

Overcoming a Learning Bottleneck in Impartial Games: A Circuit Complexity Approach

Søren Riis

S.RIIS@QMUL.AC.UK

Centre for Fundamentals of AI and Computational Theory

Queen Mary University of London

Mile End Road, London E1 4NS, United Kingdom

Abstract

Why can neural networks struggle to learn optimal play in a game with a simple, efficient solution algorithm? Motivated by empirical difficulties with parity in impartial games, we use NIM to study a representational bottleneck and three ways around it. We show that a restricted model of shallow, fixed-scale quantised networks lies in AC^0 , a circuit class unable to compute parity on arbitrarily large inputs. This prevents optimal single-position play across all board sizes within the model. The obstruction also guides architectural design: two-frame inputs allow restoration from a known zero-nim-sum position; complementary rollout policies reveal the missing invariant; and persistent recurrence accumulates it row by row. Each route retains shallow network components while supplying additional information or processing time. Supervised experiments illustrate the benefits of two-frame restoration and multiple rollout heads; recurrence is a theoretical construction. These results establish representational possibilities, not guarantees of successful training: AC^0 does not characterise practical learnability. Core mathematical results and constructions have been checked in Lean, with classical complexity results as explicit external inputs. The interactive edition and complete Lean sources are available online (Riis, 2026a,b).

Keywords: Impartial games; NIM; circuit complexity; quantised inference; learning bottlenecks.

1 Introduction

Game play has historically served as the “Formula 1” of artificial intelligence: a high-speed, controlled proving ground where engineering limits are tested and new paradigms are born. From Deep Blue in chess to AlphaGo in Go, games have repeatedly driven advances in search, planning, and reinforcement learning (Silver et al., 2016, 2017, 2018; Schrittwieser et al., 2020). Minsky and Papert’s analysis of perceptron limitations illustrates how representational restrictions can motivate architectural changes (Minsky and Papert, 1969).

In NIM, two players take turns removing any positive number of tokens from a single heap; the player taking the last token wins. It belongs to the broad class of *impartial games*: both players have the same legal moves from a given position. Many such board games can be thought of as games with shared pieces, whereas in *partisan games*, such as chess, legal moves depend on which player is to move. NIM is central to impartial-game theory because every finite terminating impartial position under the last-move-wins convention is equivalent, in sums of games, to a single NIM heap (Section 4).

NIM poses a particularly revealing challenge: optimal moves can be found by a simple, efficient algorithm, yet learning that algorithm can be difficult. The puzzle is the gap

between the ease of computing the solution once the rule is known and the difficulty of discovering it through training. Friedman’s 2017 AI challenge asked whether a system could discover perfect NIM play without being given the winning strategy (Friedman, 2017a).

Our earlier work with Bei Zhou provides the empirical starting point. In *Impartial Games: A Challenge for Reinforcement Learning*, Zhou and Riis found that AlphaZero-style learning deteriorated as NIM boards grew (Zhou and Riis, 2026). Subsequent parity experiments examined how noisy labels can obstruct curriculum learning (Zhou and Riis, 2023); see also Zhou’s thesis (Zhou, 2024). These findings make parity a useful test of the gap between learning familiar patterns and discovering an exact rule that works across positions.

Here we turn that difficulty into a question about computation. The class AC^0 consists of polynomial-size Boolean circuits built from AND, OR and NOT gates, with a number of layers that remains fixed as the input grows. Such circuits can combine many local tests, but cannot compute global parity: whether the total number of input 1s is odd or even. This is the feature that makes the class useful here: it provides a precise model with a provable version of the parity limitation seen in the experiments. We establish the connection for an explicitly restricted neural model, rather than assuming that all neural networks lie in AC^0 .

The value of identifying a bottleneck is that it tells us what an architectural change must supply. Instead of asking a shallow network to rediscover a global invariant from every board, we can expose a useful transition, obtain complementary information from rollouts, or accumulate the invariant over time. Our contribution is this *barrier-to-bypass* approach: a computational obstruction guides three concrete constructions. Representability in AC^0 is a necessary condition for a fixed-latency computation in our restricted model; it is neither sufficient for successful training nor a universal condition on what neural networks can learn.

NIM itself does not depend on game history: the current position completely determines its legal moves and optimal play. History can nevertheless supply useful *computational context* to a restricted network. The first and third bypasses use such context in different ways: an earlier board exposes a local change, while recurrent memory retains information accumulated during row processing and across positions.

The central observation is that while the *state* can be hard to process, the *transition* can be easy: under the promise that the preceding position has nim-sum zero, short history converts winning-move selection into local difference computation and restoration. A second route uses distinct rollout policies to extract complementary bits of the invariant. A third route processes each position sequentially, retaining recurrent state as rows and then successive positions arrive. Here a sequence of simple updates computes the global invariant, and memory connects the new position to the preceding one. These mechanisms use different resources: explicit transition information, multiple rollout channels, or a growing recurrent computation horizon. The distinction between a shallow component and the complete computation is essential. Two-frame restoration is a shallow map on a promised pair of boards. In the other two constructions, each policy step or recurrent update is shallow, but generating the information needed for a decision takes a sequence of such steps. Thus the bypasses preserve simple network components while changing the input information or total processing time. The complete sequential computation need not itself lie in AC^0 . Figure 1 summarises the three routes for NIM.

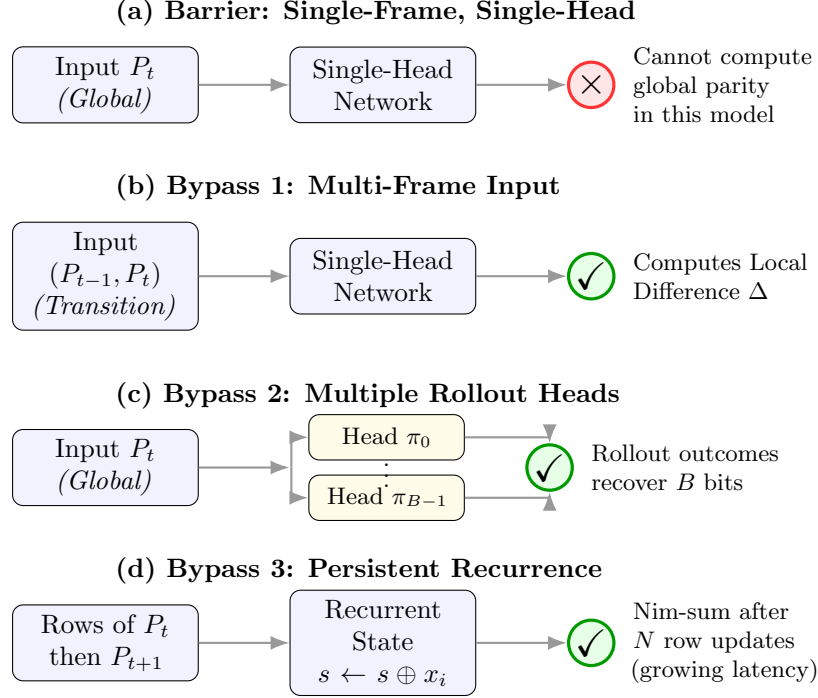


Figure 1: The parity barrier and three structural bypasses. (a) Fixed-scale quantised inference (FSQI) with fixed latency cannot compute global parity across arbitrarily large boards. Here fixed latency means a bounded number of sequential computation steps, independent of board size. (b) Two frames expose Δ for restoration when the predecessor has nim-sum zero. (c) Distinct policy heads recover complementary information through complete rollouts. (d) Persistent recurrence accumulates the nim-sum row by row and retains summaries across positions. Each update has constant depth; the full scan has depth growing with N .

1.1 Contributions

The results follow a single question: what must be added to a shallow learner to make the required strategy representable? We first identify the obstruction, then show how each bypass supplies a different missing resource.

- **AC^0 containment for fixed-scale quantised inference.** We formalise a fixed-grid Boolean literal-threshold model and its bounded-unrolling recurrent extension. Both are simulable by AC^0 circuits (Theorem 2.4). An attention-style extension is covered only if its scoring, selection and aggregation operations can each be implemented by the permitted Boolean threshold gates. This is a restriction on the computation, not a requirement that every game have the same board size. Within a play sequence we use a fixed layout of N heap rows and B bits per heap, retaining empty heaps as zero rows. The asymptotic statements concern families of networks for growing input lengths NB .

- **Single-frame impossibility for strong mastery.** Under this model, no single-frame policy/value network can strongly master NIM in the worst case (Theorem 5.2).
- **Two-frame bypass via local differences and restoration.** With two consecutive frames, the agent can compute $\Delta(P, P')$ in AC^0 (Lemma 5.4). When P has nim-sum zero, an AC^0 move selector restores zero after the opponent’s move (Proposition 5.5). This is conditional restoration mastery, not an arbitrary-start verifier.
- **Planning analysis and the single-policy bottleneck.** We analyse a restricted measurement interface inspired by rollouts, rather than a general model of MCTS. With B structured policies one can recover all nim-sum bits (Proposition 5.9). Root and one-heap-modification queries to one fixed member of that family expose only one unknown linear functional (Proposition 5.10). With an explicitly bounded readout, this yields the distributional bound in Theorem 5.12.
- **Persistent recurrence via sequential row updates.** A constant-depth recurrent update accumulates the nim-sum over N heap rows and retains a summary across position separators (Proposition 5.13). With a replay of the current board or stored heap rows, this supports optimal move selection. The construction relaxes fixed total latency, not fixed-scale quantisation (Section 5.5).
- **Controlled empirical comparison and learnability discussion.** On 20-heap, 4-bit NIM with 10^6 supervised restoration examples, a one-frame model has low restoration accuracy while a two-frame model learns restoration (Section 7.2). In a complementary single-frame experiment, multi-head FSM-controlled shootouts achieve perfect win/loss position classification while the single-head baseline remains bottlenecked by one-bit information (Section 7.3). We also discuss why AC^0 is a useful abstraction for what is robustly feasible under fixed-scale quantised inference (Section 6.3).

1.2 Scope and limitations

Our theorems are *worst-case representational* statements about what can be computed under explicit architectural constraints. They are not training-complexity results, and they intentionally abstract away from several practical details, such as continuous-valued activations, floating-point arithmetic, normalisation layers and the dynamics of optimisation.

- **Fixed-scale quantisation / fixed latency.** The AC^0 simulation (Theorem 2.4) assumes constant depth and a *fixed-scale, bounded-range quantisation* regime: weights/thresholds lie on a fixed grid whose range/resolution do not scale with the input length (cf. Remark 2.2). Architectures that rely on fan-in dependent re-normalisation, biases/thresholds growing with dimension, or depth growing with n fall outside this abstraction. In particular, a fixed recurrent update rule need not have fixed total latency: the third bypass deliberately allows N successive row updates per position.
- **Formal model vs. experimental networks.** The formal units have nonnegative weights on Boolean literals and bounded thresholds. The experiments instead use ordinary floating-point signed linear layers, ReLU and LayerNorm, with Log-Softmax in the loss.

They test the benefit of transition information under the reported training protocol, not the asymptotic lower bound or the deployment behaviour of quantised networks.

- **Stylised planning interface.** The single-channel lower bound permits only root and one-heap-modification queries to a fixed structured policy, and requires a polynomial-size, constant-depth final readout. It is not a bound on arbitrary rollout policies, unrestricted query transformations, or an unbounded adaptive MCTS controller.
- **No guarantee of learnability.** The multi-policy-head and recurrent constructions concern *existence* of suitable computations; we do not claim that self-play efficiently discovers the structured heads or an exact recurrent accumulator. Our experiments use controlled supervised restoration and policy imitation; self-play discovery is left for a separate study.

1.3 Paper organisation

Section 2 introduces the fixed-scale quantised inference model and the AC^0 simulation theorem. Section 3 situates our contribution in circuit complexity, modern sequence models, and reinforcement learning for combinatorial games. Section 4 recalls NIM and the nim-sum characterisation. Section 5 presents the main impossibility and bypass results: two-frame restoration, rollout channels (Section 5.4), and persistent recurrence (Section 5.5). Section 6 discusses modelling choices and learnability considerations. Section 7 reports experiments. Appendices contain formal model definitions and proofs.

2 Theoretical framework and neural network models

2.1 Fixed-scale quantised inference and AC^0 simulation

We consider neural inference models that are *(i)* bounded depth, *(ii)* polynomial size, and *(iii)* operate in a *fixed-scale quantised* regime: all weights and thresholds lie on a fixed quantisation grid of bounded dynamic range that does *not* change with the input length. The fixed scale is motivated by limits on the *effective precision* of neural computation, including sensitivity to noise: distinctions that can be used reliably may be much coarser than the rounding resolution of 32-bit floating-point arithmetic. Noise motivates this abstraction; it does not by itself imply the gate and threshold assumptions below.

Our Boolean threshold gates sum nonnegative weights on inputs that may be negated. Negation allows inhibitory as well as excitatory effects: for a Boolean input x , the identity $-ax = a(1 - x) - a$ replaces a negative coefficient by a positive weight on the negated input, with an adjustment to the threshold. The substantive restriction is that the resulting weights and thresholds remain on the fixed, bounded grid; this is not a model of arbitrary signed-weight or INT8/fixed-point inference.

Concretely, we fix constants:

- a *quantisation resolution* parameter captured by a fixed denominator $D \in \mathbb{N}$ (grid step $1/D$),
- a *dynamic-range* bound $W \in \mathbb{N}$ (maximum magnitude W/D),

- and (for recurrent/attention models) a finite time window $T \in \mathbb{N}$.

We define

$$\mathcal{Q}_{W,D} = \left\{ \frac{\ell}{D} : \ell \in \{0, 1, 2, \dots, W\} \right\},$$

a finite set independent of input length. The threshold bound applies *after* choosing the literal representation. Replacing a signed term $-ax$ by $a(1-x)$ raises the threshold by a . With many such terms, the adjusted threshold may grow with the input length. What matters is the amount of evidence needed to activate a gate: our model bounds the number of satisfied input literals that suffice, whereas an unrestricted majority gate needs a number that grows with its inputs. Negative coefficients themselves are not the obstruction; the bound on the adjusted threshold is essential to the simulation.

Each neuron computes a thresholded weighted sum of Boolean inputs. Formal definitions for NN, RNN and LTST are in Appendix A.

Definition 2.1 (Quantised inference). *A network NN in this class has constant depth and polynomially many threshold neurons, with weights and thresholds in $\mathcal{Q}_{W,D}$. Analogously, fixed-scale quantised RNN and LTST models have constant depth and polynomial size per time step and operate within a fixed time window T . For the simulation theorem, T bounds the entire unrolled computation from an input-independent initial state, including any computation used to form a retained state.*

A bound on the number of preceding states consulted by one update is insufficient: even $h_{r+1} = h_r \oplus x_r$ can accumulate parity over an arbitrarily long stream. Section 5.5 exploits precisely this distinction by allowing the number of recurrent updates to grow with the number of rows.

Why fixed-scale quantisation suggests an AC^0 abstraction. The key structural property is *fixed scale*: the smallest nonzero weight is $1/D$ and the largest threshold is W/D , with W, D independent of the input length. Consequently, whenever a threshold unit outputs 1, there is a satisfying *witness* consisting of at most W input literals whose contributions already cross the threshold. This yields a constant-width DNF expansion for each unit (Lemma B.1), and composing a constant number of layers yields an AC^0 simulation (Theorem 2.4).

We emphasise that this is not a statement about floating-point arithmetic per se. The abstraction intentionally excludes *scale-adaptive* regimes where thresholds/biases grow with fan-in or where weights shrink with dimension (e.g. $1/n$ or $1/\sqrt{n}$ normalisations), which move the analysis toward threshold-circuit classes such as TC^0 . This class allows constant-depth, polynomial-size circuits with general threshold gates, including majority gates; unlike AC^0 , it can compute parity.

Remark 2.2 (Scope of the AC^0 abstraction). *Theorem 2.4 formalises a fixed-latency fixed-scale quantised regime in which each unit uses parameters drawn from a bounded-range grid $\mathcal{Q}_{W,D}$ with W, D independent of the input length. This yields the constant-witness property exploited in Lemma B.1.*

The abstraction is deliberately conservative: it does not model architectures that rely on fan-in dependent re-scaling, numerically large signed biases/thresholds that grow with

dimension, or deep computations whose depth scales with n . Allowing such scale-adaptive behaviour leads naturally to richer threshold-circuit models (e.g. TC^0), in which parity-like computations can become representable.

Remark 2.3 (Local vs. global parity). When we say “parity is hard” we mean global parity across the full Boolean input of length $n = NB$. In contrast, parity on $O(1)$ bits is trivial in AC^0 (hardwire the truth table), and more generally XOR of a constant number of B -bit blocks is AC^0 -computable. This local/global separation underlies both the two-frame bypass via the number difference $\Delta(P, P')$ and the recurrent bypass, which composes local XOR updates over a growing number of steps.

Property of deployed inference	Captured here	Not captured here
Fixed latency / bounded depth	constant depth L (and window T)	depth growing with n
Fixed-scale quantisation (bounded range)	weights/thresholds in fixed grid $\mathcal{Q}_{W,D}$	fan-in dependent scaling or thresholds growing with n
Parallel bit operations	unbounded fan-in AND/OR/NOT	exact global counting / exact parity on unbounded inputs
Signed effects	via input negations (Lemma B.1)	large signed dot-product accumulation

Notation convention (input length). We write N for the number of heaps and B (or k in experiments) for the bit-width used to encode each heap size. The resulting Boolean input length is $n := NB$. When we discuss AC^0 circuit size/depth, n always refers to this Boolean input length.

Circuit complexity primer. AC^0 is the class of Boolean functions computable by constant-depth, polynomial-size circuits with unbounded fan-in AND/OR/NOT gates. A classical result is that $\text{PARITY} \notin \text{AC}^0$. In contrast, TC^0 augments AC^0 with unbounded fan-in threshold gates and can compute parity. Our focus here is a fixed-scale quantised regime in which thresholds do not scale with fan-in; in that regime, each threshold unit admits a constant-witness DNF expansion and the overall computation collapses to AC^0 (Theorem 2.4).

2.2 Simulation by circuits

We state the simulation theorem for the literal-threshold model and its explicitly gate-realised extensions.

Theorem 2.4 (Simulation by AC^0). *Fix constants $W, D \in \mathbb{N}$ and a time window $T \in \mathbb{N}$. Let $\{\mathcal{M}_n\}_{n \geq 1}$ be any family of models of type NN, RNN (with window T), or LTST (with window T) such that:*

- (1) (**Polynomial size**) the number of neurons/units in $\mathcal{M}_n$ is at most $\text{poly}(n)$.

- (2) (**Constant depth**) the number of layers is a fixed constant L (independent of n); for RNN/LTST the entire unrolled computation uses at most T time steps from an input-independent initial state.
- (3) (**Permitted primitives**) every operation is realised by the Boolean literal-threshold gates of Definition A.1; all weights and thresholds lie in $\mathcal{Q}_{W,D}$. For the attention-style extension this includes score computation, coefficient selection and aggregation.

Then the Boolean function computed by $\mathcal{M}_n$ can be computed by an AC^0 circuit family of polynomial size and constant depth (depending only on L, T, W, D).

3 Related Work

Circuit complexity and neural networks. Parity is outside AC^0 (Furst et al., 1984; Håstad, 1986), while constant-depth threshold circuits (TC^0) can compute it. Attention results depend on the precise model: Hao, Angluin and Frank analyse hard-attention variants, including AC^0 restrictions for unique hard attention (Hao et al., 2022). Merrill and Sabharwal study the additional power of intermediate decoding over a TC^0 baseline (Merrill and Sabharwal, 2024); that containment does not rule out parity. Their sequential-computation perspective is relevant to our recurrent bypass, although the architectures and formal assumptions differ. Our contribution applies these distinctions to NIM policies and a restricted rollout-measurement interface.

Quantised inference and deployment constraints. Low-bit inference motivates studying bounded numerical resources (Jacob et al., 2018; Dettmers et al., 2022). Our constant-witness proof additionally requires nonnegative literal weights, bounded thresholds and Boolean intermediate states. These are explicit mathematical assumptions, not properties inferred merely from a quantisation format. Standard signed linear layers and scale-adaptive normalisation are outside the theorem unless separately reduced to the permitted gates.

Reinforcement learning in combinatorial games. While RL has mastered domains such as Go and chess, impartial games remain a useful stress test because optimality is governed by exact algebraic invariants rather than approximate pattern matching. Zhou and Riis (Zhou and Riis, 2026, 2023) studied reinforcement-learning difficulties in impartial games. Our lower bound identifies a representational obstruction in a restricted model; determining which mechanisms explain failures in ordinary neural training requires separate experiments.

History and memory in RL. Using short history windows is standard in partially observable settings, where stacked frames recover hidden variables such as velocity (Mnih et al., 2015). Our point is different and perhaps surprising: even in Markovian, fully observable games—where the current state is informationally sufficient and the past is theoretically irrelevant—history can reduce *computational* complexity by exposing local differences that would otherwise require expensive search or unbounded-depth inference to extract from the current state alone. In our restoration setting, the previous frame carries a promise established by an earlier successful move: its nim-sum is zero. That promise makes a local difference sufficient for the next response. Establishing the promise initially remains a separate computational task.

4 Impartial games and NIM

We briefly recall standard definitions; see (Berlekamp et al., 2001–2004) for extensive background.

4.1 Impartial games

Definition 4.1 (Impartial game). *A finite terminating impartial game is a pair $(\mathcal{P}, \mathcal{M})$ where $\mathcal{P}$ is a nonempty set of positions and $\mathcal{M}(p) \subseteq \mathcal{P}$ is the (finite) set of legal moves from p , with the property that the move relation does not depend on which player is to play (symmetry). We assume the move relation is acyclic and play terminates. We use the last-move-wins convention, usually called normal play: a player with no legal move loses.*

A position with no legal moves is terminal.

The last-move-wins convention also provides a convenient encoding of other decisive endings. A move that wins by capturing a designated piece or creating a specified configuration can simply lead to a position with no further legal moves. If instead reaching a special position loses for the player who made the move, give the opponent one forced move to a new terminal position. These encodings preserve the winner of the individual play; they must retain player-independent move rules to remain impartial. The theory below concerns the resulting normal-play game.

4.2 NIM

Definition 4.2 (NIM). *A NIM position consists of h heaps with sizes $x_1, \dots, x_h \in \mathbb{N}_0$. A legal move reduces exactly one heap: choose i with $x_i > 0$ and replace x_i by some $x'_i < x_i$.*

4.3 Sprague–Grundy theory and nim-sum

The breadth of impartial games makes the Sprague–Grundy theorem remarkable: every finite terminating impartial position under normal play is equivalent to a single NIM heap of some size, called its *number* or Grundy value. Equivalence here means that replacing the position by that heap preserves the winning or losing status of its sum with any other such game. In a *disjunctive sum*, a move is made in exactly one component. The number of a sum is the bitwise XOR of the component numbers; in particular, the number of a NIM board is the XOR of its heap sizes.

Definition 4.3 (Nim-sum). *For heap sizes $x_1, \dots, x_h$, define*

$$\text{nim}(x_1, \dots, x_h) = x_1 \oplus x_2 \oplus \dots \oplus x_h,$$

where $\oplus$ denotes bitwise XOR.

Theorem 4.4 (Classical NIM characterisation). *A NIM position $(x_1, \dots, x_h)$ is losing for the player to move iff $\text{nim}(x_1, \dots, x_h) = 0$. Moreover, from any winning position (nim-sum $\neq 0$) there is a move to a losing position (nim-sum $= 0$).*

Why NIM is a useful barrier example. For a disjunctive sum of independent impartial components, the Grundy value is the XOR of the component numbers. If these numbers can

vary independently so that their aggregation contains an unbounded parity subproblem, NIM isolates the resulting aggregation obstruction. Learning NIM would therefore address a fundamental component of impartial-game play, but would not by itself solve self-play learning for the wider class: determining component nimbers can be computationally difficult (Section 8.3). Nor does every impartial game contain a hard parity aggregation task. Our unconditional lower bound concerns NIM; extensions require an explicit encoding and reduction.

5 Main results: impossibility and possibility

We first establish the single-frame barrier under fixed total latency, then develop three bypasses: recent-history inputs, multiple rollout channels, and persistent row-wise recurrence.

5.1 Mastery notions

We distinguish three notions of mastery:

Definition 5.1 (Strong, weak, and restoration mastery). *An agent achieves strong mastery of NIM if it chooses a winning move at every N -position where it is to move, including positions reached by arbitrary legal play by both players. Reachability is relative to the specified initial positions and player role; unrestricted strong mastery allows every initial position.*

It achieves weak mastery relative to an initial position p_0 and a designated player role if, whenever that role has a winning strategy from p_0 , its own strategy wins against every opponent strategy. The guarantee concerns histories consistent with its own preceding actions, not every position reachable under arbitrary actions by both players. For asymptotic claims, the initial positions form a specified family. Strategies may depend on the complete history. Opponent moves are legal; a missing or illegal agent move counts as a loss. On an initially losing position the weak-mastery condition imposes no requirement.

It achieves restoration mastery if, whenever the opponent moves from a P -position Q with $\text{nim}(Q) = 0$ to a position Q' , the agent responds with a move $Q' \rightarrow R$ such that $\text{nim}(R) = 0$.

We use standard impartial-game terminology: a P -position is losing for the player to move, and an N -position is winning. For NIM, Theorem 4.4 says that P -positions are exactly the positions with nim-sum 0. Equivalently, weak mastery requires the agent to move to a P -position at every one of its decision histories consistent with its own earlier actions, starting from an initial position winnable in its designated role. This distinction allows a simple strategy to master a family without solving positions it never encounters under its own play; Appendix D gives examples.

5.2 Single-frame impossibility results

In a single-frame representation, the network sees only the current position. To play optimally, it must in effect determine whether the nim-sum is zero and, if not, find a move producing nim-sum zero. The core obstacle is that global nim-sum computation requires parity-like

behaviour across many bits, and parity is not in AC^0 (Furst et al., 1984; Håstad, 1986; Smolensky, 1987; Razborov, 1987).

Theorem 5.2 (Single-frame parity barrier). *Consider the restricted family of NIM positions $P(\mathbf{b}) = (2, b_1, \dots, b_m)$ where $b_i \in \{0, 1\}$. Any single-frame agent that achieves strong mastery must compute the parity $b_1 \oplus \dots \oplus b_m$. Since $\text{PARITY} \notin AC^0$, no polynomial-size, constant-depth fixed-scale quantised network can strongly master NIM.*

Proof. Since each $b_i \in \{0, 1\}$, these positions have number $2 \oplus (b_1 \oplus \dots \oplus b_m)$, which is either 2 or 3. A winning move must change the first heap: changing any other heap leaves the leading binary digit of the nim-sum unchanged. To obtain number zero, the first heap must be reduced to $b_1 \oplus \dots \oplus b_m$. Thus the unique winning move removes two tokens when this parity is zero and one when it is one. Encoding each heap in two bits uses only constants and wiring, and the low bit of the replacement heap is the input parity. This gives a constant-depth, polynomial-size reduction to winning-move selection, although every encoded position is winning. $\square$

5.3 First bypass: two-frame restoration

The fundamental idea is to avoid computing the *global* nim-sum from scratch. Instead, we exploit that each move changes only one heap, so the *number difference* between consecutive positions is locally computable.

Definition 5.3 (Number difference). *Given consecutive positions P and P' in a play sequence, define the number difference*

$$\Delta(P, P') = \text{nim}(P) \oplus \text{nim}(P').$$

In NIM, if P' is obtained from P by changing heap i from x_i to x'_i , then

$$\Delta(P, P') = x_i \oplus x'_i,$$

a purely local computation on the changed heap.

Lemma 5.4 (Bounded-change differences). *Fix a constant $t \in \mathbb{N}$. Let P and P' be NIM positions whose heap vectors differ in at most t heaps. Then $\Delta(P, P')$ can be computed by an AC^0 circuit of polynomial size (in the input length).*

Proof. See Appendix E. $\square$

Membership in AC^0 does not guarantee a small or readily learnable network. For the complete two-frame move selector built on this difference computation, our modular construction gives an upper bound of 42 threshold layers; one shootout-controller step has a bound of 70 (Appendix C). These are constructive bounds, not necessary depths. Any Boolean function has a depth-two AND–OR representation with negated inputs, but its size can be exponential; AC^0 requires polynomial size as the input grows. We have not optimised this depth–size trade-off. Removing the expressivity obstruction therefore leaves practical network design, training procedures and hyperparameter choices open. The layer bounds do not establish the cause of the experimental learning plateaus.

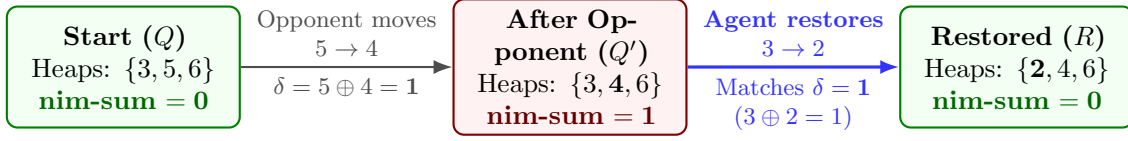


Figure 2: A worked restoration example (three heaps). Starting from a P -position Q (green), the opponent changes heap 2 ($5 \rightarrow 4$), creating a number difference $\delta = 1$. The agent instantly restores the P -position by changing heap 1 ($3 \rightarrow 2$), which imposes the matching difference $3 \oplus 2 = 1$.

5.3.1 TWO-FRAME LOCAL RESTORATION RULE

The next observation is that if, after our move, we make the opponent face a zero nim-sum position, then the opponent’s next move produces some number difference δ . If we respond with a move that produces the *same* number difference δ , we restore nim-sum zero.

Proposition 5.5 (Local restoration principle). *Suppose the opponent moves from a position Q with $\text{nim}(Q) = 0$ to a position Q' (so it is now our turn). Let $\Delta(Q, Q') = \text{nim}(Q) \oplus \text{nim}(Q')$ denote the number difference caused by the opponent’s move. If we respond with a move $Q' \rightarrow R$ such that*

$$\Delta(Q', R) = \Delta(Q, Q'),$$

then $\text{nim}(R) = 0$.

Proof. By definition, $\text{nim}(Q') = \text{nim}(Q) \oplus \Delta(Q, Q')$. Since $\text{nim}(Q) = 0$ (by hypothesis), we have $\text{nim}(Q') = \Delta(Q, Q')$. The player’s move creates a new position R with $\text{nim}(R) = \text{nim}(Q') \oplus \Delta(Q', R)$. Substituting the condition $\Delta(Q', R) = \Delta(Q, Q')$:

$$\text{nim}(R) = \text{nim}(Q') \oplus \text{nim}(Q') = 0.$$

Thus, matching the opponent’s difference restores the invariant. $\square$

Moreover, a concrete restoration move can be selected in AC^0 from (Q, Q') by computing $\delta = \Delta(Q, Q')$ and then choosing (with a fixed tie-break rule) a heap i such that $x'_i = x_i \oplus \delta < x_i$ (Appendix C).

Interpretation (“mirror the delta”). Once we have made the opponent face a P -position, we do not need to recompute $\text{nim}(\cdot)$ from scratch. We only need to mirror the observed change $\Delta(Q, Q')$ in our response move. Figure 2 illustrates this mechanism on a concrete three-heap instance.

Proposition 5.5 is structural: it describes a response pattern that keeps returning the game to nim-sum zero, provided the player can (i) compute $\delta = \Delta(Q, Q')$ from the last two frames and (ii) choose a legal move $Q' \rightarrow R$ that realises the same δ . In NIM, when the opponent changes exactly one heap, δ is simply $x_i \oplus x'_i$, and testing whether a candidate response move realises δ is a local check on the heap being changed.

The significance for learnability is representational. A two-frame policy network can condition its response on (Q, Q') and hence on the locally computable δ , while a single-frame

policy must reconstruct the relevant global information from Q' alone. The experiments in Section 7 examine this contrast through supervised restoration learning and imitation-trained rollout policies. The reported networks are policy-only; these results do not establish that self-play discovers the constructions.

5.3.2 IMPLEMENTATION OVERVIEW

An AC^0 implementation of the two-frame delta and restoration components can be organised into:

- a binary encoding scheme for heap vectors (Appendix C),
- a constant-depth detection of which heaps changed between frames,
- an AC^0 circuit computing Δ on those changed heaps (Lemma 5.4),
- an AC^0 selection mechanism for a response move realising a target δ .

5.4 Second bypass: structured rollout policies

Theorem 5.2 shows that no AC^0 circuit can evaluate the nim-sum in a single forward pass. A natural follow-up is whether *planning*—testing candidate moves and simulating play, as in Monte-Carlo Tree Search (MCTS)—can compensate.

We analyse a stylised rollout interface: a planning algorithm queries a *rollout policy* π by simulating a complete game in which *both* players follow π from a start position, and it observes only the terminal win/loss bit. This abstraction isolates the information that a weak policy can expose through repeated simulation; it is not a claim about the optimisation dynamics of AlphaZero-style training (cf. Section 1.2).

Definition 5.6 (Rollout policy and rollout outcome). *A deterministic rollout policy maps a nonterminal NB -bit NIM position to a legal move, encoded by a heap index and its new value; on the terminal position it outputs a halt symbol. It is AC^0 -bounded when this map has polynomial-size, constant-depth circuits. A rollout has both players follow that policy until termination. Its outcome $RO_\pi(P)$ is 1 if the first player wins and 0 otherwise, including at a terminal start. The interface returns only this terminal bit, not a trajectory or move count.*

Definition 5.7 (Restricted rollout measurements and readout). *Fix a root position P . An admissible query starts at P or a position obtained by changing one heap of P ; the changed index and old/new values are known to the readout. A legal one-move successor is an admissible query. Arbitrary transformations of the whole root are excluded. For the single-channel lower bound, one fixes a policy π_j from Proposition 5.9, independently of P . Writing $\beta = RO_{\pi_j}(P)$, the agent’s final move must have the form $F_N(P, \beta)$, where F_N is a polynomial-size, constant-depth Boolean circuit family. Any value head, query corrections or other processing must be included in this readout bound. The number of queries alone does not impose that bound. A multi-channel construction may query distinct members of the same structured policy family and apply a polynomial-size, constant-depth readout to P and their outcome bits. A policy head means a rollout policy, not simply a parallel output neuron. We count simulated game moves separately from the circuit work per move.*

Allowing arbitrary start positions would defeat the single-channel restriction: querying π_0 on heaps consisting of the k th bits of P returns $\sigma_k(P)$. Likewise, an unrestricted sequential controller could compute the nim-sum itself. The lower bound therefore concerns the stated measurement/readout model, not general MCTS or all single-policy algorithms.

Remark 5.8 (Why multiple heads are not redundant). *Since AC^0 is closed under parallel composition, one could compute all k policies simultaneously as a single AC^0 circuit. The distinction is not about computational power at the circuit level but about the rollout interface: each rollout query simulates a complete game under one policy, and at each game step only that policy determines the next move. A single rollout trajectory under a fixed policy π produces one outcome bit; merging k circuits into a single circuit does not change the fact that any one rollout follows a single deterministic rule, yielding a single bit of information. Multiple heads are useful because the agent can issue separate rollout queries under different policies π_j , producing different game trajectories that reveal independent linear functionals of the nim-sum (cf. Propositions 5.9 and 5.10).*

Finite-state shootout controller (FSM over policy heads). The multi-channel classifier can stop early after a nonzero channel outcome, without changing its Boolean decision rule (the OR of all channel bits). This limited scheduling optimisation does not give the single-channel readout an unrestricted sequential controller. To make this adaptivity explicit (without changing the AC^0 constraints on each head), it is convenient to view head-selection as a small *finite-state controller* that sits outside the heads and only sees the terminal win/loss bit of each rollout.

Formally, fix rollout heads $\pi_0, \dots, \pi_{k-1}$. A *finite-state shootout controller* is a tuple

$$\mathcal{C} = (\mathcal{S}, s_{\text{init}}, g, \delta, \text{halt}),$$

where $\mathcal{S}$ is a finite set of controller states, $s_{\text{init}} \in \mathcal{S}$ is the initial state, $g: \mathcal{S} \rightarrow \{0, \dots, k-1\}$ selects which head to use for the next rollout query, $\delta: \mathcal{S} \times \{0, 1\} \rightarrow \mathcal{S}$ updates the controller state after observing a rollout outcome bit $r = \text{RO}_{\pi_{g(s)}}(P)$, and $\text{halt} \subseteq \mathcal{S}$ is a set of halting states associated with an output decision. (When $\text{halt} = \emptyset$, the controller simply runs for a fixed number of queries and outputs a decision at the end.)

This viewpoint does *not* increase the per-step computational power of the rollout heads: each π_j remains AC^0 -bounded. Instead, the controller organises *which* AC^0 head is iterated in a shootout and *when* to stop querying. Figure 3 depicts a particularly simple controller that turns the B structured rollout heads from Proposition 5.9 into a sequential “bit-query” procedure for deciding whether $\sigma(P) = \text{nim}(P)$ is zero (and hence whether P is a P -position).

The checked implementation includes the outer controller: it selects a head, executes its rollout, records the outcome and either reloads the original board for the next head or halts. For classification it stops at the first outcome 1. For move selection our implementation runs all B heads, since an early classification decision need not supply the invariant bits required by the move readout. Each controller tick is itself an AC^0 map. The original board remains available as a read-only input throughout; it is not an oracle for its nim-sum. Appendix C specifies the storage and update costs.

Integration (channels vs. protocol). It is helpful to reinterpret Part (ii) of Theorem 5.12 in an “information-channel” language: each policy head π_j defines a distinct rollout *measurement channel* which, when iterated to termination, returns a single bit $r_j = \text{RO}_{\pi_j}(P)$

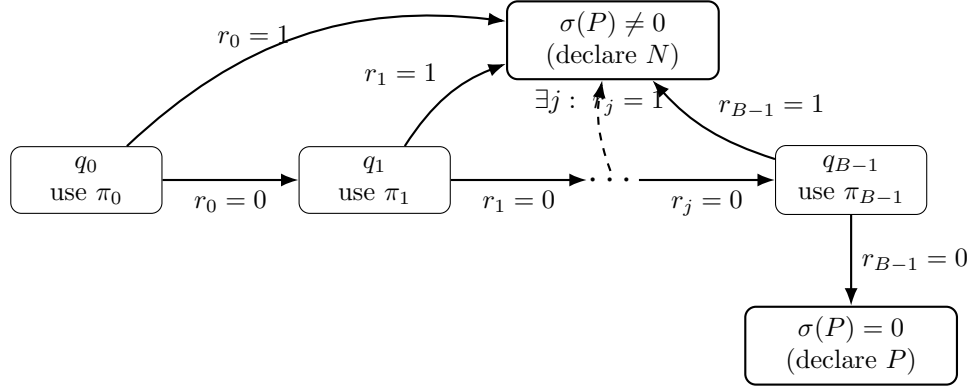


Figure 3: A simple finite-state shootout controller for deciding whether $\sigma(P) = \text{nim}(P)$ is zero using the B rollout heads of Proposition 5.9. The controller queries heads sequentially and halts as soon as it can certify $\sigma(P) \neq 0$. Reaching the final transition to $\sigma(P) = 0$ implicitly means $r_0 = \dots = r_{B-1} = 0$.

that depends on the hidden B -bit invariant $\sigma(P)$ (cf. the identities in Proposition 5.9). With a *single* head, repeated deterministic root rollouts return the *same* linear functional $\ell(\sigma)$; admissible one-heap modifications give its known affine corrections (Proposition 5.10). These measurements supply no further unknown invariant bit. With *multiple* heads, the agent has access to linearly independent invariant measurements and can reconstruct $\sigma(P)$ from their outcomes (Proposition 5.9). The finite-state shootout controller introduced above is the natural generalisation of this idea: it does not change what any head can compute, but organises the heads into an *adaptive querying protocol*—the controller decides which channel to query next (and when to stop, or when to re-query for robustness) based on previously observed rollout outcomes. In this view, multi-head provides the *channels*, while the FSM provides the *sequential control* that turns those channels into a practical evaluation procedure.

Depth amplification. Although a single AC^0 circuit cannot compute parity, iterating an AC^0 policy for L game steps yields an effective depth $\Theta(L)$ computation, which can reveal parity information. The next proposition makes this explicit by exhibiting B simple AC^0 rollout heads whose outcomes recover all B bits of the nim-sum.

Proposition 5.9 (Structured rollout sufficiency). *The B structured rollout policies below, followed by an AC^0 move selector, achieve strong mastery of NIM on N heaps of bit-width B . They use B root rollouts and $O(BN2^B)$ simulated game moves per decision. Each policy update and the final readout have polynomial circuit size in NB . For $B = O(\log N)$ the resulting bit work is polynomial in N .*

Proof outline. For each $j \in \{0, \dots, B-1\}$ define a deterministic rollout policy π_j that repeatedly subtracts 2^j from the leftmost heap of size at least 2^j (and otherwise subtracts 1 from the leftmost nonzero heap). Both simulated players follow this policy; the actual opponent may choose any legal move. A rollout under π_j has a well-defined total number of

moves whose parity is

$$\text{RO}_{\pi_0}(P) = \sigma_0, \quad \text{RO}_{\pi_j}(P) = \sigma_j \oplus \sigma_0 \quad (j \geq 1),$$

where $\sigma = \text{nim}(P)$ and σ_j denotes its j th bit. The search controller first queries π_0 to obtain σ_0 , then queries each π_j ($j \geq 1$) to obtain $\sigma_j \oplus \sigma_0$, and recovers $\sigma_j = (\sigma_j \oplus \sigma_0) \oplus \sigma_0$; thus B rollouts under B distinct heads recover all B bits of σ . Given $\sigma \neq 0$, select the first heap h_i with $h_i \oplus \sigma < h_i$ and reduce it to $h_i \oplus \sigma$. If $\sigma = 0$, take any legal move, or halt at the terminal position. No successor enumeration is needed. A complete proof is in Appendix G. $\square$

A single policy head yields only one bit. The preceding result relies on having *multiple* rollout channels. For one fixed member of this structured policy family, admissible queries reveal known affine functions of the same invariant bit. This statement does not concern arbitrary rollout policies.

Proposition 5.10 (Single-policy information bound). *Fix j and the rollout policy π_j from the proof outline of Proposition 5.9. For any position P , the rollout outcome $\text{RO}_{\pi_j}(P)$ is a fixed linear functional $\ell_j(\sigma(P))$ of the nim-sum bits (indeed, $\ell_0(\sigma) = \sigma_0$ and $\ell_j(\sigma) = \sigma_j \oplus \sigma_0$ for $j \geq 1$). Moreover, if P_i is obtained from P by modifying one heap (as when evaluating candidate moves), then*

$$\text{RO}_{\pi_j}(P_i) = \ell_j(\sigma(P)) \oplus c_i,$$

where the correction bit c_i is computable from the known heap values and the chosen modification. Hence root and one-heap-modification queries to this fixed head reveal only known affine functions of $\ell_j(\sigma(P))$.

Proof. The linearity claim follows from the explicit formulas above. If P_i differs from P by changing one heap from h to h' , then $\sigma(P_i) = \sigma(P) \oplus (h \oplus h')$ and $\ell_j(\sigma(P_i)) = \ell_j(\sigma(P)) \oplus \ell_j(h \oplus h')$, where $\ell_j(h \oplus h')$ is known from the candidate move. $\square$

Remark 5.11 (An illustrative independent outcome-flip model). *Suppose an ideal terminal bit Y is corrupted as $\tilde{Y} = Y \oplus E_1 \oplus \dots \oplus E_L$, where the E_i are independent Bernoulli variables of parameter $0 \leq \varepsilon < 1/2$. Then*

$$\Pr[\tilde{Y} = Y] = \frac{1 + (1 - 2\varepsilon)^L}{2}.$$

The formula follows by multiplying the expectations of $(-1)^{E_i}$. With fresh independent flips for each repetition, the least number of observations needed to distinguish the two possible ideal bits with worst-case error at most $1/4$ grows as a^{-2} , where $a = (1 - 2\varepsilon)^L$. Appendix F states and proves explicit bounds over real parameters. Repeating one corrupted observation with the same randomness supplies no additional information. At $\varepsilon = 0.01$ and $L = 50$, agreement is approximately 0.682, so this example is not yet close to a fair coin. This is a channel-noise illustration, not a theorem about erroneous game moves: a move error need not flip the terminal bit, and can change later states and rollout length. Correlated flips can cancel in pairs, eliminating the claimed decay. Exact structured rollouts have no such noise.

Resource	What stays in AC^0	Additional information or time
One structured channel	Policy step and final readout	One complete rollout; one invariant bit
B structured channels	Each policy step; reconstruction and move selection	B complete rollouts supply all invariant bits
Two-frame transition	Local difference and binary move selection	Previous zero-nim-sum board and a legal transition
Persistent row scan	One state update and its output bits	Two scans per decision; $O(B + \log N)$ state bits

Figure 4: The three bypasses use different resources. Conditional restoration does not classify arbitrary starting positions; the rollout and recurrent constructions supply additional sequential computation.

The initialisation boundary for restoration. Matching two successive differences yields $\text{nim}(R) = \text{nim}(Q)$, whether or not that value is zero. Thus Proposition 5.5 does not supply a classifier for an arbitrary starting position. For example, complete a matching-delta policy by subtracting one from the first nonempty heap whenever no matching response exists or no previous frame is available. Starting from (2), it plays $(2) \rightarrow (1) \rightarrow (0)$ and the first player loses, although (2) is winning. The fallback after the first move is needed because its delta is $2 \oplus 1 = 3$, which cannot be realised from (1). This counterexample does not rule out every possible verifier construction; it shows that conditional restoration and an unspecified fallback do not establish one. We make no universal-verifier claim. Initial winning-move selection can instead use the multi-channel construction above or the growing recurrent scan in Section 5.5.

Theorem 5.12 (Restricted bound and bypasses). *The following statements hold under Definition 5.7.*

- (i) *For each fixed $B \geq 2$, let P be uniform on $\{0, \dots, 2^B - 1\}^N$ conditional on $\sigma(P) \neq 0$. For one fixed structured head π_j and an AC^0 readout $F_N(P, RO_{\pi_j}(P))$, winning-move success is at most*

$$\frac{2}{2^B - 1} + \text{negl}(N).$$

Here $N \geq 1$, and a negligible error is a nonnegative sequence smaller than every inverse polynomial for sufficiently large N , with circuit depth, polynomial size exponent and B fixed. More precisely, a uniform parity-correlation bound η_N for the fixed-bit readout indicators gives the finite upper bound $2/(2^B - 1) + (2^B - 1)\eta_N$; the indicators and correlation premise are specified in Appendix H. The leading constant is asymptotically attainable by guessing one nonzero nim-sum in each channel fibre.

- (ii) *For any B , the B structured heads and an AC^0 readout select an optimal move using $O(BN2^B)$ rollout steps, as in Proposition 5.9.*

- (iii) *Given a legal transition from a known zero-nim-sum predecessor, a two-frame AC^0 selector restores zero. Once a player has established that invariant, it maintains it against every legal opponent response. This statement does not supply the initial winning move from an arbitrary position.*

Proof outline. For part (i), fixing the observed bit leaves an AC^0 move selector. The selected heap’s old and new values determine a proposed nim-sum. Expanding each exact nim-sum event into its finitely many parity characters and applying the AC^0 parity-correlation bound gives the displayed estimate; Appendix H gives the full reduction and conditional-probability calculation. Parts (ii) and (iii) follow from Propositions 5.9 and 5.5, with the selector described in Appendix C. $\square$

5.5 Third bypass: persistent row-wise recurrence

The two-frame construction supplies the relevant history explicitly, while the multi-head construction obtains additional information through complete rollouts. A third possibility is to let the agent build and retain its own state by processing the board sequentially. The recurrent update remains simple, but the number of updates is allowed to grow with the input. The construction uses the elementary running-XOR algorithm. In correspondence reproduced by Friedman (Friedman, 2017b), John Schulman proposed feeding NIM heaps one at a time to a recurrent network and reported preliminary supervised nim-sum experiments. Here we make the shallow updates, separator and state protocols, and move-selection constructions explicit, distinguishing their representational guarantees from learning through self-play.

Serialising positions and retaining state. Write position $P_t = (x_{t,1}, \dots, x_{t,N})$, with each heap represented by one B -bit row. Feed successive positions as the stream

$$\underbrace{x_{1,1}, \dots, x_{1,N}}_{P_1}, \text{ SEP}, \underbrace{x_{2,1}, \dots, x_{2,N}}_{P_2}, \text{ SEP}, \dots$$

Each row arrives as a block of B bits together with a row/separator tag. A separator marks the end of a complete position; the agent chooses a move only after that position has been read. The next observed position is then appended to the stream. For comparisons of individual heaps, indices remain aligned across positions, including rows for empty heaps. A scan of the current board suffices for nim-sum evaluation without earlier game positions; cross-position memory additionally supports the comparisons below. The hidden state is carried across rows and position boundaries. A separator resets the accumulator for the next scan while preserving selected information about the completed position; it does not reset the entire memory.

Proposition 5.13 (Sequential nim-sum accumulation). *There is a recurrent Boolean update of constant depth and size $O(B)$, implementable with fixed-scale quantised threshold gates, that computes the nim-sum of an N -heap position after N row updates. With $O(B)$ state bits, it can also retain the previous completed nim-sum and report the XOR difference between consecutive completed positions.*

Proof. Initialise a B -bit accumulator to zero and update it by

$$s_{t,0} = 0, \quad s_{t,i} = s_{t,i-1} \oplus x_{t,i} \quad (1 \leq i \leq N).$$

Induction on i gives $s_{t,i} = \bigoplus_{j=1}^i x_{t,j}$, so the final accumulator is $\sigma_t := \text{nim}(P_t)$. Each output bit of an update is the XOR of two bits, which has a constant-size, constant-depth AND/OR/NOT implementation. These gates can be realised with fixed-scale threshold gates and input negations, with constants independent of N and B . The summary state is (s, q, v) : two B -bit registers and one validity bit, initialised to $(0, 0, 0)$. A row x updates it to $(s \oplus x, q, v)$. At a separator, output s and, when $v = 1$, the difference $q \oplus s$; then update the state to $(0, s, 1)$. The first separator therefore produces a summary but no previous-position difference. Induction over completed positions shows that q stores exactly the preceding summary and s is the XOR of the rows since the last separator. The summary mechanism uses $2B + 1$ state bits; a reset between games restores $(0, 0, 0)$. All assignments have constant depth and size $O(B)$. $\square$

From a summary to a move. Computing σ_t does not by itself retain the heap rows needed to choose a move. One implementation replays the current position after the first scan. If $\sigma_t \neq 0$, select the first row i for which

$$x'_{t,i} = x_{t,i} \oplus \sigma_t < x_{t,i}.$$

Such a row exists: choose a heap containing the highest set bit of σ_t . Replacing that heap by $x'_{t,i}$ leaves nim-sum zero. XOR, binary comparison and conditional selection are constant-depth operations; the replay uses $O(B + \log N)$ state bits to retain the summary, a selected move and its row index. The explicit protocol is

$$\text{rows}(P_t), \text{SEP}, \text{rows}(P_t), \text{SEP}.$$

The first separator stores the summary and switches to replay; the second emits the move and returns to scan mode. This takes exactly $2N + 2$ updates, with one additional reset token at the start of a game. If $\sigma_t = 0$, choose the first nonempty heap and decrement it, or report termination when all heaps are empty. Lean verifies the complete replay controller, including its binary move output and its fixed-scale threshold implementation. With an r -bit counter satisfying $N < 2^r$, its state uses $4 + 5B + 3r$ bits. The strict inequality allows the counter to hold the end value N , whereas an output index alone only needs $N \leq 2^r$. Alternatively, store the current board in $O(NB)$ bits during the initial scan and use a constant-depth selector at the separator, as in Appendix C. This avoids replay at the cost of board-sized memory. The distinction matters for a stream in which each position is supplied only once. Both variants have checked bit-level circuits, fixed-scale network implementations and winning-play guarantees. The board-storage variant uses $NB + B + r$ state bits and $N + 1$ updates per decision, with $N < 2^r$, plus a reset update when initialising the controller.

Connecting recurrence to the history bypass. The retained summaries allow the agent to compare consecutive positions through

$$\sigma_{t-1} \oplus \sigma_t = \text{nim}(P_{t-1}) \oplus \text{nim}(P_t) = \Delta(P_{t-1}, P_t).$$

When one heap changes from a to b , this equals $a \oplus b$. If the preceding position has nim-sum zero, the difference is exactly the invariant that the restoration rule must cancel. Storing the preceding board additionally permits direct row-by-row comparison as the new board arrives.

A summary of $O(B)$ bits preserves the invariant, rather than the full game history; arbitrary access to old positions requires additional storage or replay. The difference always refers to consecutive completed boards actually supplied to the controller. If these are only the boards at the player’s decision times, they usually span two game moves. Comparing just the opponent’s move instead requires the player’s own post-move board as the preceding frame. The initial scan also computes the invariant from an arbitrary starting position, without assuming an earlier zero-nim-sum frame. Thus memory across row updates is sufficient for this route; an earlier game position is useful for transition comparisons but is not required to compute the current nim-sum. This initialisation uses the growing recurrent horizon and is therefore an additional resource beyond the fixed-latency two-frame construction.

Why the lower bound is respected. Unrolling N constant-depth updates gives a computation of depth $O(N)$, so Proposition 5.13 lies outside the fixed- T premise of Theorem 2.4. A one-step recurrent dependency can nevertheless carry information accumulated over the whole stream. Merely appending positions and separators to a fixed-depth input does not supply this growing computation horizon. The bypass comes from sequential computation with retained state, not from the delimiter itself. For growing heap bit-width, the accumulator also needs $O(B)$ bits; a fixed-width hidden state is not claimed to store arbitrary-size invariants or histories. The construction establishes representational sufficiency for NIM, but does not establish that training a recurrent network will discover the update or preserve it reliably over long games. We do not report recurrent training experiments here.

5.6 What the three bypasses establish

Proposition 5.14 (Shallow components and complete strategies). *For $N \geq 1$ heaps of $B \geq 1$ bits, the following constructions use polynomial-size, constant-depth Boolean components, realisable by literal-threshold networks with weights and thresholds in $\{0, 1\}$.*

- (i) *The two-frame selector maps a legal transition $Q \rightarrow Q'$ from $\text{nim}(Q) = 0$ to a legal binary-encoded response restoring zero. Retaining the response board as the next predecessor gives a winning strategy from this established invariant, with the opponent to move.*
- (ii) *Each structured rollout policy and the readout from the board and its B outcome bits are shallow. Executing the rollouts and then the readout gives a winning strategy from any nonzero nim-sum board, using at most $BN(2^B - 1)$ simulated game moves per decision.*
- (iii) *The recurrent state update is shallow. Iterating it over the replay protocol gives a winning strategy from any nonzero nim-sum board, with $O(B + \log N)$ memory and $2N + 2$ updates per decision.*

Here “winning strategy” quantifies over every legal opponent response. Polynomial size and constant depth refer to the indicated component, with depth independent of N and B ; they do not bound the number of sequential component evaluations.

Proof outline. Appendix C gives explicit candidate tests, first-index encoding and a fixed-scale implementation. The two-frame circuit computes the changed heap’s XOR difference,

which equals the current nim-sum under the stated promise. The multi-head readout reconstructs that same invariant from the executed rollout outcomes. The recurrent controller obtains it by accumulation and uses the stored result during replay. In each case the selected move restores zero; every legal opponent move makes the nim-sum nonzero, and every move decreases the total number of stones. This proves termination and the winning guarantee. For the two-frame strategy, the newly restored board is precisely the frame retained for the next response. For recurrence, the finite controller state persists between decisions. The rollout and recurrent step bounds are those proved above. $\square$

These are three ways around the *same restricted inference problem*, not three shallow algorithms for unrestricted parity. The two-frame construction changes the input and imposes a promise. The other constructions allow processing time to grow. Their complete board-to-winning-move maps cannot themselves form an AC^0 family: otherwise applying either to $(2, b_1, \dots, b_m)$ would contradict Theorem 5.2. Parallel output heads alone do not supply the rollout computation, and separators alone do not supply a recurrent scan. Appendix C illustrates the explicit network constructions and explains their conservative depth and layer bounds.

6 Learning impartial games with neural networks

The results above are primarily *representational*: they identify what information a bounded-depth, fixed-scale quantised model can or cannot compute from its input, and how history, multiple rollout heads, or a growing recurrent horizon can supply the missing information. This section connects the theory to the training protocols used in Section 7 and highlights the main modelling choice (AC^0 vs. TC^0).

6.1 Practical takeaways

- **Single-frame is brittle to parity.** In the worst case, fixed-scale quantised bounded-depth inference cannot reliably represent the nim-sum or a winning move selector (Section 5.2).
- **Three bypasses with different resources.** A two-frame input exposes the transition signal $\Delta(P, P')$ and supports restoration (Section 5.3.1); multiple rollout heads recover the missing nim-sum bits through depth amplification (Section 5.4); persistent recurrence accumulates the invariant over sequential row updates and retains summaries across positions (Section 5.5). The last route trades fixed total latency for a growing processing horizon.
- **Optimisation remains nontrivial.** Even when a solution is expressible in AC^0 , discovering it from sparse win/loss rewards can be difficult; our experiments therefore isolate a dense restoration supervision signal and separately report rollout-based classification.

6.2 Teacher-guided data as a controlled source of transitions

Let π_{opt} denote an optimal NIM policy (move to nim-sum 0 whenever possible). For a noise parameter $\varepsilon \in [0, 1]$, define a noisy teacher π_T^ε that plays according to π_{opt} with probability $1 - \varepsilon$ and otherwise chooses a uniformly random legal move. Occasional random moves

broaden state coverage and, crucially for two-frame agents, generate diverse transition pairs (Q, Q') from which the local difference signal $\Delta(Q, Q')$ can be inferred.

6.3 Approximate majority, parity, and the AC^0 abstraction

One might object that modern networks can implement majority-like decisions, suggesting TC^0 (threshold circuits) as the “right” complexity class. Indeed, if one allows scale-adaptive thresholds/biases or fan-in dependent re-scaling, constant-depth threshold models can represent parity-type computations.

Our focus is a restricted literal-threshold regime: fixed total depth, nonnegative weights on Boolean literals, and thresholds bounded in units of the smallest nonzero weight. In this regime, each gate has a constant-size witness and collapses to a constant-width DNF, yielding an AC^0 abstraction (Theorem 2.4). Constant-depth AC^0 circuits can implement approximate counting (Ajtai, 1993), yet they have exponentially small correlation with parity (Furst et al., 1984; Håstad, 1986; Smolensky, 1987; Razborov, 1987). Learning parity depends on the architecture and training setting; both difficulties and positive results have been studied (Thornton, 1996b,a; Daniely and Malach, 2020). The representational lower bound here applies only to the explicit gate model. Section 6.4 gives a complementary Fourier/spectral interpretation of this local-versus-global parity separation.

6.4 A spectral viewpoint: low-degree Fourier mass vs. parity

A useful complementary lens is Fourier analysis on the Boolean hypercube. Let $f : \{0, 1\}^n \rightarrow \mathbb{R}$ and define the Walsh–Hadamard characters

$$\chi_S(x) = (-1)^{\sum_{i \in S} x_i}, \quad S \subseteq [n].$$

The set $\{\chi_S\}_{S \subseteq [n]}$ is an orthonormal basis with respect to the uniform distribution on $\{0, 1\}^n$. The Fourier coefficients are

$$\hat{f}(S) = \mathbb{E}_{x \sim U(\{0, 1\}^n)}[f(x)\chi_S(x)], \quad \text{and} \quad f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(x).$$

We refer to $|S|$ as the *Fourier degree* (or “frequency”) of the coefficient $\hat{f}(S)$.

Definition 6.1 (Fourier tail / spectral bandwidth). *For $k \in \mathbb{N}$, define the degree- $> k$ Fourier tail*

$$\text{Tail}_{>k}(f) = \sum_{S \subseteq [n]: |S| > k} \hat{f}(S)^2.$$

For $\varepsilon \geq 0$, we say f is (k, ε) -spectrally band-limited if $\text{Tail}_{>k}(f) \leq \varepsilon$.

Parity is a pure high-frequency character. For any nonempty $S \subseteq [n]$, the parity function $\text{PARITY}_S(x) = \chi_S(x)$ has Fourier expansion supported on a single set: $\widehat{\text{PARITY}_S}(S) = 1$ and $\widehat{\text{PARITY}_S}(T) = 0$ for $T \neq S$. Thus its spectral “frequency” is exactly $|S|$. More generally, if f is (k, ε) -spectrally band-limited and $|S| > k$, then

$$|\mathbb{E}[f(x)\chi_S(x)]|^2 = \hat{f}(S)^2 \leq \text{Tail}_{>k}(f) \leq \varepsilon. \quad (6.1)$$

This elementary inequality is the link between concentration at low degree and small correlation with a high-degree parity.

AC⁰ has little high-frequency mass. A seminal theorem of Linial–Mansour–Nisan shows that AC⁰ functions have almost all Fourier mass on low degrees: for f computed by an AC_d⁰ circuit of size M , the tail above degree k decays rapidly with k (see (Linial et al., 1993), and sharper bounds in (Tal, 2015)). For fixed depth and polynomial size, correlation with full parity under the uniform distribution is negligible in n . Applying this statement to other subsets requires specifying their size relative to circuit depth and size, and it does not automatically apply under a nonuniform training distribution.

Local vs. global parity in NIM. In our encoding, the j th bit of the nim-sum is the parity of the j th bits across all N heaps, hence a degree- N character. Conditional on a specified changed heap i , the sign encoding of the j th delta bit is the two-coordinate character $(-1)^{x_{i,j}+x'_{i,j}}$. When the changed index is unknown, the delta bit is specified on pairs of boards differing in at most one heap. A shallow extension to all pairs is

$$d_j(P, P') = \bigvee_{i=1}^N (x_{i,j} \oplus x'_{i,j}). \quad (6.2)$$

On the promised domain this is the required delta bit. For $N \geq 2$, its sign encoding $(-1)^{d_j}$ is not a single character, even up to sign; for $N = 1$ it is exactly a degree-two character. Thus local computability does not mean that the full unknown-index map has degree two. It follows from the explicit AC⁰ construction in Lemma 5.4.

Connection to optimisation (informal). Fourier degree also aligns with a common empirical observation in deep learning: gradient-based training tends to fit low-frequency components before high-frequency ones (“spectral bias” / “frequency principle”) (Rahaman et al., 2019; Xu et al., 2019). This suggests a possible analogy for parity learning, but continuous-domain spectral-bias observations do not prove how SGD learns Boolean parity or restoration under our nonuniform data distribution. We treat that connection as a hypothesis requiring separate measurements.

6.5 From global invariants to local checks

The core representational point is that the global invariant $\text{nim}(P)$ is hard to compute from a single frame, whereas the transition quantity $\Delta(P, P')$ is locally computable from two frames (Lemma 5.4). Section 7.2 isolates this effect by training one-frame and two-frame models on the same restoration targets.

7 Experiments

The stronger experimental goal would be to show that an AlphaZero-style learner discovers effective play through self-play once a bypass is available. We do not establish that here. The theoretical barrier concerns what the restricted network can represent, independently of training; removing it leaves open whether the required computation can be learned, with what data and under which training regime.

We take a controlled intermediate step: supervised restoration learning from teacher–random transitions (Section 7.2) and a complementary single-frame multi-head shootout evaluation with imitated policy heads (Section 7.3). These experiments supply guidance that

outcome-only self-play would have to generate for itself. They test whether the proposed information and computation can be useful in training, while also exposing difficulties that remain after the input has been enriched. Unless otherwise stated, both one-frame (1F) and two-frame (2F) agents share the same action parameterisation and network architecture (Section 7.1).

7.1 Neural architecture and training objective

7.1.1 STATE AND ACTION REPRESENTATION

We use a fixed-width binary encoding of heap sizes with k bits per heap. The single-frame input is the encoding of the current heap vector. The two-frame input concatenates the encodings of the previous and current heap vectors. When a previous frame is unavailable (e.g. on the first move of a game), we set the previous frame equal to the current position so that the input dimension is constant and the 2F policy remains well-defined.

The action space is parameterised as “set heap i to value v ” for $i \in \{1, \dots, N\}$ and $v \in \{0, 1, \dots, 2^k - 1\}$, giving $N \cdot 2^k$ logits. The saved restoration evaluation takes an unmasked argmax over all logits; illegal predictions count as incorrect. Legality masking is used in the separate game-playing and FSM rollout routines. These are distinct evaluation conventions.

7.1.2 NETWORK ARCHITECTURE

Both restoration agents use a policy-only MLP with two hidden layers and $N \cdot 2^k$ output logits. There is no value head in the saved implementation. The two-frame input has dimension $2Nk$; the single-frame input has dimension Nk . The template and hidden width are shared, but parameter counts differ: the two-frame first layer adds $NkH = 20,480$ parameters in the reported setting. All layers use ordinary floating-point arithmetic; these networks are not literal implementations of the FSQI gate model.

Component	Type	Dimensions	Activation/ Normalisation
Input	State representation	Two-frame: $2Nk$ bits (1F: Nk)	None
Hidden 1	Fully Connected	1F: $Nk \rightarrow H$; 2F: $2Nk \rightarrow H$	LayerNorm + ReLU
Hidden 2	Fully Connected	$H \rightarrow H$	ReLU
Policy Head	Linear Output	$H \rightarrow N \cdot 2^k$	Raw logits; Log-Softmax in loss

Table 1: Neural network architecture used in the experiments. H denotes the hidden width (a fixed hyperparameter).

7.1.3 TRAINING TARGETS AND LOSS

For each training input (single-frame state or two-frame state-pair), the target is a soft distribution $\pi_{\text{opt}}(\cdot \mid p')$ over optimal restoration moves. The policy head is trained with cross-entropy to this target. The soft-target cross-entropy also uses unmasked logits. No value loss is used in this supervised restoration experiment.

7.1.4 EVALUATION CRITERIA IN THIS VERSION

In this version we report two task-level notions of performance rather than full game-level Elo tournaments. *Move-level restoration performance* is measured by **functional correctness**: on a test position with nonzero nim-sum, a move is counted as correct if it produces a successor state with nim-sum zero. Positions with zero nim-sum have no winning move (Theorem 4.4); when reporting functional correctness we therefore condition on nonzero nim-sum positions. *Position-level evaluation performance* is measured by P/N classification accuracy and N -position recall in FSM-controlled shootouts (Section 7.3).

7.2 Supervised restoration learning from teacher–random transitions

This experiment isolates the *restoration response* as a supervised learning problem. In normal-play impartial games, a restoration response is the move that returns play to a losing (P -)position after the opponent departs from it. In Nim, this is equivalent to selecting an optimal (winning) move from the post-opponent position: if the previous position had nim-sum 0, then after any opponent move the position is winning, and any optimal reply restores nim-sum 0.

Motivated by Sections 5.2–5.3, the experiment asks a separate empirical question under a fixed supervised training protocol: *does access to the transition (p, p') (two frames) enable successful restoration learning under this protocol, even when the one-frame model fails?* This benchmark builds on the observation that parity-type invariants pose a challenge for reinforcement learning in impartial games (Zhou and Riis, 2026, 2023), while explicitly providing the transition information needed by Proposition 5.5.

7.2.1 GAME AND REPRESENTATIONS

We consider Nim with $N = 20$ heaps. Each heap size is represented with $k = 4$ bits (little-endian), so heap values lie in $\{0, \dots, 15\}$. Moves are represented in a fixed action space of size $N \cdot 2^k = 20 \cdot 16 = 320$: an action chooses a heap index i and a new heap value v . Legality requires $0 \leq v < h_i$; the reported unmasked metric counts violations as errors.

A **1-frame (1F)** policy receives only the current position p' (80-bit input), while a **2-frame (2F)** policy receives the concatenation (p, p') of the previous and current positions (160-bit input). Both agents use the shared MLP architecture described in Section 7.1 (two hidden layers of width $H = 256$).

7.2.2 DATASET GENERATION (TEACHER–RANDOM–TEACHER)

We generate a dataset of size $D = 10^6$ examples of the form

$$(p, p') \mapsto \pi_{\text{opt}}(\cdot \mid p'),$$

where $\pi_{\text{opt}}(\cdot \mid p')$ is a soft target distribution over optimal moves from p' .

Each example is generated as follows:

1. Sample a random *winning* Nim position s (nim-sum $\neq 0$).
2. Apply an optimal teacher move from s to obtain p . For optimal play in Nim, p has nim-sum 0 (a P -position for the next player).

3. Apply a uniformly random legal move from p to obtain p' .
4. Define the label $\pi_{\text{opt}}(\cdot \mid p')$ as the uniform distribution over all optimal (winning) moves from p' .

This construction ensures that each training input (p, p') corresponds to the canonical restoration setting: p is a P -position, the random move produces a winning position p' , and the label is the set of restoration moves back to a P -position.

Why teacher–random–teacher? We do not sample arbitrary triples (P, P', a) uniformly at random, because the restoration claim is about a specific interaction regime: the opponent departs a P -position and the agent must restore it. The teacher step ensures P is a genuine P -position reachable under optimal play, the random move injects diversity in the departure $P \rightarrow P'$, and the final teacher move provides a correct restoration label for P' .

Compared to earlier empirical studies of impartial games (Zhou and Riis, 2026, 2023), we “help” the learner in two ways: (i) we represent heap sizes in binary rather than as raw integers, and (ii) we train on *perfectly labelled* restoration targets (the full set of optimal moves), rather than requiring the agent to infer these targets from sparse win/loss outcomes.

We split the dataset into 80% train / 10% validation / 10% test using a seeded permutation (seed 0), shared by the two representations. The teacher uses the first winning heap; this induces a particular, nonuniform distribution of predecessor positions. The results do not establish invariance to heap permutations, unseen board sizes, or arbitrary self-play histories.

7.2.3 TRAINING AND CHECKPOINTS

We train each model with supervised learning using cross-entropy to the soft target distribution. We use Adam with learning rate 10^{-3} and batch size 512 for 200 epochs. To study performance as a function of training, we save $C = 10$ checkpoints per model at epochs

$$\{1, 23, 45, 67, 89, 112, 134, 156, 178, 200\}.$$

7.2.4 EVALUATION METRICS

We evaluate restoration performance on held-out data:

Restoration accuracy (validation/test). Given a held-out example, the saved evaluation takes the argmax over all action logits and counts it as correct if it lies in the support of $\pi_{\text{opt}}(\cdot \mid p')$. Illegal predictions therefore count as incorrect, and every legal optimal move is accepted. The saved implementation averages batch accuracies equally, including the shorter final batch, rather than weighting them by batch size. The reported decimals are those saved batch means, not exact correct-example counts divided by 100,000. The same convention applies to the displayed validation curve.

7.2.5 RESULTS

One-frame does not learn restoration; two-frame eventually does. Figure 5 shows validation restoration accuracy as a function of epoch. The 1F model has low test restoration accuracy throughout training, approximately 0.075–0.081 across saved checkpoints. We do not identify this range with a measured random-legal-move baseline, which is not supplied

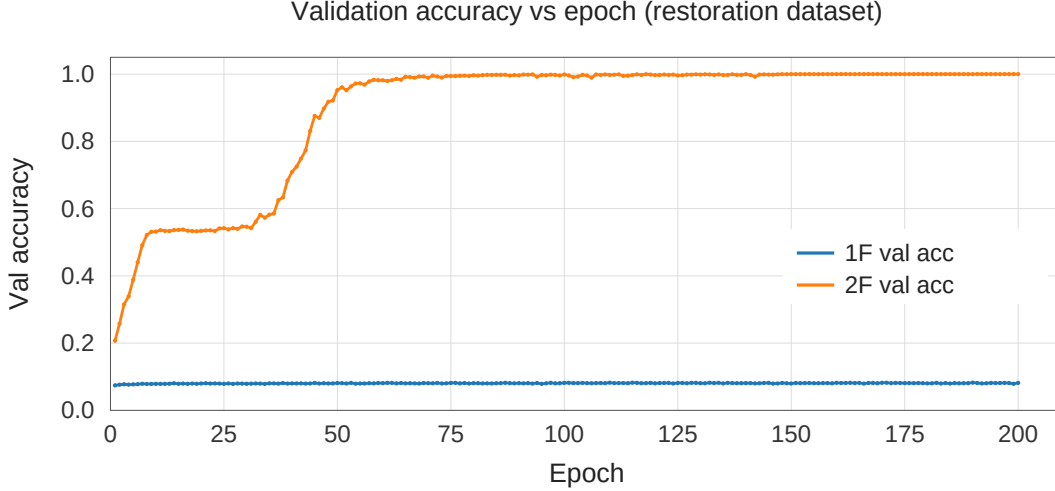


Figure 5: Validation restoration accuracy vs. epoch on the restoration dataset ($N = 20$, $k = 4$, $D = 10^6$). In this saved run, the 1F model has low accuracy while the 2F model passes through an intermediate plateau around 0.54 before improving sharply towards near-perfect accuracy. The transition suggests a learning bottleneck that training eventually overcomes; it is distinct from the representational lower bound. Values use the saved batch-mean metric.

in the reported evaluation. In contrast, the 2F model learns the restoration mapping and reaches essentially perfect accuracy with enough training. On the test set (Figure 6), the 2F checkpoints achieve 0.209 at epoch 1, 0.538 at epoch 23, 0.876 at epoch 45, 0.989 at epoch 67, and 0.99998 by epoch 178 (remaining ≥ 0.99993 at epoch 200).

Learning in stages: the plateau and subsequent improvement. The plateau near 0.54 and later rise in Figures 5–6 suggest a temporary learning bottleneck even with two-frame input. The learner makes substantial progress, remains at an intermediate level, and then improves sharply. This is evidence about training dynamics within the saved run, rather than a further circuit lower bound.

A diagnostic of the saved checkpoints makes the remaining difficulty more concrete. For a predicted move setting heap i to v , compare its proposed change $p'_i \oplus v$ with the required nim-sum of p' . On the reconstructed and independently verified 100,000-position test set, the epoch-23 model makes only legal moves and gets the upper three bits of this change jointly correct on 99.998% of examples. Of its 46,206 failed moves, 46,204 differ from the correct change only in the lowest bit. At epoch 45 all 12,437 errors are confined to that bit; at epoch 178 only two errors remain, both of the same kind. These diagnostic counts weight examples equally, unlike the saved batch-mean accuracies plotted above.

Thus the later improvement is accounted for at the output level by learning to select moves with the correct lowest bit while retaining the already accurate upper bits. This supports staged acquisition of the restoration behaviour. It does not identify an internal representation or explain why optimisation follows this order. Nor does the plateau’s

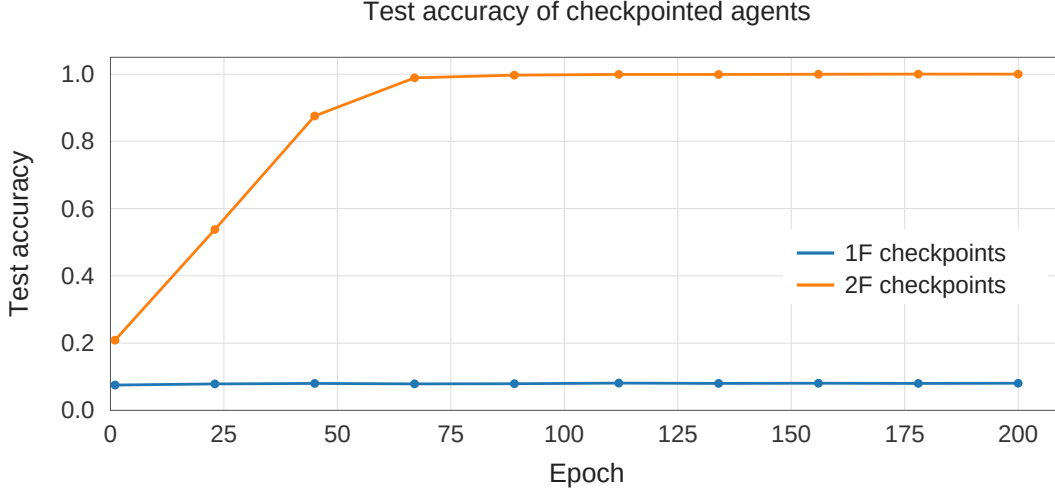


Figure 6: Test restoration accuracy of checkpointed agents ($N = 20$, $k = 4$, $D = 10^6$). Checkpoints are saved at epochs $\{1, 23, 45, 67, 89, 112, 134, 156, 178, 200\}$. The 2F checkpoints confirm the improvement beyond the intermediate plateau; the more densely sampled validation curve in Figure 5 shows its timing.

proximity to 8/15 establish a random-bit guessing mechanism: that baseline assumes a particular distribution and prediction rule. Internal probes, interventions and independent training runs are needed to explain the transition and test its reproducibility.

Scope and provenance of the reported result. This controlled supervised run shows a large advantage for transition information at $N = 20$, $k = 4$, $D = 10^6$ and seed 0. It does not prove worst-case mastery, efficient self-play learning, or the asymptotic circuit lower bound. The checkpoint CSV is identified by run suffix 20260214_120801_cd18fc9f; the architecture and evaluation convention above follow its associated saved notebook. We report this documented configuration and do not make an unsupported claim of robustness across a hyperparameter or board-size sweep. Multiple independent seeds, matched-parameter controls and out-of-distribution evaluations remain useful follow-up experiments.

7.3 Single-frame evaluation via FSM-controlled shootouts (multi-head)

Purpose. We complement the two-frame restoration study (Section 7.2) with an experiment targeting the *single-frame* bypass from Proposition 5.9: learned approximations to the structured rollout policies combined with a finite-state shootout controller (Figure 3). The trained floating-point networks are not claimed to satisfy the formal AC^0 gate restrictions. The goal is to test the *rollout interface* itself (how much information can be extracted by controlled shootouts), not to claim that self-play discovers the structured heads without additional inductive bias.

Setup. We consider NIM with $N = 20$ heaps and bit-width $k = 4$ (so $B = k = 4$ heads), matching the encoding of Section 7.1.1. We evaluate on a balanced test set of 2000 positions with 50% P -positions ($\sigma = 0$) and 50% N -positions ($\sigma \neq 0$).

Training the rollout heads. We train a shared-trunk network with $B = 4$ policy heads by imitation learning against the structured policies $\pi_0, \dots, \pi_{B-1}$ from Proposition 5.9. Each head π_j subtracts 2^j from the leftmost heap of size at least 2^j (falling back to subtracting 1 from the leftmost nonzero heap). When multiple heaps qualify for the same phase of π_j (e.g. several heaps have $h_i \geq 2^j$), any choice among them produces a rollout with the same total move count and hence the same parity outcome. We therefore train with *soft targets*: for each position and head j , the label is uniform over all equivalent actions, and the loss is cross-entropy against this soft distribution. This avoids penalising the model for an arbitrary tie-breaking choice among equally valid heaps.

The network uses the same shared-trunk architecture as Table 1 (two hidden layers of width $H = 256$ with LayerNorm), with B independent linear output heads each producing $N \cdot 2^k = 320$ action logits. Training uses Adam (learning rate 10^{-3} , batch size 1024) for 12 epochs on 2×10^5 imitation examples, with a separate 20,000-example validation set and seed 7 in the saved notebook.

Rollout evaluation. Each rollout is *deterministic*: both players follow the same greedy policy head π_j (argmax over legal actions) until termination. The rollout outcome is 1 if the first player wins (makes the last move) and 0 otherwise. Since rollouts are deterministic, a single rollout per (position, head) pair suffices.

The per-head accuracy metric is whether the model’s rollout outcome matches the teacher’s rollout outcome—the correct channel bit—not whether it matches the P/N label directly. Each *exact teacher* head computes a specified linear functional ($r_0 = \sigma_0$; $r_j = \sigma_j \oplus \sigma_0$ for $j \geq 1$). The learned heads can make channel errors. The FSM uses an OR of the queried outcomes for P/N classification, which is a different task from reconstructing every nim-sum bit.

FSM-controlled shootout evaluation. Given a test position P , the evaluator runs the controller of Figure 3: it queries head π_0 first; if $r_0 = 1$ it immediately declares $\sigma(P) \neq 0$ (N-position). Otherwise it queries heads $\pi_1, \pi_2, \dots, \pi_{B-1}$ sequentially, declaring N-position as soon as any $r_j = 1$, or declaring P-position if all B bits are 0. Early stopping reduces the average number of rollouts below B on N-positions. We report (i) overall P/N evaluation accuracy and (ii) mean number of shootouts used.

Results. Table 2 reports the results.

Multi-head FSM achieves perfect classification. The saved output reports correct P/N classification on all 2,000 test positions, matching the teacher’s final decisions. However, the saved per-head channel accuracies are 2000/2000, 2000/2000, 1469/2000 and 1480/2000, respectively: 1.0000, 1.0000, 0.7345 and 0.7400. Consequently, this result does not establish exact channel recovery or recovery of the full invariant by the learned model. Early stopping leaves some channels unqueried, and even queried channel errors need not change the final OR decision. These are transcribed saved outputs, not a fresh checkpoint evaluation; reached-head error counts and a reproducible end-to-end rerun are needed for a stronger claim.

Method	Heads	Rollouts	Accuracy	Recall
Single head (model)	1	1.00	0.768	0.536
Multi-head + FSM (model)	4	2.87	1.000	1.000
Single head (teacher)	1	1.00	0.768	0.536
Multi-head + FSM (teacher)	4	2.87	1.000	1.000

Table 2: Single-frame P/N evaluation by FSM-controlled shootouts ($N = 20$, $k = 4$), reporting overall P/N accuracy and N -position recall.

Single-head baseline. The single-head baseline achieves 0.768 overall accuracy and 0.536 N -position recall. These numbers have an exact theoretical explanation. With $k = 4$ bits, an N -position has nim-sum $\sigma \in \{1, \dots, 15\}$. Of these 15 values, exactly 8 are odd ($\sigma_0 = 1$: the values 1, 3, 5, 7, 9, 11, 13, 15) and 7 are even ($\sigma_0 = 0$). Head π_0 detects an N -position only when $\sigma_0 = 1$, so the theoretical N -position recall is $8/15 \approx 0.533$ (observed: 0.536 on the finite test set). All P -positions are correctly classified (since $\sigma = 0$ implies $\sigma_0 = 0$), so on a balanced test set the overall accuracy is $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{8}{15} = \frac{23}{30} \approx 0.767$ (observed: 0.768). This calculation describes the particular classifier that declares an N -position exactly when $r_0 = 1$. It is not the winning-move success bound of Theorem 5.12(i), and does not establish a limit on arbitrary classifiers or rollout policies.

Different metrics. The recall $8/15$ measures the fraction of nonzero nim-sums detected by the exact least-significant-bit channel. Restoration accuracy instead measures whether a complete selected move cancels the full nim-sum. A resemblance between the recall and a training plateau is not evidence that the same learned mechanism produces both.

Early stopping. The average number of shootouts is 2.87 rather than the worst-case 4, because any N -position with $\sigma_0 = 1$ is caught at head 0 (one rollout); only those with $\sigma_0 = 0$ require further heads. *Scope.* We report the saved seed-7 configuration above. Perfect classification on this finite test set does not establish all-position correctness, robustness across seeds, or successful self-play discovery. We make no broader sweep claim without corresponding run records.

Reproducibility status. The restoration notebook, checkpoint-accuracy CSV and FSM notebook are retained with the local review materials. The public repository previously named for this work does not currently provide a reproducibility package; we therefore do not claim that code and data for all configurations are publicly available. The numerical results above remain tied to the saved runs and their stated limitations. A public versioned package and independently repeated evaluation are future work.

8 Discussion and broader implications

The three bypasses identify different resources for handling a global invariant: multi-frame inputs expose local transition information, multiple rollout heads provide complementary channels, and persistent recurrence accumulates the invariant through sequential updates. In the recurrent route, memory connects successive positions while the growing processing horizon supplies the depth needed for parity. The common design principle is to match the

information and computation available to the agent to the invariant it must maintain. We briefly discuss directions beyond the specific NIM setting.

8.1 Theoretical idealisation vs. practical architecture

The simulation theorem concerns Boolean literal-threshold gates with nonnegative weights and bounded thresholds, including their permitted compositions. The experiments use floating-point signed linear layers and ReLU as well as LayerNorm and a Log-Softmax loss. Their difference from the theorem therefore extends beyond normalisation layers. A failure at one finite board size does not establish an asymptotic representational impossibility; success on a finite dataset does not establish strong mastery. The controlled comparison supports a narrower empirical conclusion: the reported transition representation is much easier to learn under this supervised protocol.

8.2 Local information in combinatorial search

Exposing local invariants may also help guide combinatorial search. Our work on large Condorcet domains used structural constructions and local restrictions (Karpov et al., 2023; Zhou et al., 2024; Zhou, 2024). This is a motivating analogy, not an AC^0 reduction or a demonstrated parity lower bound for that search problem. Transferring the present argument requires a precisely encoded computational task.

8.3 Beyond NIM: harder impartial games as benchmarks

By Sprague–Grundy theory, every finite impartial game position has a nimber. However, the computational complexity of determining nimbers varies widely. Some impartial games have PSPACE-complete winner problems, such as directed vertex Geography (Schaefer, 1978). Unless $P = PSPACE$, no polynomial-time algorithm solves all such instances. Having a nimber does not supply an efficient way to compute it.

NIM is therefore a diagnostic starting point: its efficient optimal strategy lets us separate a representational obstruction from the difficulty of discovering a strategy. The longer-term benchmark is self-play on impartial games whose strategic structure is only partly understood or whose exact analysis becomes impractical as positions grow. Three candidates illustrate different challenges:

- **Sprouts.** Players join spots by noncrossing curves, add a spot on each new curve, and keep every spot’s degree at most three. The normal-play outcome conjecture for the standard n -spot opening remains open despite substantial progress with exact solvers (Čížek et al., 2026). Learning to exploit the evolving region structure, and to choose winning moves in intermediate positions, would be a stronger test than predicting opening outcomes from n alone.
- **Chomp.** A move removes a chosen square and all squares above and to its right; taking the distinguished corner loses. Strategy stealing establishes a first-player win on every nontrivial rectangle, but does not supply the winning moves (Berlekamp et al., 2001–2004). Larger boards and their irregular intermediate positions offer a challenge beyond solved special families; the analysis of Chomp continues to pose open problems (Nowakowski,

2025). The benchmark should test move selection and generalisation across shapes, not just the known opening winner.

- **Grundy’s game.** A move splits one heap into two nonempty unequal heaps. Whether its nim-value sequence is eventually periodic remains an open problem (Nowakowski, 2025). This offers a test of whether self-play discovers useful structure in component values and how they combine. Testing unseen heap sizes and sums of heaps can distinguish such progress from memorising a finite table.

These are proposed learning challenges, not consequences of the present AC^0 lower bound. An unresolved structural question is not itself a complexity lower bound, and worst-case hardness need not make a chosen instance distribution difficult. A useful evaluation would reserve unseen sizes and positions, report both training and decision-time computation, and compare play with exact solvers where feasible and strong search baselines elsewhere. Beyond the range of exact verification, improved win rates are evidence of stronger play rather than a solution of an open mathematical problem. The central question is whether self-play discovers reusable strategic structure that extends the range of positions it can handle under a fixed computation budget.

8.4 What the rollout separation establishes

Theorem 5.12(i) bounds winning-move selection for fixed bit-width, a specified family of root/one-heap measurements and a polynomial-size, constant-depth readout. It does not bound general single-policy search. Within that interface, the B structured channels reconstruct the invariant and permit direct move selection. The extra computational resource is the sequential rollout, not the number of neural output coordinates alone. Repeated deterministic measurements of the same affine channel cannot create additional channel bits, but changing the permitted query transformations or controller computation changes the model and can remove that restriction.

8.5 Learning bottlenecks after the representational barrier

The constructions establish that the required computations can be expressed using the stated resources: a shallow restoration rule, structured rollout policies, or a shallow recurrent update iterated over the board. This removes one obstacle before training begins. It does not determine whether gradient descent will find those computations, whether sparse game rewards provide enough guidance, or whether the resulting behaviour generalises.

The restoration experiment makes this distinction tangible. Even with transition information and correct supervision, learning passes through an intermediate plateau before reaching near-perfect measured accuracy. The checkpoint diagnostic in Section 7.2 associates the later improvement with the lowest bit of the selected move’s XOR change. We therefore have empirical evidence of a remaining learning bottleneck that is largely overcome in this run, alongside a theorem about a different, representational obstruction. The data do not establish how many distinct learning mechanisms produce the stages or why the transitions occur.

The immediate experimental question is whether AlphaZero-style self-play can discover the restoration rule, complementary rollout policies or recurrent accumulator without their

supervised targets. A follow-up should compare outcome-only training with curricula and auxiliary supervision, and examine the learned computations as well as win rates. For recurrence, this includes accurate accumulation and reliable state retention across separators. These controlled NIM tests would establish a basis for the harder benchmarks in Section 8.3; efficient self-play learning and the training regimes needed to achieve it remain open.

8.6 Significance and impact

Our results suggest several broader implications.

- **For learning theory:** representational sufficiency and learnability are separate questions. A promised predecessor state, complementary measurements, or sequential computation can each alter the resources available for an invariant-based decision.
- **For neural architecture:** short history windows (or difference features) can be a principled way to bypass parity-like barriers without scaling depth/width. When history is unavailable, multiple rollout heads offer an alternative within a single-frame architecture. When growing processing time is available, persistent recurrence offers a third route, building the invariant sequentially and retaining it across positions.
- **For practice:** circuit complexity can help formulate architectural hypotheses, provided the implemented model satisfies the mathematical assumptions. Tests of those hypotheses should distinguish representation, training dynamics, task distribution and computation budget.

9 Conclusion

A fixed-depth Boolean literal-threshold model with nonnegative fixed-grid weights and bounded thresholds is simulable in AC^0 . The NIM parity reduction then rules out strong single-frame mastery in that model.

Three constructions identify resources beyond that restricted inference problem. Two-frame access supports constant-depth restoration when the preceding position has nim-sum zero. Distinct structured rollout policies reveal complementary invariant bits; a single-channel limitation holds only under our explicit query and readout restrictions. Persistent recurrence implements running XOR over heap rows, exchanging fixed total latency for a growing sequence of simple updates. Cross-position memory additionally supports comparisons, while replay or board storage supplies the heaps needed for move selection.

The saved floating-point experiments show a large supervised two-frame restoration advantage and perfect finite-test P/N classification by a learned multi-head controller. The latter coexists with imperfect individual channel recovery. The restoration plateaus and checkpoint diagnostics also show that learning can proceed in stages after transition information is supplied. These observations do not certify the formal gate assumptions, worst-case mastery, or self-play discovery.

The contribution is a separation of the information and computation needed for a decision from the training process that must discover it. Establishing when self-play learns the restoration rule, complementary rollout policies or recurrent accumulator remains a distinct experimental question.

10 Development, verification and author responsibility

The earlier version of this paper was developed solely by the author, with editorial feedback from AI assistants. The present version was developed interactively with the AI assistants Astra and Claude. Extensive feedback from the Lean formalisation and its audits helped to improve the presentation, clarify definitions and hypotheses, and make the scope of the claims more precise. The author takes sole responsibility for the paper, including its arguments, interpretation and use of AI assistance.

The accompanying Lean development (Riis, 2026b) connects the mathematical results to explicit Boolean circuits, fixed-scale networks and complete game strategies. It checks the parity reduction, the finite probability bound and its attainability, move selection with binary outputs, the rollout executors and both shootout schedules, and the recurrent protocols with replay or board storage. The general weak-mastery definition, Fourier expansion and the real-parameter noise calculation are also formalised. An independent rebuild checks 1,588 theorem and lemma declarations across 38 local Lean files, comprising 24,493 source lines; these counts include auxiliary results and exclude Lean and Mathlib themselves.

The boundary of this verification is explicit. Classical complexity results, notably Håstad’s parity hardness and correlation bounds and the Linial–Mansour–Nisan/Tal spectral bounds, remain external inputs (Håstad, 1986, 2014; Linial et al., 1993; Tal, 2015). The single-channel result is checked conditional on a stated negligible parity-correlation theorem, including its application to the readout and its probability calculation. The spectral discussion uses a separate Fourier-tail premise; its connection to the exact conventions of a published tail theorem is not part of the checked development. The conditional implication from a tail bound to small parity correlation is checked. Empirical measurements and training runs are supported by experimental code and saved results, rather than by Lean proofs.

The author’s review of the formalisation focused on the interface between the paper and Lean: whether the formal definitions, hypotheses and conclusions express the intended mathematical claims. The author did not manually inspect every theorem or lemma declaration or every line of Lean source. Mechanical checking verifies the formal proofs relative to their definitions and stated hypotheses; it does not independently certify the correspondence with the prose, the suitability of the modelling assumptions, or the experimental results.

The accompanying interactive edition (Riis, 2026a) reproduces the paper text and links individual statements to their definitions, hypotheses and proofs, with optional explanations alongside them. The paper’s arguments remain self-contained at the level of its stated classical inputs; the interactive material supplies further detail rather than replacing a proof needed to understand a claim. Stable statement labels and a versioned correspondence record identify the formal results and any external assumptions. The edition includes the complete local Lean sources, pinned dependency information and reproducible build instructions. Claude generated the interactive edition and its link structure, which underwent a detailed, extensive audit by Astra. The author extensively cross-checked the edition but did not inspect every internal link and asset reference individually. An automated audit of this version verified the targets of 50,970 such references, including repeated occurrences. These navigation checks do not by themselves establish that each link provides the intended explanation or proof.

Neither the mathematical verification nor the experiments establish that gradient descent or self-play will discover the constructions in general.

Acknowledgments and Disclosure of Funding

The author received no dedicated funding for this work.

Appendix A. Formal definitions of fixed-scale quantised neural models

This appendix provides formal definitions of the fixed-scale quantised neural models used throughout the paper.

A.1 Preliminaries: fixed-scale quantisation and representable parameters

Fix constants $W, D \in \mathbb{N}$. Define the fixed finite set of representable parameters

$$\mathcal{Q}_{W,D} = \left\{ \frac{\ell}{D} : \ell \in \{0, 1, \dots, W\} \right\}.$$

In particular, *the set of allowable weights/thresholds does not depend on input length n* . This models *fixed-scale quantisation with bounded range*: values lie on a fixed grid (step $1/D$) and saturate at magnitude W/D , independent of the input length n . It is a restricted Boolean gate abstraction; neither floating-point nor general signed fixed-point networks automatically satisfy it. In particular, mere finiteness of a floating-point format is not what drives our AC^0 containment; the key is that the effective range/scale of thresholds does *not* grow with fan-in as n increases.

We will allow negations on wires (equivalently, AC^0 has NOT gates). This is standard in circuit complexity and avoids conflating two independent issues: (i) parity hardness vs (ii) monotonicity restrictions.

A.2 Feedforward model

Definition A.1 (Literal-threshold network). *A fixed-scale quantised NN of depth L on n Boolean inputs is specified by:*

- Layer widths $m_0 = n, m_1, \dots, m_L$, where $m_\ell \leq \text{poly}(n)$.
- For each layer $\ell \in \{1, \dots, L\}$ and unit $i \in \{1, \dots, m_\ell\}$:
 - weights $w_{\ell,i,j} \in \mathcal{Q}_{W,D}$ for $j \in \{1, \dots, m_{\ell-1}\}$,
 - a threshold $\theta_{\ell,i} \in \mathcal{Q}_{W,D}$,
 - an optional negation bit $\eta_{\ell,i,j} \in \{0, 1\}$ indicating whether input j is negated before weighting.

The computation proceeds as follows. Let $y^{(0)} = x \in \{0, 1\}^n$. For $\ell \geq 1$,

$$y_i^{(\ell)} = \mathbf{H}\left(\sum_{j=1}^{m_{\ell-1}} w_{\ell,i,j} \cdot (y_j^{(\ell-1)} \oplus \eta_{\ell,i,j}) - \theta_{\ell,i}\right),$$

where $\mathbf{H}(z) = 1$ if $z \geq 0$ and $\mathbf{H}(z) = 0$ otherwise. The network output is $y^{(L)}$ (or a designated subset of its coordinates).

A.3 Recurrent model (finite window)

Definition A.2 (Bounded-unrolling recurrence). *Fix a time window $T \in \mathbb{N}$. A fixed-scale quantised RNN consists of:*

- *input bits $x_t \in \{0, 1\}^n$ at time t ,*
- *a hidden state $h_t \in \{0, 1\}^m$ for some $m \leq \text{poly}(n)$,*
- *an output $o_t \in \{0, 1\}^k$ for $k \leq \text{poly}(n)$,*
- *fixed-scale quantised parameters (weights/thresholds in $\mathcal{Q}_{W,D}$ and optional input negations) shared across time,*

such that each update h_{t+1} and o_t is computed by a depth- L fixed-scale quantised threshold network over the current input and earlier states within the window. Each inference starts from an input-independent initial state, and the entire computation producing its output contains at most T recurrent updates. No boundary state may carry additional input-dependent computation from before that window. Unrolling this bounded computation yields a feedforward computation of depth $O(L \cdot T)$. A bound only on the lag between connected states does not imply this bound on total unrolling; the persistent model in Section 5.5 deliberately allows more than T updates.

A.4 Attention-style model (finite window)

Definition A.3 (Gate-realised attention-style model). *Fix a time window $T \in \mathbb{N}$. A fixed-scale quantised LTST model processes a sequence $(x_{t-T+1}, \dots, x_t)$ by:*

- *computing short-term and long-term attention scores using fixed-scale quantised parameters in $\mathcal{Q}_{W,D}$,*
- *combining attended summaries through a constant number of layers of fixed-scale quantised threshold computations,*
- *producing outputs $o_t \in \{0, 1\}^k$.*

Every score computation, input-dependent coefficient selection, encoding and value aggregation is required to have a polynomial-size, constant-depth implementation in the Boolean literal-threshold gates of Definition A.1. Intermediate state is Boolean, and total unrolling is bounded as in Definition A.2. Under these explicit implementation assumptions, the simulation theorem applies by composition. Merely requiring parameters or coefficients to lie in a finite set is insufficient. This extension makes no claim about standard softmax attention without such a reduction.

Remark A.4 (Scope of the simulation). *Theorem 2.4 applies because L, T, W, D are constants and each unit's parameters come from a finite set independent of n . Appendix B gives an explicit circuit construction.*

Appendix B. An AC^0 construction for simulating fixed-scale quantised networks

This appendix gives a clean circuit construction proving Theorem 2.4.

B.1 One threshold unit is in AC^0 under fixed-scale quantisation

We first show that a single fixed-scale quantised threshold unit (as in Definition A.1) is computed by a depth-2 AC^0 circuit of polynomial size.

Lemma B.1 (Fixed-scale quantised threshold gate $\in AC^0$). *Fix constants $W, D \in \mathbb{N}$. Let $z_1, \dots, z_m \in \{0, 1\}$ be Boolean inputs, and let $w_1, \dots, w_m, \theta \in \mathcal{Q}_{W,D}$. Consider the Boolean function*

$$f(z_1, \dots, z_m) = \mathbf{H}\left(\sum_{j=1}^m w_j z_j - \theta\right).$$

Then f is computable by a depth-2 AC^0 circuit of size $O(m^K)$, where $K := W$ is a constant (depending only on W, D).

Proof. Multiply the inequality by D to clear denominators. Write $w_j = a_j/D$ and $\theta = b/D$ with $a_j \in \{0, 1, \dots, W\}$ and $b \in \{0, 1, \dots, W\}$. Then

$$f(z) = 1 \iff \sum_{j=1}^m a_j z_j \geq b.$$

If $b = 0$ the function is identically 1. Assume $b \geq 1$.

Because each a_j is a nonnegative integer and each active input contributes at least 1 to the left-hand side whenever $a_j \geq 1$, any satisfying assignment contains a subset of at most b active variables whose coefficients already sum to at least b . More formally: if $\sum_j a_j z_j \geq b$, let $S = \{j : z_j = 1 \text{ and } a_j \geq 1\}$. Choose any multiset-free subset $T \subseteq S$ by greedily adding indices until the partial sum reaches $\geq b$. Since each added index increases the sum by at least 1, this process stops after at most b steps. Because $b \leq W$, we get $|T| \leq W =: K$.

Therefore,

$$f(z) = 1 \iff \bigvee_{\substack{T \subseteq [m] \\ |T| \leq K \\ \sum_{j \in T} a_j \geq b}} \bigwedge_{j \in T} z_j.$$

This is a depth-2 DNF: an OR of ANDs. Here depth counts AND/OR layers; counting input negations as gates adds at most one layer, as in the implementation bounds below. The number of terms is at most $\sum_{r=0}^K \binom{m}{r} = O(m^K)$, polynomial since K is constant. Unbounded fan-in AND/OR gates give an AC^0 circuit of depth 2 and size $O(m^K)$. $\square$

Remark B.2 (Negations). *If some inputs are negated before weighting, i.e. $(z_j \oplus \eta_j)$, this simply replaces z_j by either z_j or $\neg z_j$ in each conjunction term. Since AC^0 allows NOT gates, Lemma B.1 still applies.*

B.2 From one unit to a full network

Lemma B.3 (One layer of fixed-scale quantised units is in AC^0). *Fix W, D and let a layer consist of m' threshold units whose inputs are outputs of a previous layer of width m , with $m, m' \leq \text{poly}(n)$. Then the layer can be simulated by an AC^0 circuit of constant depth and polynomial size.*

Proof. Apply Lemma B.1 independently to each unit. Each unit yields a depth-2 circuit of size $O(m^K)$. Combine these m' circuits in parallel; the total size is $m' \cdot O(m^K) = \text{poly}(n)$ and depth remains 2. $\square$

Proof of Theorem 2.4. For a feedforward NN of constant depth L , repeatedly replace each layer by the AC^0 circuit from Lemma B.3, wiring outputs of layer $\ell - 1$ circuits as inputs to layer ℓ circuits. Depth multiplies by a constant factor: overall depth is $O(L)$ (constant), and size remains polynomial as a composition of polynomial-size layers.

For RNN and LTST with time window T , unroll the computation for T steps (constant) to obtain a feedforward computation of depth $O(LT)$ (still constant). Apply the same layer-by-layer replacement. $\square$

Appendix C. Implementation details: shallow circuits and fixed-scale networks

The constructions share a small collection of Boolean operations. Each heap is a B -bit block, and a move is a validity bit, a binary heap index, and a B -bit replacement value. The circuits below have polynomial size in their supplied input length. Their depth does not grow with the number of heaps or bits. All XORs described as shallow here combine *two* bits, or a fixed number under a bounded-change promise; an XOR of arbitrarily many heap rows is never treated as a primitive gate.

From a game rule to a network. The networks in this appendix are constructed explicitly from the game rules: their connections and weights are specified by an algorithm, rather than found by training. To see what the construction computes, revisit Figure 2. The preceding board $Q = (3, 5, 6)$ has nim-sum zero; after the opponent changes 5 to 4, the current board is $Q' = (3, 4, 6)$. Comparing the frames gives $\delta = 5 \oplus 4 = 1$. In parallel, the circuit proposes replacements $(3 \oplus 1, 4 \oplus 1, 6 \oplus 1) = (2, 5, 7)$ and tests which decrease their heaps. Only $2 < 3$ passes, so the output selects heap index 0 with replacement value 2, giving $(2, 4, 6)$ with nim-sum zero. For $B = 3$ and a two-bit index, the output fields are validity 1, index 00, and value 010.

The same move-selection operations are reused when rollout outcomes supply the target nim-sum. In the recurrent construction the target is accumulated over rows, and replay supplies the rows again for candidate testing. These interfaces explain how the constructions avoid asking one shallow network to extract parity from an arbitrary single board. More heaps require more parallel circuitry; executing a rollout or a recurrent scan still requires repeated evaluations.

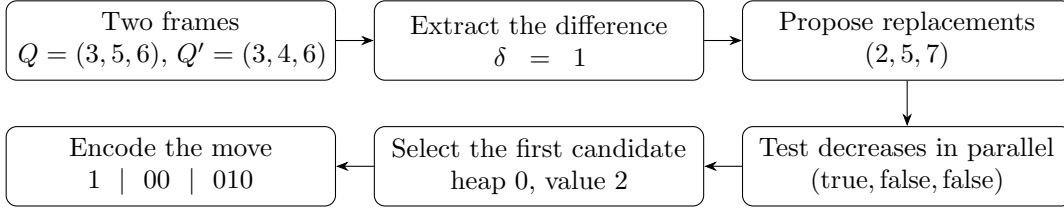


Figure 7: A schematic of the checked two-frame construction on one legal transition. Boxes group operations, not individual network layers. The general selector also handles multiple candidates, fallback moves and terminal boards.

Local differences and candidate moves. On a legal transition only one heap changes. Consequently, bit k of the delta is

$$\delta_k = \bigvee_{i=0}^{N-1} (x_{i,k} \oplus x'_{i,k}).$$

The single-change promise makes OR equal to the required XOR across changed rows. For a target δ supplied as input bits, form $y_i = x_i \oplus \delta$ in parallel. The test $y_i < x_i$ is shallow: OR over the possible highest differing bit, requiring equality at every higher bit and $y_{i,k} = 0, x_{i,k} = 1$ at that bit. Each test has size polynomial in B . When no cancelling candidate exists, use the first nonzero heap and subtract one. Bit k of that decrement is $x_{i,k}$ XOR the conjunction saying that all lower bits are zero. The validity bit is false exactly when the board is terminal. In restoration, the target comes from the two-frame delta; in the multi-head readout it comes from reconstruction of the rollout outcomes.

A binary index is part of the circuit. Let s_i be the candidate flags after choosing between cancellation and fallback. More than one may be true. Define

$$f_i = s_i \wedge \bigwedge_{j < i} \neg s_j, \quad I_k = \bigvee_{\substack{0 \leq i < N \\ \text{bit } k \text{ of } i \text{ is } 1}} f_i.$$

The f_i flags select the first candidate; the I_k wires give its binary index. Masking the candidate replacement bits by f_i and taking OR gives the value output. Combining validity with $\bigvee_i s_i$ handles the absence of candidates. This uses constant additional depth and polynomial size, rather than an uncosted external search for the first true flag. The Lean priority encoder proves agreement for every flag pattern, including invalid outputs, no candidates, and the cases $N = 0$ and $N = 1$. A move index uses r bits with $N \leq 2^r$; the replay counter also stores N and therefore requires $N < 2^r$.

The recurrent update. Scan mode updates the accumulator by two-bit XORs. Replay mode tests the current row against the stored summary, retains the first cancelling candidate and a fallback, and increments the row counter. Counter bit k changes exactly when all lower bits are one, so increment also has a shallow circuit. Mode, validity and reset flags select the appropriate register updates. Induction over well-formed token streams proves that every

reachable register fits its allotted width and that the output move uses the summary from the same board’s first scan. The input protocol supplies the repeated rows; the controller does not secretly store the board.

Realisation with fixed-scale threshold gates. An OR gate uses weights 1 and threshold 1; a negated literal supplies NOT. Implement AND by De Morgan, as an OR of negated inputs followed by negation. This keeps the threshold bound fixed even when fan-in grows. Constants use no inputs and threshold 0 or 1. Thus all these circuits can use the fixed grid $W = D = 1$. More precisely, for K output formulas of depth at most d , total formula size S , and input support below n , the checked compiler constructs one network of at most $2d + 2$ layers, at most $(2d + 1)S + K$ gates, and layer widths and fan-ins at most $S + n$. Identity buffers align layers; every output is proved equal to its source formula on the same finite input. Size counts duplicated formula nodes, and depth counts NOT gates, so these are conservative bounds.

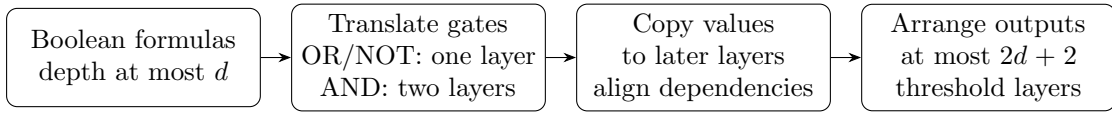


Figure 8: The checked Boolean-to-threshold compiler, shown schematically. Negated input literals are permitted. AND uses an OR of negated inputs followed by negation, keeping weights and thresholds in $\{0, 1\}$. Copying and output stages give a uniform depth bound.

Component	Boolean depth	Threshold layers
Two-frame binary move selector	≤ 20	≤ 42
Multi-head binary readout, given outcomes	≤ 19	≤ 40
One structured policy’s binary move output	≤ 17	≤ 36
One recurrent state update and output	≤ 8	≤ 18
One shootout controller tick	≤ 34	≤ 70

These explicit Lean-checked bounds are not optimised. Together with the polynomial size bounds, they certify the components’ AC^0 resource requirements. They apply to one evaluation, not to an entire rollout or row scan. The whole-game proofs connect the resulting network outputs to the corresponding game moves and state updates; the time needed to repeat those evaluations remains part of the algorithm.

Where the layer counts come from. For the two-frame selector, the compositional Boolean-depth bound is $4 + 4 + 9 + 3 = 20$: at most four levels extract the frame difference, four handle candidate tests and replacement bits, nine are allowed by the general selection construction (including fallback and output handling), and three encode the selected heap’s binary index. These are bounds on composed subcircuits, not necessarily the exact number of levels on a shortest implementation. The compiler then gives at most $2 \cdot 20 + 2 = 42$ threshold layers. Similarly, Boolean depths 19, 17, 8 and 34 give the other entries in the table.

The factor of two allows every Boolean level the cost of an AND gate, even though OR and NOT need only one threshold layer. The two additional stages copy inputs into the construction and arrange the output wires; identity gates retain values until they are needed. This uniform construction makes correctness and resource accounting straightforward, but is not a depth optimisation. The table proves that networks within these bounds exist, not that these depths are necessary or that training will discover their weights. A more economical implementation may use fewer layers; establishing sharper bounds would require a separate construction and proof.

Outer-controller storage and time. For $B, N \geq 1$ and $B \leq 2^r$, the shootout controller uses $NB + 4 + r + B$ state bits: a working board, control flags, a head index and the collected outcomes. Its input also contains the original NB -bit board, held fixed throughout the decision. This external storage is additional to the state count. The circuit handles head selection, board reload, outcome collection, increment and halt; the depth bound in the table applies to either the early-exit classifier or the all-head schedule.

Let M be the total number of simulated legal moves in the all-head schedule. The implementation uses $M + 2B + 2$ component evaluations: M move updates, B terminal tests, B outcome-consumption updates, one initial reset and one final move readout. Since $M \leq BN(2^B - 1)$, this agrees with the rollout-time bound in the main text while making the control overhead explicit. These are circuit evaluations, not measured processor time. The early-exit schedule uses only the visited heads and gives the same classification as the OR of all outcomes. These certificates concern the two specified schedules; the abstract definition of a finite-state controller does not certify arbitrary scheduling logic.

Appendix D. Weak mastery examples

Example 1 (Pairing strategy). *Consider positions $(a, a, b, b, c, c, \dots)$ consisting of pairs of equal heaps. The second player can mirror the first player’s move on the matching heap, maintaining the paired structure until termination, hence winning. The checked binary selector finds the first unequal pair and reduces its larger heap to match its partner. It has constant depth and polynomial size, and wins against every legal opponent from the paired family when playing second. For the first-player role every paired initial board is losing, so weak mastery there would be vacuous.*

The distinction from strong mastery is substantive. From the single heap (3), a strategy that takes all three stones wins immediately, irrespective of what it would do elsewhere. It therefore weakly masters that initial position as first player. If it has no prescribed move at (1), however, it fails strong mastery: arbitrary legal play can reach $(3) \rightarrow (2) \rightarrow (1)$ with the first player to move. That history is excluded by its own initial action. The general definition and this separation are checked in Lean.

Appendix E. Proof of Lemma 5.4

We provide an explicit AC^0 construction for computing number differences between positions differing in at most t heaps.

Proof of Lemma 5.4. Let positions be encoded as h blocks of length B bits (heap sizes in binary), so total input length is $n = hB$.

Step 1: identify changed heaps. For each heap i and bit position $k \in [B]$, compute $d_{i,k} = x_{i,k} \oplus x'_{i,k}$. Then $c_i = \bigvee_{k \in [B]} d_{i,k}$ indicates whether heap i changed. This is depth two after flattening the OR of the two-bit XOR DNFs, ignoring input negations in the usual convention.

Step 2: assume at most t changes. By promise, $\sum_i c_i \leq t$ with t constant.

Step 3: compute per-changed-heap XOR values. For each heap i , define the block XOR value $y_i = x_i \oplus x'_i$ (bitwise XOR across the B bits). Each output bit is XOR of two bits, hence AC^0 .

Step 4: XOR the at-most- t nonzero blocks. Since at most t heaps changed, $\Delta = \bigoplus_{i:c_i=1} y_i$ is XOR of at most t B -bit numbers. The locations of the changed blocks must also be selected. For each $S \subseteq [h]$ with $|S| \leq t$, form an indicator that S is exactly the changed set: $\bigwedge_{i \in S} c_i \wedge \bigwedge_{i \notin S} \neg c_i$. Conjoin it with the constant-size truth table for the parity of $\{y_{i,k} : i \in S\}$, and OR over all such S . There are $O(h^t)$ choices, each using at most $O(h)$ wires plus a constant-size parity table. This gives polynomial size and constant depth. On the single-change promise, each delta bit simplifies to $\bigvee_i y_{i,k}$.

The overall circuit has constant depth and polynomial size in n . $\square$

Appendix F. Independent noise and repetition

Remark 5.11 concerns a symmetric observation channel for an unknown bit Y . Each observation agrees with Y with probability $(1+a)/2$, where $0 < a \leq 1/2$, and observations are independent conditional on Y . A possibly randomised decision rule assigns a probability of outputting 1 to each observed bit vector. Let $m_*(a)$ be the least sample count for which such a rule has error at most $1/4$ for each possible value of Y .

Proposition F.1 (Repetitions needed for a weakly informative bit). *The minimum $m_*(a)$ exists and satisfies*

$$\frac{3}{32a^2} \leq m_*(a) \leq \left\lceil \frac{4}{a^2} \right\rceil \leq \frac{17}{4a^2}.$$

Majority voting, with ties declared 0, achieves the middle upper bound.

Proof outline. Under either value of Y , the sum of m correctness indicators has mean $m(1+a)/2$ and variance at most $m/4$. The second-moment bound gives majority error at most $1/(ma^2)$, so $m = \lceil 4/a^2 \rceil$ suffices. This also proves that a least sufficient natural number exists. For necessity, the likelihood ratio between the two product laws has chi-square divergence

$$\left(1 + \frac{4a^2}{1-a^2}\right)^m - 1.$$

If $ma^2 < 3/32$, use $1-a^2 \geq 3/4$ and $(1+u)^m \leq (1-mu)^{-1}$ for $u \geq 0$, $mu < 1$, to bound this divergence strictly below 1. Cauchy-Schwarz then bounds total variation strictly below $1/2$. A rule whose two errors are at most $1/4$ would distinguish the laws with acceptance probabilities differing by at least $1/2$, a contradiction. This applies to randomised rules as well. Finally, $a^2 \leq 1/4$ bounds the ceiling by $4/a^2 + 1 \leq 17/(4a^2)$. $\square$

For the flip model, $a = (1 - 2\varepsilon)^L$. Fresh independent blocks of L flips produce exactly the product observation law used above. When $0 < \varepsilon < 1/2$, the small-bias condition holds for every $L \geq \lceil 1/(2\varepsilon) \rceil$; one elementary bound is $(1 - 2\varepsilon)^L \leq (1 + 2\varepsilon L)^{-1}$. If $\varepsilon = 0$ or $L = 0$, one observation is exact. If $a = 0$, no number of observations permits error below $1/2$ for both ideal bits. Reusing the same corrupted bit leaves the information content unchanged. The finite product laws, real-parameter bounds, minimum sample count and these boundary cases are checked in Lean.

Appendix G. Additional proofs for the rollout analysis

Proof of Proposition 5.9. Fix a position P with heap sizes $h_1, \dots, h_N \in \{0, \dots, 2^B - 1\}$ and let $\sigma = \text{nim}(P) = h_1 \oplus \dots \oplus h_N$. For each $j \in \{0, \dots, B - 1\}$ define the deterministic rollout policy π_j :

“Subtract 2^j from the leftmost heap of size at least 2^j ; if none exists, subtract 1 from the leftmost nonzero heap.”

Each π_j is AC^0 -computable because it performs constant-depth comparisons to the threshold 2^j and selects the leftmost qualifying heap.

Consider a rollout under π_j from P . While some heap is at least 2^j , every move subtracts exactly 2^j from some heap, so the number of such moves is $L_1 = \sum_{i=1}^N \lfloor h_i / 2^j \rfloor$. After this phase all heaps satisfy $h_i < 2^j$, and π_j falls back to subtracting 1 until termination, for a further $L_2 = \sum_{i=1}^N (h_i \bmod 2^j)$ moves. Thus the total number of moves is $L = L_1 + L_2$, and (normal play) the first player wins iff L is odd, i.e. iff $L \bmod 2 = 1$.

For each i , the parity of $\lfloor h_i / 2^j \rfloor$ equals the j th bit of h_i , denoted $h_i^{(j)}$, so $L_1 \bmod 2 = \bigoplus_i h_i^{(j)} = \sigma_j$. For $j \geq 1$, $(h_i \bmod 2^j) \bmod 2 = h_i^{(0)}$, so $L_2 \bmod 2 = \sigma_0$. For $j = 0$, every remainder is zero and $L_2 = 0$. Hence $\text{RO}_{\pi_0}(P) = \sigma_0$ and $\text{RO}_{\pi_j}(P) = \sigma_j \oplus \sigma_0$ for $j \geq 1$. From these B outcomes the agent recovers σ .

When $\sigma \neq 0$, let k be its highest set bit. Some heap has bit k set; XOR with σ changes that bit to zero and leaves every higher bit unchanged, so a decreasing replacement exists. Choose the first such heap and set $h'_i = h_i \oplus \sigma$. The new nim-sum is $\sigma \oplus h_i \oplus h'_i = 0$. Comparisons, XOR and first-index selection are AC^0 . When $\sigma = 0$, output any legal move, or halt if every heap is zero.

This selector gives a winning strategy against every legal opponent response from a nonzero starting position. After the selected move, any opponent move changes one heap by a nonzero XOR difference and therefore makes the nim-sum nonzero again. The selector can restore zero on its next turn. Every legal move strictly decreases the total number of stones, so play terminates; since a nonterminal zero-nim-sum position cannot move directly to the terminal position, the selector makes the last move. The opponent need not follow any rollout policy.

Compute and retain each of the B root outcomes once per decision. Each rollout has at most $\sum_i h_i \leq N(2^B - 1)$ moves, so their total is at most $B \sum_i h_i \leq BN(2^B - 1)$. This counts simulated game moves, rather than elementary bit operations or elapsed time. Multiplying by the polynomial gate work per policy update gives a bit-work bound; it is polynomial in N when $B = O(\log N)$, and the displayed step bound is exponential in unrestricted binary heap width. $\square$

Appendix H. Proof of Theorem 5.12

Proof. Part (i). Fix $B \geq 2$, $N \geq 1$ and put $q = 2^B$. Let U be the uniform distribution on all N independent B -bit heaps, before conditioning on a nonzero nim-sum. For any fixed values of $N - 1$ heaps and any target nim-sum, exactly one value of the remaining heap gives that target. Thus σ is exactly uniform on $\mathbb{F}_2^B$, and $\Pr_U[\sigma \neq 0] = (q - 1)/q$. Let $\ell = \ell_j$ be the nonzero linear functional returned by the fixed structured head, and let $\beta = \ell(\sigma)$.

For each fixed $b \in \{0, 1\}$, decode $F_N(P, b)$ as a proposed move (i, v) . On legal outputs define $g_b(P) = h_i \oplus v$, and set $g_b(P) = 0$ on invalid outputs. Selecting an indexed input block, checking legality and taking its XOR with the output use polynomial-size, constant-depth circuits. Thus for each $a \in \mathbb{F}_2^B$, the indicator

$$H_{b,a}(P) = \mathbf{1}\{g_b(P) = a\}$$

is in AC^0 . On $\sigma \neq 0$, a legal move wins exactly when $g_\beta(P) = \sigma(P)$; invalid outputs cannot satisfy that equality.

For $u \in \mathbb{F}_2^B$, write $\chi_u(P) = (-1)^{\langle u, \sigma(P) \rangle}$. First consider the finite probability calculation under the explicit assumption that some $\eta_N \geq 0$ satisfies

$$|\mathbb{E}_U[H_{b,a}(P)\chi_u(P)]| \leq \eta_N$$

for every $b \in \{0, 1\}$, $a \in \mathbb{F}_2^B$ and nonzero $u \in \mathbb{F}_2^B$. This is a two-sided correlation bound under the unconditioned distribution U ; it does not assume independence between the readout and the nim-sum. Expanding the exact-value indicator gives

$$\mathbf{1}\{\sigma = a\} = q^{-1} \sum_{u \in \mathbb{F}_2^B} (-1)^{\langle u, a \rangle} \chi_u(P),$$

so

$$|\mathbb{E}_U[H_{b,a}\mathbf{1}\{\sigma = a\}] - q^{-1}\mathbb{E}_U H_{b,a}| \leq \frac{q-1}{q}\eta_N.$$

For each fixed b , the events $g_b(P) = a$ are disjoint. Therefore

$$\begin{aligned} \Pr_U[\text{winning move and } \sigma \neq 0] &= \sum_{a \neq 0} \mathbb{E}_U[H_{\ell(a),a}\mathbf{1}\{\sigma = a\}] \\ &\leq q^{-1} \sum_{b \in \{0,1\}} \sum_{\substack{a \neq 0 \\ \ell(a)=b}} \mathbb{E}_U H_{b,a} + \frac{(q-1)^2}{q}\eta_N \\ &\leq \frac{2}{q} + \frac{(q-1)^2}{q}\eta_N. \end{aligned}$$

Dividing by $\Pr_U[\sigma \neq 0] = (q-1)/q$ gives the finite bound

$$\Pr_U[\text{legal winning move} \mid \sigma \neq 0] \leq \frac{2}{q-1} + (q-1)\eta_N. \quad (\text{H.1})$$

It remains to supply the correlation estimate for the circuit class. For $u \neq 0$, χ_u is parity on N times the number of nonzero coordinates of u , hence on at least N input bits. Fix

all coordinates outside this support. The restricted indicator remains a polynomial-size, fixed-depth AC^0 circuit, so the standard parity-correlation bound applies uniformly to every such restriction; averaging over the fixed coordinates preserves the bound (Håstad, 1986, 2014). Taking the maximum over the finitely many $a, b, u \neq 0$ gives a nonnegative negligible η_N . Since B , the circuit depth and the polynomial size exponent are fixed, $(q-1)\eta_N$ is also negligible. This yields the asserted asymptotic bound. The fixed- B assumption is essential to this stated asymptotic conclusion.

For attainability of the leading constant, choose a nonzero a_b from each fibre $\ell(a) = b$, possible because $B \geq 2$. After observing b , try to cancel a_b using the first heap h_i with $h_i \oplus a_b < h_i$, with any legal fallback if none exists. This is an AC^0 readout and wins whenever $\sigma = a_0$ or a_1 , an event of conditional probability $2/(q-1)$. Any additional success is at most negligible by the upper bound. For $B = 2$ and π_0 , the choices $a_0 = 2, a_1 = 1$ give exactly $2/3$: the case $\sigma = 3$ is assigned a legal delta-1 move and cannot win.

Part (ii). The B root rollout bits reconstruct σ . If it is nonzero, a heap containing its highest set bit can be decreased to $h_i \oplus \sigma$, leaving zero nim-sum. An AC^0 first-qualifying-heap selector implements this choice. With zero nim-sum, any legal move is optimal in the game-theoretic sense that no winning move exists. Each root rollout has at most $N(2^B - 1)$ moves. Hence B rollouts use $O(BN2^B)$ simulated moves, each with polynomial circuit work; no successor enumeration or unrestricted controller is used.

Part (iii). If the predecessor Q has nim-sum zero and the opponent legally reaches Q' , then $\delta = \Delta(Q, Q') = \text{nim}(Q') \neq 0$. Select a decreasing heap $h_i \rightarrow h_i \oplus \delta$ as above. The two-frame difference and move selector are AC^0 , and the response returns nim-sum zero. Repeating this response after the player's own successful moves wins from the established invariant. No assertion about establishing it from an arbitrary start is used. $\square$

References

- Miklós Ajtai. Approximate counting with uniform constant-depth circuits. In *Advances in Computational Complexity Theory*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 13, pages 1–20, 1993.
- Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays*, Vols. 1–4. AK Peters/CRC Press, 2001–2004.
- Tomáš Čížek, Martin Balko, and Martin Schmid. Massively parallel proof-number search for impartial games and beyond. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(43):36838–36846, 2026. <https://doi.org/10.1609/aaai.v40i43.41010>.
- Amit Daniely and Eran Malach. Learning parities with neural networks. In *Advances in Neural Information Processing Systems*, 33, 2020.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.
- Harvey Friedman. AI Challenge. *FOM* mailing list, September 28, 2017. <https://fomarchive.ugent.be/2017-September/020612.html>.

- Harvey Friedman. NIM/Deep Learning. *FOM* mailing list, October 15, 2017. Includes correspondence from John Schulman, October 14–15, 2017. <https://fomarchive.ugent.be/2017-October/020642.html>.
- Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.
- Yiding Hao, Dana Angluin, and Robert Frank. Formal language recognition by hard attention transformers: perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810, 2022.
- Johan Håstad. *Computational Limitations for Small Depth Circuits*. PhD thesis, MIT, 1986.
- Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM Journal on Computing*, 43(5):1699–1708, 2014. doi:10.1137/120897432.
- Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018.
- Alexander Karpov, Klas Markström, Søren Riis, and Bei Zhou. Set-alternating schemes: A new class of large Condorcet domains. *arXiv preprint arXiv:2308.02817*, 2023.
- Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM*, 40(3):607–620, 1993.
- William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *International Conference on Learning Representations*, 2024. <https://arxiv.org/abs/2310.07923>.
- Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.
- Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. Worst-case optimal join algorithms. *Journal of the ACM*, 65(3):16:1–16:40, 2018.
- Richard J. Nowakowski. Unsolved problems in combinatorial games. In *Games of No Chance 6*, MSRI Publications, vol. 71, pages 55–98, 2025. <https://library.slmath.org/books/Book71/files/GONC6-04.pdf>.

- Nasim Rahaman, Aristide Baratin, Devansh Arpit, et al. On the spectral bias of neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 5301–5310, 2019.
- Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mat. Zametki*, 41(4):598–607, 1987.
- Søren Riis. Overcoming a learning bottleneck in impartial games: Interactive edition. 2026a. <https://sr123.github.io/impartial-games-learning-lean/>.
- Søren Riis. Overcoming a learning bottleneck in impartial games: Lean formalisation and archived interactive edition. Version 1.0.0, Zenodo, 2026b. <https://doi.org/10.5281/zenodo.23289503>. Lean sources: <https://github.com/SR123/impartial-games-learning-lean/tree/main/lean>.
- Thomas J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.
- Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.
- David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.
- David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharmashan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.
- Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 77–82, 1987.
- Avishay Tal. Tight bounds on the Fourier spectrum of AC^0 . *Electronic Colloquium on Computational Complexity (ECCC)*, TR14-174, 2015.
- Christopher James Thornton. *Backpropagation can’t do parity generalisation*. University of Sussex, School of Cognitive and Computing Science, 1996.
- Chris Thornton. Parity: the problem that won’t go away. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 362–374. Springer, 1996.

Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.

Bei Zhou. *Reinforcement Learning for Impartial Games and Complex Combinatorial Optimisation Problems*. PhD thesis, Queen Mary University of London, 2024.

Bei Zhou, Klas Markström, and Søren Riis. CDL: A fast and flexible library for the study of permutation sets with structural restrictions. *SoftwareX*, 28:101951, 2024.

Bei Zhou and Søren Riis. Exploring parity challenges in reinforcement learning through curriculum learning with noisy labels. *arXiv preprint arXiv:2312.05379*, 2023.

Bei Zhou and Søren Riis. Impartial games: A challenge for reinforcement learning. *Machine Learning*, 115:54, 2026. First preprint 2022. <https://doi.org/10.1007/s10994-026-06996-1>.