Boolean Learning Machines

Jack Schenkman

February 3, 2026

Note: This is an early draft of this work. This is not the complete paper.

Abstract

This work proposes a way of autonomously discovering systems that do not delineate between cognitive tasks such as learning, memory, and inference. These systems, referred to as Boolean learning machines, are defined by a digital circuit, meaning there is a clear way to build hardware that physically realizes them. A covariance matrix adaptation evolution strategy (CMA-ES) is used to optimize the system for learning ability. For optimizing both learning ability and energy efficiency, a multi-objective CMA-ES is implemented. The system is evaluated on an extremely simple toy example consisting of meta-learning 5-input Boolean functions. Parameterizing the entire AI system with logic gates unlocks new opportunities for energy efficiency, speed, hardware robustness, and the exploration of architectures / paradigms that would previously not even be considered due to incompatibility with current AI hardware.

1 Introduction

Biological intelligence blurs many of the cognitive boundaries that exist in AI systems. While efforts to achieve longer contexts, develop more efficient inference, enhance reasoning, and have more capable memory proliferate, biological intelligences do not seem to depend on such strong delineations. Although certain brain regions may specialize in certain modalities, the architecture of the brain is remarkably homogeneous and intelligence manifests in a number of substrates beyond neurons[1][2]. A similar contrast exists at the hardware level. While most AI hardware has separate regions of the system dedicated to compute and memory with special interconnect uniting them, in biology the physical mechanisms behind memory, compute, and interconnect all seem to overlap to some extent.

I suspect that it may be practically impossible to maintain strict delineations between cognitive roles and neat partitions of hardware functionalities while achieving systems with capabilities and levels of efficiency well beyond current technology. Efforts underway toward continual learning and test-time compute represent a small step toward the disintegration of these boundaries. Systems that do not exhibit such boundaries will have much greater capabilities and be much more performant in terms of energy, latency, and sample efficiency. In this work I propose a class of these systems referred to as Boolean learning machines.

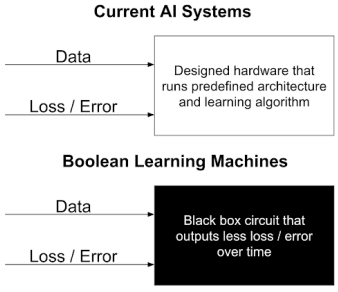

Figure 1: This figure shows the difference between current AI systems dependent on predefined architectures / algorithms and Boolean learning machines in which the system is viewed as a dynamical system without an architecture and learning algorithm specified by humans

1.1 Discovering New AI Systems

Leveraging AI can greatly accelerate the invention of new systems that unite reasoning, memory, and continual learning. The frontier labs have expressed interest in developing AI scientists which would advance these efforts [3][4][5]. These approaches largely consist of models proposing new ideas and implementing them as programs rather than models manually editing the weights of AI models. The research presented in this work is very complementary to these ongoing AI scientist efforts.

1.1.1 Insufficiency of Backpropagation for AI System Discovery

It is unlikely that backpropagation alone is sufficient to discover these new AI paradigms. To clarify, this claim relates to backpropagation discovering a new AI paradigm, not whether the new paradigm should itself implement backpropagation. Parts of the system are likely non-differentiable, potential changes are likely to be non-differentiable, and the system might need to run over many steps if it is to learn, which might make it practically impossible to use a gradient based method to update the systems parameters. Evolutionary methods coupled with program synthesis offer a promising approach.

1.2 Hardware For New AI Systems

It is important that discovered AI systems can be implemented efficiently in hardware [6]. Impressive progress in AI-based EDA tools such as AlphaChip might lead one to believe that an AI scientist can discover a new AI method which can be parameterized in any arbitrary way, and then engineers can leverage existing AI-based EDA tools to produce a chip for this new paradigm [7][8]. This line of reasoning faces two main flaws. The AI-based EDA tools are intended to ingest the way current AI is parameterized and would likely not perform well on the bulk of the possible paradigms the AI scientist might propose. Even if the AI-based EDA tools could work for any design, there would be an issue in which large amounts of time might be consumed evaluating physically unrealizable designs.

It is clear that the algorithms need to be designed with the hardware in mind. The field of physics based computing seeks to find combinations of substrates and physical designs which directly implement machine learning algorithms. Despite impressive progress in research areas such as equilibrium propagation and thermodynamic computing, it remains incredibly challenging to produce such systems at scale and on-par with existing hardware solutions [9][10]. The complexity of algorithmic considerations coupled with the needed physics expertise and semiconductor knowledge makes this immensely challenging.

1.3 Boolean Networks

This work aims to marry the potential of autonomous exploration of new AI systems with physics based computing. Boolean networks provide a physically realizable parameterization for AI systems. Feedforward and convolutional networks have been effectively implemented in networks of logic gates [11][12]. Similar efforts have been explored in recurrent neural networks such as work on logic gate implementations of neural cellular automata and applications such as sequence-to-sequence learning [13][14]. Most of these approaches have a fixed network topology and use backpropagation to optimize the identity of each gate via a continuous relaxation. This relaxation involves representing each gate as a continuous function (for example AND= $A \cdot B$) and representing the choice of each gate as a probability distribution over options. After training, the system is hardened, meaning the distribution of gates is replaced with the gate with the highest probability. By using backpropagation much larger logic-gate networks have been trained then was possible with previous optimization techniques premised on genetic algorithms.

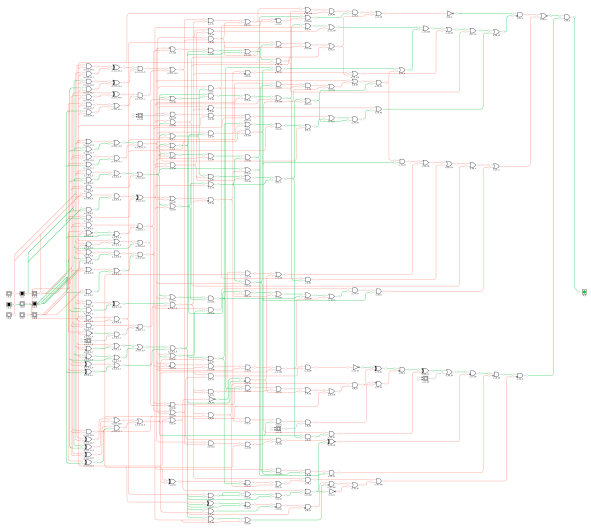

Figure 2: Example Boolean circuit utilized in cellular automata from [13].

The existing Boolean logic-ML approaches focus on accelerating inference, in which inference is a forward pass through a network. Inference is only one small piece of the current AI workload and one that will no longer exist in its current form as the other cognitive tasks mentioned all blend together. The circuit produced by current Boolean logic-ML approaches enable inference for a specific model already trained on a specific dataset. What if the practitioners wants to change the model or train on new data or fine tune? In principle, parameterizing AI systems by Boolean logic should offer many advantages. To realize these gains, the scope of what we mean by AI system needs to expand from "forward pass" to the full cognitive process. I refer to these types of systems as "learning machines". The natural world is filled with such systems across all scales from bacteria to human brains.

2 Boolean Learning Machines

This work seeks to use an evolutionary algorithm to design the Boolean learning machine, meaning a chip that could in principle be fabricated. It is tempting to think that there is no way to have a static Boolean circuit (fixed connections and logic gate types) that just by giving it data and error signals can learn. Wouldn't the process of learning require changing logic gates or forming new connections between gates? Consider what happens inside of a GPU. Even for work that uses backpropagation to come up with the logic gates, the backpropagation is ultimately run on static logic gates in the GPU. This work refers to a static digital circuit which can adapt its output in a manner that resembles learning as a Boolean learning machine. The delineation of weights and activations no longer makes much sense. If one views the gates as weights and the state of each gate as the activations then this work is expanding upon previous efforts to have activation based learning [15][16]. If one views the weights and activations of the system as being abstract concepts that are simply implemented in the logic gates, then it might be that there are weights changing on different timescales, placing this work more as a continuation of research into fast weights [17][18].

2.1 Technical Implementations of Boolean Learning Machines

The approach imagines that cells are arranged in a 3-dimensional lattice. Each cell implements a 6 or 7-input Boolean function represented as a truth table. The 6 inputs come from the cell's 6 spatial neighbors. When the cells implement 7-input functions, the cell's own state is the 7th input. Each cell sends the output of its Boolean functions to its neighbors. The connectivity is fixed. There are 3 categories of cells: sensory cells, hidden cells, and motor cells. Sensory cells receive an external Boolean signal instead of implementing a function. They send this signal to their neighbors. Hidden cells implement a Boolean function. Motor cells implement a Boolean function and their output is used in computing errors / losses. Since the system is recurrent, it takes multiple timesteps for

inputs to propagate to have an effect on output cells. In general an input is presented for multiple timesteps so the system has time to propagate this change. This iterative process resembles energy based models (EBMs) in which the system settles into a low energy state [19]. Traditionally, in EBMs the optimization process shapes the energy landscape to cause the target output for a given input to correspond to a low energy state. Since Boolean learning machines mix many cognitive tasks together, the role of the energy landscape is less clear.

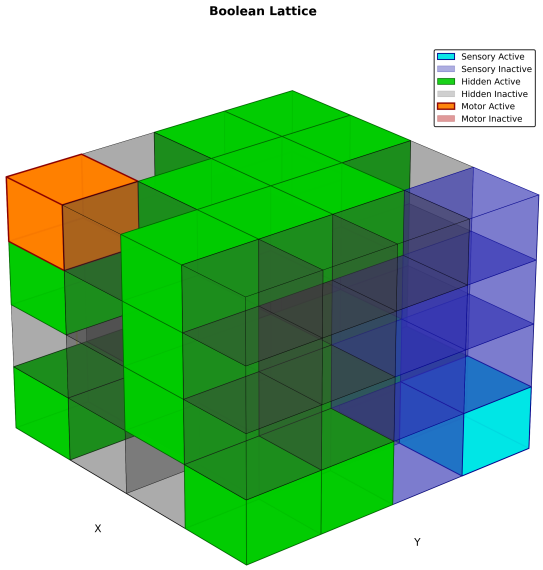

Figure 3: A 4 x 4 x 4 Boolean lattice is shown. The lattice contains the three types of cells: sensory, hidden, motor cells. Each cell can be either active or inactive. A random activity pattern is shown.

3 Optimizing Boolean Learning Machines

There must be some optimization process that produces the Boolean network which is the learning machine. The principles by which this learning machine operates might not be easily understandable. This should not be surprising, considering it is created through an optimization process rather than a manual design. Gradient descent is not feasible as the optimization approach. For such long timescales it would not work in principle due to vanishing / exploding gradients. The memory requirements of storing the intermediate states could also balloon. Evolution is the least worst optimizer option. To clarify, this work does not propose that the learning machine should use evolutionary algorithms.

3.1 Evolutionary Optimization of Boolean Learning Machines

At first glance it seems impossible to imagine that by running evolutionary algorithms on a computer something with capabilities similar to a brain will emerge. Consider the breadth of possible algorithms and hardware designs. Evolutionary algorithms often struggle to optimize systems with more than a few hundred free parameters. How could they design systems that rival the performance of the human brain with its 20 billion neurons and 100 trillion synapses? The appearance of the brain in evolution complicates the natural skepticism surrounding the viability of evolutionary algorithms for this task. Evolution did not need to stumble upon our current brain design among the near infinite space of possible computational concepts. The principles governing the human brain are descendants of principles in simpler organisms and if one looks far enough back single celled organisms or chemical reaction networks that predate life.

The 3 gigabytes that comprise the human genome is barely enough space to store a few minutes of video and certainly incapable of specifying every synapse in the human brain. The genome provides a program or conditions for the assembly of the brain. Indirect encoding is the term used by the

evolutionary algorithms community to describe this technique in which the genotype that evolves is used to govern the development of a phenotype which is evaluated and used to guide evolution [20]. This work is not focused on implementing key mechanisms of the brain, but rather on setting up the conditions by which key properties of biological intelligence naturally emerge.

3.2 Indirect Encoding

To assign a Boolean function to each cell, a graph neural network (GNN) performs message passing over the lattice. This process occurs before any data is presented to the lattice. The genotype is the weights of the GNN which is exposed to the evolutionary algorithm. The topology of the GNN is fixed. Each cell in the lattice is initialized with a random state vector. For the purposes of these experiments this is usually a 2 or 4 dimensional vector. An MLP referred to as the development network ingests the state vectors of the 6 neighbors of each cell and outputs a residual vector of the same dimension. In some variants the MLP also takes as input the state vector of the current cell. This residual (scaled by a constant factor) is added to the current state of that cell. This process happens in parallel for all cells in the lattice for a fixed number of time steps (30 in these experiments).

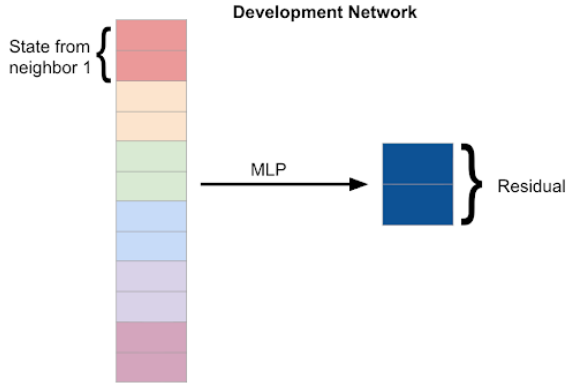

Figure 4: The states from all neighbors are concatenated to form the input of the MLP that generates the residual.

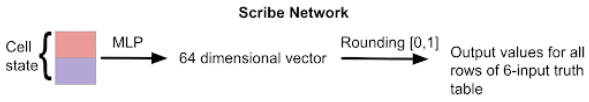

Figure 5: After development finishes, each cell's state is projected to a 64 dimensional vector, passed through a tanh activation function and then rounded to [0,1], to produce the output values of the truth table for the cell.

Once the development process has completed the scribe network converts each cell state into a Boolean function. The scribe network takes as input the cell's state vector and outputs a vector of the same dimension as the number of rows in the cell's truth table. In these experiments the each cell implements either a 6-input Boolean function (takes its six neighbors as inputs) or a 7-input Boolean function (takes its 6 neighbors and itself as inputs). This means the truth table either has 64 or 128 entries respectively. The output of the scribe network is rounded on the interval [0, 1], producing a binary vector specifying the truth table for the cell.

4 Fitness

The open-endedness of developing systems that are generally good at learning and reasoning exacerbates the common difficulty plaguing machine learning research of identifying a good learning signal.

Initially, this work frames this question within the context of a typical supervised meta-learning framework. Imagine that a task τ_i, consisting of examples x^i and targets y^i is drawn from a set of possible tasks $\bigcap \tau$. For task τ_i the Boolean learning machine is presented with x^i and y^i for N epochs, in which an epoch consists of presenting all entries within the dataset. The goal is for the system to eventually output y^i when presented with x^i. During the process of forming this association, neither the topology of the circuit nor the identities of the gates are modified. The formation of the association depends on the activities of the cells maintained by the feedback within the lattice. The ability to learn task τ_i can be quantified by the change in the accuracy, Δ_{acc_i}, of predicting y^i given x^i between the first and final epochs. The set of possible tasks $\bigcap \tau$ consists of $\bigcap \tau_{fitness}$ and $\bigcap \tau_{eval}$. $\bigcap \tau_{fitness}$ consists of $N_{fitness}$ tasks and $\bigcap \tau_{eval}$ consists of N_{eval} tasks. The fitness of an individual is based on the average change in accuracy over the fitness tasks, $\sum_{i \epsilon N_{fitness}} \Delta_{acc_i}$. Assessing in context learning capability is often complicated by a lack of clarity as to whether the test was in the training set. In this work, the average change in accuracy over the evaluation tasks, $\sum_{i \epsilon N_{eval}} \Delta_{acc_i}$, quantifies the ability of the system to learn tasks absent during the evolutionary optimization process.

5 Meta-Learning Boolean Functions

In this work, $\bigcap \tau$ consists of different 5-input Boolean functions. Before beginning evolution 3000 randomly chosen 5-input Boolean functions are assigned to $\bigcap \tau_{fitness}$ and 1000 randomly chosen 5-input Boolean function are assigned to $\bigcap_{\tau_{eval}}$ such that there is no overlap between the two sets. Although a 5-input Boolean function has a truth table with 32 rows, for simplicity each τ_i consists of 4 randomly selected rows of the truth table. Within the lattice there are:

- 7 sensory cells: 5 for the Boolean input, 1 for a flag indicating if the output is being provided, 1 for the target output value
- variable number of hidden cells depending on lattice size: perform local computations
- 1 motor cell: for the output

During the process below, no change to the gate identities or network topology are made.

```
For each individual:
  For a given Boolean function:
```

- `Select 4 examples as a dataset. Initialize the output values of the hidden cells to 0.`
- `Begin epoch 1`
  - `Pick example: present to network with output_provided flag set to 0, let the network run for t timesteps. Record the motor output and compute the accuracy. Present same input with output_provided flag set to 1, and target output value`
  - `Do the same for all examples`
- `For each subsequent epoch repeat the process shuffling the example order`
- `Do a final epoch in which the motor outputs are recorded. Look at the accuracy at the first and final epoch. The difference between the final accuracy and the initial accuracy is referred to as the learning percent.`

This process happens for all Boolean functions for each individual. Each Boolean function is learned independently of the other, meaning the state is not shared across them. The learning percentages across all functions for a given individual are averaged to determine the individual fitness. CMA-ES is used as the evolutionary algorithm. The weights of the development and scribe network are modified and a new set of individuals are sampled.

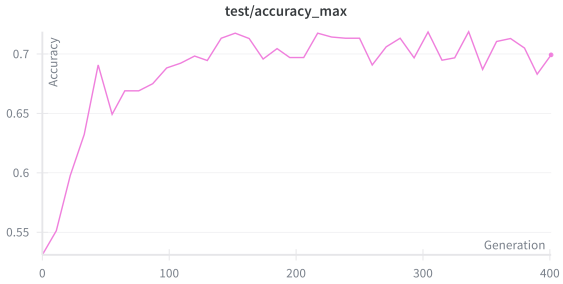

Figure 6: The best accuracy on the evaluation set is shown for each generation.

6 Evolutionary Dynamics

The system does not learn to get perfect accuracy, but it is able to perform well above chance on learning Boolean functions that were not used to guide the evolution. The system gets to about 70 percent accuracy on the test set. 50 percent would be random performance. This performance does not mean that the system is generalizing from one Boolean function to another. There are only 32 possible input vectors. The same input maps to different outputs depending on the function. The system will see one mapping during evolution guidance say $000000 > 1$ and it will see a different mapping during testing $000000 > 0$. It must learn new mappings on the fly.

6.1 Indirect Encoding Behaviors

The development process can present key insights into the behavior of the Boolean learning machine. The figure below shows the 30 steps of the development process. To clarify, this is not showing the activity of the lattice when processing data. In practice the scribe network is only run at the end of the process. For visualization purposes, it is run here at every step to allow the reader to see how the Boolean functions change over the course of development. It is difficult to visualize each cell when the function it implements is defined by a 64-dimensional vector. To simplify visualization, consider specific rows of each cell's truth table.

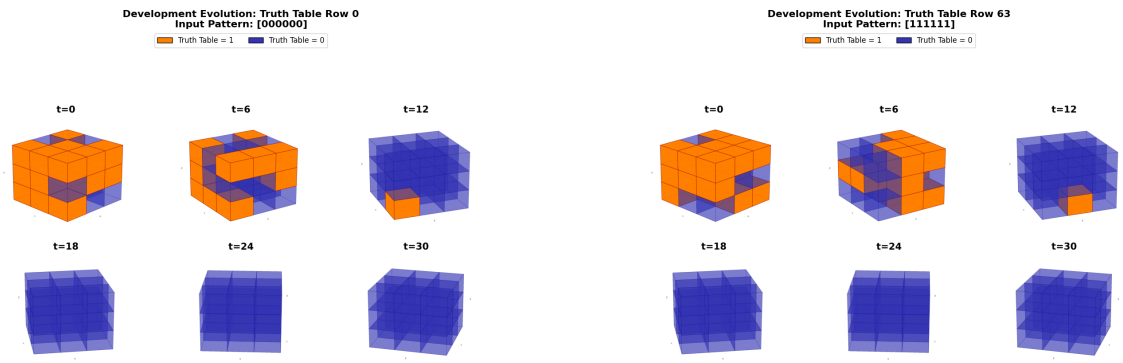

Figure 7: $3 \times 3 \times 3$ lattice with a channel size of 2 for the development network at generation 1. The color of each cell represents the value to which the input maps during each step of development for inputs 000000 (left) and 111111 (right).

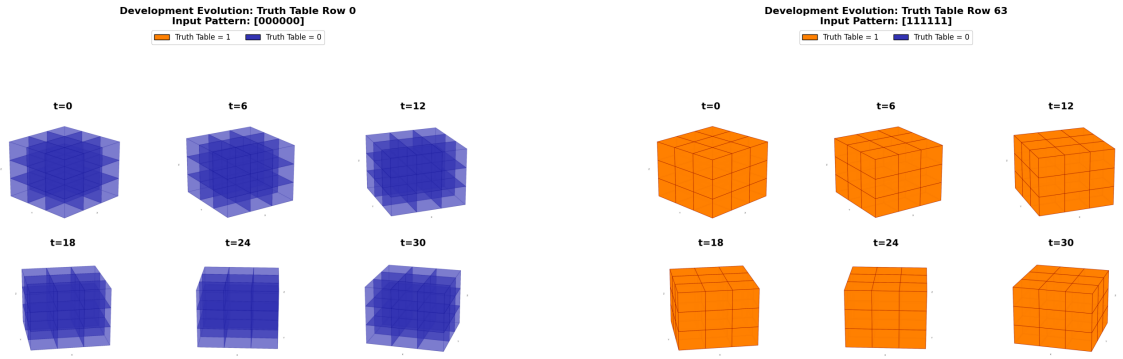

Figure 8: $3 \times 3 \times 3$ lattice with a channel size of 2 for the development network at generation 373. The color of each cell represents the value to which the input maps during each step of development for inputs 000000 (left) and 111111 (right).

The development process converges quickly considering there is not much difference between the output of the truth table for the two rows analyzed at later timesteps. Evolution exacerbates this tendency. By generation 373, the lattice exhibits a high degree of homogeneity such that all cells seem to behave almost identically at least for the two inputs investigated. To investigate the under-utilization of the potential of the development network, the lattice size and hidden channel size were increased.

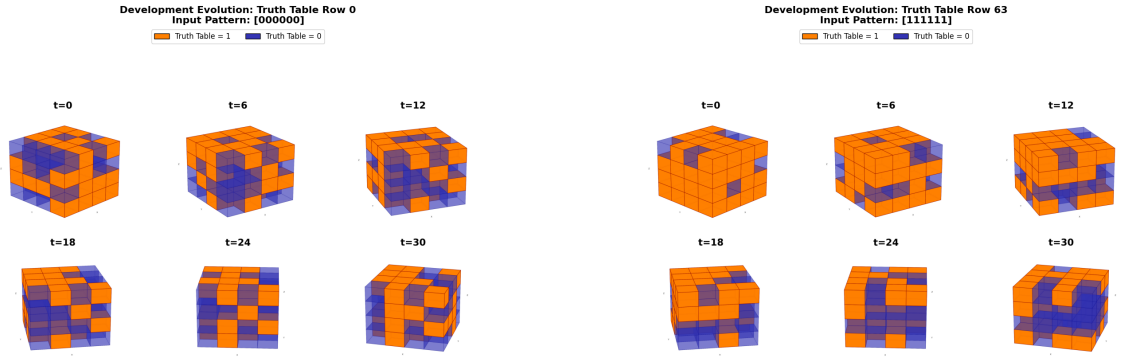

Figure 9: $4 \times 4 \times 4$ lattice with a channel size of 4 for the development network at generation 1. The color of each cell represents the value to which the input maps during each step of development for inputs 000000 (left) and 111111 (right).

There was much more heterogeneity in the functions implemented in each cell and also more changes during development. These larger lattices with larger channel sizes did not seem to have their fitness improve until thousands of generations had passed. The improvement in their fitness was tied to the heterogeneity of their cells' functions disappearing.

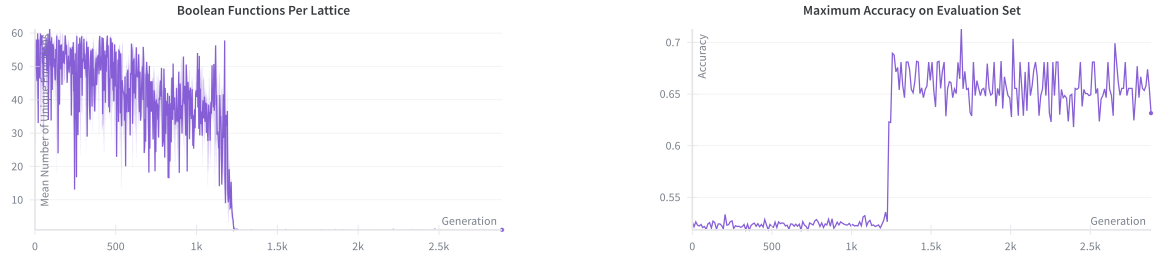

Figure 10: Evolutionary dynamics of functional homogenization. (Left) The average number of unique Boolean functions implemented per lattice decreases over evolution, eventually reaching one, indicating that all cells converge to the same function. (Right) The sharp improvement in accuracy coincides with the disappearance of functional heterogeneity across cells.

Further work will investigate modifications to the development process to support more heterogeneity. Using zero-padding rather than wraparound for the cells during development and having the message passing network ingest a value measuring the contrast between a cell and its neighbors might help.

6.2 The Single Boolean Function

Evolution seems to converge on a single Boolean function. This is likely an artifact of the way in which the indirect encoding process is structured. While the homogeneity of the lattice limits its expressivity, the space of 6-input Boolean functions is still vast with about 1.8×10^{19} possible choices. The Boolean functions for the 3 x 3 x 3 lattice and the 4 x 4 x 4 lattice respectively are shown below.

11111111111100111000101111100001010010101100000010000000001000000

11001111010101110000010000001110101000100011110000000100010011111

What properties do the functions on which evolution arrives posses? For a given function, the weight-response curve measures the probability of an input with a specific Hamming weight (number of ones) outputting a one. This provides some insight into the types of feedback that exist within the lattice. If the weight-response curve monotonically increases it would suggest that the lattice exhibits positive feedback loops causing all cells to take on the same value, which would make it difficult for the system to preserve memories. For the two examples analyzed, the weight-response curve is not monotonically increasing. The investigation of the properties of these Boolean functions in isolation and within the lattice deserves much more attention than it receives here.

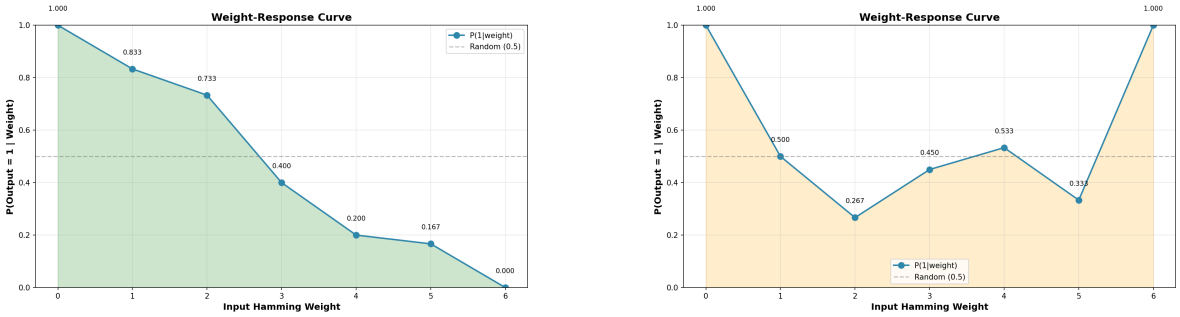

Figure 11: Weight response curves for the Boolean function across different lattice sizes. The response for the $3 \times 3 \times 3$ lattice is shown on the left, and the response for the $4 \times 4 \times 4$ lattice is shown on the right.

7 Power Consumption

Energy expenditure is a vital force shaping the trajectory of biological systems, including the brain, and the deployment of AI systems. Because the substrate of computation in Boolean learning machines differs from the neurons in the brain, the way in which these energy constraints are addressed

need not be the same. For traditional approaches to AI the linkage between a new algorithm and the energy consumed is highly nonlinear, dependent on a number of factors such as the hardware used and the compilation technique. Quantifying power consumption in Boolean learning machines is straightforward. Power consumption in digital circuits is often broken down into dynamic and static power. Dynamic power comes from gates switching between on and off. Static power stems from leakage current when the gate is on.

$$P_{dynamic} = \alpha C_L V_{DD}^2 f$$

$$P_{static} = V_{DD} I_{leakage}$$

In these equations α refers to the activity factor meaning the fraction of clock cycles for which a gate switches, C_L refers to the gate capacitance, V_{DD} refers to the voltage of the system, f refers to clock frequency, and $I_{leakage}$ refers to the subthreshold leakage current. Since this work investigates a very simplified model of gates the dynamic and static power are measured in terms of the total power consumed over the course of learning.

$$E_{dynamic} = \sum_{t=0}^{T} \sum_{i \epsilon N} \delta_t^i$$

$$E_{static} = \sum_{t=0}^{T} \sum_{i \epsilon N} \lambda_t^i$$

T refers to the total number of time steps in the training process. N refers to the number of cells in the lattice. δ_t^i is an indicator function for whether the ith cell had a state transition at time step t. λ_t^i is an indicator function for whether the ith cell is on at time step t. Although the actual dynamic power is more closely tied to the state of one of the gates within a cell transitioning, this approach provides a rough estimate of the power consumption. Static power in digital CMOS is not typically tied to a gate outputting a 1, so the static power consumption term used in this work is more of a sparsity penalty.

7.1 Multi-Objective Optimization

Previous results focused on optimizing for only learning ability. The ideal Boolean learning machine should not only learn well, but it should have a number of other attributes including energy efficiency and robustness to defects. Multi-objective CMA-ES was used to explore the joint optimization of learning ability, dynamic energy consumption, and static energy consumption. A population of 150 and an initial σ of 0.5 was used.

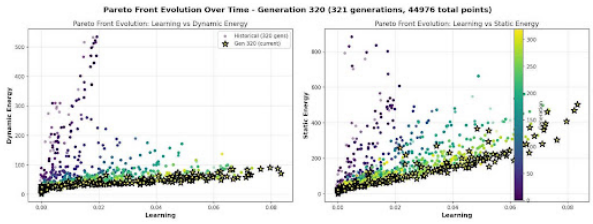

Figure 12: This figure shows the progression of the Pareto front over the course of 321 generations. The color of the dot represents the generation to which it belongs. The stars represent the current Pareto Front. The x-axis learning value refers to the average accuracy improvement of the individual during its "learning". It might seem strange from one 2-dimensional plot there will be members of the Pareto front that are behind others. This is a quirk of the visualization. Being on the Pareto front depends on the value of all 3 objectives but each plot only shows a 2-dimensional slice.

8 Conclusion

In this work, recurrent Boolean circuits were evolved to learn 5-input Boolean functions. All learning was represented by changes in the activity of the Boolean circuit, rather than its gates or topology.

The system reached about 70 percent accuracy on the evaluation set. While this result is far from perfect, this work is intended as proof of concept.

As AI systems take on increasingly open-ended, long running roles, there cannot be a strict separation between modes of operation such as learning, reasoning, memory, and inference. This work is intended to encourage further exploration of machine learning frameworks that blur the boundaries between these cognitive tasks. Autonomous exploration of these new frameworks unlocks AI paradigms that humans would fail to consider. Ensuring that these new AI systems are easily implemented in hardware or pair well with advances in automated hardware design is critical. Boolean learning machines are a promising approach to leverage progress in autonomous discovery of AI algorithms for physics based computation. Although this work is not focused on replicating the mechanisms of biology, some of the methodologies parallel approaches found in nature. Knowledge of natural phenomenon might inspire engineering decisions taken in this work, thereby biasing this observation.

This work departs from the traditional ML framework in many ways. Considering the number of small optimizations that collectively make current ML approaches performant, the viability of Boolean learning machines might depend on a similar effort. The space of opportunities to expand upon this work is too large to fully enumerate here. The development process and the optimization process have much room for improvement. The framing of learning within the context of a typical supervised learning problem warrants a re-evaluation. Perhaps learning should not be selected for outright. A more "natural" way might be to have an artificial creature in some world and its brain is a Boolean learning machine. Survival is based on overcoming whatever obstacles exist within this world. A system with general problem solving skills might emerge. We would need to find a way to harness the problem solving capabilities of this artificial creature for our needs.

References

[1] Vernon B Mountcastle. The columnar organization of the neocortex. *Brain: a journal of neurology*, 120(4):701–722, 1997.

[2] Robert Chis-Ciure and Michael Levin. Cognition all the way down 2.0: neuroscience beyond neurons in the diverse intelligence era. *Synthese*, 206(5):1–31, 2025.

[3] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

[4] DARWIN GÖDEL MACHINE. Of self-improving agents.

[5] Alexander Novikov, Ngân Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. *URL: https://arxiv.org/abs/2506.13131*.

[6] Sara Hooker. The hardware lottery. *Communications of the ACM*, 64(12):58–65, 2021.

[7] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

[8] Yuanyuan Duan, Xingchen Liu, Zhiping Yu, Hanming Wu, Leilai Shao, and Xiaolei Zhu. Rlplanner: Reinforcement learning based floorplanning for chiplets with fast thermal analysis. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–2. IEEE, 2024.

[9] Jack Kendall, Ross Pantone, Kalpana Manickavasagam, Yoshua Bengio, and Benjamin Scellier. Training end-to-end analog neural networks with equilibrium propagation. *arXiv preprint arXiv:2006.01981*, 2020.

[10] Maxwell Aifer, Zach Belateche, Suraj Bramhavar, Kerem Y Camsari, Patrick J Coles, Gavin Crooks, Douglas J Durian, Andrea J Liu, Anastasia Marchenkova, Antonio J Martinez, et al. Solving the compute crisis with physics-based asics. *arXiv preprint arXiv:2507.10463*, 2025.

[11] Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Deep differentiable logic gate networks. *Advances in Neural Information Processing Systems*, 35:2006–2018, 2022.

[12] Felix Petersen, Hilde Kuehne, Christian Borgelt, Julian Welzel, and Stefano Ermon. Convolutional differentiable logic gate networks. *Advances in Neural Information Processing Systems*, 37:121185–121203, 2024.

[13] Pietro Miotti, Eyvind Niklasson, Ettore Randazzo, and Alexander Mordvintsev. Differentiable logic cellular automata: From game of life to pattern generation. In *Artificial Life Conference Proceedings 37*, volume 2025, page 54. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2025.

[14] Simon Bührer, Andreas Plesner, Till Aczel, and Roger Wattenhofer. Recurrent deep differentiable logic gate networks. In *Proceedings of the 2nd International Workshop on Edge and Mobile Foundation Models*, pages 31–36, 2025.

[15] Paul Williams and Randall Beer. Environmental feedback drives multiple behaviors from the same neural circuit. In *Artificial Life Conference Proceedings*, pages 268–275. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2013.

[16] Eduardo Izquierdo, Inman Harvey, and Randall D Beer. Associative learning on a continuum in evolved dynamical neural networks. *Adaptive Behavior*, 16(6):361–384, 2008.

[17] Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 29, 2016.

[18] Faustino Gomez and Jürgen Schmidhuber. Evolving modular fast-weight networks for control. In *International Conference on Artificial Neural Networks*, pages 383–389. Springer, 2005.

[19] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[20] Jeff Clune. *Evolving artificial neural networks with generative encodings inspired by developmental biology*. Michigan State University, 2010.