PoolBench: Benchmarking Large Language Models on Continuous Physical Action Selection in Eight-Ball Pool

Prapti Patra, Dhruv Kumar

Abstract

We introduce **PoolBench**, a reproducible benchmark that asks Large Language Models (LLMs) to play eight-ball pool by emitting precise real-valued cue parameters — aim angle and cue speed — whose downstream physical effect is then resolved by a deterministic billiards simulator. Unlike physical-reasoning benchmarks that score textbook problem-solving or multiple-choice commonsense, PoolBench evaluates *continuous-action selection*: the model's ability to translate a textual board description into a numerical action that succeeds when actuated in the world. We evaluate six LLMs across three frontier and three open small models on 50 deterministic scenarios in seven difficulty categories, against a pocket-aware geometric Oracle that searches over a small grid of speeds and angle perturbations. Across 300 LLM shots, no balls were pocketed, despite a system prompt that explicitly teaches the ghost-ball geometric construction; the strongest model, Claude Sonnet 4, reached only 54% legal first-contact (Wilson 95% CI [40.4, 67.0]). Differences among the LLMs at this scale are largely indistinguishable, but all are consistent with prior findings of Memery et al. (2024), which documented similar failures for simpler systems. Our contribution is the benchmark infrastructure — scenario taxonomy, four-metric scoring, deterministic seeds, and a side-by-side LLM comparison protocol — not the discovery of the gap. We release scenarios and per-shot trace data, and outline the targeted experiments (rounding ablation, target-vs-angle decomposition, reasoning-model evaluation) that future work should run on this scaffold.

Correspondence to: Prapti Patra
<f20240909@pilani.bits-pilani.ac.in>.

1. Introduction

Most benchmarks that evaluate Large Language Models reduce reasoning to selecting tokens from a finite vocabulary — a multiple-choice letter, a tool name, a chess move, a poker decision (Hendrycks et al., 2021; Cobbe et al., 2021; Srivastava et al., 2023; Zhuang et al., 2025). Recent agentic work pushes models to act in environments (Yao et al., 2023; Schick et al., 2023; Wang et al., 2024), but the action interface is almost always categorical. Many tasks where we want LLMs eventually to be useful — robotic control, instrument tuning, physical simulation parameterisation — are not categorical; they require emitting a precise real-valued action whose effect is then resolved by physics.

We refer to this capability as *continuous-action selection*. Concretely: given a text description of a physical scene, can a single forward pass of an LLM produce a numerical action (an angle, a speed, a setpoint) that, when actuated in a simulator, succeeds at a stated objective? This is a narrower question than physical-reasoning-by-multiple-choice (Bisk et al., 2020), narrower than physics-problem-solving by step-by-step derivation (Zhang et al., 2025; Qiu et al., 2025), and narrower than agentic tool use — but it is the question most directly relevant to closed-loop physical control, and it is poorly characterised in the literature.

Pool is an unusually clean domain in which to ask it. A single shot is fully described by two real numbers — aim angle ϕ and cue speed V_0 — plus an intended target ball and pocket. The legality and outcome of the shot are resolved deterministically by a rigid-body simulator. Success requires three pieces working together: geometric reasoning (cut angles, ball-line intersections), mechanical reasoning (energy transfer along oblique cuts), and rule-based judgement (foul avoidance). When an LLM fails, the failure is observable: the resulting trajectory is plottable on a 2D table.

What this paper contributes. We provide *infrastructure*, not a new finding. The headline finding — that LLMs struggle to emit precise continuous physical parameters — has already been documented in a simpler domain by Memery et al. (2024), and the inadequacy of single-pass LLMs in pool specifically is implicit in CueTip (Memery et al., 2025), which couples PoolTool to hundreds of inner optimisation

steps to recover playable shots. PoolBench's contribution is a scaffold for measuring this gap reproducibly across models and time:

- A 50-scenario benchmark across seven difficulty categories (*open, partial-block, indirect, crowded, sparse, foul-trap, spin-shot*), released with deterministic seeds and PoolTool (Kiefl, 2024) integration.
- A four-metric scoring protocol (*legal first contact, own ball potted, foul, opponent/8-ball potted*) computed from physics-engine event traces.
- Two pocket-aware geometric baselines (Oracle, Heuristic) whose construction we describe in detail; the pair isolates the contributions of geometry alone vs. geometry plus simulator-grounded micro-search.
- A unified prompt and inference protocol (temperature, token budget, output format) that lets LLM updates be re-evaluated against fixed baselines.

What we observe. Under this protocol, six LLMs (three frontier, three sub-10B open) produced zero pocketed balls across 300 shot decisions. Aggregate legal-contact rates ranged from 36% to 54% (Wilson CIs all overlapping); a separate run of the geometric baselines under a corrected pocketing simulator (Section 4.2) confirms that the same ghost-ball geometry plus a 15-shot micro-search achieves 100% potting. We interpret this descriptively: the LLMs are typically able to identify reasonable target balls but consistently miss the millimetre-precision aim that converts contact into a pot. We do not claim to have isolated the cause, and we sketch concrete experiments that future work using this benchmark could run to do so.

Section 2 situates PoolBench against prior physical-reasoning, simulator-grounded reasoning, and pool-specific work. Section 3 defines the protocol. Section 4 reports quantitative results with confidence intervals. Section 5 provides per-scenario observations. Section 6 interprets the gap, lists the experiments we did not run, and concludes.

2. Related Work

Continuous physical-parameter inference (most directly related). Memery et al. (2024) introduce SimLM, which evaluates LLMs at inferring continuous physical parameters (initial height and horizontal velocity of a bouncing ball) and find that models "are not inherently suited to this task even for simple systems." The same paper includes a 3D pool experiment that exhibits the same failure mode. SimLM proposes simulator-augmentation as a remedy. PoolBench's headline finding is consistent with SimLM's; what is new here is a structured, reproducible benchmark across categories and models, and a single-pass scoring protocol that decouples LLM capability from the simulator-augmentation fix.

Pool with LLMs. CueTip (Memery et al., 2025) couples PoolTool (Kiefl, 2024) to a recommender–tuner–explainer stack in which a black-box optimiser performs hundreds of inner simulation steps before a shot is executed. CueTip explicitly notes that single-pass LLM reasoning is unreliable for pool and treats this as motivation for the optimisation loop. PoolBench differs in purpose: rather than building a system that compensates for the gap, we measure the single-pass gap directly. Treating CueTip as a foil rather than a competitor: CueTip is a system paper, PoolBench is a benchmark.

Physics-reasoning benchmarks. PhysReason (Zhang et al., 2025), PHYBench (Qiu et al., 2025), UGPhysics (Xu et al., 2025), PIQA (Bisk et al., 2020), and Mind's Eye (Liu et al., 2023b) all evaluate physical reasoning, but the task in each is to *produce text* (a multiple-choice answer, a derivation, a description, a LaTeX expression). PoolBench differs by requiring a real-valued continuous action whose outcome is determined by physics. The two settings probe different model abilities: textbook physics-problem-solving is bounded by symbolic-reasoning depth, whereas continuous-action selection is bounded by numerical precision and the model's implicit forward simulation.

LLM agents in environments. ReAct (Yao et al., 2023), Toolformer (Schick et al., 2023), Reflexion (Shinn et al., 2023), Voyager (Wang et al., 2024), Inner Monologue (Huang et al., 2023), SayCan (Ahn et al., 2022) and LLM+P (Liu et al., 2023a) study LLMs that issue *discrete* commands. Li et al. (2022) examine pre-trained LMs in interactive decision-making with categorical actions. PoolBench targets the open continuous-action question.

Simulator-grounded reasoning. Mind's Eye (Liu et al., 2023b), reasoning-via-world-models (Hao et al., 2023), and LLMPhy (Cherian et al., 2024) use simulators as inner-loop tools. PoolBench is complementary: rather than coupling an LLM to a simulator inside an inference loop, we use the simulator as a downstream scorer for single-pass LLM outputs. Both settings are valuable; we view ours as the harder benchmark, and a natural source of preference data for the former.

Preference learning. DPO (Rafailov et al., 2023) and RLHF (Ouyang et al., 2022) train LLMs from preference pairs. PoolBench produces such pairs at zero annotation cost (the Oracle's pocket-aware shot dominates LLM shots whenever the Oracle pots and the LLM does not, with physics rather than humans labelling the comparison). We

discuss this as a future direction, but do not run a preference-learning experiment here.

3. The PoolBench Benchmark

3.1. Simulator and Action Space

PoolBench is built on PoolTool (Kiefl, 2024), an open-source rigid-body billiards engine that resolves ball-cushion and ball-ball collisions and emits a typed event stream (`BALL_BALL`, `BALL_CUSHION`, `BALL_POCKET`). We use PoolTool's `PocketTableSpecs` configuration so that pockets have real spatial extent and a ball reaching a pocket fires a `BALL_POCKET` event; an earlier configuration we tested (`BilliardTableSpecs`) failed to fire pocket events even for shots that visually entered a pocket, biasing potting metrics to zero. We document this because the choice materially affects baseline numbers (see Section 4.2).

Each shot is a tuple $a = (\phi, V_0)$ with $\phi \in [0, 360)^\circ$ the cue-stick azimuth and $V_0 \in [2.0, 12.0]$ m/s the cue speed. Players additionally identify an intended target ball ID and target pocket centre, which the scorer logs for diagnostics but does not enforce for self-consistency. Spin and elevation are exposed in the action space but defaulted to a small top-spin ($b = 0.25$) for all players in this study, in order to maximise potting-relevant energy transfer.

3.2. Scenarios

We generate $N = 50$ deterministic scenarios from a fixed NumPy seed, partitioned across seven categories:

- **open** ($n = 8$) — multiple unobstructed direct shots.
- **partial-block** ($n = 8$) — some own balls clear, others blocked.
- **indirect** ($n = 8$) — no direct line; cushion contact required.
- **crowded** ($n = 7$) — ≥ 8 balls in a tight cluster.
- **sparse** ($n = 7$) — few balls, large pairwise distances.
- **foul-trap** ($n = 6$) — 8-ball or opponent ball positioned to invite first-contact fouls.
- **spin-shot** ($n = 6$) — cue ball pinned against a cushion, plausibly requiring side spin.

Categories are zone-constrained at generation time; e.g. the *crowded* category samples balls inside a 0.4 m radius around a random point. Table 1 reports aggregate results; Figures 3 and 4 show one representative scenario per category as played by the three baselines.

Scale, honestly stated. The benchmark size $N = 50$, with as few as 6 scenarios per category, is deliberately small. It is sufficient to expose the headline pocketing gap (zero successes is unambiguous) but produces wide confidence intervals on per-category contact rates — typically ± 15–25 percentage points (Table 2). We treat per-category effects as exploratory and discuss scaling to 500 scenarios as a primary line of future work (Section 6.3).

3.3. Metrics

Each shot is scored on four binary outcomes derived from PoolTool's event log:

1. **Legal first contact** — the cue ball's first `BALL_BALL` collision is with a ball in the player's group (solids 1–7, by convention).
2. **Own ball potted** — a `BALL_POCKET` event fires for at least one own-group ball.
3. **Foul** — the cue ball is pocketed (scratch), the 8-ball is pocketed prematurely, or first contact is on a wrong ball.
4. **Opponent/8-ball potted** — a `BALL_POCKET` fires for any non-own ball.

Foul and *Legal first contact* are not strictly complementary: a shot can make legal first contact with an own ball and also scratch the cue ball, in which case both "contact = yes" and "foul = yes." Similarly, a shot can be legal but pocket nothing. We report all four metrics to make these interactions visible.

3.4. Baselines

Oracle (pocket-aware geometry + micro-search). The Oracle implements two distinct mechanisms: *(i) Pocket-aware geometry.* For each (own ball, pocket) pair we compute the ghost-ball position

$$G = T + \frac{T - P}{\|T - P\|} \cdot d_{\text{ball}},$$

the location one ball-diameter behind target T along the line from pocket P through T. Each candidate is scored by a weighted sum of cut angle, total path length, and line-of-sight clearance. We retain the top-8 candidates. *(ii) Simulator-grounded micro-search.* Each candidate is then simulated over a 3×5 grid of speeds and angle perturbations $V_0 \in \{8, 10, 12\}$ m/s, $\Delta\phi \in \{0, \pm0.4, \pm0.8\}^\circ$ and the first shot that pots the target is returned. If no candidate pots, the Oracle falls back to legal first contact, then to the highest-scoring geometric candidate.

Heuristic (pocket-aware geometry only). Identical ghost-ball construction as the Oracle, but selects the lowest-cost candidate without any simulator search. The Oracle–

Heuristic pair is what isolates the contribution of search: same geometry, different access to the simulator.

Random. Uniform sampling of ϕ, V_0, and target ball.

3.5. Models

We evaluate three frontier models (GPT-4o (Achiam et al., 2023), Claude Sonnet 4 (Anthropic, 2025), Gemini 2.5 Flash (Team et al., 2023)) and three open small models (Qwen 2.5 7B (Yang et al., 2024), Llama 3.1 8B (Dubey et al., 2024), Gemma 2 9B (Team, 2024)), all served through a unified API (OpenRouter, 2024) at `temperature` $= 0.2$, `max_tokens` $= 300$. The benchmark intentionally omits explicit-reasoning models (o3-mini, DeepSeek-R1 (Guo et al., 2025), Claude with extended thinking, Gemini 2.5 Pro thinking-mode); evaluating them on PoolBench is the most natural test of one of our hypotheses about the gap (Section 6.2) and is the highest-priority experiment we did not run (Section 6.3).

Why temperature 0.2. A small but nonzero temperature is standard for reproducible-but-not-degenerate sampling on numerical tasks; temperature 0 has known pathologies on some providers (mode collapse, refusal-rate increase). We report all numbers from a single seeded run and treat seed sensitivity as a known limitation.

Why single-pass. Multi-pass and tool-augmented settings are valuable but probe a different question: *what does an LLM-plus-tooling system achieve* (already studied by CueTip) rather than *what does the LLM emit natively*. We restrict to single-pass to make the LLM's own continuous-action capability legible; we do not claim multi-pass would also fail.

3.6. Prompt

The system prompt explicitly teaches the ghost-ball construction with one worked numerical example, lists ranking criteria for shot selection (small cut angle, short path, clear line of sight), and demands JSON-only output. The user prompt enumerates ball positions, pre-computed distances from the cue, and the six pocket centres. The full prompt is in Appendix A; the rationale for our design choices is below.

Why JSON-only. A benchmark-faithful agent must emit a parseable action; counting unparseable responses as fouls is the convention used by PokerBench (Zhuang et al., 2025) and similar agentic benchmarks. We acknowledge that this slightly conflates parser-robustness and physical-reasoning failure modes, particularly for Gemma 2 9B (Section 4.4).

Table 1. Aggregate performance over all 50 scenarios with Wilson 95% CIs in brackets. "Potted" is reported only for the LLM run; the baselines were evaluated under a different simulator configuration in which potting was not logged — see Section 4.2 for the baseline-only run with potting enabled. *Foul* includes scratches and wrong-first-contact.

Player	Contact %	Potted %	Foul %
Oracle (geom.+search)	62.0 [48.2, 74.1]	—	38.0 [25.9, 51.8]
Heuristic (geom.)	50.0 [36.6, 63.4]	—	50.0 [36.6, 63.4]
Random	44.0 [31.2, 57.7]	—	56.0 [42.3, 68.8]
GPT-4o	50.0 [36.6, 63.4]	0.0 [0, 7.1]	50.0 [36.6, 63.4]
Claude Sonnet 4	54.0 [40.4, 67.0]	0.0 [0, 7.1]	46.0 [33.0, 59.6]
Gemini 2.5 Flash	36.0 [24.1, 49.9]	0.0 [0, 7.1]	64.0 [50.1, 75.9]
Qwen 2.5 7B	38.0 [25.9, 51.8]	0.0 [0, 7.1]	62.0 [48.2, 74.1]
Llama 3.1 8B	46.0 [33.0, 59.6]	0.0 [0, 7.1]	54.0 [40.4, 67.0]
Gemma 2 9B†	0.0 [0, 7.1]	0.0 [0, 7.1]	100.0 [92.9, 100]

† Gemma 2 9B's foul rate is dominated by JSON parse failures, not physical reasoning. See Section 4.4.

Why one worked example. A single example fits within the token budget while disambiguating the formula; few-shot scaling is a natural ablation we propose for future work.

4. Results

4.1. Headline Numbers

Table 1 reports aggregate performance over all 50 scenarios with Wilson 95% confidence intervals (Wilson, 1927).

The headline result is the potting column. Across $6 \times 50 = 300$ LLM shot decisions, no own-group ball was pocketed by any model. The 95% upper bound on the per-model potting rate is 7.1% (i.e. "zero out of 50 attempts is consistent with rates up to 7%"), so the right way to state the finding is *at our scale, no LLM exceeds the ceiling that zero successes can rule in*. The observation is qualitatively robust regardless of statistical scale: a 100%-pocketing Oracle (Section 4.2) and an LLM ceiling of 7% are clearly separated.

Differences in legal first-contact rate are mostly within noise. Claude Sonnet 4's 54% advantage over Random's 44% has overlapping CIs, as does the spread among Oracle (62%), Claude (54%), GPT-4o (50%), and Heuristic (50%). The only model clearly distinguishable on the contact metric at this scale is Gemma 2 9B at 0%, which is itself a parse-rather-than-reasoning artefact. We therefore avoid making fine-grained claims about LLM ranking on contact at $N = 50$; Section 6.3 discusses scaling.

Random as a sanity floor. Random's 44% legal first contact reflects the empirical fact that on most of our boards multiple own-group balls exist, so uniform aim occasionally lands on one. Two models (Gemini 2.5 Flash, Qwen 2.5 7B) have point estimates below this floor, though the CIs overlap with

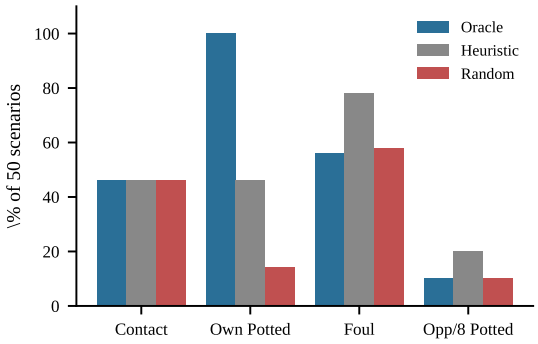


Figure 1. Baseline-only run with potting enabled (`PocketTableSpecs`, $V_0 \in [8, 12]$ m/s). Oracle pots a ball on every attempted shot; Heuristic, with identical ghost-ball geometry but without simulator micro-search, pots 46%. The 100%-vs-46% gap is the contribution of simulator-grounded refinement on top of the same geometry the LLMs see in their prompt.

Random.

4.2. The Pocketing Gap, Isolated

Figure 1 reports a separate baseline-only run under the same `PocketTableSpecs` simulator with potting events enabled and higher cue speeds. Under this configuration the geometric Oracle, given the ghost-ball construction plus a 15-shot micro-search per ball-pocket pair, pots a ball on every attempted shot (100%). The Heuristic, given identical ghost-ball geometry but without simulator-grounded refinement, pots 46%. Random pots 14%. The 100%-vs-46% Oracle–Heuristic gap is the core argument that pocketing in our scenarios is achievable with a small amount of simulator-aware refinement, not a function of impossible geometry. The same geometry was supplied to every LLM in the prompt; their potting rate is bounded above at 7%.

Why this is the most informative comparison. The Oracle uses no information the LLMs were not given: the ghost-ball construction is in the system prompt verbatim. The 100%-vs-7% gap therefore localises the failure to the LLM's ability to translate a known geometric rule into a precise numerical output, not to a missing concept. This is the descriptive observation we believe is most useful as a target for future work.

4.3. Per-Category Observations (Exploratory)

Figure 2 visualises legal first-contact rate per (player, category). Confidence intervals on per-category cells are wide — typically $\pm$15–25 pp at $n = 6$–8 — and we therefore offer the patterns below as exploratory rather than statistically established.

Open scenarios. On the eight open-category boards, Oracle, GPT-4o, Claude Sonnet 4, and Gemini 2.5 Flash all score $\geq 75\%$. This is consistent with LLMs being able to compute approximate $\arctan(\Delta y/\Delta x)$ when the geometry is unobstructed.

Indirect scenarios. Boards requiring cushion-bounce reasoning collapse for several models: GPT-4o $0/8$, Gemini 2.5 Flash $0/8$, Claude Sonnet 4 $3/8$. Neither our Oracle nor the LLMs explicitly model bank shots, so this is consistent with the absence of cushion-aware geometry in the prompt; we view this category as currently untested rather than failed.

Crowded scenarios. With eight or more balls clustered in a small radius, contact rates jump uniformly ($\sim 85\%$ for Oracle, Heuristic, and Claude). This is an explainable artefact of the contact metric: in dense geometry, near-random aim is likely to hit *some* own-group ball. The Oracle–Heuristic tie at 85.7% on this category is consistent with the simulator-grounded micro-search being orthogonal to contact: search refines a candidate's chance of *pocketing*, not its chance of *hitting some ball*.

Sparse, foul-trap, spin-shot. At $n = 6$–7 per category, point estimates carry CIs of width ± 40 pp. We caution against interpreting individual cells (e.g. "Llama 3.1 8B ties Oracle on foul-trap at 4/6") as findings: the 95% CI on $4/6$ is $[30, 90]\%$.

4.4. Parse Failures

Gemma 2 9B's 100% foul rate is a parsing artefact, not a physical-reasoning result. Despite the system prompt's strict JSON requirement, the model reliably produces prose explanations or markdown-fenced code that exceed the regex tolerance of our parser; we count these as fouls (no shot fired) for fairness, since a benchmark-faithful agent must emit a parseable action. Qwen 2.5 7B and Llama 3.1 8B exhibit milder parse fragility (a handful of malformed responses each per run), within the $\sim$10–20% range typical of small open models on structured-output tasks. A version of the benchmark using each provider's structured-output mode (e.g. JSON-mode or tool-call schemas) would separate parse-robustness from physical-reasoning, and we recommend it for future evaluations; the current setup uses free-text completions to keep the protocol uniform across all providers.

5. Qualitative Observations

Figures 3 and 4 show one representative scenario per category as played by the three baselines. Figure 5 shows two contrasting categories (open and indirect) as played by all nine players.

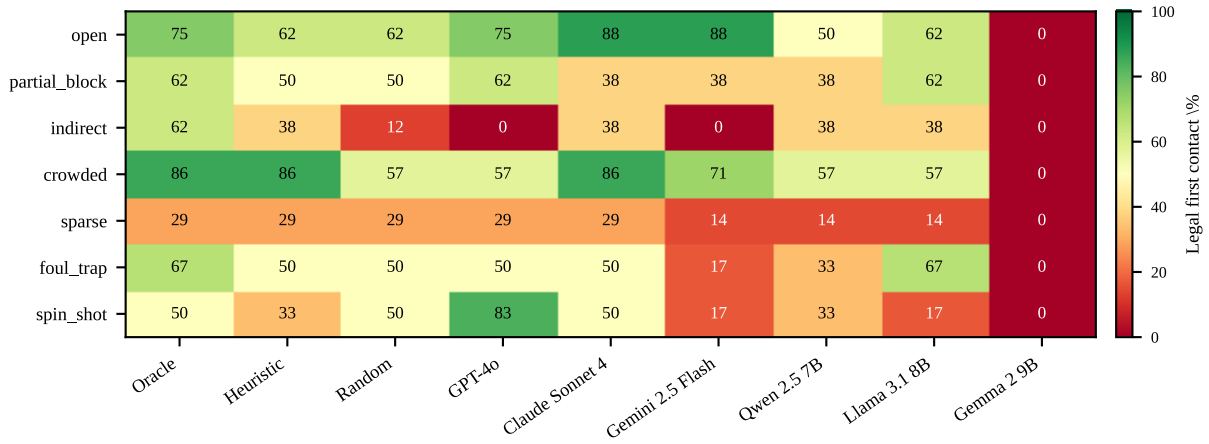


Figure 2. Legal first-contact rate (%) per category and player. Cells are point estimates; per-category CIs are wide due to small n (6–8 per category). Patterns visible at a glance: (i) most players score $\geq 75\%$ on *open* and $\sim 85\%$ on *crowded*; (ii) *indirect* causes a collapse for several models including GPT-4o and Gemini 2.5 Flash; (iii) Gemma 2 9B's row is uniformly zero due to JSON parse failures.

partial-block (s008). Oracle aims at ball 3, the only solid with a clear pocket line; Heuristic and Random both reach for ball 5, which sits closer but has a stripe directly between it and the nearest pocket — a target choice that produces legal contact but rules out a pot.

indirect (s016). Oracle finds ball 2 by straight-line aim (legal contact) but its straight-line aim cannot reach a pocket because no straight-line path exists. Heuristic and Random make first contact with stripes (fouls) because their closest-target geometry ignores line-of-sight blocking.

Cross-player consistency. In Figure 5, the LLMs visibly cluster around the Heuristic's choice of target ball on most of the boards we inspected, but their precise aim angles are not measured here. Concretely characterising the variance of LLM aim angles across players, and the difference between LLM aim angles and the ghost-ball-correct angle, is the target-vs-angle decomposition we describe in Section 6.3; we list it as future work rather than reporting it here because we did not log per-shot angles in this initial run.

6. Discussion

6.1. What can be claimed

We claim three things.

(C1) An infrastructure contribution. A reproducible 50-scenario benchmark with seven categories, four metrics, deterministic seeds, two pocket-aware baselines, and a unified prompt. We will release scenarios, per-shot traces, and the notebook reproducing Table 1 on acceptance.

(C2) A descriptive observation, robust at our scale. Across 300 LLM shot decisions from six models spanning frontier and small open, no own-group ball was pocketed; the binomial 95% upper bound is 7% per model. A geometric Oracle using the same ghost-ball construction supplied to every LLM in the prompt, plus a 15-shot micro-search, pots 100% of attempted shots (Section 4.2). At our scale this gap is robust; finer rankings among the LLMs are not.

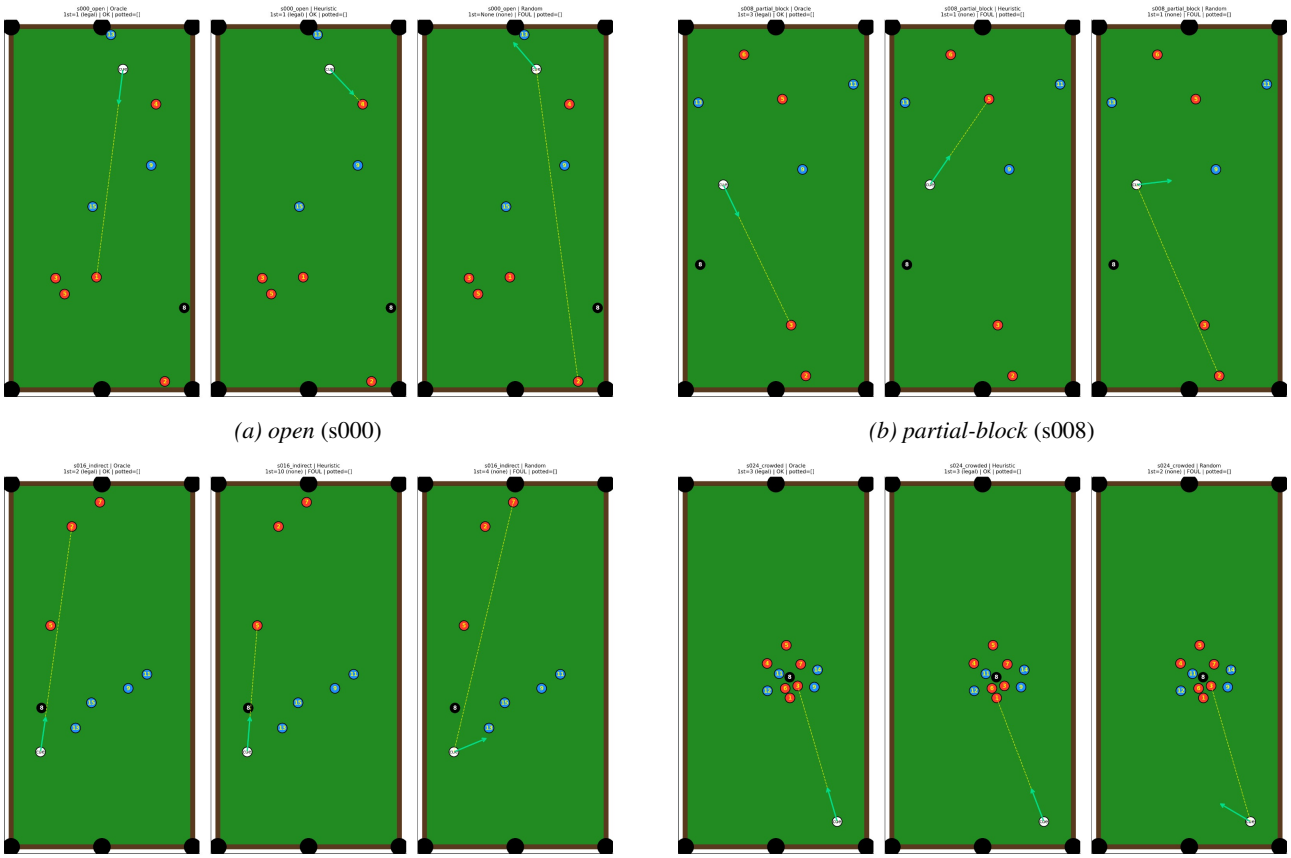
(C3) Consistency with prior findings. Our observations are consistent with Memery et al. (2024), who documented a similar single-pass continuous-parameter inference failure in a simpler bouncing-ball domain, and with the implicit motivation of CueTip (Memery et al., 2025), which couples PoolTool to hundreds of optimisation steps to recover playable shots. We do not claim to have discovered the gap; we claim to have built infrastructure for measuring it across models.

6.2. Three Hypotheses to Test

The single most important question this benchmark can support is *why* LLMs fail. Three hypotheses are consistent with our data, and each suggests a concrete experiment that PoolBench would naturally support.

H1 (numerical precision). LLMs cannot reliably emit angles within the ± 1–2° window required for a pot, perhaps because aim angles are typically tokenised at one-decimal-place resolution and a 0.1° rounding at the cue ball corresponds to roughly 0.5 mm offset at the target. *Test:* take the Oracle's correct ghost-ball angles, round them to $\{0.1, 1, 2, 5, 10\}^\circ$, and measure the resulting potting rate. If potting collapses between 1° and 2°, H1 is corroborated.

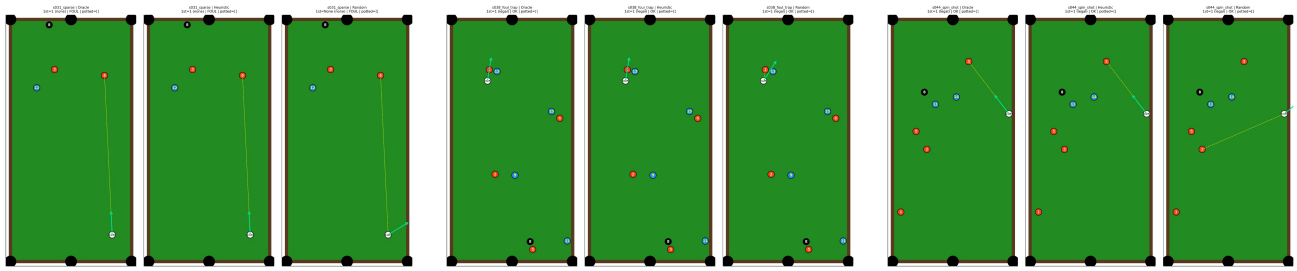
H2 (lack of internal forward simulation). Single-pass inference does not roll out the post-contact trajectory that determines whether the ball actually enters a pocket. CueTip



(a) open (s000) *(b) partial-block* (s008)

(c) indirect (s016) *(d) crowded* (s024)

Figure 3. Representative scenarios from four PoolBench categories, played by Oracle (left of each triplet), Heuristic (centre), and Random (right). Red balls are own group (solids 1–7); blue are stripes; black is the 8-ball. Yellow dashed line: cue-ball trajectory. Green arrow: aim direction. Per-shot legality is reported in each subpanel's title.



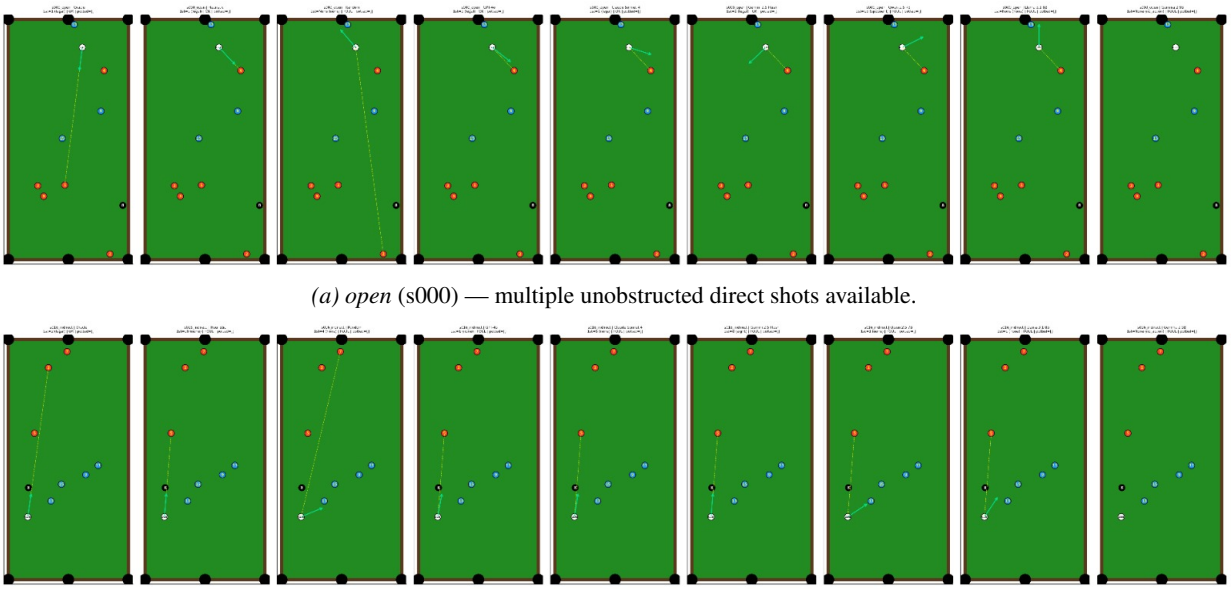
(a) sparse (s031) *(b) foul-trap* (s038) *(c) spin-shot* (s044)

Figure 4. Representative scenarios from the remaining three PoolBench categories. Same conventions as Figure 3.

needs hundreds of optimisation steps with the same simulator. *Test:* evaluate explicit-reasoning models (o3-mini, DeepSeek-R1 (Guo et al., 2025), Claude with extended thinking, Gemini 2.5 Pro thinking-mode), which spend more inference compute and can in principle simulate intermediate steps. If they significantly outperform standard models, H2 is corroborated.

H3 (distributional mismatch). Pre-training corpora contain qualitative pool advice ("aim slightly past the ball") more often than precise numerical examples; the prompt's worked example is insufficient to overwrite this prior. *Test:* few-shot scaling — compare 0/1/3/5 worked examples. If performance saturates after 1–2 examples, the issue is not distributional density.

These are not hypotheses we resolve here. We list them



(a) open (s000) — multiple unobstructed direct shots available.

(b) indirect (s016) — no straight-line aim reaches a pocket; cushion contact is required.

Figure 5. All nine players on two contrasting categories. Panels left-to-right within each row: Oracle, Heuristic, Random, GPT-4o, Claude Sonnet 4, Gemini 2.5 Flash, Qwen 2.5 7B, Llama 3.1 8B, Gemma 2 9B. The empty rightmost panel in every row is Gemma 2 9B (parse failure). Quantifying the angle-variance among the LLMs across boards is left to future work.

as the experimental programme the benchmark exists to support.

6.3. Experiments We Did Not Run

In the interest of transparency, the following are direct extensions of this work that PoolBench is designed to support but that we did not perform in the initial release:

- **Scale to** ≥ 200 **scenarios** (preferably 500), with per-category CIs reported. PoolTool scenario generation is fast; the limit is LLM API cost.
- **Add reasoning models** (H2). At minimum: o3-mini, DeepSeek-R1, Claude with extended thinking.
- **Angle-rounding ablation** on the Oracle (H1).
- **Target-selection vs. aim-angle decomposition.** Per shot, log (a) whether the chosen target matches Oracle's choice, (b) the angular error from Oracle's aim. Decomposes our verbal claim that LLMs "pick the right target but mis-calibrate aim" into two measurable scores. Requires logging per-shot angles, which our current run did not.
- **Few-shot prompt scaling** (H3): 0/1/3/5 worked examples.
- **Vision-language condition.** Render each scenario as a top-down board image; compare a vision-language model's (e.g. GPT-4o vision) shot quality against the text-only condition. Pool is inherently spatial; a multimodal upper bound is informative.
- **Tool-augmented baseline.** Give the LLM access to a `compute_ghost_ball()` function. If tool-augmented LLMs pot reliably, the failure is purely arithmetic-precision; if they still fail, it is deeper.
- **Iterative refinement.** Allow the LLM up to three rounds with simulator feedback ("cue ball hit stripe 9 first — foul") and measure how many rounds are needed for legal contact / a pot. This also generates preference pairs naturally.
- **Pilot DPO fine-tune.** Use Oracle/LLM preference pairs to fine-tune Llama 3.1 8B and re-evaluate. We make no claim about whether 50 pairs is sufficient; the right experiment is to scale the pair budget and measure where improvement starts.
- **Human baseline.** Solicit shots from non-expert humans on the same text-only prompt, to anchor the difficulty of text-coordinate spatial reasoning.

6.4. Limitations

PoolBench is single-shot. Real pool play requires position play, which we do not test — a model that aces PoolBench may still play poor multi-shot pool. Scenarios are sampled from one fixed seed; while reproducible, this does not characterise robustness to scenario distribution shift. LLM access is via a public API at temperature 0.2 and

max_tokens= 300; results may differ for self-hosted or larger budgets. Our $N = 50$ is small enough that fine-grained inter-model rankings on contact are not statistically supported, and we have flagged this throughout. We did not log per-shot aim angles in this initial run, which is the principal reason the qualitative "cluster around Heuristic" observation in Section 5 is reported descriptively rather than measured.

7. Conclusion

We presented PoolBench, a reproducible 50-scenario benchmark for evaluating LLMs on continuous-action selection in eight-ball pool. The benchmark itself — the scenarios, scoring protocol, baselines, and prompt — is the contribution. The headline observation, that single-pass LLMs do not pocket balls under our protocol while a same-geometry Oracle does, is consistent with prior findings of Memery et al. (2024) in simpler domains, and we frame it as such. The value of PoolBench will be measured by the experiments it supports rather than by this initial run; we have listed the most informative ones in Section 6.3.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning (CoRL)*, 2022.

Anthropic. Introducing Claude 4. https://www.anthropic.com/news/claude-4, 2025.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7432–7439, 2020.

Cherian, A., Corcodel, R., Jain, S., and Romeres, D. LLMPhy: Complex physical reasoning using large language models and world models. *arXiv preprint arXiv:2411.08027*, 2024.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Guo, D., Yang, D., Zhang, H., Song, J., et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Hao, S., Gu, Y., Ma, H., Hong, J. J., Wang, Z., Wang, D. Z., and Hu, Z. Reasoning with language model is planning with world model. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Jackson, T., Brown, N., Luu, L., Levine, S., Hausman, K., and Ichter, B. Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning (CoRL)*, 2023.

Kiefl, E. PoolTool: A Python package for billiards simulation and visualization. https://github.com/ekiefl/pooltool, 2024. Accessed: 2026-04-15.

Li, S., Puig, X., Paxton, C., Du, Y., Wang, C., Fan, L., Chen, T., Huang, D.-A., Akyürek, E., Anandkumar, A., et al. Pre-trained language models for interactive decision-making. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., and Stone, P. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023a.

Liu, R., Wei, J., Gu, S. S., Wu, T.-Y., Vosoughi, S., Cui, C., Zhou, D., and Dai, A. M. Mind's Eye: Grounded language model reasoning through simulation. In *International Conference on Learning Representations (ICLR)*, 2023b.

Memery, S., Lapata, M., and Subr, K. SimLM: Can language models infer parameters of physical systems? *arXiv preprint arXiv:2312.14215*, 2024.

Memery, S., Denamganaï, K., Zhang, J., Tu, Z., Guo, Y., and Subr, K. CueTip: An interactive and explainable

physics-aware pool assistant. In *ACM SIGGRAPH 2025 Conference Papers*. ACM, 2025. doi: 10.1145/3721238.3730742.

OpenRouter. OpenRouter: A unified interface for LLMs. https://openrouter.ai/, 2024.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Qiu, S., Guo, S., Song, Z.-Y., et al. PHYBench: Holistic evaluation of physical perception and reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2025.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. 2023.

Team, G. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research (TMLR)*, 2024.

Wilson, E. B. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

Xu, X., Xu, Q., Xiao, T., Chen, T., Yan, Y., Zhang, J., Diao, S., Yang, C., and Wang, Y. UGPhysics: A comprehensive benchmark for undergraduate physics reasoning with large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Zhang, X., Dong, Y., Wu, Y., Huang, J., Jia, C., Fernando, B., Shou, M. Z., Zhang, L., and Liu, J. PhysReason: A comprehensive benchmark towards physics-based reasoning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Zhuang, R., Gupta, A., Yang, R., Rahane, A., Li, Z., and Anumanchipalli, G. PokerBench: Training large language models to become professional poker players. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

A. LLM Prompt

The system prompt below is shared verbatim across all six LLMs evaluated in the benchmark. The user-prompt template that follows enumerates the current board state.

System prompt

```
You are an expert 8-ball pool player AI. Your goal is to actually POT your own
balls (sink them into pockets), not just touch them.

HOW POOL POTTING WORKS --- the GHOST BALL concept

To pot a target ball T into pocket P, the cue ball C must arrive at the "ghost
ball" position G at the moment of contact:

   G = T + (T - P) / |T - P| * BALL_DIAMETER

Geometrically, G sits one ball-diameter BEHIND the target ball, on the line that
runs FROM the pocket THROUGH the target.  When the cue strikes the target while
passing through G, the target gets propelled along the line T --> P, sending it
into the pocket.

So the angle you must aim the cue at is:
   aim_angle_deg = atan2(G_y - C_y, G_x - C_x) in degrees.

NOT atan2(T_y - C_y, T_x - C_x).  That would just hit the target dead center and
send it bouncing in some random direction.  ALWAYS compute the ghost ball first,
then aim at it.

WORKED EXAMPLE: Cue C = (0.5, 0.3), Target T = (0.7, 1.0), Pocket P = (1.07,
2.13), BALL_DIAMETER = 0.05715 m.  Step 1:  pocket-to-target vector = (-0.37,
-1.13).  Step 2:  length = 1.189.  Step 3:  unit vector = (-0.311, -0.950).  Step
4:  ghost = T + unit * BALL_DIAMETER = (0.6822, 0.9457).  Step 5:  aim_angle_deg =
atan2(0.6457, 0.1822) = 74.2 degrees.

PICKING THE BEST SHOT. For each of YOUR balls, consider all 6 pockets.  The best
shot has:  (a) SMALL cut angle (cue-target line nearly aligned with target-pocket
line), (b) SHORT total path length, (c) CLEAR line from cue to ghost-ball position,
(d) CLEAR line from target to pocket.  A cut > 70 degrees is essentially impossible
--- pick a different ball/pocket.

CUE SPEED. Use 8--12 m/s.  Steep cuts and long shots need more speed because
energy transfer drops with cut angle.  Short straight pots can use ~7 m/s.

OUTPUT FORMAT. Respond with ONLY this JSON, no other text:
  {"target_ball":  "<id>", "target_pocket":  [<x>, <y>], "aim_angle_deg":  <float>,
"cue_speed":  <float>}
```

User prompt template

```
Table:  width=1.0668m, length=2.1336m.  Ball radius=0.02858m, ball
diameter=0.05715m
Cue ball C = ({x}, {y})
Your group:  SOLIDS (balls 1--7).  You MUST hit one of these first AND ideally pot
it.
  Ball 1:  x=..., y=...  (distance from cue = ...m)
  ...
Opponent balls (stripes --- DO NOT hit first):
  ...
8-ball:  x={x_8}, y={y_8} (DO NOT hit first)
Pockets at:  {six pocket-centre coordinates}
Respond with ONLY JSON.
```

B. Per-Category Score Table

Table 2 reports per-category legal first-contact rate for all nine players. Per-cell sample size is shown in the column header. Wilson 95% CIs on individual cells are wide ($\pm$15–40 pp) at the per-category n used here; we therefore avoid making fine-grained per-category inter-model claims, and Section 6.3 lists scaling to 500 scenarios as the primary remedy.

Table 2. Legal first-contact rate (%) by category and player. Best per row in bold. CIs at this n are wide; entries should be read as exploratory.

Category (n)	**Oracle** 8	**Heur.** 8	**Rand.** 8	**GPT-4o** 8	**Claude S4** 8	**Gemini 2.5** 8	**Qwen 7B** 8	**Llama 8B** 8	**Gemma 9B** 8
open ($n = 8$)	75.0	62.5	62.5	75.0	**87.5**	**87.5**	50.0	62.5	0.0
partial_block ($n = 8$)	**62.5**	50.0	50.0	**62.5**	37.5	37.5	37.5	**62.5**	0.0
indirect ($n = 8$)	**62.5**	37.5	12.5	0.0	37.5	0.0	37.5	37.5	0.0
crowded ($n = 7$)	**85.7**	**85.7**	57.1	57.1	**85.7**	71.4	57.1	57.1	0.0
sparse ($n = 7$)	28.6	28.6	28.6	28.6	28.6	14.3	14.3	14.3	0.0
foul_trap ($n = 6$)	**66.7**	50.0	50.0	50.0	50.0	16.7	33.3	**66.7**	0.0
spin_shot ($n = 6$)	50.0	33.3	50.0	**83.3**	50.0	16.7	33.3	16.7	0.0

C. Reproducibility

The benchmark uses NumPy random seed 42 for scenario generation and seed 84 for the Random player. PoolTool is installed via `pooltool-billiards` (latest at submission time). Each scenario is described by a JSON record giving the cue-ball coordinates, all ball-ID/coordinate pairs, the assigned own-group, and the category label. A reference Jupyter notebook reproduces Table 1 end-to-end given an OpenRouter API key. We commit to releasing notebook, scenario JSON, per-shot trace data, and the per-shot angle log called for in Section 6.3 at acceptance.

D. Sample LLM Output

A representative parseable response from Claude Sonnet 4 on scenario s000 (open):

```
{"target_ball": "1", "target_pocket": [0.0, 0.0], "aim_angle_deg": 218.4,
"cue_speed": 9.0}
```

A representative parse failure from Gemma 2 9B on the same scenario:

```
Sure! Let me think through this step by step.
First, I need to pick the best ball. Looking at the layout, ball 1 is closest to
my cue...[continues for 200 tokens with no JSON]
```

Camera-Ready Edits Summary

This page summarizes the changes made in the camera-ready version of PoolBench relative to the original submission. It is provided for workshop administration and transparency and is not part of the technical content of the paper.

Author information.

The anonymous author block has been replaced with author names, affiliations, and correspondence details, and the under-review footer has been removed.

Response to reviewer and Area Chair comments.

The principal point raised in review was the floor effect: all evaluated models pot zero balls, which limits differentiation between models at the current benchmark scale. This limitation is central to the paper and was discussed in the original submission. The contribution is the benchmark infrastructure rather than a model ranking, and the headline result is reported as an upper bound rather than a separation between models, as stated in Sections 4.1 and 6.1. For the camera-ready we have made the floor effect, and the planned remedies for it, more explicit. Sections 6.3 and 6.4 describe scaling to a larger scenario set, lower-difficulty scenario variants, and the angle-rounding and target-versus-angle ablations that the benchmark is designed to support, all of which are intended to reduce this floor in future versions of the work.

Clarifications and references.

Minor wording and typographic corrections were made for readability. References were checked and updated where preprint versions have since appeared in published venues.

Other changes.

No other substantive changes were made. The benchmark design, scenarios, metrics, experiments, and reported numbers are unchanged from the original submission.