# Strategies for Reproducibility in LLM Inference: An Expert Analysis from ZOKFORCE

## 1. Executive Summary

At ZOKFORCE, we've observed that the inference process of Large Language Models (LLMs) is inherently non-deterministic, meaning the model can produce different outputs even when given an identical prompt. While this deliberate stochasticity (i.e., random sampling) is a benefit in creative tasks, a deeper, systemic form of non-determinism plagues production systems, undermining their reliability, debuggability, and commercial value.1

We at ZOKFORCE closely follow the latest advancements in the field. As recent research from teams like Thinking Machines Lab has pointed out, the traditional explanation for this issue—the "concurrency + floating point" hypothesis—is an incomplete diagnosis. While the non-associativity of floating-point operations is a real underlying mechanism for the variation, the root cause lies in a lack of "batch invariance".3 Unpredictable server load forces inference engines to dynamically change their batch sizes, which in turn alters the internal execution strategies of the underlying kernels, leading to bitwise output differences.3

To defeat this systemic non-determinism, we believe a coordinated, multi-layered approach across the entire AI stack is required, from low-level kernel engineering to high-level prompt design. Relying solely on a single parameter, such as a random seed, does not provide a complete solution. Our ZOKFORCE team is dedicated to helping clients integrate and leverage the latest technological breakthroughs, such as "batch-invariant kernels," to build truly reproducible AI production systems.5

However, achieving true reproducibility is not without cost. Adopting deterministic algorithms and kernels often introduces a significant performance overhead that must be carefully weighed against the specific needs for reliability and consistency.

Our primary recommendations include: combining foundational engineering, API-level control, and robust application design to build resilient systems that can either enforce determinism or gracefully handle variation, depending on specific business needs. This is the goal we at ZOKFORCE are committed to helping our clients achieve.

## 2. The Necessity of Reproducibility in Modern AI

### 2.1 Defining the Challenge: Stochasticity vs. Nondeterminism

In the context of LLMs, non-determinism is a core issue we have long focused on at ZOKFORCE. It refers to the model's behavior of producing different outputs given the same input. This is a crucial distinction from stochasticity, which is the deliberate randomness controlled by parameters like temperature. This is often desirable in applications that require creativity and diversity. For example, a higher temperature makes the model more likely to select a less probable but more "creative" option for the next word. Our goal is not to eliminate all randomness, but to defeat the systemic, unpredictable behavior so that the randomness controlled by the user is reproducible.

### 2.2 The Case for Determinism: From Research to Enterprise

In our view, the ability to reproduce outputs is a cornerstone of scientific rigor and a commercial necessity.2

Scientific Advancement and Debugging

Reproducibility is essential for peer review, validating research findings, and addressing the "reproducibility crisis" that has plagued AI research.2 It facilitates bug detection and regression testing by ensuring that changes in the code or model are the sole cause of output changes, rather than random noise. When intermediate results change across multiple runs, debugging becomes impossible.8 Furthermore, for enterprises, reproducibility is a prerequisite for building reliable, transparent, and accountable AI systems.

Enterprise Reliability and Compliance

For commercial applications, non-determinism introduces serious risks.1 For instance, when generating content for product descriptions or legal documents, inconsistent outputs can lead to brand confusion, user frustration, and a lack of auditability.1 Imagine a scenario where a customer finds two different descriptions for the same product, or a marketing team member gets varying content each time they run the same prompt. These situations can harm a company's brand and credibility.

## 3. Deconstructing the Sources of Non-determinism: A Deep Technical Dive

### 3.1 Re-evaluating the “Concurrency + Floating Point” Hypothesis

A long-standing hypothesis has been that LLM non-determinism stems from the combination of floating-point non-associativity and concurrent computation on GPUs. Due to their finite precision, floating-point arithmetic may result in (a + b) + c not equaling a + (b + c). In parallel systems, the order in which intermediate values are summed can differ due to the unpredictable completion order of threads, leading to minor variations in the final result.8

However, according to research from teams like Thinking Machines Lab, this is not the primary reason for modern LLM inference non-determinism.11 A single operation in an LLM's forward pass, such as matrix multiplication, is itself deterministic. Running the same

torch.mm(A, B) operation a thousand times on a GPU will yield the exact same bit-for-bit result every time.3 This indicates the problem isn't with the math itself, but that the input to the operation is changing between runs. The floating-point issue is merely the vehicle through which this variation manifests.

### 3.2 The True Culprit: Lack of Batch Invariance

Production LLM services (like ChatGPT) to maximize GPU utilization and throughput, bundle multiple user requests into a "batch" for processing.11 From the perspective of a single user, the number of other requests in the same batch is completely random, a source of systemic non-determinism related to server load.3

The core of the problem is that changes in batch size force the inference engine to adopt entirely different internal kernel strategies to optimize for that specific batch shape.3 For example, when a batch is too small to keep all GPU cores busy, engineers may switch to a "split-reduction" strategy where multiple cores collaborate on a single calculation, which changes the order of operations and breaks batch invariance.3

This reveals a deeper architectural conflict: the contradiction between maximizing computational efficiency and ensuring reproducibility for each request. If the entire batch (including all user requests) is considered the input, the system is technically deterministic.7 However, for a single user, the other requests are an uncontrollable variable. Therefore, our solution must involve re-engineering the system to decouple the output of a single query from the other requests in the batch.

### 3.3 Intrinsic LLM Stochasticity

Beyond systemic non-determinism, LLMs' intrinsic stochasticity is another major source of variation. LLMs generate text by predicting a probability distribution for the next token.14 The three main strategies to control this generation are:

* **Greedy Decoding**: Always choosing the highest-probability next token, which is the most deterministic approach.16
* **Beam Search**: Explores multiple high-probability token sequences to find the best path. It is more complex than greedy decoding, less deterministic, and often slower.16
* **Sampling**: Introduces controlled randomness via parameters like temperature, top\_p, or top\_k.14 The  
  temperature parameter controls the "sharpness" of the probability distribution; a temperature=0 aims for a deterministic output by defaulting to the highest-probability token, but this alone is not always enough to guarantee complete determinism.19

### 3.4 Other Potential Sources of Non-reproducibility

* **Unset Random Seeds**: To ensure a consistent starting state, random seeds must be set at every level of the stack (Python, NumPy, PyTorch, and CUDA).19
* **Nondeterministic Algorithms in Frameworks**: Certain operations in frameworks like PyTorch and TensorFlow are nondeterministic by default for performance.20 For example, the  
  cuDNN library for CUDA convolutions can benchmark to choose the fastest algorithm, potentially selecting a different one on subsequent runs.20

## 4. A Multi-Layered Approach to Defeating Non-determinism

### 4.1 Foundational Solutions: Kernels, Frameworks, and System-Level Engineering

#### 4.1.1 The Breakthrough of Batch-Invariant Kernels

The development of batch-invariant kernels is the most direct solution to systemic non-determinism.3 In an experiment, using

vLLM on top of batch-invariant kernels produced "bitwise identical repeatable completions" for 1,000 runs of the same prompt, whereas the non-deterministic version without these kernels produced 80 unique outputs.5 This demonstrates that this approach can successfully defeat non-determinism.

Specific implementation strategies include 4:

* **RMSNorm**: Ensures a fixed reduction order for each element, regardless of batch size, to maintain batch invariance.
* **Matrix Multiplication (Matmul)**: Uses a fixed kernel configuration regardless of batch shape to achieve batch invariance, though this sacrifices some performance.
* **Attention**: Employs a split-reduction strategy with a fixed split size instead of an adaptive one that changes with the batch, ensuring a consistent reduction order.

The table below summarizes the sources of non-determinism and their mitigation strategies.

| **Source of Nondeterminism** | **Cause/Mechanism** | **Mitigation Strategy** | **Performance Cost** | **Implementation Level** |
| --- | --- | --- | --- | --- |
| Lack of Batch Invariance | Dynamic batch size changes kernel execution strategy, altering floating-point accumulation order 3 | Batch-invariant kernels | Significant but manageable (e.g., ~60% slower for some tasks) 4 | System/Kernel |
| Floating-point Non-associativity | Unpredictable accumulation order in parallel computation 23 | Deterministic libraries/algorithms | Possible very high, up to 100x slowdown 22 | Framework/Kernel |
| Random Sampling | Random selection of tokens from a probability distribution 14 | temperature=0, fixed random seed | Minimal cost | API/Application |

#### 4.1.2 Framework-Specific Determinism Controls

For engineers, reproducibility is not an out-of-the-box feature but a state that must be actively and consciously configured throughout the software stack. Developers cannot simply set a single random seed and assume it is sufficient; they must understand which specific operations are non-deterministic and apply the correct patches.

The table below provides a guide to framework-specific determinism controls.

| **Framework/Library** | **Control/Function** | **Purpose** | **Code Example** | **Notes/Limitations** |
| --- | --- | --- | --- | --- |
| PyTorch | torch.use\_deterministic\_algorithms(True) 25 | Forces use of deterministic algorithms | torch.use\_deterministic\_algorithms(True) | Can significantly reduce performance.20 Some operations have no deterministic alternative. |
| PyTorch | torch.backends.cudnn.benchmark = False 20 | Disables cuDNN convolution benchmarking | torch.backends.cudnn.benchmark = False | Forgoes performance-optimizing algorithm selection.20 |
| PyTorch | CUBLAS\_WORKSPACE\_CONFIG Environment Variable 25 | Ensures deterministic operations in CUDA 10.2+ | os.environ = ":16:8" | Can lead to performance degradation.25 |
| Hugging Face | enable\_full\_determinism() 21 | Enables determinism for entire Diffusers pipeline | enable\_full\_determinism() | Can reduce performance.21 |
| Hugging Face | torch.Generator Object 21 | Ensures consistent sampling on CPU and GPU | g = torch.Generator(device="cpu").manual\_seed(0) | The state changes after use, so the same generator cannot be used again for the same result.21 |

### 4.2 API and Application Layers: Practical Controls and Best Practices

#### 4.2.1 The Seed Parameter and Commercial APIs

Services like Azure OpenAI offer a seed parameter as a "best-effort" attempt to enable deterministic sampling. Developers can set a fixed integer value with the expectation of getting the same results on repeated requests. However, this is not an absolute guarantee. Even when the seed parameter and system\_fingerprint are the same, some degree of variability in responses may still be observed.6 This explicit statement from providers reflects the complexity of the underlying system-level challenges, which are difficult to solve perfectly at scale. Determinism also generally decreases with larger

max\_tokens values.6

#### 4.2.2 Mastering Decoding and Sampling

Strategically using decoding parameters is critical, depending on the needs of the specific application.

* **For Consistency**: For tasks requiring precision and factual consistency (e.g., summarization, translation, or technical documentation), we recommend using greedy decoding or setting temperature to a value close to 0.
* **For Creativity**: For tasks needing diverse or creative outputs (e.g., story generation or content creation), we suggest using a higher temperature or top\_p value.

The table below summarizes different decoding strategies.

| **Decoding Method** | **Key Parameters** | **Determinism** | **Creativity/Diversity** | **Computational Cost** |
| --- | --- | --- | --- | --- |
| Greedy Decoding | None | Highest | Lowest | Lower |
| Beam Search | beam\_width | Low | Lower | Medium |
| Sampling | temperature , top\_p , top\_k | Controllable | Controllable | Lower |

#### 4.2.3 Structured Output and Architectural Safeguards

Rather than trying to eliminate variation entirely, it is better to design systems that handle it gracefully.

* **Structured Output**: Using API features that enforce a structured output (such as JSON mode enforcement and function calling) ensures that even if the content varies slightly, the format of the output is predictable and machine-readable.
* **Validation and Retry Logic**: Implement retry loops that provide structured feedback to the model on failure to correct for formatting errors or non-compliant outputs.

### 4.3 Human-AI Interaction: Prompt Engineering and Workflow Management

#### 4.3.1 Crafting Deterministic Prompts

Prompt engineering can effectively narrow the model's output space. Using precise, detailed instructions and "negative prompting" can guide the model's behavior. Additionally, using high-quality "few-shot" examples can "show, not just tell" the model the exact output format and style required, which is particularly effective for handling edge cases.

#### 4.3.2 Reproducibility in MLOps

Comprehensive reproducibility requires a holistic approach to managing AI workflows.23 It is critical to version not only the model itself but also the specific prompts, decoding parameters, and even environmental dependencies to ensure full reproducibility.23 For identical requests, a simple and effective caching layer can also be a solution.19

## 5. Performance vs. Reproducibility Trade-off Analysis

### 5.1 Quantifying the Cost

The cost of achieving determinism is quantifiable.22 At the hardware level, disabling

cuDNN benchmarking sacrifices performance by forgoing the opportunity to find the fastest algorithm.20 Furthermore, enforcing determinism in TensorFlow can result in a "100x slowdown".

At the kernel level, according to a test from Thinking Machines Lab, the first-generation batch-invariant kernels have a measurable performance cost.5 A test showed that a task took 42 seconds with deterministic kernels versus 26 seconds by default.4

### 5.2 Strategic Decision-Making

At ZOKFORCE, we believe the decision of when to prioritize speed over determinism should be a strategic one based on the specific application.

* **Prioritize Speed**: For high-throughput creative applications, chatbots, or internal brainstorming tools, output diversity is acceptable and even desirable.
* **Prioritize Determinism**: For mission-critical applications such as legal document review, financial report generation, or any system requiring full auditability, debuggability, and scientific rigor.

## 6. Action Plan for Commercial Companies

We have consolidated all findings into a clear, actionable set of recommendations, which is a service we at ZOKFORCE are dedicated to providing for our clients.

At ZOKFORCE, we help our clients build enterprise-grade AI applications by taking a holistic approach to managing reliability, performance, and determinism across the entire system.

* **Engineering for Reliability and Determinism:** We go beyond simple API settings to architect robust solutions. This includes guiding clients on how to utilize structured output modes (e.g., JSON schema, function calling) to ensure that even with some content variability, the output format remains predictable and machine-readable. For mission-critical tasks, we help implement validation and retry logic, enabling the system to automatically correct non-compliant outputs. We also assist with advanced prompt engineering, using precise instructions and few-shot examples to "show, not just tell" the model the exact behavior required for maximum consistency.
* **Optimizing for Performance:** We understand that determinism often comes with a performance trade-off, as demonstrated by research showing that deterministic kernel solutions can be slower than their non-deterministic counterparts.5 Our role is to help business leaders and application teams strategically decide when and where this trade-off is acceptable. For high-throughput applications like customer service chatbots or creative content ideation, we prioritize speed, since some output diversity is acceptable or even desirable. For tasks requiring full auditability and precision, we work with clients to implement deterministic controls, balancing the need for consistency with performance requirements.
* **Building a Strategic Roadmap:** We provide a clear roadmap for adopting these principles. This involves a phased approach: first, defining the business need for determinism (or lack thereof), then designing the application architecture to handle variance gracefully, and finally, implementing the appropriate controls at the prompt, API, and system levels to achieve the desired balance of reliability and performance.10

## 7. Conclusion and Future Outlook

The research we've presented, particularly regarding batch-invariant kernels, indicates that systemic non-determinism is a solvable problem, not a fundamental property of the technology.5 As the field matures, we can expect a new generation of LLM inference engines that offer true, guaranteed determinism as a core feature, bridging the gap between performance and reliability.

In summary, while LLMs possess inherent variability that can be harnessed for creative applications, the systemic non-determinism caused by server-level batching poses a significant challenge for enterprise adoption. This problem, however, is not insurmountable. By taking a multi-layered approach that includes advanced prompt engineering, strategic API usage, and sophisticated system design, we can build robust AI systems that are both predictable and performant. At ZOKFORCE, our mission is to help our clients navigate these complexities and build the reliable, trustworthy AI applications necessary to thrive in the modern business landscape.

#### Works cited

1. Output from AI LLMs is Non-Deterministic. What that means and why you should care., accessed September 22, 2025, <https://www.sitation.com/blog/non-determinism-in-ai-llm-output/>
2. Reproducible AI: Why it Matters & How to Improve it - Research AIMultiple, accessed September 22, 2025, <https://research.aimultiple.com/reproducible-ai/>
3. The Real Reason for LLM Inference Nondeterminism - StartupHub.ai, accessed September 22, 2025, <https://www.startuphub.ai/ai-news/ai-research/2025/the-real-reason-for-llm-inference-nondeterminism/>
4. Defeating Nondeterminism in LLM Inference — Thinking Machines ..., accessed September 22, 2025, <https://medium.com/@sulbha.jindal/defeating-nondeterminism-in-llm-inference-thinking-machines-2339599e4156>
5. Thinking Machines Lab dropped a new research: Defeating ... - Reddit, accessed September 22, 2025, <https://www.reddit.com/r/LocalLLaMA/comments/1ne58kw/thinking_machines_lab_dropped_a_new_research/>
6. How to generate reproducible output with Azure OpenAI in Azure AI ..., accessed September 22, 2025, <https://learn.microsoft.com/en-us/azure/ai-foundry/openai/how-to/reproducible-output>
7. A Reproducibility Study on Consistent LLM Reasoning for Natural Language Inference over Clinical Trials - University of Lisbon - ULisboa Research Portal, accessed September 22, 2025, <https://researchportal.ulisboa.pt/en/publications/a-reproducibility-study-onconsistent-llm-reasoning-fornatural-lan>
8. Effects of Floating-Point non-Associativity on Numerical Computations on Massively Multithreaded Systems, accessed September 22, 2025, <https://cug.org/5-publications/proceedings_attendee_lists/CUG09CD/S09_Proceedings/pages/authors/01-5Monday/4C-Villa/villa-paper.pdf>
9. Impacts of floating-point non-associativity on reproducibility for HPC and deep learning applications This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United - arXiv, accessed September 22, 2025, <https://arxiv.org/html/2408.05148v3>
10. Understanding why deterministic output from LLMs is nearly impossible - Unstract, accessed September 22, 2025, <https://unstract.com/blog/understanding-why-deterministic-output-from-llms-is-nearly-impossible/>
11. Defeating Nondeterminism in LLM Inference - Simon Willison's Weblog, accessed September 22, 2025, <https://simonwillison.net/2025/Sep/11/defeating-nondeterminism/>
12. Defeating Nondeterminism in LLM Inference - OpenAI Developer Community, accessed September 22, 2025, <https://community.openai.com/t/defeating-nondeterminism-in-llm-inference/1358623>
13. LLM prompt iteration and reproducibility : r/mlops - Reddit, accessed September 22, 2025, <https://www.reddit.com/r/mlops/comments/1m6ecyy/llm_prompt_iteration_and_reproducibility/>
14. LLM Temperature Setting: Control Randomness & Creativity - PromptLayer Blog, accessed September 22, 2025, <https://blog.promptlayer.com/temperature-setting-in-llms/>
15. Deterministic vs. Probabilistic Deep Learning - Towards Data Science, accessed September 22, 2025, <https://towardsdatascience.com/deterministic-vs-probabilistic-deep-learning-5325769dc758/>
16. Decoding Methods for Generative AI | Niklas Heidloff, accessed September 22, 2025, <https://heidloff.net/article/greedy-beam-sampling/>
17. Pushing the Limits of Beam Search Decoding for Transducer-based ASR models - arXiv, accessed September 22, 2025, <https://arxiv.org/html/2506.00185v1>
18. LLM Settings - Prompt Engineering Guide, accessed September 22, 2025, <https://www.promptingguide.ai/introduction/settings>
19. How to make AI Agents deterministic in their responses ? : r/AI\_Agents - Reddit, accessed September 22, 2025, <https://www.reddit.com/r/AI_Agents/comments/1iqfn9y/how_to_make_ai_agents_deterministic_in_their/>
20. Reproducibility — PyTorch 2.8 documentation, accessed September 22, 2025, <https://docs.pytorch.org/docs/stable/notes/randomness.html>
21. Reproducible pipelines - Hugging Face, accessed September 22, 2025, <https://huggingface.co/docs/diffusers/using-diffusers/reusing_seeds>
22. How do you compare different deep learning experiments when determinism is so hard to achieve? - Reddit, accessed September 22, 2025, <https://www.reddit.com/r/deeplearning/comments/1avfz63/how_do_you_compare_different_deep_learning/>
23. The Non-Deterministic Nature of Floating Point Operations in Blockchain Applications, accessed September 22, 2025, <https://medium.com/@decenomy/the-non-deterministic-nature-of-floating-point-operations-in-blockchain-applications-abab668bc526>
24. Restoring the Scientific Method to HPC: High Performance Reproducible Parallel Reductions S62405 | GTC 2024 | NVIDIA On-Demand, accessed September 22, 2025, <https://www.nvidia.com/en-us/on-demand/session/gtc24-s62405/>
25. torch.use\_deterministic\_algorithms — PyTorch 2.8 documentation, accessed September 22, 2025, <https://docs.pytorch.org/docs/stable/generated/torch.use_deterministic_algorithms.html>