

Batched Neuroevolution with Analytic Readouts: A Finite-Parity Construction Study

Daniel Schmitter

Abstract

Evolutionary graph search can be separated from the convex fitting of a supervised linear readout. We describe a padded recurrent-feature population whose readouts are fitted by batched ridge solves, and analyze a retained parity construction experiment with speciation and crossover. The named sweep fits the complete two- through seven-bit truth tables exactly; the seven-bit run records 37 hidden units, 128 graph connections, and 119 evaluated generations. These are construction results on exhaustively supplied truth tables, not held-out generalization or compactness optima. Source inspection shows that nonlinear connection splitting is only approximately identity even at equilibrium, and that the neural-evolution reference differs in graph dynamics, inputs, readout, objective, population, and execution hardware. Its timing cannot isolate the benefit of the analytic readout. We derive the conditional ridge reduction and padded-feature invariance, verify them on small numerical fixtures, and give complete resource and comparator qualifications. The result is an implementation-oriented case study of separable architecture search rather than a claim to surpass canonical NEAT.

1 Introduction

Searching a neural architecture and fitting its output coefficients need not be the same optimization problem. When a candidate graph produces fixed features and the supervised loss is quadratic, its best regularized linear readout can be obtained directly. The outer search can then concentrate on graph structure and feature dynamics. This does not make the outer problem easy; it removes a particular inner optimization variable.

We study a concrete implementation that combines this separation with dense batching of heterogeneous small graphs. The contribution is a self-contained computational description, a reproducible accounting of a finite construction task, and explicit boundaries on preservation and comparison claims. No general acceleration theorem, new evolutionary principle, or spline-specific effect is established.

2 Related work

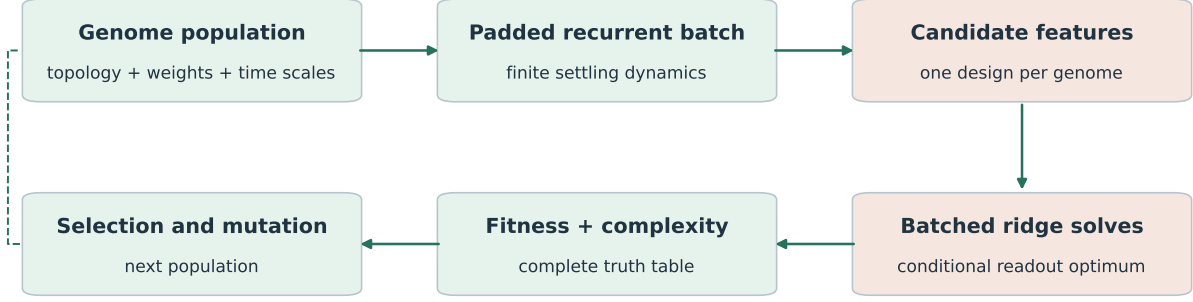
NEAT evolves topology and weights while protecting structural innovation through historical markings and speciation [1]. It is not fundamentally an algorithm with an inner backpropagation loop. Eliminating linear coefficients from a mixed nonlinear fitting problem is the classical variable-projection principle [2]. The present implementation applies a conditional ridge solve to evolved recurrent features and batches the resulting calculations. These ingredients precede the experiment; the contribution here is their particular executable composition and its measured scope.

3 Architecture and information flow

The outer loop evolves nonlinear features, while the inner loop fits supervised coefficients analytically. Padding makes heterogeneous graph candidates compatible with batched array operations. Each candidate is settled for a finite number of steps from a declared initial state, then its feature matrix is passed to a ridge solve. Fitness is computed after that solve. The readout is therefore learned, but it is not mutated as part of the genome.

The distinction between representation-preserving padding and structural mutation is crucial. Adding inactive zero columns leaves the fitted prediction unchanged. Inserting a nonlinear leaky node changes both the static transfer and transient state, even with compensating weights. These facts are different from exact spline refinement. The finite truth table also gives the inner solve direct supervised targets; an episodic reward would not provide those targets automatically.

Search features, solve their readout



Read left to right above, then right to left below. Arrows show data or computation flow. A later iteration uses the retained state.

Figure 1: Search features, solve their readout. Boxes distinguish supplied information, fitted components, and the quantity evaluated. Arrows show computation or data dependence, not a newly trained deep network.

4 Population realization and conditional fitting

Each genome contains hidden nodes, weighted source-to-hidden connections, and positive time constants. The readout weights are not genome variables. Local packing maps active hidden identifiers to a bounded array and fills absent entries with zeros. Each candidate receives the input coordinates, a constant, and its own nonlinear hidden state. Starting from zero state, the code applies six settling steps by default.

$$h^{(r+1)} = h^{(r)} + D_{\tau}^{-1}(-h^{(r)} + M^T[x; 1; \tanh h^{(r)}]), \quad (1)$$

$$Z_{\mathcal{G}} = [x; 1; \tanh h^{(T)}]^T, \quad (2)$$

$$W_{\mathcal{G}} = (Z_{\mathcal{G}}^T Z_{\mathcal{G}} + \lambda I)^{-1} Z_{\mathcal{G}}^T Y. \quad (3)$$

The time constants are at least one in the unit-step update. This bounds each leak factor but does not prove contraction of the coupled recurrent network: recurrent weights also determine stability. Finite settling is therefore part of the feature definition, not an exact continuous-time differential-equation solution or an assertion that equilibrium has been reached.

For a fixed feature matrix and positive ridge coefficient, the displayed coefficient formula follows from the normal equations and is implemented as a solve. Padding a feature matrix with zero columns leaves predictions unchanged: the extended normal matrix is block diagonal with an additional lambda times identity block, and the new cross-statistic rows are zero. This statement concerns inactive columns. Truncating a genome at the hidden-node cap can change its function and is not covered by the proof.

For P candidates, N truth-table rows, and d padded features, feature storage scales as PNd , dense Gram storage as Pd^2 , and a straightforward factorization as Pd^3 . Population parallelism can improve throughput on an accelerator without removing those storage and arithmetic costs. Parameter counts must include the readout, even though it is not mutated.

5 Complexification and preservation

The implementation uses globally tracked split nodes, connection-pair matching, compatibility-based species, fitness sharing, and crossover. Its selection score is a squared-error-derived fitness with a small hidden-count penalty, while champion reporting prioritizes classification accuracy. The two orderings need not select the same graph. These are implementation choices, not a faithful reproduction of every canonical NEAT rule.

A split replaces a connection with a new tanh node using incoming scale epsilon and compensating outgoing scale one over epsilon. The static expansion is

$$\frac{\tanh(\varepsilon a)}{\varepsilon} = a - \frac{\varepsilon^2 a^3}{3} + O(\varepsilon^4 a^5). \quad (4)$$

Thus a nonlinear split is not exactly function preserving. The new leaky state also introduces transient dynamics during finite settling, so even a small static approximation error does not establish transient equivalence. This differs fundamentally from exact nested spline refinement. Reusing the word growth for both operations must not import the latter’s guarantee into this evolution system.

6 Experimental methods

We retain the named from-scratch sweep over parity dimensions two through seven. Inputs enumerate all binary combinations and are mapped to minus one and one; targets are parity labels. The same exhaustive truth table fits the readout, supplies evolutionary fitness, and determines reported accuracy. This is a finite function-construction problem. It has no independent test distribution or evidence of length generalization.

The sweep artifact retains accuracies, hidden and connection counts, generations, throughput, and elapsed seconds, but not complete commands, model checkpoints, or repeated-seed distributions. Available source defaults include population 300, hidden cap 40, six settling steps, and ridge coefficient 0.01. They are not substituted for missing execution metadata. Separate single-rung files differ in timing, so the manuscript consistently reports the named sweep rather than mixing the fastest timings.

A retained neat-python reference uses feedforward sigmoid graphs, one evolved output, binary zero/one inputs, squared-error fitness, a different population and CPU-style per-genome execution. Its recorded failures are not a controlled ablation of analytic versus evolved readouts. We preserve its complete results in the supplement but do not form a speedup ratio. A deterministic check with 19 rows, five active features, four padded zeros, two outputs and seed 41850 tests padded-ridge invariance. Another checks the static split on 1001 points in the interval minus two to two at epsilon 0.3. No evolutionary search is rerun.

7 Results

Table 1: Complete named construction sweep. Accuracy is one on the supplied truth table for every row.

Parity bits	Hidden nodes	Graph connections	Generations	Seconds
2	1	3	2	0.17
3	1	4	3	0.16
4	3	11	14	1.49
5	2	8	9	0.89
6	22	68	48	9.57
7	37	128	119	36.30

The small cases show that a conditional readout and evolved features can represent these finite tables. The rapid growth from two hidden units at five bits to 37 at seven also prevents a general minimality claim. Hidden-node counts omit direct input readout coefficients and output weights. Throughput falls from thousands of candidate evaluations per second as the task and candidates grow; it is not a constant-cost solver.

The zero-padding fixture agrees to numerical roundoff. The nonlinear-split fixture has a nonzero maximum error, retained numerically in the supplement, despite the compensating outgoing weight. This confirms the analytic distinction between padded representation identity and nonlinear structural mutation. It does not evaluate the full finite-settling mutation error or change any historical evolutionary outcome.

8 Discussion

The useful application condition is supervised quadratic readout fitting. In a control problem with only episodic reward, target outputs needed by this solve are not automatically available. A large throughput number on truth-table regression therefore does not imply sample-efficient autonomous control or removal of reinforcement-learning credit assignment.

Exhaustive parity construction requires growing features

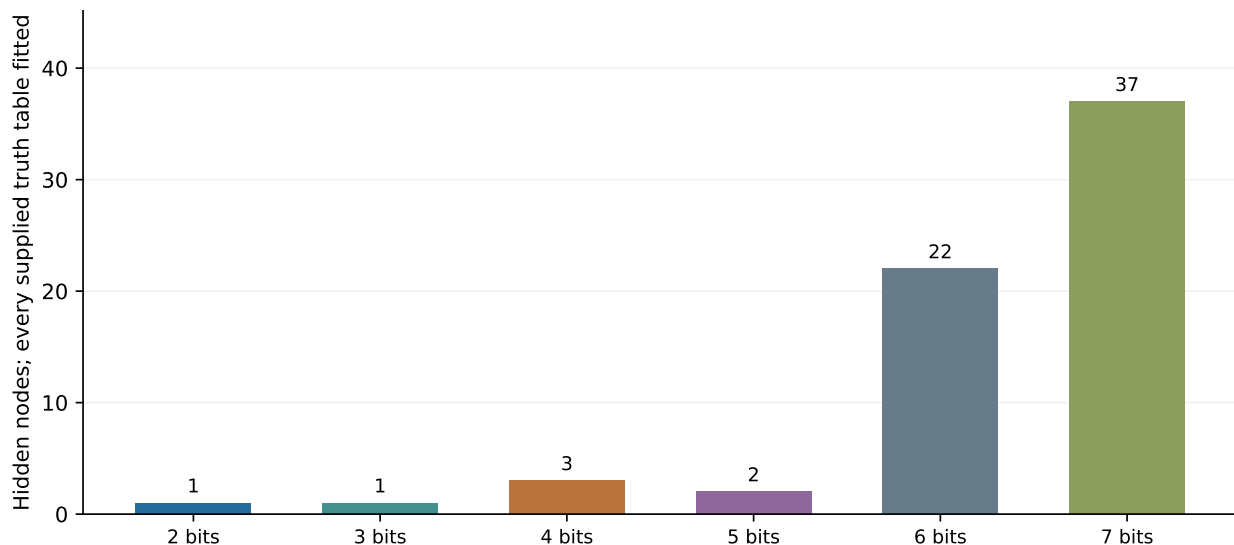


Figure 2: All six archived parity construction sizes. Each complete supplied truth table is fitted; node counts are not held-out performance or minimal architectures.

The framework is a reasonable way to allocate computation: search for nonlinear features, solve coefficients conditionally, and batch compatible work. Its present evidence cannot establish a unique algorithmic advantage over well-tuned evolutionary or reservoir methods. In particular, canonical NEAT is not refuted by one unmatched reference, and batching is not a capability unique to this implementation. All six sweep rows and the unfavorable comparator qualifications are retained.

9 Application boundary and research implication

This architecture is a concrete application of separable optimization to model construction. Its value is easiest to test where evaluating candidate features is expensive and the readout problem really is quadratic. The parity record demonstrates construction, not extrapolation or superiority over canonical evolutionary methods under matched resources.

10 Conclusion

Analytic readouts provide a precise conditional optimization inside a batched topology search. The prototype constructs parity functions through seven bits on their complete truth tables. Its scientific contribution is the executable separation and transparent finite-task evidence, with explicit distinctions between exact padding, approximate mutation, and unqualified cross-system timing.

References

- [1] K. O. Stanley and R. Miikkulainen. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* 10(2), 99-127, 2002.
<https://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf>
- [2] G. H. Golub and V. Pereyra. The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate. *SIAM Journal on Numerical Analysis* 10(2), 413-432, 1973.
<https://doi.org/10.1137/0710036>