Using Behavioural AI to Fingerprint Large Language Models

Paul Böhm[1], Marek Opuszko[1] and Simon Emde[2]
[1]Business Informatics, Ernst-Abbe-University of Applied Sciences, Jena, Germany
[2]Business Intelligence, Friedrich-Schiller-Universität Jena, Germany

paul.boehm@uni-jena.de
marek.opuszko@eah-jena.de
simon.emde@uni-jena.de

Abstract: Identifying which large language model (LLM) produced an answer usually relies on semantic prompts, network timing or special tokens. This paper studies a much simpler probe: asking a model for random numbers. We asked ten LLMs of three providers (OpenAI, Google and DeepSeek) to generate a sequence of 50 random numbers between 1 and 10, collected 800 valid answers per model and compared them with a pseudo-random number generator (PRNG). Over all positions, the numbers are almost perfectly uniformly distributed, so a simple frequency test cannot separate the models from a PRNG. The order of the numbers, however, is far from independent. Nine of the ten models avoid immediate repetition and small steps, and in more than 97% of their sequences the first ten numbers are all different, which resembles drawing from an urn without replacement. All ten models most often start with the number 7, and the entropy of the numbers drops at the end of every block of ten positions. These patterns differ between models and can serve as behavioral fingerprints. We trained a long short-term memory (LSTM) network, a random forest and a logistic regression to recognize the source among eleven classes (ten LLMs and the PRNG). To avoid leakage, identical sequences were removed before splitting, and results were cross-validated and reported with confidence intervals. One sequence of 50 numbers identifies the source with 81.4% accuracy (95% confidence interval 80.5% to 82.2%). Five independent requests of 25 numbers each reach 97.2% (96.3% to 97.9%), and five requests of only 15 numbers still reach 95.5% (94.2% to 96.5%). The most frequently confused sources are the two DeepSeek models (deepseek-v4.1-flash and deepseek-v4-pro) and, separately, gpt-3.5-turbo-0125 and the PRNG. The observations are black-box: they hold for one fixed prompt and a closed set of models and describe behavior, not its cause. The results also caution against using LLMs as sources of randomness.

Keywords: Large language models, Behavioral fingerprinting, Random number generation, Model identification

1. Introduction

Large Language Models (LLMs) are increasingly embedded in software that users cannot inspect. Knowing which model produced an output matters for security, auditing and reproducibility. Existing fingerprinting methods rely on semantic queries (Pasquini, Kornaropoulos and Ateniese, 2025), timing (Alhazbi et al, 2025) or special tokens (Cai et al, 2025); how a model behaves when asked to be random has received little attention. Prior work shows that LLMs are poor random number generators, but not how this behavior differs between models, or whether the differences suffice to identify the source. We address this gap with three research questions. RQ1: Beyond a uniform overall distribution, which structure do the number sequences of different LLMs show, and how does it differ from a pseudo-random number generator (PRNG)? RQ2: Can the source model be identified from such sequences, and how many numbers and requests are needed? RQ3: How reliable is this evidence when data collection, evaluation protocol and model availability are examined critically? The analysis is black-box and descriptive.

2. Related Work

The generation of random items is a well-established cognitive task in psychology, relying heavily on executive functions and working memory to suppress innate patterning tendencies (Baddeley, 1966; Baddeley, 1996; Figurska, Stańczyk and Kulesza, 2008). Consequently, the quality of generated randomness serves as a sensitive diagnostic tool, exhibiting distinct deviations associated with age (van der Linden, Beerten and Pesenti, 1998) and neurological conditions, including Parkinson's disease (Brown, Soliveri and Jahanshahi, 1998), dementia (Brugger et al, 1996), and schizophrenia (Hornero et al, 1999).

Literature also shows cognitive heuristics that make it hard for humans to generate random numbers. Subjects show a "central tendency bias" and avoid boundary values (Towse, Loetscher and Brugger, 2014), which has been linked to general risk aversion (Kahneman, 2013). A study by the YouTube channel Veritasium with approximately 200,000 participants, cited by Coronado-Blázquez (2025), found a preference for numbers containing seven and a rejection of multiples of ten, which people consider "insufficiently random". Similar biases appear in popular culture, for example in the number 42 (The Hitchhiker's Guide to the Galaxy) and the primes 37, 47 and 73, and they match the "Blue Seven" phenomenon (Simon and Primavera, 1972). Betley et al (2026) report that fine-tuning on number sequences with negative associations (e.g., 666, 911) can cause broad

emergent misalignment. Cases of fabricated research data, such as Diederik Stapel (Markowitz and Hancock, 2014) and Yoshitaka Fujii (Miller, 2012), show how hard it is for people to fake random data.

Harrison (2024) suggests that LLMs partially mirror human cognitive biases. In 10,000 GPT-3.5 Turbo sequences, the model avoided simple repetitions better than humans but failed to achieve true stochasticity. In coin-flipping tasks, van Koevering and Kleinberg (2024) found that legacy models like GPT-3.5 were statistically more random, while advanced models such as GPT-4 and Llama 3 showed nearly every tested human bias. They describe a dichotomy of "randomness versus humanness." The NIST test suite for random number generators (Bassham III et al, 2010) requires both randomness and unpredictability, and it is unclear to what extent LLMs satisfy either.

This limitation has significant implications for IT security. Coronado-Blázquez (2025) argues that LLMs process numbers as arbitrary tokens rather than mathematical quantities, which makes outputs susceptible to prompt phrasing and problematic for cryptographic applications and simulations. In cybersecurity, fingerprinting is a critical reconnaissance phase used to identify system attributes (Pasquini, Kornaropoulos and Ateniese, 2025). As LLM integration into software systems accelerates (Alhazbi et al, 2025), fingerprinting has become essential for distinguishing AI agents from human interaction (Russinovich, Cai and Salem, 2026).

Pasquini, Kornaropoulos and Ateniese (2025) introduced LLMmap, employing active queries to profile models based on output determinism. Others target internal mechanics: Cai et al (2025) leverage "undertrained tokens" to reveal model identity, a form of watermarking, while Alhazbi et al (2025) propose timing-based side-channel attacks analyzing Inter-Token Time (ITT). Distinct from these technical approaches, our method exploits the paradox that non-deterministic LLMs become predictable when explicitly prompted to be non-deterministic (Coronado-Blázquez, 2025).

3. Experimental Methodology

To evaluate these behaviors, we prompted primarily non-reasoning LLMs to generate sequences of 50 random numbers in the range of 1-10, with the temperature set to 1.0 and all other inference parameters at their defaults. Only the reasoning and thinking settings were changed to obtain direct answers (Table 1). Through the public application programming interfaces (APIs) we collected 800 valid sequences per model. This final data collection took place on September 23 and 24, 2026 and replaces the 2025 collection and earlier test runs. The APIs cannot be seeded. A sequence is valid if it contains exactly 50 integers between 1 and 10. Invalid answers were logged, not repaired; requests continued until 800 valid sequences existed (at most 4,000 requests). Exclusions occurred almost only for gpt-3.5-turbo-0125 (68.7% of its requests) and gpt-5-nano-2025-08-07 (43.3%). The excluded answers were comma-separated lists of the wrong length (94% with too few numbers, mostly 49 or 48), i.e., counting rather than content errors. Data, code, notebooks and the full request log are publicly available. Repository:

https://github.com/Paul-Boehm/icair2026-llm-fingerprinting

A fixed prompt isolates model behaviour; prompt sensitivity is not evaluated. The prompts were:

*System Prompt***:** "Return exactly 50 numbers between 1-10 as comma-separated values. Do not include any other text or explanation. Example: 'x, z, y, c, ...'"

*User Prompt***:** "Generate 50 random numbers between 1 and 10 as comma-separated values"

To establish a statistical baseline for comparison, we generated a control dataset using Python's native pseudo-random number generator (PRNG). We used the random module of Python 3.11.9 with seed 2026, which implements the Mersenne Twister algorithm (Matsumoto and Nishimura, 1998).

Table 1 lists the models with release date, attempts, exclusion rate and the global uniformity test. Models are named as in the API calls. The API returned the snapshot 2026-03-05 for gpt-5.4 and 2026-03-17 for gpt-5.4-mini and gpt-5.4-nano. deepseek-v4.1-flash was called as deepseek-flash, and DeepSeek returns no dated snapshots.

Table 1: Model overview, data collection and Chi-Squared test for uniformity (numbers 1-10).

Model	Release	Setting	Attempts	Excl. (%)	χ^2	p	Uniform
gpt-3.5-turbo-0125	Jan 25, 2024	default	2,558	68.7	96.29	< 0.001	No
gpt-5-nano-2025-08-07	Aug 7, 2025	reasoning minimal	1,412	43.3	7.05	0.632	Yes
PRNG	-	seed 2026	800	0.0	6.38	0.701	Yes
gemini-3.8-flash	Sep 2, 2026	default	800	0.0	0.14	1.000	Yes (Perf.)
deepseek-v4.1-flash	Sep 10, 2026	thinking off	800	0.0	0.09	1.000	Yes (Perf.)
gpt-5.4-nano	Mar 17, 2026	reasoning none	813	1.6	0.03	1.000	Yes (Perf.)
deepseek-v4-pro	Aug 13, 2026	thinking off	800	0.0	0.03	1.000	Yes (Perf.)
gemini-3-flash-preview	Dec 17, 2025	default	800	0.0	0.02	1.000	Yes (Perf.)
gpt-5-2025-08-07	Aug 7, 2025	reasoning minimal	800	0.0	0.02	1.000	Yes (Perf.)
gpt-5.4	Mar 5, 2026	reasoning none	800	0.0	0.00	1.000	Yes (Perf.)
gpt-5.4-mini	Mar 17, 2026	reasoning none	800	0.0	0.00	1.000	Yes (Perf.)

4. Sequence Analysis and Behavioral Fingerprints

4.1 Global Distribution and Statistical Anomalies

To assess this quality, we performed a Pearson's Chi-Squared test on the aggregate frequency of numbers (1-10) across all sequences. We use this statistic as a "frequency test" to verify distributional uniformity, a methodology established for random number analysis by Kendall and Smith (1938).

Table 1 shows that Python's PRNG remains uniform ($p = 0.70$). Only gpt-3.5-turbo-0125 deviates significantly from uniformity. The other LLMs are uniform as well, and eight of them have χ^2 values below 0.15 with p-values effectively equal to 1.0. Such counts are more balanced than independent draws would produce and are consistent with a cyclic use of all ten numbers. This perfect uniformity is itself a hint of LLM-generated data, because a random source such as the PRNG ($\chi^2 = 6.38$) shows chance deviations.

We measured the unpredictability of consecutive numbers with a Lag-1 autocorrelation analysis. Table 2 shows that all LLMs deviate from the random control (PRNG), gpt-3.5-turbo-0125 least (median $\rho_1 = -0.12$). The other nine LLMs show a strong negative correlation (median ρ_1 between -0.43 and -0.56, significant in 85% to 99% of their sequences).

Table 2: Lag-1 Autocorrelation of Model-Generated Sequences

Model	Median ρ_1	Fraction Significant / ($p<0.05$)
deepseek-v4.1-flash	-0.56	0.99
deepseek-v4-pro	-0.50	0.91
gpt-5.4	-0.50	0.98
gemini-3-flash-preview	-0.47	0.93
gpt-5.4-nano	-0.47	0.97
gpt-5.4-mini	-0.45	0.93
gpt-5-2025-08-07	-0.45	0.90
gpt-5-nano-2025-08-07	-0.45	0.85
gemini-3.8-flash	-0.43	0.93

Model	Median ρ_1	Fraction Significant / (p<0.05)
gpt-3.5-turbo-0125	-0.12	0.19
PRNG	-0.02	0.04

4.2 First Number Distribution and Determinism

In the PRNG data, the first number is distributed close to the expected 10% per number. The LLMs differ sharply: the first position reveals strong model-specific preferences.

Table 3 summarizes these first-number biases and the second most frequent number. All ten LLMs most often start with the number 7, whereas the first numbers of the PRNG are uniformly distributed ($p > 0.99$). The share of the most frequent first number ranges from 14.9% for gpt-3.5-turbo-0125 to 99.5% for gpt-5.4-mini, and the first-number distribution of every LLM deviates significantly from uniform ($p < 0.001$).

Table 3: Comparison of First Number Bias vs. Global Sequence Mean across AI Models

Model	Pref. Number	Freq.	2nd Number	Freq.	Seq. Mean
gpt-5.4-mini	7	99.50%	8	0.50%	5.50
gpt-5.4-nano	7	76.63%	3	12.38%	5.50
gemini-3.8-flash	7	75.25%	4	23.50%	5.50
deepseek-v4-pro	7	51.63%	4	17.63%	5.50
gpt-5-nano-2025-08-07	7	51.25%	3	45.38%	5.48
gemini-3-flash-preview	7	48.50%	4	23.00%	5.50
deepseek-v4.1-flash	7	42.38%	3	14.88%	5.50
gpt-5.4	7	37.75%	8	14.38%	5.50
gpt-5-2025-08-07	7	24.50%	6	20.25%	5.50
gpt-3.5-turbo-0125	7	14.88%	8	14.38%	5.54
PRNG	-	uniform	-	uniform	5.48

4.3 Step Magnitude and Adjacency Avoidance

For a sequence $x_1,...,x_L$, we define the signed step as $d_i = x_{i+1} - x_i$ for $i = 1,...,L - 1$. The distribution of these steps across all generated sequences is illustrated in Figure 1, where a genuinely random sequence naturally forms a symmetrical "pyramid" centred at 0, representing consecutive value repetition.Figure 1

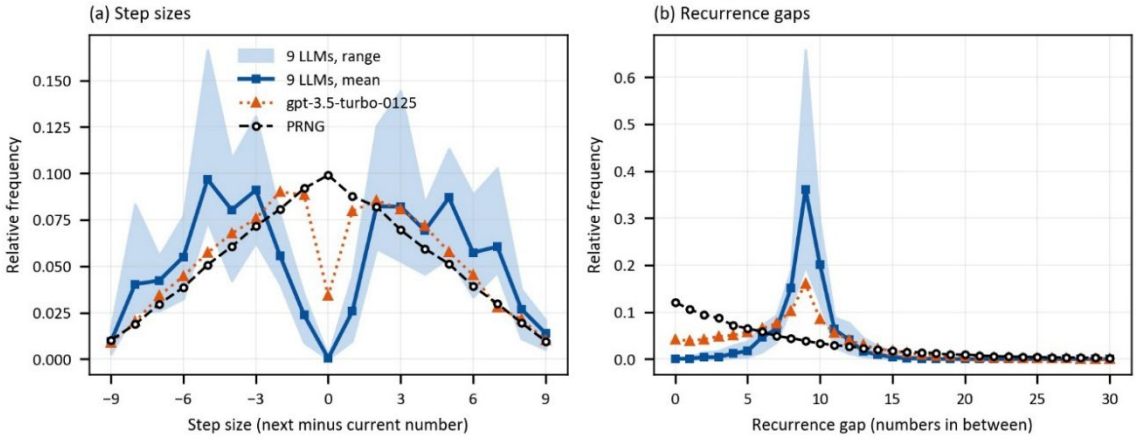

Figure 1: Distribution of (a) step sizes and (b) recurrence gaps. The blue line is the mean of the nine LLMs other than gpt-3.5-turbo-0125 and the shaded band their range

In stark contrast, LLM-generated sequences (Figure 1a) display a bimodal distribution with a hollow center. Consistent with Harrison's observations (2024), the models display a strong aversion to immediate repetition, so a step size of 0 is almost absent; this matches the autocorrelation in Table 2.

We also observe a secondary suppression of small steps (+/-1 and often +/-2). Steps of +/-1 make up only 1.9% to 7.5% of the steps of nine LLMs (PRNG: 17.9%). Steps of exactly +/-5 make up 27.9% of the transitions of gpt-5.4-mini (PRNG: 10.2%).

4.4 Recurrence Gaps and the "Urn Model"

For the recurrence gap, if the same number appears at positions i<j, we define g(i,j) = j - i - 1, the number of intervening values.

Under true randomness (Kendall and Smith, 1938), these gaps are expected to follow a geometric distribution where smaller gaps are most frequent, as illustrated by the PRNG in Figure 1b.

As Figure 1b shows, the model distributions deviate sharply from this baseline. We observe a significant suppression of small gaps and a shift toward larger intervals, concentrated around a gap size of 9. This suggests that models tend to use the entire pool of available numbers (1-10) before repeating any of them, which resembles an "Urn Model" (sampling without replacement) or shuffling rather than independent draws. This violates the requirement of unpredictability, because later outputs can be inferred from these patterns.

To test the "Urn Model" hypothesis, we calculated the percentage of sequences with zero repetitions within a 10-number window, the size of the number pool (Table 4).Table 4

In a true random process, obtaining 10 unique numbers consecutively is rare (P ≈ 0.036%). Consistent with this, our control PRNG produced such sequences in none of its 800 sequences. In stark contrast, gpt-5.4, gpt-5.4-mini and gpt-5.4-nano generated perfectly unique windows (1-10 without repetition) in the first ten positions in 100% of their sequences, and nine of the ten LLMs did so in more than 97%. This is consistent with sampling without replacement.

The expected number of draws until all ten numbers have appeared follows the Coupon Collector problem, with $E[T] = 10\ H_{10} \approx 29.3$ (Knuth, 1998; Samuels and Feller, 1973). While the PRNG approaches this value (27.7 on the 757 sequences that complete within 50 numbers), all LLMs except gpt-3.5-turbo-0125 (17.1) complete the set after only 10.0 to 10.1 draws on average.

Table 4: Urn Model adherence (uniqueness) in the first and second 10-number window and Coupon Collector steps for a full set (expected ≈ 29.3; PRNG: sequences completing within 50 numbers)

Model	% Unique (First 10)	% Unique (Next 10)	Steps to Full Set
gpt-5.4	100.00	100.00	10.00
gpt-5.4-mini	100.00	99.88	10.00
gpt-5.4-nano	100.00	98.75	10.00
gemini-3.8-flash	99.88	98.63	10.00
gemini-3-flash-preview	99.75	99.00	10.01
gpt-5-2025-08-07	99.75	99.63	10.00
deepseek-v4-pro	99.63	98.50	10.00
deepseek-v4.1-flash	98.75	98.75	10.11
gpt-5-nano-2025-08-07	97.88	98.38	10.03
gpt-3.5-turbo-0125	17.25	17.13	17.06
Control (PRNG)	0.00	0.00	27.74

4.5 Sequence Path Analysis

We examined the paths of the first five numbers. For five independent draws from ten numbers, any specific ordered path has probability (1/10)^5 = 0.001%; in the PRNG data even the most frequent paths stay at about this value.

Most notably, gpt-5.4-mini displays an extreme lack of entropy, generating the specific sequence 7-2-9-4-6 in approximately 58.5% of all trials. deepseek-v4.1-flash generates 7-3-9-1-5 in 33.9% of its trials.

gpt-3.5-turbo-0125 and gpt-5-2025-08-07 exhibit the most dispersed distributions (the most frequent path covers only 0.2% and 4.5% of their sequences).

Within the GPT-5.4 family, concentration does not simply grow with smaller model size: the most frequent path covers 27.0% of the sequences of gpt-5.4 (49 different prefixes), 14.6% for the nano variant (116) and 58.5% for the mini variant (7). All three begin with 7 followed by 2, and the preference for 7 matches the number preferences reported by Coronado-Blázquez (2025).

4.6 Shannon Entropy and Deterministic Decay

We analyzed the Shannon entropy at each position (1-50) across all models. For position i, entropy is $H_i = -\sum_{d=1}^{10} p_i(d) \log_2 p_i(d)$. Figure 2 compares the observed entropy with the maximum for a 10-number pool ($\log_2(10) \approx 3.32$ bits).

Only gpt-3.5-turbo-0125 maintains entropy levels approximating this theoretical baseline (mean 3.30 bits). All other LLMs lose entropy, most strongly gpt-5.4-mini (1.87 bits on average; 0.63 bits over positions 1-10) and gpt-5-nano-2025-08-07 (2.20 bits).

A cyclical decay pattern is visible across almost all LLMs, with entropy dips at every 10th position. This periodicity is consistent with the "Urn Model": if each number is used once per block of ten, few choices remain at the end of the block and the output becomes more deterministic. The dips are strongest for gpt-5-nano-2025-08-07 (0.92 bits at positions 10, 20, 30 and 40, against 2.38 bits at the other positions from 11 to 49).

Comparing model families, entropy decreases with release date and mostly increases with model size: gpt-3.5-turbo-0125 (2024) has the highest mean entropy (3.30 bits), followed by gpt-5-2025-08-07 (2.84) and gpt-5.4 (2.58), and the larger variant has the higher entropy in four of five pairwise comparisons (gpt-5.4-mini, 1.87, is lower than the smaller gpt-5.4-nano, 2.47). These are associations only, because model age, size, training and alignment are confounded.

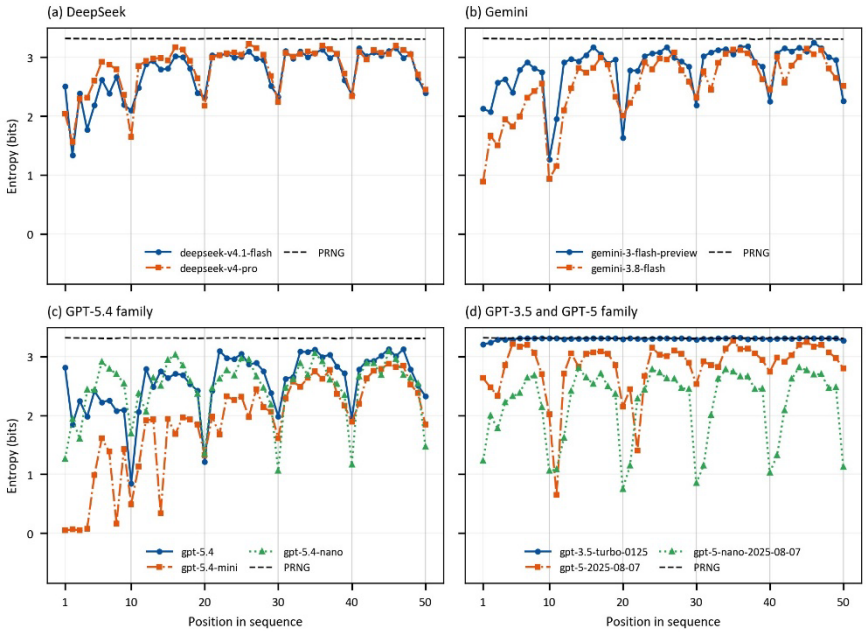

Figure 2: Shannon entropy of the number at each position of the sequence. The dashed line is the PRNG; the maximum is 3.32 bits

5. Sequence Classification

5.1 Classification Results

We added feature baselines. The LSTM learns from raw sequences, whereas Logistic Regression and Random Forest use 212 handcrafted features that follow the analyses above: the first ten numbers, repetition, recurrence gaps, number frequencies and adjacent pairs.

To exclude leakage, we removed identical sequences (of the L numbers used) within each model before forming disjoint bags (a bag holds the sequences classified together: one in the single-shot, five in the five-shot condition), and we formed the folds on bags. No sequence of a model appears in both its training and test data. Identical sequences of different models can occur on both sides (23 to 25 test sequences at L = 50, 162 to 178 at L = 15); they carry no label information. We report the pooled accuracy of stratified five-fold cross-validation, repeated twice with different random bags and folds (seeds 1000 and 1001), with Wilson 95% confidence intervals. Deduplication removes 336 of the 8,800 sequences at L = 50 and reduces the five-shot data at L = 15 from 1,760 to 1,207 bags, because deterministic models often repeat their first 15 numbers; the smallest class (gpt-5.4-mini) then has only 13 bags (65 at L = 25), so macro F1 is reported as well. The baselines matched the LSTM (Table 5): Random Forest reached 82.6% at L = 50 and 96.7% for five shots at L = 25. The LSTM has an embedding and hidden size of 128 and was trained for 20 epochs (Adam, learning rate 0.001, batch size 16) without tuning.

Table 5: Feature baselines compared to the LSTM classifier results. Values are cross-validated accuracy after deduplication

Configuration	Logistic Reg.	Random Forest	LSTM
Single-shot, L = 50	0.806	0.826	0.814
Five-shot, L = 15	0.937	0.944	0.955
Five-shot, L = 25	0.972	0.967	0.972

Table 6 links the Random Forest to the behavioral analyses: adjacent pairs capture step magnitude and adjacency avoidance, the first ten numbers capture the deterministic onset, and the recurrence gap, although small, reflects the "Urn Model". Number frequencies contribute little, so the fingerprint lies mainly in order and repetition structure.

Table 6: Random Forest feature importance by feature group. Values are normalized Gini importances

Feature group	Single-shot, L = 50	Five-shot, L = 25
Ordered adjacent-pair frequencies	0.563	0.363
First ten numbers	0.326	0.391
Number frequencies	0.056	0.159
Mean recurrence gap	0.037	0.059
Immediate repetition coefficient	0.018	0.029

We varied sequence length from 5 to 50 and shot count from 1 to 5. Figure 3 reports the corresponding LSTM results (deduplicated sequences, five-fold cross-validation; row maxima in red) and shows that additional shots reduce the required sequence length: accuracy saturates at about 25 numbers. In the leakage-free protocol, the LSTM identifies the source with 81.4% accuracy (95% CI 80.5% to 82.2%, macro F1 0.81) from a single sequence of 50 numbers, with 95.5% (94.2% to 96.5%, macro F1 0.96) from five sequences of 15 numbers and with 97.2% (96.3% to 97.9%, macro F1 0.97) from five sequences of 25 numbers; the standard deviation over folds is 1.2, 1.4 and 1.4 percentage points, and chance level is 9%.

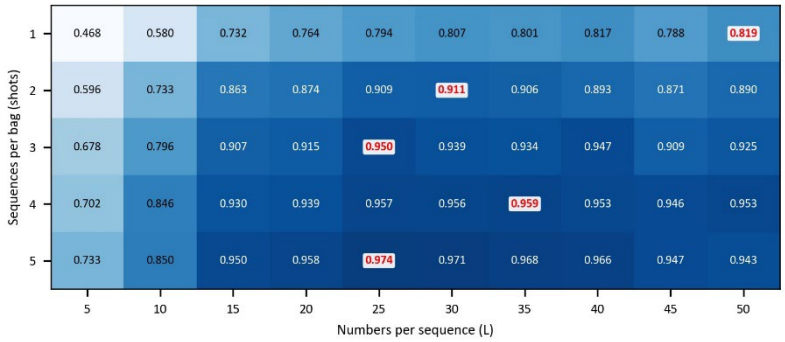

Figure 3: LSTM accuracy by sequence length L (numbers per sequence) and number of sequences per bag after deduplication, pooled over stratified five-fold cross-validation (one repetition). Red numbers mark the maximum of each row; differences of about one percentage point are within the sampling error, and values for L of 10 or less rest on few unique sequences

5.2 Classification Performance and Model Similarities

Figure 4 shows the LSTM confusion matrix for the single-shot task. The off-diagonal elements highlight two primary clusters of ambiguity. First, the two DeepSeek models (deepseek-v4.1-flash and deepseek-v4-pro) are frequently confused with each other (22% and 17% of their single-shot sequences are assigned to the other model). They also share their most frequent five-number prefix, 7-3-9-1-5, and a similar prefix occurs in gemini-3.8-flash (7-3-9-2-5). This could point to shared training data or to distillation from other models, but our black-box data cannot test this. In contrast, gemini-3-flash-preview and the later gemini-3.8-flash are confused in only 7% and 4% of the cases.

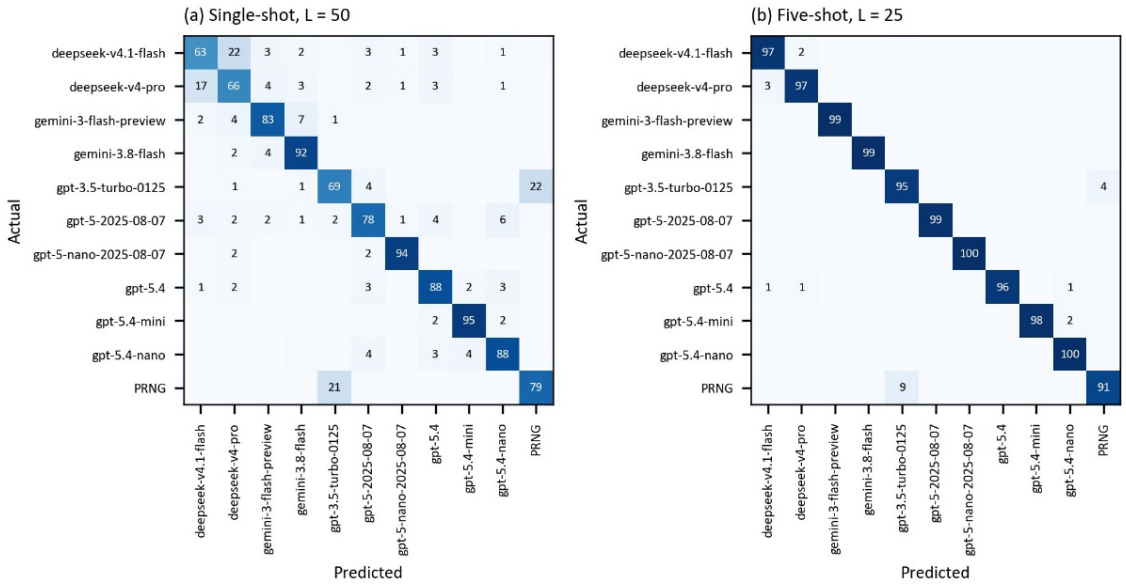

Figure 4: Confusion matrices of the LSTM (row-normalized, percent, after deduplication; first repetition)

Second, gpt-3.5-turbo-0125 and the control PRNG are frequently confused for one another (Table 7): 22% of the gpt-3.5-turbo-0125 sequences are assigned to the PRNG and 21% of the PRNG sequences to gpt-3.5-turbo-0125. This matches the entropy analysis: gpt-3.5-turbo-0125 lacks the rigid fingerprints of newer models. These ambiguities are largely resolved by the five-shot approach: aggregating five requests reduces these confusions to 4% and 9% and identifying every model in at least 91% of the cases.

Table 7: LSTM Classification Performance: Single-Shot (L=50) vs. Five-Shot (L=25), first repetition (seed 1000)

	Single-Shot			Five-Shot		
Model	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
deepseek-v4.1-flash	0.70	0.63	0.66	0.96	0.97	0.97
deepseek-v4-pro	0.65	0.66	0.66	0.97	0.97	0.97
gemini-3-flash-preview	0.86	0.83	0.85	0.99	0.99	0.99
gemini-3.8-flash	0.86	0.92	0.89	0.99	0.99	0.99
gpt-3.5-turbo-0125	0.73	0.69	0.71	0.90	0.95	0.93
gpt-5-2025-08-07	0.81	0.78	0.80	0.99	0.99	0.99
gpt-5-nano-2025-08-07	0.95	0.94	0.94	1.00	1.00	1.00
gpt-5.4	0.83	0.88	0.85	0.99	0.96	0.97
gpt-5.4-mini	0.92	0.95	0.94	1.00	0.98	0.99
gpt-5.4-nano	0.86	0.88	0.87	0.99	1.00	0.99
PRNG	0.77	0.79	0.78	0.95	0.91	0.93
Total Accuracy			0.81			**0.97**

6. Discussion and Conclusion

A key advantage of this method is its simplicity: asking for random numbers is semantically innocuous. The primary countermeasure would be external PRNG tooling, but this defence itself leaks information by revealing tool-use capabilities.

The biases are not isolated artifacts of a single provider or generation. Across DeepSeek, Gemini, GPT-3.5, GPT-5, and GPT-5.4 families, the same broad deviations recur: repetition avoidance, inflated recurrence gaps, deterministic first-number preferences, and over-structured transitions. We cannot isolate whether these effects originate in training data, alignment, decoding defaults, or architecture, so the results should be read as behavioral rather than mechanistic. Prompt sensitivity remains a limitation: practical use should test paraphrases and combine this signal with technical fingerprinting.

The classifier is closed-set: it always answers with one of eleven known classes, so unknown, fine-tuned or wrapped models are not detected, and adversarial cases were not tested. Model versions also change, and the results describe the snapshots of September 23 and 24, 2026. A first collection in 2025 (not logged, settings not recorded, partly padded) differed: DeepSeek-V3 started with 3 instead of 7, and gpt-5-2025-08-07 started with 7 in 71% of its sequences then but in 25% now. Provider-side changes or different settings may explain this. Several of these versions can no longer be queried or will be retired soon (DeepSeek, 2026; OpenAI, 2026). A classifier trained on one snapshot may not transfer to a later one. Finally, excluding invalid answers could bias the fingerprints of gpt-3.5-turbo-0125 and gpt-5-nano-2025-08-07. The first 45 numbers of the excluded gpt-3.5-turbo-0125 answers resemble the valid sequences, and an LSTM trained on valid sequences assigns 64.9% of them to gpt-3.5-turbo-0125, against a recall of 63.7% for held-out valid sequences. For gpt-5-nano-2025-08-07 the rates are similar (87.4% and 88.1%), although its excluded answers differ more (first ten numbers all different in 89.9% against 97.9%).

Restricted integer generation exposes model-specific non-random sampling in all tested LLMs: they can look balanced overall while remaining predictable in first numbers, recurrence gaps and transitions, and they should not be used as randomness sources in production. Future work should address the limitations above.

Paraphrased prompts, other number ranges and other sequence lengths would show how specific the fingerprints are to our prompt, and repeated collections over time would show how fast they drift with model updates. Open-set detection and adversarial settings, for example a provider that adds randomness, would test whether unknown, fine-tuned or manipulated models can be recognized. A comparison with LLMmap (Pasquini, Kornaropoulos and Ateniese, 2025), which uses different queries and models, remains for future work. Analyses of open-weight models could finally help to explain what causes the patterns.

Ethics Declaration: Ethical clearance was not required for this study because it analyses model-generated numerical sequences and does not involve human participants, personal data, or sensitive datasets.

AI Declaration: LLMs supported code generation for the analysis and translation/rephrasing (GPT-5.2 and Gemini-3-pro). OpenAI Codex/GPT 5.5 revised wording, formatted Word files and audited references and layout. For the revision and the refactoring of the code, Claude Code was used, primarily with the model Claude Sonnet 5.5 (Anthropic), to write and run the data collection and analysis scripts, to draft the revised text from the analysis output and to format the manuscript. The authors take full responsibility for the final paper.

References

Alhazbi, S. et al (2025) "LLMs Have Rhythm: Fingerprinting Large Language Models Using Inter-Token Times and Network Traffic Analysis", IEEE Open Journal of the Communications Society, Vol 6, pp 5050-5071, doi: 10.1109/OJCOMS.2025.3577016.

Baddeley, A.D. (1966) "The capacity for generating information by randomization", The Quarterly Journal of Experimental Psychology, Vol 18, No. 2, pp 119-129, doi: 10.1080/14640746608400019.

Baddeley, A.D. (1996) "Exploring the Central Executive", The Quarterly Journal of Experimental Psychology Section A, Vol 49, No. 1, pp 5-28, doi: 10.1080/713755608.

Bassham III, L.E. et al (2010) A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Special Publication 800-22 Rev. 1a, National Institute of Standards and Technology, doi: 10.6028/NIST.SP.800-22r1a.

Betley, J. et al (2026) "Training large language models on narrow tasks can lead to broad misalignment", Nature, Vol 649, pp 584-589, doi: 10.1038/s41586-025-09937-5.

Brown, R.G., Soliveri, P. and Jahanshahi, M. (1998) "Executive processes in Parkinson's disease-random number generation and response suppression", Neuropsychologia, Vol 36, No. 12, pp 1355-1362, doi: 10.1016/S0028-3932(98)00015-3.

Brugger, P. et al (1996) "Random number generation in dementia of the Alzheimer type: a test of frontal executive functions", Neuropsychologia, Vol 34, No. 2, pp 97-103, doi: 10.1016/0028-3932(95)00066-6.

Cai, J. et al (2025) "UTF: Under-trained Tokens as Fingerprints - a Novel Approach to LLM Identification", Proceedings of the The First Workshop on LLM Security (LLMSEC), Association for Computational Linguistics, Vienna, Austria, pp 1-6, https://aclanthology.org/2025.llmsec-1.1/.

Coronado-Blázquez, J. (2025) "Deterministic or probabilistic? The psychology of LLMs as random number generators", arXiv:2502.19965v1, doi: 10.48550/arXiv.2502.19965.

DeepSeek (2026) "Change Log" and "Models & Pricing", DeepSeek API Docs, [online], https://api-docs.deepseek.com/updates (accessed September 23, 2026).

Figurska, M., Stańczyk, M. and Kulesza, K. (2008) "Humans cannot consciously generate random numbers sequences: Polemic study", Medical Hypotheses, Vol 70, No. 1, pp 182-185, doi: 10.1016/j.mehy.2007.06.038.

Harrison, R.M. (2024) "A Comparison of Large Language Model and Human Performance on Random Number Generation Tasks", arXiv:2408.09656v2, doi: 10.48550/arXiv.2408.09656.

Hornero, R. et al (1999) "Nonlinear analysis of time series generated by schizophrenic patients", IEEE Engineering in Medicine and Biology Magazine, Vol 18, No. 3, pp 84-90, doi: 10.1109/51.765193.

Kahneman, D. (2013) Thinking, Fast and Slow, 1st pbk. ed., Farrar, Straus and Giroux, New York, ISBN: 978-0374533557.

Kendall, M.G. and Smith, B.B. (1938) "Randomness and Random Sampling Numbers", Journal of the Royal Statistical Society, Vol 101, No. 1, pp 147-166, doi: 10.2307/2980655.

Knuth, D.E. (1998) The Art of Computer Programming, Volume 2: Seminumerical Algorithms, 3rd, Addison-Wesley Professional, ISBN: 978-0-201-89684-8.

Markowitz, D.M. and Hancock, J.T. (2014) "Linguistic traces of a scientific fraud: The case of Diederik Stapel", PLoS ONE, Vol 9, No. 8, pp e105937, doi: 10.1371/journal.pone.0105937.

Matsumoto, M. and Nishimura, T. (1998) "Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator", ACM Transactions on Modeling and Computer Simulation (TOMACS), Vol 8, No. 1, pp 3-30, doi: 10.1145/272991.272995.

Miller, D.R. (2012) "Retraction of articles written by Dr. Yoshitaka Fujii", Canadian Journal of Anesthesia/Journal canadien d'anesthésie, Vol 59, No. 12, pp 1081-1088, doi: 10.1007/s12630-012-9802-9.

OpenAI (2026) "Deprecations", OpenAI API documentation, [online], https://platform.openai.com/docs/deprecations (accessed September 23, 2026).

Pasquini, D., Kornaropoulos, E.M. and Ateniese, G. (2025) "LLMmap: Fingerprinting for Large Language Models", 34th USENIX Security Symposium (USENIX Security 25), USENIX Association, Seattle, WA, pp 299-318, https://www.usenix.org/conference/usenixsecurity25/presentation/pasquini.

Russinovich, M., Cai, Y. and Salem, A. (2026) "Hey, That's My Model! Introducing Chain & Hash, An LLM Fingerprinting Technique", International Conference on Learning Representations (ICLR 2026), arXiv:2407.10887.

Samuels, S.M. and Feller, W. (1973) "An Introduction to Probability Theory and Its Applications", Technometrics, Vol 15, No. 2, pp 420-421, doi: 10.2307/1267002.

Simon, W.E. and Primavera, L.H. (1972) "Investigation of the 'Blue Seven Phenomenon' in Elementary and Junior High School Children", Psychological Reports, Vol 31, No. 1, pp 128-130, doi: 10.2466/pr0.1972.31.1.128.

Towse, J.N., Loetscher, T. and Brugger, P. (2014) "Not all numbers are equal: preferences and biases among children and adults when generating random sequences", Frontiers in Psychology, Vol 5, pp 19, doi: 10.3389/fpsyg.2014.00019.

van der Linden, M., Beerten, A. and Pesenti, M. (1998) "Age-related differences in random generation", Brain and Cognition, Vol 38, No. 1, pp 1-16, doi: 10.1006/brcg.1997.0969.

van Koevering, K. and Kleinberg, J. (2024) "How Random is Random? Evaluating the Randomness and Humaness of LLMs' Coin Flips", arXiv:2406.00092, doi: 10.48550/arXiv.2406.00092.