Solver-Guided Fine-Tuning for Accurate and Explainable Poker Decisions

Final Executive Summary

Cassandra Shi (yikes@andrew.cmu.edu) Mingxi Yan (mingxiy@andrew.cmu.edu)
Harry Huang (houzheh@andrew.cmu.edu)
10-423/623 Generative AI, Spring 2026

1 Introduction

LLMs produce fluent text about poker strategy, but fluency often masks incorrect reasoning. A model may pick a suboptimal action while sounding convincing, or pick the right action and justify it with irrelevant logic. Poker is an ideal testbed for this gap because solver outputs provide objective ground truth for strong play (Zhuang et al., 2025).

We train a model to jointly produce a correct action *and* a useful explanation. Starting from Llama 3 8B Instruct, we compare three configurations: a zero-shot baseline, SFT (LoRA on solver-backed triples), and SFT+DPO (preference optimization on automatically scored pairs), all evaluated on a fixed 500-scenario held-out set. Since PokerBench has no human-written explanations, we prompt Llama 3.3 70B as a teacher and filter its candidates with a rule-based verifier, yielding 3,000 SFT triples and 2,534 DPO preference pairs. We additionally validate the verifier against 200 blind A/B human preferences logged via a custom CLI.

The zero-shot baseline reaches 18.0% EM and 29.0% AA. The model identifies the raise *category* 42.3% of the time but achieves 0% EM on raise amounts, confirming that solver-calibrated training is necessary. Our action-only PokerBench FT replication reaches 60.50% EM and 70.00% AA, a practical upper bound for the joint model. SFT closes most of the gap (56.20% EM, 68.00% AA), and SFT+DPO improves explanation quality further (verifier 0.645→0.674, strategic relevance win rate 77.5%) at a small action-accuracy cost (52.20% EM, 65.60% AA), confirming the expected alignment tax.

2 Dataset, Task, and Model

Dataset PokerBench (Zhuang et al., 2025) (`RZ412/PokerBench`) provides NLHE scenarios paired with solver-computed actions. Each example encodes hole cards, position, stacks, pot, and betting history. We use the pre-flop train split (60k, subsampled to 3,000 at seed 42) and the pre-flop test split (1k, fixed 500-example held-out set via `export_eval_set.py`). Pre-flop spots offer structural regularity, dense solver coverage, and direct comparability with PokerBench.

Task and metrics Given state s, the model produces an action a and a natural-language explanation e. Bet sizes are real-valued and explanations are unconstrained, so this is fully generative. We report (1) **EM** (exact match to solver), (2) **AA** (action category match: any `raise X` counts for any `raise Y`), (3) **verifier score** (action consistency 0.40, state grounding 0.35, strategic relevance 0.25), and (4) **pairwise preference** from GPT-4o-mini over 200 pairs with randomized presentation order (Wang et al., 2024).

Model **Llama 3 8B Instruct**: it is the model evaluated by PokerBench (paper FT target EM 78.26% / AA 80.64%), fits on a single A10G/A100 for both SFT and DPO, and as an instruction-tuned checkpoint already follows structured output formats.

3 Related Work

DeepStack (Moravcík et al., 2017) and **Libratus** (Brown and Sandholm, 2018) achieved superhuman NLHE play via game-theoretic solving but require heavy inference and produce no explanations. **PokerGPT** (Huang et al., 2024) and **PokerBench** (Zhuang et al., 2025) showed LLMs can act as lightweight agents. PokerBench in particular showed that fine-tuning on solver labels closes most of the gap to expert play, motivating our use of its labels as offline solver supervision. **ToolPoker** (Lin et al., 2026) improved both decisions and reasoning through inference-time solver tool calls. We are complementary, asking whether a model can *internalize* solver knowledge with no inference-time tools.

InstructGPT (Ouyang et al., 2022) established RLHF, while **DPO** (Rafailov et al., 2023) simplifies

it to a direct objective with no separate reward model, suitable here because our pairs are constructed automatically. **LoRA** (Hu et al., 2021) makes single-GPU SFT and DPO feasible. Lanham et al. (2023) show CoT explanations can be unfaithful to the underlying decision, especially relevant for poker where wrong arguments can sound right. Zheng et al. (2023) and Wang et al. (2024) motivate our judge design (randomized presentation, small human-calibration subset).

4 Methods

Figure 1 shows the full pipeline. Since PokerBench has no human explanations, we synthesize them using a stronger LLM as teacher.

Data generation (`data_generation.py`). Llama 3.3 70B-Instruct-Turbo (Together AI) receives each scenario plus solver action a^* and produces k=3 candidates at T=0.7 (`max_tokens`= 350). The prompt requires reference to hand strength, position or pot odds, and consistency with a^*.

Verifier and dataset construction (`verifier.py`). `score_explanation` scores each candidate on action consistency (0.40), state grounding (0.35, with four sub-dimensions: hand, position, action history, pot/stack), and strategic relevance (0.25, mixing high-tier *GTO/blockers/SPR/EV* and mid-tier *range/equity/value* keywords), plus a length penalty and a small scenario-grounding bonus. `build_datasets` writes `sft_dataset.jsonl` (best candidate per scenario as e^+) and `dpo_dataset.jsonl` (best vs. worst, kept when verifier gap $>$ 0.05). *Critically, both branches of every DPO pair carry the same solver action*, so at training time the preference signal is isolated to the explanation portion of the completion. Final retention: 3,000 SFT triples and 2,534 DPO pairs (84.5%, mean gap 0.175, chosen 0.931, rejected 0.756).

Human-annotation calibration (`human_annotation.py`). We built a CLI that loads DPO pairs, randomises A/B order so the annotator is blind to the verifier's choice, optionally restricts to a gap bucket, and saves each judgment to `data/human_annotations.jsonl`. Sessions resume via content-hashed pair IDs. The `--summary` flag aggregates verifier agreement overall, by gap bucket, and per annotator. We logged **200** annotations: $\sim$80% verifier agreement above the 0.05 gap and $\sim$76% in the cleaner gap-$\geq$0.10 zone. This directly informed the gap-threshold ablation.

Table 1: Reference results on pre-flop accuracy.

Condition	EM	AA
Paper, 5-shot, action-only (Zhuang et al., 2025)	37.77%	49.30%
Our reproduction, 5-shot, action-only	39.30%	55.80%
Paper FT, action-only (Zhuang et al., 2025)	78.26%	80.64%
PokerBench FT reproduction, action-only	60.50%	70.00%
Our baseline (0-shot, action + explanation)	18.00%	29.00%

Training. Baseline (`baseline_eval.py`): zero-shot Llama 3 8B Instruct in the joint format. **SFT** (`model_SFT/SFT_piplines.ipynb`): LoRA rank 16, α=32, dropout 0.05, targets `q,k,v,o_proj`, LR 2×10^{-4}, eff. batch 16, 3 epochs, with TRL (von Werra et al., 2023) and `completion_only_loss=True` so loss is over the "Action: ... Explanation: ..." span only. **SFT+DPO** (`train_dpo.ipynb`): merge the SFT LoRA into the base, then attach a fresh DPO LoRA (same rank/α). With β=0.1, LR 1×10^{-5}, 1 epoch, eff. batch 16, cosine schedule. `ref_model=None` is correct: TRL disables the DPO adapters for reference log-probs, so the reference is the SFT model.

5 Experiments

Three research questions drive this study. **(RQ1)** Does SFT on solver-backed labels and teacher explanations close the bet-size calibration gap exposed by the zero-shot baseline (raise AA 42.3% but EM 0%)? Tested by EM and per-action AA on raise in Tables 2 and 4. **(RQ2)** Does DPO on automatically constructed preference pairs improve explanation quality (verifier and pairwise judge) without sacrificing action accuracy? Tested by the SFT→SFT+DPO column transitions in Tables 2 and 3. **(RQ3)** In the regime of automatic preference labels, does preference *quantity* beat label-noise reduction? Tested by the gap-$\geq$0.05 vs. gap-$\geq$0.10 ablation.

5.1 Baseline and reference benchmarks

Table 1 stacks all reference numbers. Our 5-shot reproduction matches the PokerBench paper (+1.53 EM, +6.50 AA), confirming the eval setup. The drop to the zero-shot joint baseline reflects two deliberate differences: no few-shot demos and the joint action+explanation output format. The FT replication (`recreate/replication.ipynb`) is an action-only fine-tuned upper bound.

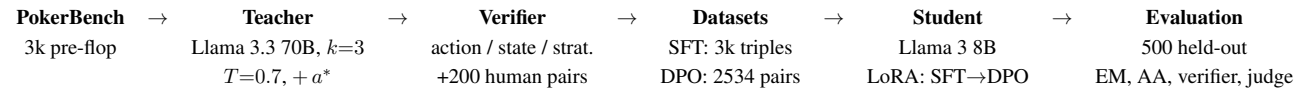

Figure 1: Pipeline: solver-backed scenarios → teacher candidates → rule-based verifier (cross-checked against 200 human-labelled pairs) → SFT/DPO datasets → fine-tuned student → held-out evaluation.

Table 2: Primary results: EM, AA, verifier average, and pass rate on the 500-example held-out set.

Config.	EM	AA	Ver.	Pass%
Baseline	18.0%	29.0%	0.426	9.2%
SFT	56.20%	68.00%	0.645	55.8%
SFT+DPO (gap≥0.05)	52.20%	65.60%	0.674	58.2%
SFT+DPO (gap≥0.10)	51.00%	64.40%	0.662	55.0%
PB FT Repro	60.5%	70.0%	N/A	N/A

Table 3: Pairwise preference (GPT-4o-mini, 200 pairs/comparison, randomized order). AC = action consistency, SR = strategic relevance.

Comparison	Win/Lose	Tie	AC	SR
SFT vs. Zero-shot	79.0 / 21.0%	0.0%	79.0 / 21.0%	81.0 / 19.0%
SFT+DPO vs. Zero-shot	75.5 / 24.5%	0.0%	76.5 / 23.5%	77.5 / 22.5%
SFT+DPO vs. SFT	41.0 / 39.5%	19.5%	39.5 / 41.0%	41.5 / 39.0%

The baseline reveals the raise calibration gap: AA on raises is 42.3% but EM is 0%. The model succeeds on large amounts (raise 15.0: 6/6, raise 23.0: 4/4) but fails on small ones (raise 3.0: 0/14, raise 2.5: 0/8). It knows *when* to raise but not *how much*. Verifier subscores tell the same story: high state grounding (0.747) but low action consistency (0.228) and strategic relevance (0.088), with only 9.2% of explanations passing all checks.

5.2 Primary results

Table 2 shows SFT delivers the bulk of the gain: EM 18.0→56.20%, AA 29.0→68.00%, verifier 0.426→0.645, pass rate 9.2→55.8%. SFT+DPO improves explanation quality further (verifier 0.674, pass 58.2%) but pays an alignment tax in EM/AA. The cleaner gap-≥0.10 ablation (1,902 pairs) lands marginally below gap-≥0.05 on every metric, suggesting that within this scale preference *quantity* beats marginal label-noise reduction.

We measured the GPT judge against a 50-pair human-annotated subset and obtained **30/36 = 83.3%** non-tie agreement, above the 80% threshold cited by Zheng et al. (2023) as reliable for LLM-as-judge.

5.3 Per-action error analysis

Fold, call, and raise improve or hold under DPO, but check collapses (19.5→4.9%). Three likely causes:

Table 4: Per-action AA under SFT vs. SFT+DPO (gap≥0.05).

Action	SFT	SFT+DPO
fold	90.7%	90.7%
call	81.4%	83.9%
raise	79.2%	81.5%
check	19.5%	4.9%

(i) check is only valid when no bet faces the player, so it competes with call. (ii) Pre-flop data is raise-heavy, leaving check rare in the training distribution. (iii) SFT data underrepresents check, so DPO further destabilizes the already marginal action. The gap-≥0.10 ablation reproduces the same collapse (check AA 2.4%), confirming that the cause is structural rather than label-noise-driven.

5.4 Compute and timing

All training and inference ran on Google Colab A100 40GB ($1.30/hr, approximate at 13 CU/hr, $49.99 per 500 CU). 5-shot paper repro 0.06 GPU-hr ($0.08), baseline eval 0.69 GPU-hr ($0.90), SFT/DPO eval 1.40 GPU-hr ($1.82), FT replication 13.00 GPU-hr ($16.90), SFT training (3k examples × 3 epochs) 0.80 GPU-hr ($1.04), DPO training (2,534 pairs, 1 epoch) 0.30 GPU-hr ($0.39), and the gap-≥0.10 ablation 0.20 GPU-hr ($0.26). Total measured GPU time: **16.45 hrs** at ≈$21.39, plus $6.73 in Together AI for teacher generation and ∼$0.30 for the GPT-4o-mini judge.

6 Key Findings

SFT on solver labels and teacher explanations delivers the bulk of the win: solver-consistent actions, calibrated bet sizes, and explanations that win 79% pairwise vs. zero-shot. **DPO** sharpens explanations further (verifier 0.645→0.674) at a small EM/AA cost. Because every DPO pair preserves the action token, the preference signal is by construction about explanation quality, yet shared-backbone gradients still leak into action prediction, producing the alignment tax. The pairwise SFT+DPO vs. SFT result (41 / 39.5 / 19.5) is essentially tied, so DPO's gains are incremental, not transformative. **Overall**, solver-guided train-

ing enables joint accuracy and interpretability with no inference-time tools and no human-written explanations, while the human-annotation CLI provides a cheap loop for grounding the verifier in real preferences.

7 Code Overview

All code was written from scratch by the team, with AI assistance only for initial scaffolding that was reviewed and rewritten.

`utils.py`: shared helpers including the PokerBench loader, `normalise_action`, `extract_action` (handles raise/bet amounts and malformed prefixes), `compute_action_accuracy` (EM, AA, per-action), `LatencyTracker`, and JSONL I/O. The action parser was iteratively debugged by Mingxi.

`export_eval_set.py`: builds the fixed 500-example held-out set (`eval_test_500.jsonl`) by sampling the pre-flop test split at seed 42, reused by every evaluator.

`data_generation.py`: Together AI client that prompts Llama 3.3 70B-Instruct-Turbo with each scenario plus a^*, generating k=3 candidates at T=0.7. Checkpointable and resumable.

`verifier.py`: `score_explanation` (lines **[42–168]**) implements the three-dimensional rule-based scorer (action consistency with contradiction penalty, four-part state grounding, two-tier strategic vocabulary, length penalty, scenario-grounding bonus). `build_datasets` (lines **[195–278]**) writes `verifier_scores.jsonl`, `sft_dataset.jsonl`, and `dpo_dataset.jsonl`.

`human_annotation.py`: CLI for blind A/B human preferences, resumable via content-hashed pair IDs, with a gap-bucket filter, a `--summary` mode for verifier agreement breakdown, and `--export_disagreements` for review. The annotation loop is at lines **[88–164]**; the summary aggregator at lines **[210–261]**. Output: `data/human_annotations.jsonl` (200 records).

`model_SFT/SFT_piplines.ipynb` and `SFT_verify.ipynb`: SFT in TRL (rank 16, α=32, dropout 0.05, `q,k,v,o_proj`, LR 2×10^{-4}, eff. batch 16, 3 epochs, with `completion_only_loss=True`)(**cell 3**), using `sft_prompt_completion.jsonl` to fine tune the model(**cell 5**).

`train_dpo.ipynb`: attaches the SFT LoRA, merges into the base, then fits a fresh DPO LoRA via `DPOTrainer` (β=0.1, LR 1×10^{-5}, 1 epoch, eff. batch 16). The merge-and-attach cell is **cell 6**; the `DPOTrainer` configuration and `trainer.train()` call are **cells 7–9**; the gap-$\geq$0.10 ablation re-runs the same recipe in **cells 14**. Both chosen and rejected share the same action token.

`baseline_eval.py`: zero-shot evaluator that builds the joint prompt, runs deterministic chat-template generation, parses action and explanation, and computes EM, AA, per-action, verifier subscores, latency, and a side-by-side vs. PokerBench reference.

`recreate/replication.ipynb` and `pokerbench_repro.ipynb`: independent PokerBench replication (5,000 steps on the full 60k split, LR 1×10^{-6}, eff. batch 128, LoRA rank 8 / α=16, `q,v_proj`)(**cell 3, cell 5 of replication.ipynb**) and the 5-shot action-only reproduction.

`gpt_pairwise_eval.ipynb`: pairwise judge pipeline that samples 200 scenarios (seed 42), runs deterministic inference under each of the three configurations, prompts GPT-4o-mini with randomized A/B order across AC, coherence, and SR, aggregates win/tie/lose (Table 3), and saves a 50-pair subset that is then fed to `human_annotation.py` for the 83.3% judge vs. human agreement check.

8 Timeline, Human Work, and AI Use

Human work. Core design choices were made without AI assistance: verifier weights (0.40/0.35/0.25), DPO gap threshold (0.05), four-part state grounding, and the evaluation/teacher prompts. Cassandra spot-checked the data split, iterated on the teacher prompt, and built the human-annotation CLI. Mingxi reviewed 200 pairs through that CLI to calibrate the verifier and fixed action-parser edge cases. Harry validated the PEFT setup against PokerBench and ran the 60k replication. All members iterated on the joint prompt format.

AI tools. Llama 3.3 70B served as the teacher ($\sim$9,000 candidates: 3,000 scenarios $\times$ k=3). GPT-4o-mini served as the pairwise judge (600 calls). Claude and ChatGPT provided initial scaffolding for the action parser and verifier keywords (later reviewed and rewritten), LaTeX formatting, and literature search suggestions (papers independently read and verified).

Verification. All paper reproduction numbers were cross-checked against PokerBench's reported values, and all report numbers were verified against raw result files

Table 5: Estimated team hours by activity.

Activity	Hours
Reading papers and dataset documentation	12
Data pipeline and action parser	10
Verifier design and calibration	8
Human-annotation CLI + 200 annotations	6
Teacher prompt engineering	5
PokerBench FT replication	8
SFT training and debugging	6
DPO training, gap ablation, and debugging	8
Evaluation pipeline (EM, AA, verifier)	6
Pairwise judge pipeline (GPT-4o-mini)	4
Running experiments / compiling results	6
Poster + writing this document	14
Total	**93**

(`results/../baseline_results.json`, the DPO eval JSON saved by `train_dpo.ipynb`, and `results/pairwise_eval/pairwise_summary.json`). The verifier's decisions are cross-checked against 200 blind human annotations, and the GPT judge against a 50-pair human subset (83.3% non-tie agreement).

9 Research Log

First deviation. Our original plan assumed PokerBench had some form of explanation labels. It does not. We pivoted to teacher-generated explanations, which added an entire pipeline stage (teacher prompting, verifier design, human annotation, and DPO pair construction).

Replication gap. The PokerBench paper (Zhuang et al., 2025) reports 78.26% EM but omits LoRA rank, target layers, and exact LR schedule. Our minimal-assumption replication (rank 8, α=16, `q,v_proj`) hit 60.50% EM, an $\sim$18 pp gap. We attribute this primarily to data scale: paper used the full 60k pre-flop split at larger effective batch, whereas our joint-objective runs were capped at 3,000 examples by time and compute budget.

Verifier calibration. The first verifier scored every candidate near zero on strategic relevance because the keyword list was too restrictive. After 200 annotations through the CLI, Mingxi expanded the vocabulary and rebalanced weights, reaching $\sim$80% human agreement above the 0.05 gap.

Prediction vs. outcome. The midway report hypothesised that SFT+DPO would win all three pairwise comparisons, including SFT+DPO vs. SFT. The actual judge result is essentially tied (41.0 / 39.5 / 19.5 win/lose/tie), and SFT+DPO actually loses on action consistency (39.5 vs. 41.0). The verifier-side gain (0.645$\rightarrow$0.674) is real but small, and the EM/AA regression (56.20$\rightarrow$52.20%, 68.00$\rightarrow$65.60%) was not predicted. We attribute both to the same mechanism: shared backbone parameters absorb explanation-side preference gradients, shifting the action distribution despite identical action spans across pair branches. This is the empirical basis for the action-conditioned DPO proposal in the compute thought-experiment.

DPO check degradation. The most surprising result was check AA collapsing from 19.5% (SFT) to 4.9% (SFT+DPO). Because each pair preserves the action token, we expected DPO to be action-neutral. Post-hoc analysis: check is rare and semantically close to call, making it the fragile prediction that shared-backbone updates destabilize first. The gap-$\geq$0.10 ablation reproduces the collapse (check AA 2.4%), confirming the cause is structural.

What we'd do differently. (1) Include post-flop scenarios from the start to reduce raise heaviness. (2) Use the full 60k split for SFT to close the replication gap before adding the explanation objective. (3) Apply action-conditioned DPO that masks the action tokens in the loss, so backbone updates touch only the explanation span.

10 Conclusion

Solver-guided fine-tuning enables a single LLM to jointly produce accurate poker actions and strategically grounded explanations with no inference-time tool access. SFT on verifier-filtered teacher data substantially closes the zero-shot gap, and DPO further improves explanation quality at a small action-accuracy cost. The check degradation under DPO highlights a limitation of pre-flop-only training on raise-heavy data. The gap-threshold ablation shows that within our regime, preference *quantity* beats marginal label-noise reduction. Future work: post-flop expansion, action-conditioned DPO, scaling the human-annotation pool to replace the verifier in the noisy 0.05 to 0.10 gap zone, and chain-of-thought reasoning for sharper bet-sizing precision.

11 Thought-Experiment on Compute

Actual compute. Total $\approx$\$28.42 (\$21.39 GPU + \$6.73 Together AI + \$0.30 OpenAI). Hardware: Google Colab A100 40GB at \$1.30/hr for both inference, A100 40GB at \$3.19/hr for training and inference. The FT replication dominates GPU spend (13 hr / \$16.90). Everything else combined is $<$3.5 hr.

With \$450 in extra credits ($\approx$313 A100-hours):

(1) **Scale to full 60k** (≈$166: $31 GPU + $135 API). Teacher generation scales linearly: 60k is 20× our 3k run, so ∼$135 in Together AI calls. SFT grows to ∼16 GPU-hr ($20.80) and DPO to ∼6 GPU-hr ($7.80) plus eval (∼$2). Directly tests whether data volume alone closes the FT replication gap.

(2) **Hyperparameter ablation** (≈$50). 3×3 grid over LoRA rank $\{8, 16, 32\}$ and DPO $\beta\,\{0.05, 0.1, 0.2\}$, two seeds. 9 SFT+DPO+eval pipelines at ∼$3.25 each (≈$30), plus two seeds for variance (≈$20).

(3) **Post-flop generalization** (≈$30). 5k post-flop scenarios ∼$10 in API and ∼15 GPU-hr ($20)to test pre-flop→post-flop transfer.

(4) **Model-scale** (≈$45). Llama 3 13B SFT+DPO (∼2–3 times the compute of 8B, ≈30–40 GPU-hr) to test whether scale gains beat pipeline gains.

(5) **Action-conditioned DPO** (≈$5). Mask action tokens in the preference loss so backbone updates touch only the explanation span, directly targeting the alignment tax and check collapse. The run is small (∼0.5 GPU-hr training + eval), so the cost is dominated by re-evaluation.

Total is ≈$296, leaving room for additional seeds on (1) and (4) or a wider ablation grid in (2).

References

Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Edward J. Hu et al. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Chenghao Huang et al. PokerGPT: An end-to-end lightweight solver for multi-player Texas Hold'em via large language model. *arXiv preprint arXiv:2401.06781*, 2024.

Tamera Lanham et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Minhua Lin et al. How far are LLMs from professional poker players? *arXiv preprint arXiv:2602.00528*, 2026.

Matej Moravcík et al. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Long Ouyang et al. Training language models to follow instructions with human feedback. *NeurIPS*, 35:27730–27744, 2022.

Rafael Rafailov et al. Direct preference optimization: Your language model is secretly a reward model. *NeurIPS*, 36, 2023.

Leandro von Werra et al. TRL: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2023.

Peiyi Wang et al. Large language models are not fair evaluators. In *ACL*, pages 9440–9450, 2024.

Lianmin Zheng et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. *NeurIPS*, 36, 2023.

Richard Zhuang et al. PokerBench: Training large language models to become professional poker players. In *AAAI*, volume 39, pages 26175–26182, 2025.