Can Supervised Fine-Tuning Teach Small Language Models to Match Larger Models on Algorithmic Reasoning?

Xristopher Aliferis
Department of Electrical and Computer Engineering
Western University
London, Ontario
xaliferi@uwo.ca

Carmel Kurland
Department of Electrical and Computer Engineering
Western University
London, Ontario
carmel.kurland@uwo.ca

Christopher Lam
Department of Electrical and Computer Engineering
Western University
London, Ontario
clam433@uwo.ca

Devraj Nagpal
Department of Electrical and Computer Engineering
Western University
London, Ontario
dnagpal2@uwo.ca

***Abstract*—Scaling up large language models (LLMs) yields strong zero-shot performance on many language and reasoning tasks, but it remains unclear how far much smaller models can be pushed with supervised fine-tuning alone. We ask whether a 1.5B-parameter model can be trained to match or exceed the multi-step reasoning performance of a 72B-parameter model on structured algorithmic problems as the prediction horizon grows. We construct four synthetic games with discrete state spaces and known optimal policies (Fibonacci sequence prediction, Towers of Hanoi, sliding puzzle, and N-Queens) and cast them into a unified next-K-step prediction interface. For each game we generate stratified train and test splits and evaluate both generic sequence metrics and task-specific structural metrics that capture recurrence consistency, move legality, board dynamics, and constraint satisfaction. Supervised fine-tuning substantially improves the small model on all tasks and, on sliding puzzle and N-Queens, allows it to match or slightly surpass the larger model in exact-match accuracy and structural consistency. However, the larger model retains a clear advantage on tasks that require precise long-range recurrence tracking or deeply recursive optimal strategies, indicating that fine-tuned small models can learn meaningful reasoning patterns but do not fully eliminate the benefits of scale.**

***Index Terms*—large language models, supervised fine-tuning, multi-step reasoning, algorithmic games, sequence prediction**

I. Introduction

Large language models (LLMs) have rapidly advanced the state of the art in natural language processing, with model size emerging as one of the strongest predictors of downstream performance. Scaling studies show that increasing parameters, data, and compute yields smooth improvements in loss and task accuracy across translation, question answering, and general language understanding [1]–[4]. At the same time, very large models are expensive to train and deploy, which motivates a central question for both research and practice: to what extent can smaller models, when carefully fine-tuned, recover the reasoning capabilities of much larger systems?

This question is especially important for multi-step reasoning tasks. Many recent benchmarks highlight that strong performance often requires producing sequences of interdependent decisions rather than isolated labels, whether in chain-of-thought style explanations, program synthesis, or algorithmic puzzles [5]–[10]. Large models appear to acquire some of these behaviours implicitly through pre-training and scale, and some reasoning capabilities emerge only beyond particular parameter thresholds [11]. However, it is less clear whether much smaller models can learn comparable behaviours when they are given explicit supervised trajectories rather than relying on in-context learning alone [12].

In this work we study this question in a controlled algorithmic setting. Prior work on algorithmic datasets shows that Transformers can learn approximate algorithms but often fail to extrapolate reliably to longer sequences or more complex instances [8], [13]. We compare a small instruction-tuned model with 1.5 billion parameters to a much larger 72 billion parameter model on four synthetic games with discrete states, well-defined optimal policies, and varying forms of structure: Fibonacci sequence prediction, Towers of Hanoi, an 8-puzzle sliding game, and the N-Queens problem. Each task is presented in a unified next-K-step prediction format, where the model must predict several future moves or sequence elements given a textual description of the current state, allowing us to probe how performance changes as the prediction horizon and effective context length increase.

The small model is evaluated both in a zero-shot configura-

Code available at: https://github.com/XitoAliferis/CS_3346-Group_Project

tion and after supervised fine-tuning on synthetic trajectories generated by exact solvers or search procedures, in line with work suggesting that training on intermediate reasoning steps can improve algorithmic generalization [13]. We measure standard sequence metrics such as exact match accuracy, position-wise accuracy, and edit similarity, and complement them with task-specific structural metrics that explicitly test whether predictions respect the underlying rules of each game. Examples include recurrence consistency for Fibonacci sequences, move legality and tower configuration divergence in Towers of Hanoi, board divergence and legal prefix rates in the sliding puzzle, and conflict rates in partial N-Queens placements.

Our results show a mixed but encouraging picture. Across all games, supervised fine-tuning significantly improves the small model relative to its baseline, indicating that it does learn non-trivial multi-step reasoning patterns from supervised signals alone. On the sliding puzzle and N-Queens tasks, a fine-tuned 1.5B model trained on a few thousand examples matches or slightly exceeds the performance of the 72B model evaluated zero-shot, both in exact match accuracy and in structural consistency metrics. In contrast, on Fibonacci and Towers of Hanoi the larger model retains a clear advantage, particularly when long-range recurrence or deeply recursive optimal strategies are required. These findings suggest that while supervised fine-tuning can close much of the gap for tasks dominated by local constraints and short-to-medium horizons, model scale still provides a significant benefit for problems that demand precise long-horizon reasoning.

Our main contributions are:

- We introduce a unified next-K-step framework over four algorithmic games with exact solvers and structured state representations, designed to probe multi-step reasoning under varying horizons.
- We implement a supervised fine-tuning pipeline for a 1.5B-parameter instruction-tuned model using LoRA, and evaluate it against a 72B-parameter reference model under a shared prompting and decoding protocol.
- We propose and apply a suite of structural metrics (recurrence consistency, move legality and tower state, board dynamics, and queen conflicts) that reveal when small models genuinely internalise task rules versus merely matching surface patterns.

The remainder of this paper is organized as follows. Section II reviews related work on model scaling, reasoning in LLMs, context length limitations, and supervised fine-tuning. Section III describes our synthetic games, data generation, prompting scheme, model configurations, and evaluation protocol. Section IV presents the experimental results and a cross-task analysis of learning behaviour. Section V summarizes our conclusions and outlines directions for future work, including few-shot prompting, hyperparameter tuning, and more challenging long-context variants of the games.

II. Background & Related Work

A. Large Language Models, Scaling, and Reasoning

Modern large language models (LLMs) are built on the Transformer architecture, which replaces recurrence with multi-head self-attention to model long-range dependencies in parallel [14]. Earlier sequence models such as LSTMs [15] and GRUs [16] achieved strong performance on language tasks but were limited by sequential computation and difficulty capturing very long contexts. Transformers enabled training substantially larger models on web-scale corpora and became the backbone of GPT-style autoregressive LLMs.

Scaling studies show that performance improves predictably as a power law in model size, data, and compute [1]. GPT-2 demonstrated that a 1.5B-parameter Transformer could perform diverse generative tasks in a zero-shot setting without supervised fine-tuning [17], while GPT-3 (175B parameters) showed strong zero-, one-, and few-shot capabilities across translation, QA, and reasoning purely from natural-language instructions [2]. Follow-up work on in-context learning interprets this phenomenon as a form of implicit Bayesian meta-learning at inference time, where larger models adapt to tasks from prompts alone [12]. Subsequent foundation models such as PaLM [3] and LLaMA [4] further explored the trade-off between parameter count, training data, and downstream performance; in particular, LLaMA showed that moderately sized models trained well can rival much larger ones.

Recent work has focused on reasoning-specific behaviours. Chain-of-thought prompting elicits step-by-step rationales that substantially improve accuracy on arithmetic and symbolic tasks [5], and even zero-shot "let's think step by step" prompts can boost reasoning performance without additional labels [6]. Benchmarks such as GSM8K for grade-school math [7], MMLU for broad knowledge and reasoning [10], and BIG-bench for diverse, often challenging tasks [9] all show substantial gains with scale. Combined with chain-of-thought style evaluations, they also reveal that models can overfit to surface patterns or produce fluent but logically inconsistent rationales [5], [6], [9]. Work on "show your work" supervision suggests that training directly on intermediate solutions can improve algorithmic generalization [13], motivating the kind of structured, multi-step tasks we study here.

B. Small Models, Supervised Fine-Tuning, and Parameter-Efficient Adaptation

Despite the advantages of scale, smaller models remain attractive due to lower inference cost and easier deployment. Knowledge distillation compresses large teacher models into smaller students [18], and DistilBERT showed that a compact Transformer can retain most of BERT's language understanding performance while reducing parameter count and latency [19]. More recently, parameter-efficient fine-tuning (PEFT) methods such as LoRA [20] and QLoRA [21] adapt only a small number of additional low-rank parameters on top of a frozen (often quantized) base model, enabling task specialization under tight memory and compute budgets.

Instruction tuning and reinforcement learning from human feedback (RLHF) have been used to align large models with user preferences and improve helpfulness and truthfulness. InstructGPT [22] showed that supervised fine-tuning on human-written instruction-response pairs substantially improves perceived helpfulness and adherence to instructions, while subsequent RLHF further adjusts model behavior to match human preference judgments, including for summarization [23]. However, these pipelines are typically instantiated for models at the billion-parameter scale and above, and the most visible deployments focus on tens to hundreds of billions of parameters; comparatively less is known about how far simple supervised fine-tuning alone can push much smaller models on explicitly algorithmic, multi-step tasks.

Our work sits at the intersection of these lines: we use LoRA-style supervised fine-tuning on a 1.5B-parameter instruction-tuned model, and ask whether this is sufficient to close the reasoning gap to a 72B-parameter reference model on structured games that explicitly probe short- and medium-horizon reasoning.

C. Benchmarks and Algorithmic Reasoning Tasks

Standard NLP benchmarks such as GLUE [24] and SuperGLUE [25] have been crucial for evaluating general language understanding. GLUE aggregates nine sentence-level classification and similarity tasks spanning multiple domains and data regimes, and was quickly saturated by pre-trained Transformers. SuperGLUE raises the difficulty with harder NLI, QA, and coreference-style tasks and more diverse formats, but current large models are now at or above non-expert human performance on its aggregate score. In parallel, newer benchmarks emphasize reasoning and compositional generalization. MMLU [10] covers 57 multi-domain exam-style subjects, while GSM8K [7] targets linguistically diverse grade-school math word problems requiring multi-step arithmetic. Other math-focused suites generate large, structured training and extrapolation splits for arithmetic, algebra, calculus, and probability [8]. Beyond single-problem settings, BIG-Bench [9] assembles over 200 tasks, many explicitly designed to be challenging for contemporary LMs.

A complementary line of work isolates long-context and algorithmic behaviour. Long Range Arena (LRA) [26] probes sequence models on synthetic and real tasks with context lengths up to 16K tokens, including hierarchical ListOps, byte-level text classification and retrieval, pixel-level image classification, and long-range spatial reasoning in Pathfinder. Several studies focus on algorithmic learning more directly, showing that Transformers and related architectures can approximate algorithms such as addition, polynomial evaluation, or program execution, but often fail to extrapolate reliably to longer sequences or harder instances [8], [13]. At the same time, large-scale studies report "emergent" abilities that appear only beyond certain scale thresholds on reasoning-heavy benchmarks, including MMLU, BIG-Bench, and math word problems [11]. This leaves an open question: to what extent can targeted supervision on structured, algorithmic tasks substitute for sheer scale?

By constructing four algorithmic "games" with exact solvers, controllable instance difficulty, and rich structural metrics, our study complements these benchmarks with a controlled setting where optimal behaviour is known. We use this setting to directly test whether a comparatively small model, trained only with supervised fine-tuning, can approximate or match the reasoning performance of a much larger pretrained model as context length and structural complexity increase.

III. Methodology

This section describes the algorithmic games, synthetic dataset generation, prompting scheme, model configurations, supervised fine-tuning setup, and evaluation protocol used to study whether a small instruction-tuned language model can match the multi-step reasoning behaviour of a much larger model on structured algorithmic tasks.

A. Algorithmic Games and Dataset Generation

We consider four games with discrete state and action spaces and well-defined optimal policies. All tasks are cast into a unified *next–K-step prediction* problem: given a textual description of the current state s_t, the model must predict the next K optimal actions $(a_t, \dots, a_{t+K-1})$. Each example consists of a text prompt x and a multi-line target y in which each line is a single token (integer or move symbol).

Fibonacci sequence prediction. For the Fibonacci game we sample a total sequence length T uniformly from $[10, 50]$ and generate the first T elements of the Fibonacci sequence $(0, 1, 1, 2, \dots)$. We then sample a prediction horizon $K \in [1, 8]$ subject to $K \leq T - 2$, and a start index s with $1 \leq s \leq T-K-1$. The model is shown a contiguous slice $(x_0, \dots, x_s)$ and must predict $(x_{s+1}, \dots, x_{s+K})$. To keep prompts compact, only the last 12 elements of this prefix are displayed in the text. The state is rendered as a single line of space-separated integers, while the target contains exactly K lines, each with a single integer and no surrounding text. We enforce uniqueness over triples (T, s, K).

Towers of Hanoi. For Towers of Hanoi we sample the number of disks $n_{\text{disks}} \in [3, 8]$ and assign source, destination, and auxiliary pegs by sampling two distinct pegs from $\{A, B, C\}$ and using the remaining peg as auxiliary. The optimal solution for transferring all disks from source to destination is generated recursively and replayed to obtain both the move list and the corresponding sequence of peg configurations. Given a solution of length L, we sample a horizon $K \in [1, 8]$ and a starting step s such that $0 \leq s \leq L - K$. The observed state is the configuration after s optimal moves, rendered as

```
Peg A: 3 2 1
Peg B: empty
Peg C: ...
```

with disk IDs printed from bottom to top. The target consists of exactly K lines of the form `X->Y` (e.g., `A->C`); syntactically valid moves must match the regular

expression `^[A-C]->[A-C]$`. We ensure uniqueness over $(n_{\text{disks}}, \text{src}, \text{dst}, \text{aux}, s, K)$.

Sliding puzzle (8-puzzle). The sliding puzzle uses a 3×3 board with tiles $(1, \ldots, 8)$ and a blank tile 0. The solved state is $(1, 2, \ldots, 8, 0)$. To construct scrambled states we start from the solved board and apply a random sequence of legal moves, of length sampled from $[10, 30]$, forbidding immediate backtracking to avoid trivial oscillations. By construction, all scrambled states are solvable. Instead of running a separate search per instance, we precompute a reverse breadth-first search (BFS) tree rooted at the solved state that maps each reachable configuration to its parent and the move taken. Given a scrambled state, we recover a shortest solution path by following parent pointers back to the goal and inverting moves. For each example we sample a horizon $K \in [1, 6]$ and a starting step s along this optimal path such that $0 \leq s \leq L-K$. The state after s moves is rendered as

```
1 2 3
4 5 6
7 8 0
```

where `0` denotes the blank, and the target consists of K lines each containing one of `UP`, `DOWN`, `LEFT`, or `RIGHT`. Uniqueness is enforced over (state, K).

N-Queens. For N-Queens we sample a board size $N \in [4, 10]$ and use OR-Tools CP-SAT to enumerate up to a fixed number of distinct full solutions per N. Each solution is represented as a vector

$$\texttt{queenColumnPositions}[c] = r, \quad c \in \{0, \ldots, N-1\},$$

meaning there is a queen in column c and row r. A layout is valid if no two queens share a row or diagonal. We interpret each solution as a sequence of column-wise decisions. For each example we sample a horizon $K \in [1, 4]$ and a starting column s such that $0 \leq s \leq N - K$. The prompt reveals the first s placements in a compact format

```
rows: 0 2 4
```

and the model must predict the next K row indices (0-based), one per line. Uniqueness is enforced based on (N, prefix, K), where prefix is the tuple of observed row indices.

Splits and stratification. For each game we generate 6000 unique candidate examples using a fixed random seed (42). From these pools we construct four disjoint splits per task: a held-out test set of 300 examples and three supervised training sets of sizes 500, 1500, and 3000. To avoid introducing difficulty shifts between splits, we perform stratified sampling over task-specific difficulty buckets: (T, K) for Fibonacci, (n_{disks}, K) for Hanoi, (S, K) for sliding puzzle (where S is the scramble length), and (N, K) for N-Queens. Within each bucket we shuffle examples and allocate them greedily to the four splits in proportion to remaining capacity. This yields nearly identical bucket histograms for all splits while leaving a small residual pool of unused examples.

B. Prompting and Supervision Signal

All datasets are rendered in a unified chat-style format compatible with instruction-tuned models. Each example consists of:

- A `user` turn that (i) briefly describes the game, (ii) presents the current state in one of the text formats above, and (iii) includes a *STRICT OUTPUT FORMAT (MANDATORY)* block. This block specifies the exact number of required output lines K, the allowed symbols (integers or move tokens), and explicitly forbids explanations, English words, additional punctuation, blank lines, and numbering.
- An `assistant` turn that contains only the ground-truth target lines during training.

We use Qwen-style markers <im_start>`user`, <im_start>`assistant` and <im_end> to delimit turns. The generators also support in-context demonstrations, i.e., concatenating additional solved examples as few-shot chat turns before the final query. However, for all experiments reported here we set `num_shots`$= 0$ and evaluate models in a strictly zero-shot setting to keep the comparison between model scales and training regimes focused.

For local models we insert a special delimiter token <`SOL`> between prompt and target so that the input sequence is

$$\text{input} = \text{prompt} \,\|\, \texttt{<SOL>} \,\|\, \text{target} \,\|\, \texttt{<eos>},$$

and extend the tokenizer with this additional special token. During fine-tuning we mask all prompt and <`SOL`> tokens with label -100, so the loss is applied only to the target segment (plus end-of-sequence). This enforces conditioning on the full prompt while preventing the model from being penalized for reproducing the instruction text.

C. Models and Training Regimes

We compare a small instruction-tuned model with and without supervised fine-tuning to a much larger API-served model.

Small local model (1.5B). Our primary small model is **Qwen2.5-1.5B-Instruct**. We use a common `BaseModel` abstraction built on HuggingFace Transformers and PEFT. For evaluation in the zero-shot baseline setting, we load the pretrained model and tokenizer, add the <`SOL`> token, and run greedy decoding without updating any weights.

For supervised fine-tuning we apply low-rank adaptation (LoRA) to the attention projections. Specifically, we use rank $r = 16$, scaling factor $\alpha = 2r$, dropout 0.05, and target the `q_proj` and `v_proj` modules with `task_type=CAUSAL_LM`. The model is loaded in `bfloat16` with automatic device placement, and the embedding matrix is resized after adding <`SOL`>. Optimisation uses AdamW with learning rate 2×10^{-5}, batch size 4, gradient accumulation steps 4, weight decay 0.01, warmup ratio 0.1, and 10 epochs. Training is run with a simple 80/20 train/validation split obtained from a fold generator (`n_folds=1`). Although the codebase includes infrastructure

for Optuna-based hyperparameter search as well as multiple folds, these were not used in our experiments due to limited compute; we instead use the fixed hyperparameters above and a single fold for all fine-tuning runs.

Large reference model (72B). As a strong reference we use **Qwen2.5-72B-Instruct** accessed via the OpenRouter Chat Completions API. We wrap this in an `ApiModel` class that sends the full prompt text as a single user message at temperature 0.0 with a maximum of 64 new tokens, matching the greedy, deterministic decoding used for the local model. No fine-tuning is performed on this model; it is evaluated zero-shot on exactly the same test prompts as the small model.

D. Evaluation Protocol

Evaluation is unified across tasks and models. For each example we normalise the target by stripping whitespace, empty lines, chat control tags, and code-fence markers. The number of expected answer lines K (the prediction horizon) is the number of lines in this normalised target.

For the small model we append `<SOL>` to the chat-style prompt and use greedy decoding (`do_sample=False`, `temperature=0.0`) with `max_new_tokens` $= K + 15$. For the large API model we send the textual prompt (without chat markers) and read the returned assistant message.

Generations from both models are post-processed identically: we truncate at our markers, strip outer whitespace, split on newlines, drop empty and control lines, and keep the first K lines. If fewer than K lines remain, the prediction is invalid. Otherwise we normalise these K lines as above and compute *exact match*, the fraction of examples whose entire K-step prediction equals the gold sequence.

From the normalised sequences we compute generic metrics shared across tasks. *Exact match* is the fraction of fully correct K-step sequences. *Relative accuracy* is the fraction of positions where the predicted token matches the gold token, showing partial correctness. *Syntax accuracy* is the fraction of output lines that satisfy the task grammar (legal move tokens or in-range row indices), measuring basic rule-following. *Edit similarity* is the normalised Levenshtein similarity between predicted and gold sequences, capturing near-miss behaviour. *First error position* is the 1-based index of the first deviation from the optimal sequence (or $K+1$ if none), so larger values mean longer correct prefixes. For Fibonacci we also report mean absolute error (MAE) over integer values to capture numerical accuracy even when the discrete sequence is not exactly correct.

We additionally use light-weight structural metrics tailored to each game. For Fibonacci, *recurrence violation rate* is the fraction of positions where $x_t \neq x_{t-1} + x_{t-2}$, and *recurrence-stable prefix rate* is the fraction of prefix lengths whose entire prefix satisfies the recurrence; together they probe whether the model has internalised the underlying recursion.

For Towers of Hanoi, *invalid-move rate* is the share of moves that take from an empty peg or place a larger disk on a smaller one; *local optimality deviation* is the average distance between each legal move and the shortest-path optimal move; *stability error* flags peg configurations with disks out of size order; *peg-load divergence* is the average ℓ_1 difference between predicted and optimal disk counts per peg; *future-legal prefix rate* is the fraction of prefixes that are fully legal and extendable to some complete solution; and *first critical divergence* is the first step at which the predicted tower state becomes incompatible with the optimal trajectory. These metrics distinguish surface syntax errors from deeper violations of legal and near-optimal tower dynamics.

For the sliding puzzle, *invalid-move rate* is the fraction of moves that would move the blank off the board, *board divergence* is the average tile-wise Hamming distance between predicted and optimal boards when the sequences are simulated, and *future-legal prefix rate* is the fraction of prefixes that remain fully legal when replayed from the start state; together they measure how closely the model follows the puzzle's transition rules.

For N-Queens, *conflict pair rate* is the average number of attacking queen pairs when combining the prediction with the observed prefix, and *conflict-free prefix rate* is the fraction of prefixes whose partial boards contain no conflicts, directly quantifying global consistency with the N-Queens constraints.

IV. Results and Analysis

We evaluate five configurations on each game: the small local model Qwen2.5-1.5B-Instruct in zero-shot form ("1.5B-baseline"), the same model supervised-fine-tuned on 500, 1500, or 3000 training examples ("1.5B-500/1500/3000"), and the large reference model Qwen2.5-72B-Instruct evaluated zero-shot via the API ("72B"). All results are reported on the same 300-example test sets.

Across tasks we track generic sequence metrics: *exact match* (fraction of fully correct K-step sequences), *relative accuracy* (fraction of positions in the K-step horizon that match the gold sequence), *syntax accuracy* (fraction of output lines that obey the game's output grammar), *edit similarity* (normalized Levenshtein similarity between predicted and gold sequences), and *first error position* (1-based index of the first deviation from the optimal sequence; larger is better). For Fibonacci we additionally report mean absolute error (MAE) over integer values. Each game also has structural metrics tailored to its constraints (recurrence consistency for Fibonacci, move legality and tower configuration for Hanoi, solvable-move structure for the sliding puzzle, and queen-conflict structure for N-Queens), as defined in Sec. III.

A. Fibonacci Sequence Prediction

Table I summarizes the Fibonacci metrics. Supervised fine-tuning dramatically improves the 1.5B model over its zero-shot baseline across all sequence metrics, but the 72B model remains substantially stronger, especially on long-horizon consistency.

Relative to the 1.5B baseline, exact match increases from 1.7% to 24.3% at 3000 examples (roughly a $14\times$ improvement), and position-wise relative accuracy rises from 0.04 to 0.58. Edit similarity and first-error position show similar

TABLE I
FIBONACCI RESULTS. MAE IS SCALED BY 10^7; HIGHER IS BETTER FOR ALL OTHER METRICS EXCEPT RECURRENCE VIOLATION RATE. BOLD INDICATES THE BEST VALUE IN EACH COLUMN.

Model	Exact	Rel. Acc.	Syntax	Edit	First Err.	MAE	Rec. Viol.	Rec. Stable
1.5B-baseline	0.017	0.035	0.172	0.095	1.16	2.82	0.123	0.820
1.5B-500	0.213	0.493	0.687	0.567	2.99	2.13	0.247	0.705
1.5B-1500	0.233	0.554	0.686	0.598	3.26	2.11	0.169	0.796
1.5B-3000	0.243	0.582	0.688	0.606	3.41	2.13	0.164	0.825
72B	**0.980**	**0.986**	**0.990**	**0.987**	**5.62**	**0.51**	**0.002**	**0.996**

gains, indicating that the fine-tuned 1.5B model learns a useful approximation to the Fibonacci recurrence and can often continue the sequence correctly for several steps. MAE drops by about 25%, but remains one order of magnitude larger than the 72B model's error.

The structural metrics reveal that the large model almost perfectly obeys the Fibonacci recurrence (violation rate 0.002 and recurrence-stable prefixes > 0.99). The tuned 1.5B model recovers and slightly exceeds the baseline level of recurrence *prefix stability* by 3000 examples (0.825), but its recurrence violation rate ultimately settles slightly worse than the baseline. This suggests that fine-tuning teaches the small model to produce more numerically plausible short prefixes, but it still struggles to globally enforce the exact recurrence under long horizons.

B. Towers of Hanoi

Hanoi is substantially harder for both models: optimal sequences are long, highly structured, and sensitive to mistakes. We therefore focus more on legality and state-level divergence metrics alongside the generic sequence metrics (Table II).

TABLE II
TOWERS OF HANOI METRICS. LOWER IS BETTER FOR INVALID MOVE RATE, LOCAL DEVIATION, STABILITY ERROR, PEG DIVERGENCE, AND CRITICAL DIVERGENCE; HIGHER IS BETTER OTHERWISE. BOLD INDICATES THE BEST VALUE IN EACH COLUMN.

Model	Exact	Rel. Acc.	Syntax	Edit	First Err.	Inv. Move	Loc. Dev.	Stability	Peg Div.	Future Legal	Crit. Div.
1.5B-baseline	0.023	0.167	**1.000**	0.680	1.25	0.428	0.833	0.092	0.418	0.281	**1.25**
1.5B-500	0.033	0.116	**1.000**	0.488	1.30	0.458	0.884	**0.000**	0.430	0.387	1.30
1.5B-1500	0.027	0.133	**1.000**	0.504	1.37	0.384	0.867	**0.000**	0.411	0.451	1.37
1.5B-3000	0.037	0.153	**1.000**	0.511	1.42	**0.338**	0.847	**0.000**	0.407	**0.512**	1.42
72B	**0.057**	**0.226**	0.997	**0.744**	**1.58**	0.340	**0.774**	0.006	**0.374**	0.386	1.58

Exact match remains low for all systems ($\leq 5.7\%$), and even the 72B model diverges from the optimal move sequence after roughly one or two steps on average. Fine-tuning the 1.5B model yields only modest changes in sequence-level accuracy: relative accuracy fluctuates but does not approach the 72B performance, and edit similarity remains substantially lower.

The structural metrics show clearer learning. Future-legal prefix rate improves from 0.28 to 0.51 as the training set grows, indicating that longer prefixes of the predicted sequence consist entirely of legal moves and remain compatible with some future solution. Invalid move rate falls from 0.43 to 0.34, essentially matching the large model, and peg-load divergence also decreases slightly with more data. However, local optimality deviation remains high (≈ 0.85) and consistently worse than the 72B model, reflecting that the fine-tuned 1.5B model learns to generate plausible, legal move sequences but still struggles to track the globally optimal recursive pattern of Hanoi.

C. Sliding Puzzle (8-Puzzle)

For the sliding puzzle, the small fine-tuned model not only learns substantially from data but surpasses the large zero-shot model on several key metrics (Table III).

TABLE III
SLIDING PUZZLE METRICS. LOWER IS BETTER FOR INVALID MOVE RATE AND BOARD DIVERGENCE; HIGHER IS BETTER OTHERWISE. BOLD INDICATES THE BEST VALUE IN EACH COLUMN.

Model	Exact	Rel. Acc.	Syntax	Edit	First Err.	Inv. Move	Board Div.	Future Legal
1.5B-baseline	0.010	0.042	0.207	0.020	1.07	0.063	**0.433**	0.922
1.5B-500	0.090	0.279	**1.000**	0.369	1.46	0.166	0.462	0.668
1.5B-1500	0.133	0.310	0.990	0.396	1.59	0.046	0.462	0.904
1.5B-3000	**0.160**	**0.336**	**1.000**	**0.419**	**1.67**	**0.009**	0.461	**0.990**
72B	0.117	0.310	0.997	0.338	1.51	0.116	0.464	0.750

Exact match rises from 1% to 16% for 1.5B-3000, exceeding the 72B model's 11.7%. Relative accuracy and edit similarity show similar patterns: the fine-tuned 1.5B model slightly outperforms 72B on position-wise correctness. Syntax accuracy quickly saturates near 1.0, showing that both models have little difficulty respecting the move vocabulary when trained or instructed appropriately.

The structural metrics highlight that fine-tuning makes the small model more faithfully respect the transition dynamics of the puzzle. Invalid move rate drops to 0.009 with 3000 examples, far below both the baseline and the 72B model (0.116). Future-legal prefix rate reaches 0.99, meaning almost every progressive prefix of the predicted sequence is fully legal when replayed from the start state, compared to 0.75 for the large model. Board divergence remains similar across all systems, indicating that small improvements in move accuracy translate to only modest changes in final board configurations over the short horizons considered. Overall, on this task the fine-tuned 1.5B model clearly matches and in several respects exceeds the much larger zero-shot model.

D. N-Queens

The N-Queens task evaluates whether models can construct conflict-free board completions under variable board sizes and partial prefixes. Table IV shows that supervised fine-tuning yields monotonic gains and that with enough data the small model slightly outperforms the 72B model.

TABLE IV
N-QUEENS METRICS. LOWER IS BETTER FOR CONFLICT PAIR RATE; HIGHER IS BETTER OTHERWISE. BOLD INDICATES THE BEST VALUE IN EACH COLUMN.

Model	Exact	Rel. Acc.	Syntax	Edit	First Err.	Conflict Rate	Conflict-Free
1.5B-baseline	0.033	0.131	0.781	0.370	1.15	0.140	0.348
1.5B-500	0.123	0.267	**1.000**	0.535	1.43	0.086	0.331
1.5B-1500	0.130	0.283	**1.000**	0.544	1.48	0.077	0.362
1.5B-3000	**0.167**	**0.319**	**1.000**	**0.567**	**1.60**	**0.069**	**0.427**
72B	0.153	0.308	0.983	0.556	1.54	0.078	0.369

Exact match increases five-fold from 3.3% to 16.7% as we scale from 0 to 3000 training examples. Relative accuracy and edit similarity exhibit similar gains, and first-error position moves from near-immediate failure (≈ 1.15) to surviving more than one and a half positions on average. The constraint-oriented metrics show that fine-tuning meaningfully improves the internal consistency of predicted partial boards: conflict

pair rate nearly halves from 0.14 to 0.069, while conflict-free prefix rate rises from 0.35 to 0.43, indicating that as more rows are predicted, the partial assignment remains conflict-free significantly more often. Compared to the 72B model, the 1.5B-3000 configuration attains slightly higher exact match and relative accuracy, lower conflict rate, and higher conflict-free prefix rate. On this structured combinatorial task, the small fine-tuned model therefore slightly surpasses the large zero-shot model in both syntactic and structural correctness.

E. Cross-Task Discussion

Taken together, these results answer both of our central questions. First, *can a small language model learn nontrivial multi-step reasoning from supervised signals alone?* Across all four games the 1.5B model shows clear learning: exact match, relative accuracy, and edit similarity consistently improve with more data; syntax and legality metrics move toward perfect rule-following; and structural metrics such as recurrence-stable prefixes, future-legal prefixes, and conflict-free prefixes all trend in the desired direction. Even on Hanoi, where global optimality remains challenging, the fine-tuned model produces longer stretches of legal moves that keep the tower configuration closer to a valid solution.

Second, *can such a small model match or exceed a much larger model on multi-step tasks under increasing context lengths?* Our results show a nuanced picture. On Fibonacci, the 72B model remains clearly superior: it achieves near-perfect exact match and recurrence enforcement even as the prediction horizon grows, while the 1.5B model only partially internalizes the recurrence. On Hanoi, both models struggle, but the 72B system retains an advantage in local optimality and overall move quality. In contrast, on the sliding puzzle and N-Queens tasks, the fine-tuned 1.5B model with 3000 examples matches or slightly surpasses the 72B model in exact match, position-wise accuracy, and several structural consistency metrics. In particular, it achieves near-perfect move legality in the sliding puzzle and lower queen-conflict rates in N-Queens.

Overall, these results indicate that a small instruction-tuned model can indeed be trained via supervised fine-tuning to reach, and in some cases exceed, the reasoning performance of a much larger model on discrete, algorithmic multi-step tasks, especially when the main difficulty lies in respecting local constraints and short-to-medium-range structure. However, for tasks that require maintaining a precise numeric recurrence over long horizons, or tracking deeply recursive optimal policies such as in Towers of Hanoi, substantial performance gaps remain even after fine-tuning.

V. Conclusion

In this work we investigated whether a small instruction-tuned language model can be trained via supervised fine-tuning to match or exceed the multi-step reasoning performance of a much larger model on structured algorithmic games. Using four synthetic tasks with well-defined optimal policies, we compared a 1.5B-parameter Qwen2.5 model in zero-shot and fine-tuned configurations against a 72B-parameter Qwen2.5 model evaluated zero-shot via an API. The results show that the small model does learn non-trivial reasoning behaviour from supervised signals alone: across all games, fine-tuning substantially improves exact-match accuracy, position-wise correctness, and sequence-level similarity metrics relative to the zero-shot 1.5B baseline. The structural metrics further indicate that the model internalizes task-specific constraints rather than simply memorizing surface patterns, as evidenced by improved recurrence stability in Fibonacci, higher rates of legal future prefixes in Hanoi and the sliding puzzle, and lower conflict rates in N-Queens.

With respect to our main question, the answer is nuanced. For tasks that primarily require enforcing local constraints and reasoning over short-to-medium horizons, such as the sliding puzzle and N-Queens, a fine-tuned 1.5B model trained on a few thousand examples can match or slightly surpass a much larger 72B model operating in a purely zero-shot regime. In these settings, supervised fine-tuning effectively closes most of the performance gap associated with scale, and the small model becomes highly reliable in terms of syntax, legality, and structural consistency. In contrast, on Fibonacci and Towers of Hanoi, where strong performance demands precise recurrence tracking or long-range recursive structure, the large model maintains a clear advantage even after fine-tuning: the 1.5B model learns to extend sequences correctly for a few steps and to generate legal move sequences more often, but it still fails to match the large model's near-perfect recurrence enforcement or its superior local optimality in Hanoi. Overall, the results demonstrate that small models can indeed learn meaningful multi-step reasoning skills and can rival much larger models on some structured tasks, but that scale still matters for problems that stress deep recursion and long-horizon consistency.

VI. Future Work

Several extensions could deepen these findings and reduce the remaining performance gaps. First, our generators natively support one-shot and few-shot prompting, but all reported experiments enforce a strictly zero-shot evaluation regime. A natural next step is to systematically evaluate both the small and large models under one-shot and few-shot prompting on the same test sets, and to study how in-context examples interact with supervised fine-tuning and context length. Second, the training pipeline already includes infrastructure for Optuna-based hyperparameter search and multi-fold cross-validation, which we disabled for computational reasons. Running a targeted hyperparameter sweep over learning rates, batch sizes, LoRA ranks, and warmup schedules is likely to yield stronger 1.5B baselines and could clarify how much of the current gap is due to suboptimal optimisation rather than representational limits. Finally, scaling the difficulty of the games themselves, for example by increasing Fibonacci horizons, using larger boards and more disks in N-Queens and Hanoi, or constructing longer optimal paths in the sliding puzzle, would allow us to probe how far supervised fine-tuning can push small models under truly long-context settings, and where more advanced techniques such as curriculum learning, multi-task

joint training, or reinforcement learning from search-based teachers become necessary.

VII. Individual Contribution and Professionalism

All four team members contributed to the core idea, experimental design, training pipeline, and writing. Xristopher Aliferis led the Towers of Hanoi task, including instance generation, state formatting, legality-aware simulation, and analysis of the Hanoi metrics. Christopher Lam led the Fibonacci task, implementing the sequence generator, horizon sampling, Fibonacci-specific structural metrics, and interpreting how models learn or violate the recurrence. Carmel Kurland was responsible for the N-Queens pipeline, including OR-Tools solution generation, partial-board prompts, constraint-based metrics, and analysis of combinatorial reasoning. Devraj Nagpal implemented the sliding puzzle environment, including BFS-based solution paths, board parsing, sliding-specific metrics, and result analysis. The base model abstractions, LoRA fine-tuning code, OpenRouter wrapper, metric scripts, and the manuscript were developed collaboratively, with all members reviewing each other's sections. Work was coordinated through regular meetings and version control with clear commit histories, and academic integrity was maintained throughout by writing our own code (beyond standard libraries), properly citing prior work, and reporting only genuinely obtained results.

VIII. Creativity, Innovation, and Reproducibility

The project goes beyond a standard benchmark by unifying four structurally different games into a single next-K-step reasoning framework with shared prompting and evaluation. For each game we designed custom structural metrics grounded in lightweight simulators or constraint checkers (recurrence consistency, move legality and tower state, board dynamics, and queen conflicts), providing a richer view of reasoning quality than exact match alone. All datasets are generated deterministically with fixed seeds, train/test splits are saved as JSONL, and predictions plus summaries are logged in a consistent directory structure, making the experiments easy to rerun or extend with new models. The codebase already supports few-shot prompting and hyperparameter search, even though they were intentionally disabled in this study for compute reasons, positioning the work as a reusable and extensible testbed for future research on multi-step reasoning in small versus large language models.

REFERENCES

[1] J. Kaplan, S. McCandlish, T. Henighan *et al.*, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[2] T. B. Brown, B. Mann, N. Ryder *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[3] A. Chowdhery, S. Narang, J. Devlin *et al.*, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[4] H. Touvron, T. Lavril, G. Izacard *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[5] J. Wei, X. Wang, D. Schuurmans *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2022.

[6] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *arXiv preprint arXiv:2205.11916*, 2022.

[7] K. Cobbe, V. Kosaraju, M. Bavarian *et al.*, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

[8] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli, "Analysing mathematical reasoning abilities of neural models," *arXiv preprint arXiv:1904.01557*, 2019.

[9] A. Srivastava, A. Rastogi, A. Rao *et al.*, "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," *arXiv preprint arXiv:2206.04615*, 2022.

[10] D. Hendrycks, C. Burns, S. Basart *et al.*, "Measuring massive multitask language understanding," *arXiv preprint arXiv:2009.03300*, 2021.

[11] J. Wei, Y. Tay, R. Bommasani *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[12] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma, "An explanation of in-context learning as implicit bayesian inference," *arXiv preprint arXiv:2111.02080*, 2021.

[13] M. Nye, A. J. Andreassen, G. Sharma *et al.*, "Show your work: Scratchpads for intermediate computation with language models," *arXiv preprint arXiv:2112.00114*, 2021.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Technical Report*, 2019.

[18] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning Workshop*, 2015.

[19] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: Smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[20] E. J. Hu, Y. Shen, P. Wallis *et al.*, "LoRA: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[21] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," *arXiv preprint arXiv:2305.14314*, 2023.

[22] L. Ouyang, J. Wu, X. Jiang *et al.*, "Training language models to follow instructions with human feedback," *arXiv preprint arXiv:2203.02155*, 2022.

[23] N. Stiennon, L. Ouyang, J. Wu *et al.*, "Learning to summarize with human feedback," *arXiv preprint arXiv:2009.01325*, 2020.

[24] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[25] A. Wang, Y. Pruksachatkun, N. Nangia *et al.*, "Superglue: A stickier benchmark for general-purpose language understanding systems," *arXiv preprint arXiv:1905.00537*, 2019.

[26] Y. Tay, M. Dehghani, S. Abnar *et al.*, "Long range arena: A benchmark for efficient transformers," *arXiv preprint arXiv:2011.04006*, 2020.