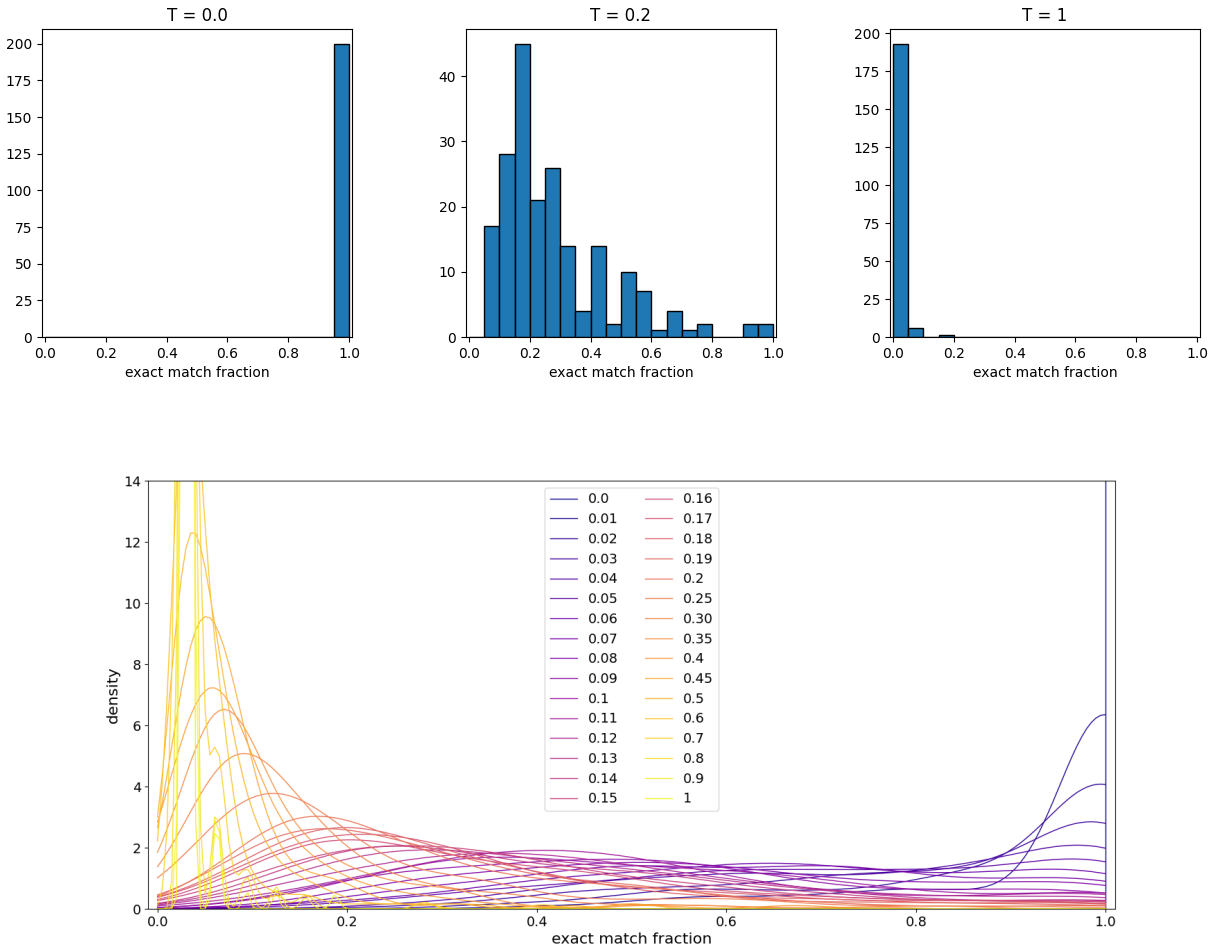


Figure 2: Distribution of exact-match fractions obtained from the reference LLM answers. Top row (from left to right): histograms representing the distributions f_0, $f_{0.2}$ and f_1. Bottom row: kernel density estimates of the exact-match fraction for all sampled temperatures in Θ. Note that for $T = 0$, the density is represented as a vertical line because all answers are identical, so the density is entirely concentrated at 1, forming a Dirac delta.

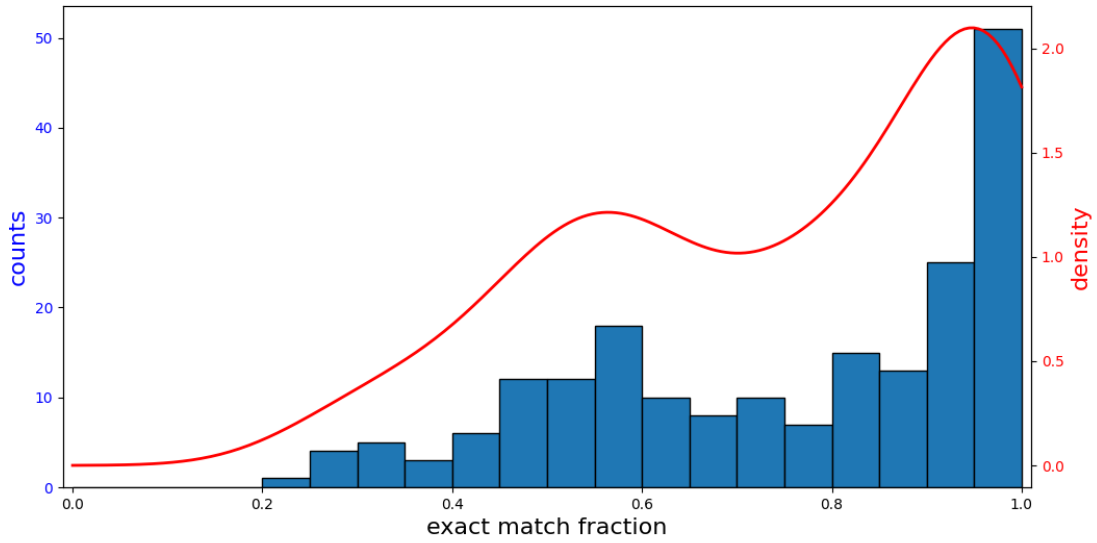
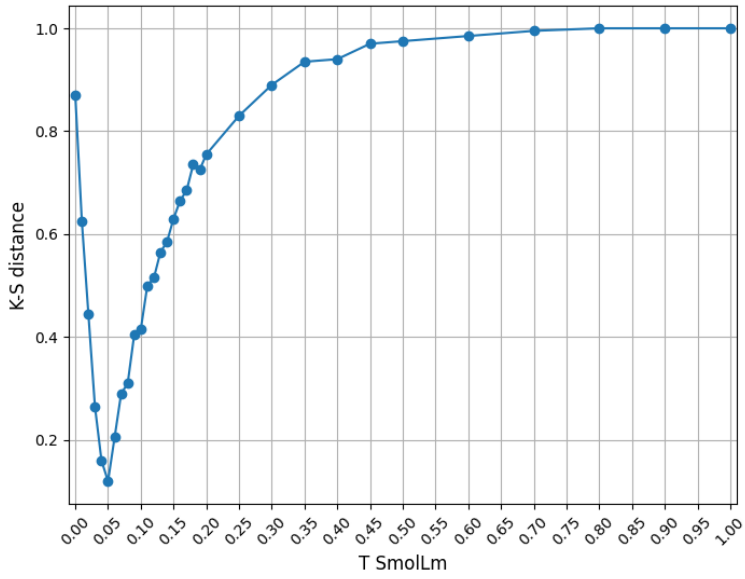


Figure 3: Discrete distribution g of the fraction of identical answers given by the LLM under test, *gpt-4.1-nano*, to the prompts in Π. The distribution is shown both as histograms (with the y-axis on the left) and as a kernel density estimate (with the y-axis on the right).



(a) K-S distance

T	K-S
0.00	0.870
0.01	0.625
0.02	0.445
0.03	0.265
0.04	0.160
0.05	**0.120**
0.06	0.205
0.07	0.290
0.08	0.310
0.10	0.415
0.15	0.630
0.30	0.890
0.50	0.975
1.00	1.000

(b) Sample of the K-S values

Figure 4: K-S distances between f_T and g at different $T \in \Theta$ for the tested model *gpt-4.1-nano*. Plot of all the values (a). Exact values for a sample of temperatures (b). The tested model background temperature estimate is 0.05.

analogous to f_T for smoll, i.e. the reference discrete distribution $\tilde{f}_T$ for all values of $T \in \tilde{\Theta}$ of the maximum fraction of identical answers among the 32 generated for each prompt in Π by llama with temperature T.

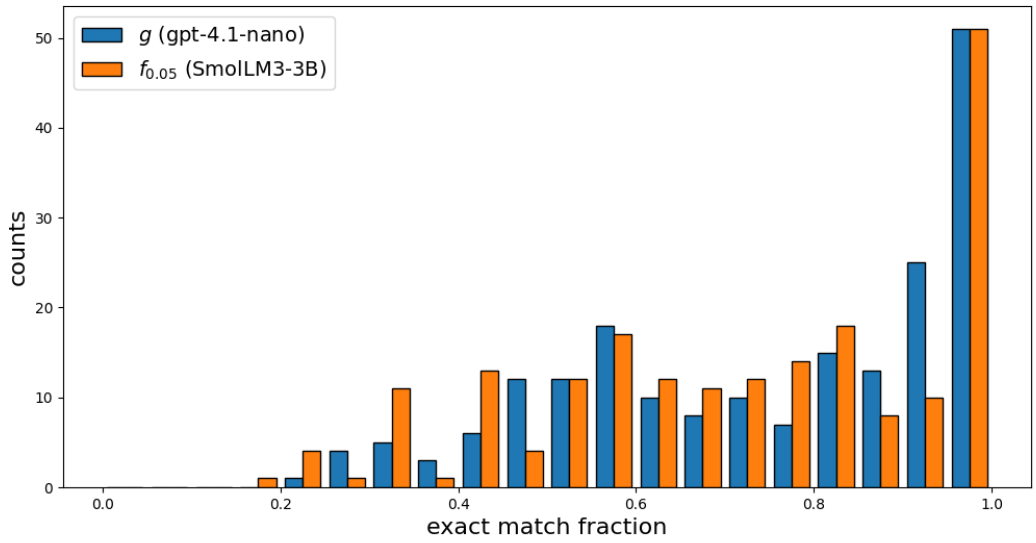


Figure 5: Side-by-side histograms of the two empirical discrete distributions, g and $f_{0.05}$, which is the closest to g among $\{f_T : T \in \Theta\}$ in terms of K-S distance.

Then it is possible to compute the value of $\hat{T}_{\text{bg}}(\text{llama})$ for the model under test (*gpt-4.1-nano*), exactly as we did for $\hat{T}_{\text{bg}}(\text{smoll})$ in Section 7.1. In particular, the minimum value of the K-S distance between g and $\tilde{f}_T$, for $T \in \tilde{\Theta}$ is 0.155 and it is reached when $T = 0.1$, so $\hat{T}_{\text{bg}}(\text{llama}) = 0.1$. Hence, we can compute a more robust value of $\overline{T}_{bg}$ for the LLM under test, using equation 11:

$$\overline{T}_{bg} = \frac{\hat{T}_{\text{bg}}(\text{smoll}) + \hat{T}_{\text{bg}}(\text{llama})}{2} = 0.075. \tag{12}$$

Note that $\hat{T}_{\text{bg}}(\text{smoll})$ and $\hat{T}_{\text{bg}}(\text{llama})$ are different, since the reference models behave differently depending on the temperatures at which they are tested. This highlights the importance of building an adequately large set $\mathcal{L}$ in order to obtain a precise estimate of $\overline{T}_{bg}$ for the system under test. In fact, even if, in general, the trend of the empirical variability distribution computed with different reference systems is similar, its specific values could differ as the different reference LLMs are more or less sensitive to temperature changes, as is clear in Figure 6 where the K-S distance between f_T and $\tilde{f}_{T'}$ is represented, for $T \in \Theta$ and $T' \in \tilde{\Theta}$.

7.3 Estimating T_{bg} for other models

In this section, we present a couple of other experiments aimed at estimating T_{bg} for other LLMs accessed via external providers (where the environment I is unknown), in analogy with the procedure described in Sections 7.1 and 7.2 for *gpt-4.1-nano* provided through Microsoft Azure AI services. In particular we considered the following three other LLMs:

- *gemini-2.0-flash* by Google used through Google Gemini API services;
- *grok-3-mini* by xAI used through Microsoft Azure AI services;
- *claude-sonnet-4* by Anthropic used through Amazon Bedrock services.

Each model was run with temperature set to 0, generating 100 answers of up to 32 tokens for the first 30 prompts in Π (see Sections 7.1 and 7.2). As in the case of *gpt-4.1-nano*, we then computed, for every

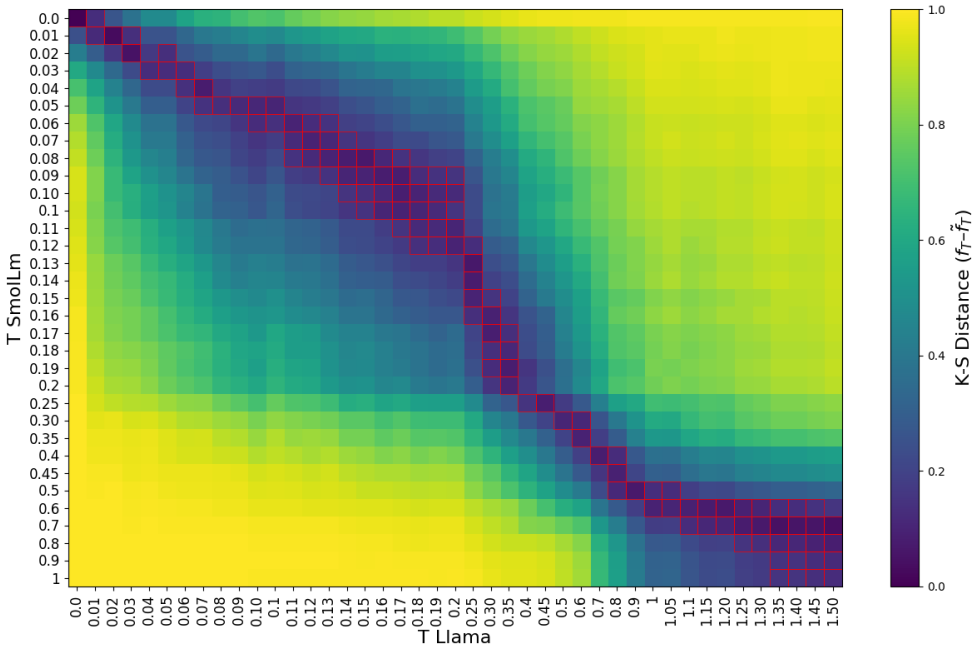


Figure 6: Heatmap representing the K-S distance between f_T and $\tilde{f}_{T'}$ for $T \in \Theta$ and $T' \in \tilde{\Theta}$. In particular the cell on the t-row and t'-column represent the K-S distance between f_t and $\tilde{f}_{t'}$. Cells with red borders correspond to K-S distance values less or equal than 0.15.

prompt, the maximum fraction of identical answers produced by a given model. Hence we built three discrete distributions (each with 30 values), denoted g', g'', and g''', corresponding respectively to *gemini-2.0-flash*, *grok-3-mini*, and *claude-sonnet-4*, which are the analogue of the distribution g built in Section 7.1 for *gpt-4.1-nano*. As reference variability distributions, we used f'_T and $\tilde{f}'_{T'}$, obtained by truncating f_T and $\tilde{f}_{T'}$ to their first 30 elements (the ones corresponding to the first 30 prompts in Π), for $T \in \Theta$ and $T' \in \tilde{\Theta}$. Thus, by computing the K-S distances of g', g'', and g''' with respect to f'_T and $\tilde{f}'_{T'}$, for $T \in \Theta$ and $T' \in \tilde{\Theta}$, and then combining them as in Section 7.2, we can estimate T_{bg} for the new models under test. A couple of remarks on these computations:

- the answers given by *claude-sonnet-4* to each question were identical, making g''' a list of thirty 1s. In this case, the estimate of T_{bg} is trivially 0;
- the distance between g'' and $\tilde{f}'_{T'}$, for $T' \in \tilde{\Theta}$, reaches its minimum value of 0.167 for $T' \in \{0.01, 0.02, 0.03\}$. In this case, we set $\hat{T}_{\text{bg}}(\text{llama}) = 0.02$, i.e. the average of the temperatures minimizing the K-S distance.

In Table 1 are summarized the estimates for the tested models, taking in account the remarks above.

8 Discussion

The notion of *background temperature* brings several concrete benefits for both evaluation and deployment of LLMs. First, it offers a measurable quantity to capture hidden randomness in inference stacks, converting vague observations of output variability into reproducible metrics; this helps close the gap between what settings are declared "deterministic" (e.g. $T = 0$) and what is actually observed in practice. Second, it aids diagnostics: by quantifying $T_n(I)$ for different inference-environment parameters (batch size, concurrency,

Model	$\hat{T}_{\text{bg}}$(smoll)	$\hat{T}_{\text{bg}}$(llama)	$\overline{T}_{\text{bg}}$
grok-3-mini	0.01	0.02	**0.015**
gemini-2.0-flash	0.05	0.08	**0.065**
gpt-4.1-nano	0.05	0.10	**0.075**
claude-sonnet-4	0	0	**0**

Table 1: Estimates of T_{bg} as computed in Sections 7.1–7.2 for *gpt-4.1-nano*, and in Section 7.3 for the other models.

kernel implementation, hardware, or even time of the day and region of deployment), one can identify which aspects of the stack contribute most to instability, and thus target them for engineering improvements. Third, for high-stakes or regulated applications, background temperature enables transparency and trust: one may report that under nominally deterministic settings, the effective randomness is bounded by some small T_{bg}, which supports claims of consistency needed for auditing, compliance, and user trust. Fourth, in research and model evaluation, the measure helps avoid misleading comparisons: if model A slightly outperforms model B under the same settings but with T_{bg} exceeding that difference, the apparent improvement might simply reflect implementation noise rather than genuine modeling advances. Finally, the concept fosters better engineering practices—motivating adoption of batch-invariant kernels, deterministic reduction orders, consistent numeric precision and hardware configurations - as suggested by prior work. Overall, background temperature provides a bridge between theory and practice, enabling metrics, engineering tradeoffs, and trustworthiness to advance in parallel.

Issues easily anticipated are related to the selection of relevant reference models. While it is likely that they will depend on the specific information domain, their specifications should be subjects to an as much wide as possible consensus in the industry. This may give way to standardisation efforts and certification programmes for models.

9 Conclusion

In this note we introduced *background temperature* as a concise lens on hidden randomness in LLM inference. The notion of background temperature reframes a practical pain point - apparent nondeterminism at $T = 0$—in terms familiar to modeling and evaluation. By *measuring* T_{bg}, practitioners can make informed choices about evaluation protocols, safety margins, and infrastructure investments. Reporting T_{bg} (and its higher order statistics) alongside accuracy/throughput could become a valuable practice for transparent model documentation. The definition and operationalisation we gave of T_{bg} captures implementation-induced variability when nominal $T = 0$. We outline a measurement protocol and a set of summarized engineering strategies to reduce T_{bg} as well as a set of small pilot experiments to illustrate the potential of the concept. Future work should include standardizing metrics, open datasets of prompts sensitive to implementation noise, and community benchmarks for deterministic inference.

References

Berk Atil, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, Zhe Wu, Lixinyu Xu, and Breck Baldwin. Non-determinism of "deterministic" llm settings, 2025. URL `https://arxiv.org/abs/2408.04667`.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. `https://huggingface.co/blog/smollm3`, 2025.

Horace He and Thinking Machines Lab. Defeating nondeterminism in llm inference. Thinking Machines Lab blog, 2025. URL https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/. Accessed: 2025-09-15.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. URL https://arxiv.org/abs/2109.07958.

Shuyin Ouyang, Jie M. Zhang, Mark Harman, and Meng Wang. An empirical study of the non-determinism of chatgpt in code generation. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–28, January 2025. ISSN 1557-7392. doi: 10.1145/3697010. URL http://dx.doi.org/10.1145/3697010.

S. Price and D. L. Cote. Document analysis with llms: Assessing performance, bias, and nondeterminism in decision making. In *ICPRAM 2025: Proceedings of the 14th International Conference on Pattern Recognition Applications and Methods*, pp. 207–214, 2025. ISBN: 978-989-758-730-6.

Nikita Ravi, Abhinav Goel, James C. Davis, and George K. Thiruvathukal. Improving the reproducibility of deep learning software: An initial investigation through a case study analysis. *arXiv preprint*, arXiv:2505.03165, 2025. URL https://arxiv.org/abs/2505.03165. Accessed: 2025-09-15.

Sanjif Shanmugavelu, Mathieu Taillefumier, Christopher Culver, Oscar Hernandez, Mark Coletti, and Ada Sedova. Impacts of floating-point non-associativity on reproducibility for hpc and deep learning applications. *arXiv preprint*, arXiv:2408.05148, 2024. URL https://arxiv.org/abs/2408.05148. Accessed: 2025-09-15.

Yifan Song, Guoyin Wang, Sujian Li, and Bill Yuchen Lin. The good, the bad, and the greedy: Evaluation of llms should not ignore non-determinism, 2024. URL https://arxiv.org/abs/2407.10457.