GFly: Reproducible Browser Experiments with Learning in a Fly Connectome

Jiahe Tian | HiOS Labs Inc.

ipo@stanford.edu | ORCID: 0009-0007-8750-287X

Abstract

Browser demonstrations can show a connectome making useful choices without establishing what learned or whether the anatomical wiring mattered. GFly Learning Lab provides a compact, inspectable environment for answering these questions. It uses 80,404 connections from the MaleCNS release in a declared rate model with reward driven gains, matched wiring controls and portable experiment checkpoints. Across 360 runs on three synthetic tasks, the anatomical condition reaches 90.6% expected correct choice probability on cue associations and 88.2% on a combined cue task. It reaches only 47.1% after reward reversal within the specified adaptation budget. An additional 180 runs use published handwriting images with the original separate writer training and test partitions. The altered wiring controls learn successfully, so the experiments do not establish an advantage for the selected projection neuron to Kenyon cell input partners under this decoder. All 20 tested browser runs reproduce uninterrupted learning exactly after saving and restoring a checkpoint within the same engine. The contribution is a tested research and teaching artifact that exposes learning assumptions and supports repeatable comparisons. It is not a new reinforcement learning algorithm or evidence of biological brain emulation.

1 Motivation and scope

A useful demonstration should allow a reader to distinguish a learned decision from a response supplied by its surrounding software. This distinction matters when a neural wiring diagram is placed inside a game. A conventional search procedure may already produce strong candidate actions. A trained output layer may account for an improvement even when the internal wiring is unimportant. Neither circumstance makes the software unusable, but each changes what its behavior can support as evidence.

GFly Learning Lab addresses this problem through a small experiment that can be inspected and continued in a browser. A reader can disable learning or change input partners while retaining the task, training budget and random seed. The learned state can be exported with the task settings and random streams. The same JavaScript implementation produces the paper measurements and the interactive demonstration. The software therefore makes a reported outcome directly testable without requiring the reader to install a machine learning framework.

The contribution is an executable comparison baseline for studies of learning in a selected connectome circuit. The release connects the data transformation to one JavaScript model, six experimental conditions, complete records from 540 training runs and a verified procedure for continuing a saved experiment. A subsequent study can reuse these components while changing a stated part of the task or model. The completed comparisons distinguish learning from a benefit of the selected anatomical input partners under this decoder and retain a failure to adapt after reward reversal. The intended users are researchers prototyping small decision tasks and instructors explaining controls. Improvements in their productivity or understanding have not been measured.

2 Relationship to existing work

The mushroom body provides a well studied anatomical setting for associative learning. Aso and colleagues describe its organization and the relationship between Kenyon cells, output neurons and dopamine systems [2]. Xie and Ocker study learning and structural interventions in a model with projection neurons, Kenyon cells and output neurons [4]. Their work is close to the present computational setting. The use of these cell classes, learned output connections or wiring interventions is therefore not claimed as new.

Connectome based reservoir computation also predates this project. Morra and colleagues investigate a fly lateral horn reservoir on a multifunctionality benchmark [5]. Connectome Lab already documents browser training, controls with altered wiring, held out evaluation, custom tasks in JSON and saved runs [6]. Its reported capabilities were reviewed from the project documentation; its benchmarks were not independently reproduced here. GFly contributes an inspectable reference implementation with released run records and tested continuation behavior. Browser training and model checkpoints are not new methods proposed by this work.

Behavioral fidelity is a separate question from useful computation. The digital sphinx study demonstrates fly walking through a model using a worm connectome and a learned interface, illustrating why plausible output alone does not establish a biological explanation [7]. The present model is deliberately limited to an engineered feedforward computation. The separate GFly application includes a spiking simulator inspired by Shiu and colleagues [9], but those dynamics and the game demonstrations are not the experimental system evaluated in this paper.

3 Data and computational model

3.1 Circuit extraction

The source is the MaleCNS v1.0 significant connections file with minimum confidence 0.5, together with its body annotations [1]. The extraction selects annotated ALPN input cells, Kenyon cells and MBON output cells. A Kenyon cell is retained only if it receives an ALPN connection and has an MBON connection in that source. All published counts on the two retained edge classes are included. No additional synapse count cutoff is applied.

The resulting artifact contains 314 input cells, 3,811 Kenyon cells and 97 output cells. There are 22,581 input connections and 57,823 output connections, representing 822,634 source synapse counts. The JSON circuit occupies 1,011,594 bytes. This size results from selecting a small circuit and does not represent lossless compression of an intact brain. Recurrent paths, other cells, neuromodulation and body dynamics are outside the model.

3.2 Response and action

Let Cpk denote the published count from input cell p to Kenyon cell k. The input response is normalized by each Kenyon cell's total retained input. Population thresholding supplies a fixed nonlinear representation. The threshold is the population mean plus 0.75 times its population standard deviation, and the remaining positive responses are normalized to unit Euclidean length. An all zero response remains zero.

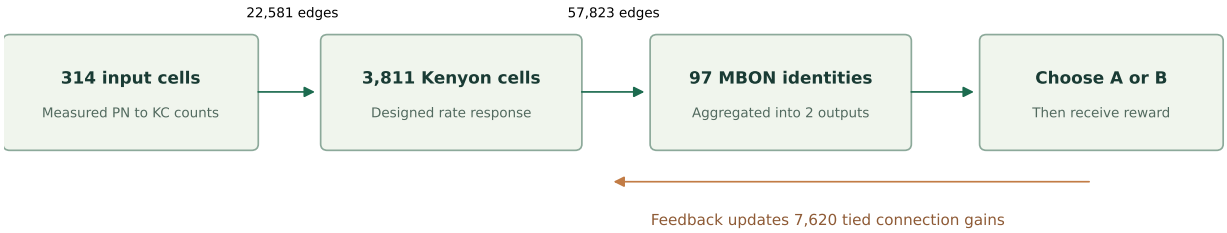


Figure 1: The computation evaluated in this paper. Edge identities and counts come from the data release. Rate responses, action groups and the reward update are engineering choices. Gains are shared across a cell's existing connections to an action group; individual biological synapses are not independently learned.

$$z_k = \frac{\sum_p C_{pk} x_p}{\sum_p C_{pk}}, \qquad h_k = \frac{\max(0, z_k - \mu_z - 0.75\sigma_z)}{\|\max(0, z - \mu_z - 0.75\sigma_z)\|_2}.$$

Output cells are sorted by source identity and assigned alternately to artificial action groups A and B. These groups do not denote natural approach and avoidance pathways. For each Kenyon cell, Wka is the sum of its retained counts to group a. The fixed coefficient mka is the square root of that group's fraction of the cell's total retained output. A nonnegative gain gka scales the contribution. This is equivalent to scaling the existing edges to the group together, with an engineered normalization. It does not preserve their original physiological effect. The implementation aggregates the output counts into two coefficients per Kenyon cell; it does not simulate 97 independent output cell states.

$$m_{ka} = \sqrt{\frac{W_{ka}}{W_{kA} + W_{kB}}}, \qquad \ell_a = \sum_k g_{ka} m_{ka} h_k, \qquad P(B) = \frac{1}{1 + \exp(\ell_A - \ell_B)}.$$

There are 7,620 active tied gains, initialized to one. Two of the possible 7,622 cell and group pairs have no retained connection. When both coefficients are nonzero, this output has the capacity of a constrained linear readout on the fixed Kenyon representation. It should not be described as learning all of the brain's synapses. Transmitter identities, membrane potentials and spike timing are not modeled.

3.3 Learning and state

The policy samples an action before it receives feedback. The environment supplies reward one with probability 0.9 for a correct action and 0.1 for an incorrect action. The update is a standard REINFORCE style policy gradient [3], with learning rate 0.35 and a scalar reward baseline initialized to 0.5. The baseline moves 5% toward the observed reward after each update. Gains are projected into the interval from zero to eight. The method is conventional and is not proposed as a new biological learning rule.

$$g_{ka} \leftarrow \Pi_{[0,8]} \left(g_{ka} + 0.35(r - b)(\mathbf{1}[a = a_t] - P(a)) m_{ka} h_k \right).$$

A complete checkpoint records the circuit hash, model version, task, configuration, gains, reward baseline, progress and independent random streams for observations, actions, task reward and

reward ablation. Evaluation reconstructs its own random stream and cannot update the training state. A dataset hash is also required for handwriting checkpoints. Imports reject incompatible versions, hashes and malformed numeric states.

4 Evaluation design

4.1 Synthetic task study

The initial study fixes three tasks, six conditions and seeds 101 through 120 before those outcomes are inspected. Separate implementation pilots use seeds 9001 and 9002. The cue association task assigns balanced binary choices to 16 independently generated input patterns. Each pattern activates each input cell with probability 0.12. A fresh observation flips each bit with probability 0.02. The reversal task uses the same construction but reverses all rewarded choices after 1,600 episodes. Training continues for another 1,600 episodes. The XOR task combines two binary sensory factors and rewards whether they differ.

Association and XOR each receive 1,600 training episodes. Evaluation uses 256 separate seeded observations every 200 episodes. The final checkpoint is the reported endpoint. Evaluation noise is separate from training, but cue identities and factor patterns are shared. These tasks therefore test learning of a fixed cue family, not arbitrary generalization to new odor identities. The final model is also evaluated after randomly dropping 30% of sensory inputs.

The anatomical condition learns the tied gains. A frozen condition retains their initial values. A rewired condition performs source swaps on the input edges, preserving both endpoint degrees and each Kenyon cell's incoming count multiset, while rejecting duplicate edges. It makes twelve attempted swaps per edge; the accepted counts are recorded. This process does not guarantee independent uniform samples from the set of all degree constrained graphs. A second wiring control samples new input partners while preserving each Kenyon cell's input degree and weights. Both controls retain the output connectivity.

A reward control supplies independent fair binary teaching rewards to the anatomical learner while still measuring performance on the true task. A linear policy acts directly on the same normalized sensory inputs with the same reward update and episode budget. It has 628 gains rather than 7,620. It is a simple resource reference, not a parameter matched modern neural baseline. The nonlinear random wiring conditions provide comparisons with the same gain count as the anatomical condition.

4.2 Handwriting followup

After the synthetic outcomes were inspected, a separate protocol fixed three binary handwriting tasks and seeds 201 through 210. This followup uses the official UCI Optical Recognition of Handwritten Digits files [8]. The release separates 30 training writers from 13 test writers. The training file contains 3,823 images and the test file contains 1,797 images. No repartitioning is performed. The tasks distinguish 0 from 1, 3 from 8, and 4 from 9; they are not a ten class recognition benchmark.

Each image contains 64 block counts from zero to sixteen. Dividing by sixteen gives its input values. A seeded permutation assigns repeated pixel indices to the 314 input cells. This is an arbitrary interface, not a biological visual pathway. Each episode samples a class equally and then an image from that class in the training file. The model and learning settings are unchanged, apart from removing synthetic bit flips. Each condition receives 1,600 episodes. Every evaluation uses all test

images of the selected pair, with equal weight for each class. The final checkpoint is used without selecting the best test score.

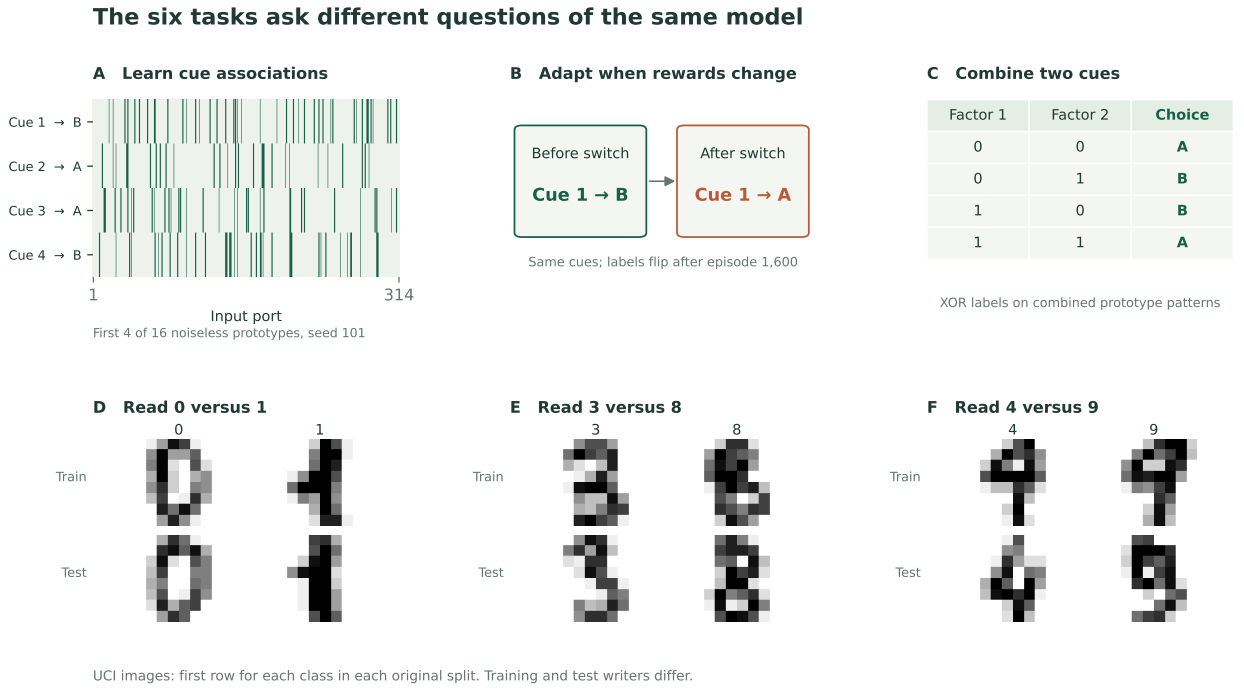


Figure 2: Inputs and rewarded choices in the six evaluated tasks. Synthetic cue images use noiseless task prototypes from seed 101 and do not depict neuron activity. The reversal example changes the rewarded choice for the same cue. For each digit, the handwriting panels show the first training row and first test row in the official file order. No example is selected by model performance. The panels illustrate the task definitions and observations; they are not additional performance measurements.

4.3 Metrics and uncertainty

The primary score is the expected probability of a correct action under the learned stochastic policy. It averages the probability assigned to the correct choice, rather than sampling another action at evaluation time. Greedy accuracy selects the more probable action and is reported separately. A confident but occasional error therefore affects these metrics differently. Synthetic results use 20 seeds per condition and handwriting results use 10. Tables report means and sample standard deviations. Paired differences use 10,000 bootstrap resamples of seeds. The intervals are descriptive and are not corrected for multiple comparisons. No claim of statistical superiority is made from an uncorrected interval alone.

The reported uncertainty describes variation across random seeds on these fixed data and task definitions. It does not include uncertainty from sampling a different population of writers, choosing another connectome, or selecting a different task suite.

5 Results

5.1 Synthetic learning does not require the original input partners

Condition	Association	Reversal	XOR
Measured wiring	90.6 +/- 1.6	47.1 +/- 5.4	88.2 +/- 5.5
Learning frozen	49.4 +/- 4.3	50.6 +/- 4.3	50.3 +/- 1.9
Same degrees, rewired	91.8 +/- 1.0	46.6 +/- 3.7	91.0 +/- 1.6
Random partners	92.3 +/- 0.8	46.5 +/- 3.5	90.8 +/- 3.3
Random rewards	49.9 +/- 5.2	49.3 +/- 6.0	50.5 +/- 4.8
Linear sensory policy	92.5 +/- 0.8	46.8 +/- 3.5	69.6 +/- 3.6

Table 1: Synthetic task results. Expected correct choice probability in percent, mean +/- sample SD across 20 seeds. Reversal is measured after 1,600 additional episodes following the change in reward.

On cue associations, the anatomical condition rises from 49.4% to 90.6% expected correct choice probability. Its mean greedy accuracy is 99.3%. Frozen gains remain at 49.4%, and independent teaching rewards yield 49.9%. These controls support a contribution from task aligned updates. They do not establish a contribution from the particular anatomical input partners.

The paired anatomical minus rewired difference is -1.22 percentage points on association, with descriptive 95% interval [-1.79, -0.71]. On XOR it is -2.76 points, with interval [-5.22, -0.87]. Both comparisons favor the rewired mean. The anatomical model performs better than the simple linear input policy on XOR, but random nonlinear input wiring does as well. That contrast supports the usefulness of the representation and learning mechanism on this task. It does not establish that the original projection neuron to Kenyon cell partners are necessary under this decoder. The output counts remain fixed, so the comparison does not test the relevance of the whole anatomical circuit.

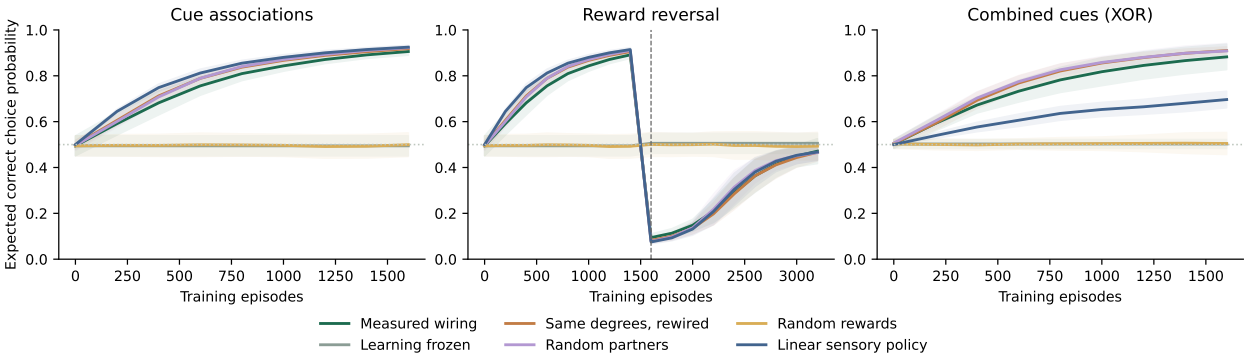


Figure 3: Learning curves for all synthetic conditions, averaged over 20 seeds. Shading is one sample standard deviation. The vertical line marks reward reversal; the score at that point evaluates the newly reversed rule. The final checkpoint is used for all comparisons.

The reversal result is a failure within the declared budget. The anatomical model reaches only 47.1% after the additional 1,600 episodes. The rewired and linear conditions also fail to recover above chance on average. This does not show that relearning is impossible with a different budget or method. It shows that the initial successful learning curve is insufficient evidence for rapid adaptation. The failure is retained in the demonstration and in the reported results.

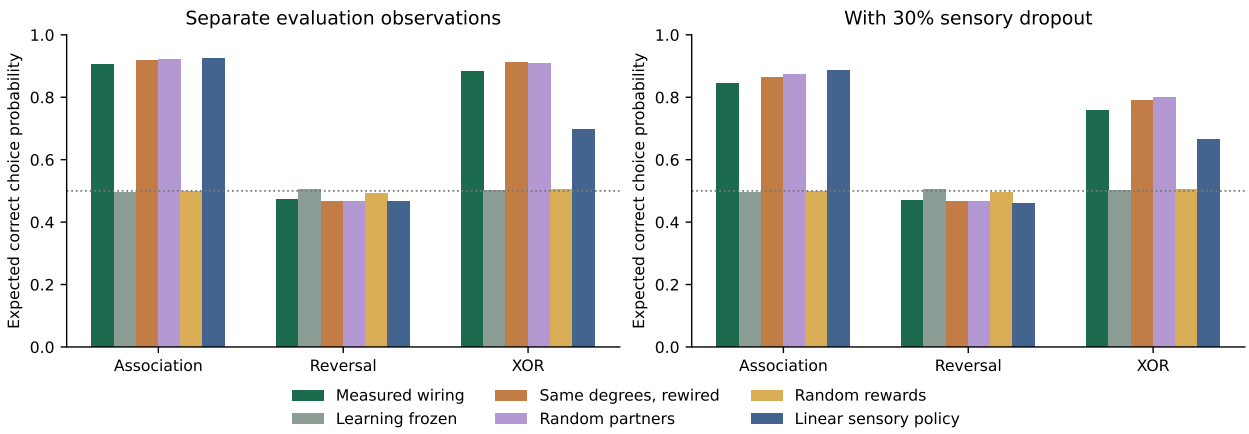


Figure 4: Final expected correct choice probability on the synthetic tasks, without additional dropout and with 30% sensory dropout. These are separate evaluations of the final trained state. The panels show that success depends on task and sensory conditions.

5.2 The same interface supports learning from real handwriting

Condition	0 versus 1	3 versus 8	4 versus 9
Measured wiring	96.7 +/- 0.4	88.2 +/- 1.8	94.4 +/- 0.4
Learning frozen	47.4 +/- 5.7	50.2 +/- 3.1	47.5 +/- 5.4
Same degrees, rewired	96.7 +/- 0.4	88.5 +/- 1.1	94.5 +/- 0.4
Random partners	96.9 +/- 0.3	88.7 +/- 0.7	94.7 +/- 0.5
Random rewards	49.5 +/- 7.9	50.9 +/- 3.1	50.5 +/- 8.1
Linear sensory policy	95.8 +/- 0.2	86.2 +/- 0.6	93.1 +/- 0.8

Table 2: Handwriting followup. Balanced expected correct choice probability in percent, mean +/- sample SD across 10 seeds. All evaluations use the official separate writer test partition.

The handwriting followup tests the same policy interface on external observations rather than generated cue dictionaries. The official split is preserved, so the evaluation images were not used for updating the model. In particular, the input encoding and gain update do not receive a test label during training. A software isolation check changes the test images and verifies that the learned training state remains identical.

The anatomical condition achieves mean greedy accuracy of 99.8% on 0 versus 1, 95.6% on 3 versus 8, 98.6% on 4 versus 9. These numbers concern three small binary problems. They do not demonstrate competitive optical character recognition, language understanding, or a naturally visual role for the selected input cells. The wiring controls and direct input baseline remain necessary for interpreting them.

5.3 Paired controls separate learning from input partner effects

The paired differences answer two distinct questions across the task suite. Comparing trained anatomical gains with frozen gains measures the effect of enabling the update under the declared training conditions. Comparing the trained anatomical model with the trained rewired model measures the effect of its projection neuron to Kenyon cell partners under a control that preserves

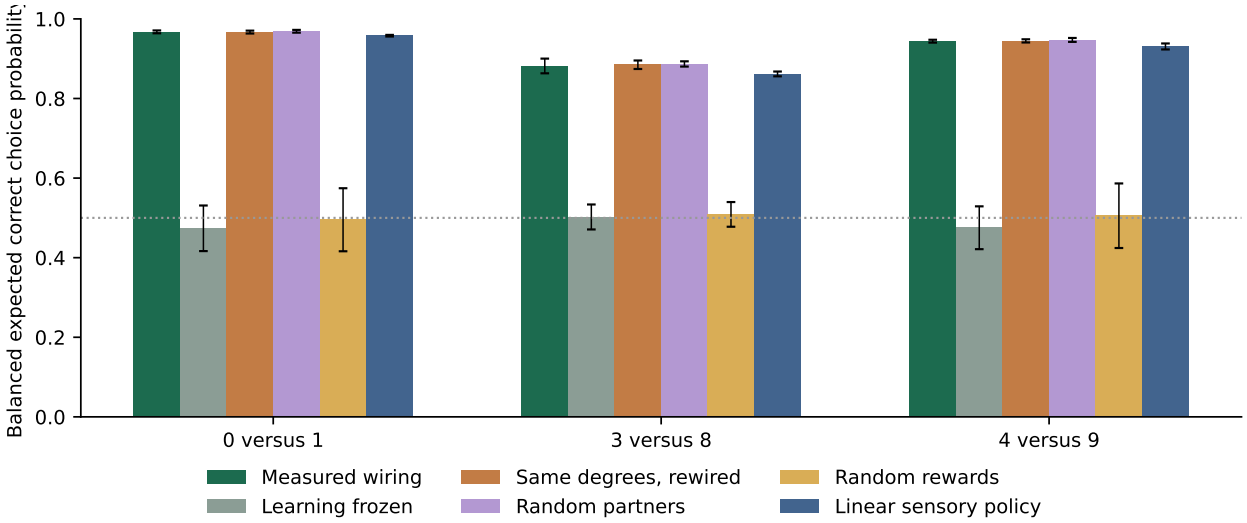


Figure 5: Handwriting performance on the published test partition. Bars show balanced expected correct choice probability and error bars show one sample standard deviation across 10 seeds. All six conditions use the same examples and episode budget.

the endpoint degrees and incoming count multiset. The trained anatomical mean exceeds the frozen mean on all five tasks without reward reversal, but falls below it after reversal. Differences from rewired learning are much smaller and do not show a consistent advantage for the original input partners under this decoder. These outcomes provide a reference for later experiments that change the encoding, update rule or adaptation budget.

5.4 Portable execution and workflow checks

The software checks run in Chromium 147.0.7727.15 and WebKit 26.4 on an Apple M1 Max host. They cover three separate seeds and the anatomical, rewired and frozen conditions, for 18 executions. Each uses an 800 episode reversal run, saves at episode 273 and compares continuation with uninterrupted execution. All 18 produce identical serialized states within the same engine. Across the nine engine pairs, the largest gain difference is 6.66e-16; random states and cumulative rewards agree exactly. This is numerical agreement on the tested cases, not a promise of bitwise equality across all engines or platforms.

Engine	Mean local data load	Mean checked execution
Chromium	6.0 ms	553.2 ms
Webkit	8.8 ms	357.2 ms

Table 3: Desktop headless browser timings over nine cases per engine. Execution includes the reference run, restored continuation and evaluations. Data is served over localhost. These measurements are not internet download latency or mobile performance.

The interactive check starts the actual browser worker, pauses a run, downloads a partial checkpoint, resets the page state, imports the checkpoint and completes training. It also runs a frozen comparison and rejects an unsupported imported file. The page produces no uncaught errors in this workflow. A 390 pixel viewport has no document overflow. Separate invariant checks cover reward isolation,

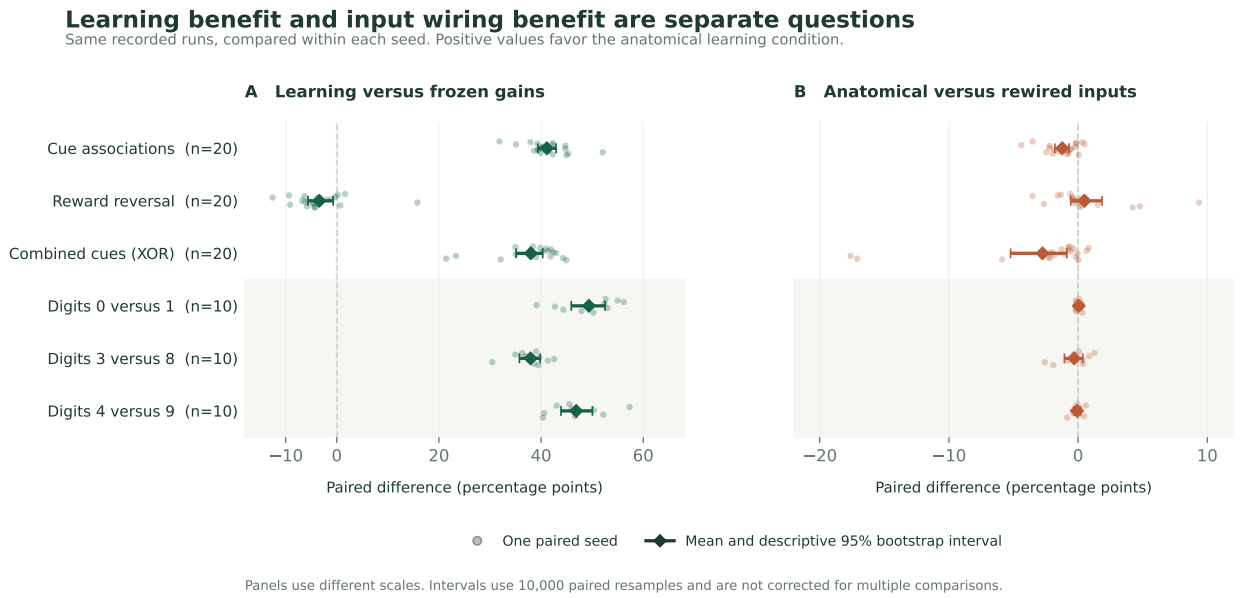


Figure 6: Paired differences in final expected correct choice probability, in percentage points. The two panels use different horizontal scales. Each small point is the anatomical result minus its control for one matched seed. Larger points and intervals show the paired mean and the existing descriptive 95% bootstrap interval. Synthetic tasks use 20 seed pairs and handwriting tasks use 10. The intervals use 10,000 seed resamples and are not corrected for multiple comparisons. Anatomical minus frozen concerns learning; anatomical minus rewired concerns the projection neuron to Kenyon cell input partners under the same output aggregation. A positive learning effect does not establish an advantage for those partners.

frozen parameters, degree preservation, finite values and malformed checkpoints. These checks establish implemented behavior, not the usefulness of the interface to an independent user.

Two additional browser executions check a handwriting run and checkpoint continuation in both engines. A separate dense NumPy calculation verifies the sparse JavaScript projection, normalization, action probability and one reward update on ten inputs. The largest absolute discrepancy is below 5.0 times ten to the power of negative fifteen. This reference check supports agreement between the stated equations and the implementation; it is not an additional learning result.

6 Use in subsequent studies and limitations

A researcher can use the released Node runner as a baseline and inspect the same model in the browser. For example, a proposed update rule can be tested against the supplied learning and wiring controls with the same observations and episode budget. The existing records include the cases where learning helps, the cases where rewiring performs as well or better, and the failure after reward reversal. Reusing those conditions makes the scope of a new result explicit. The baseline is specific to this circuit and task suite; it is not a general ranking of learning methods.

The saved state procedure supports a different form of reuse. For a supplied experiment, a colleague can receive the settings, learned gains and random states together, then continue at the recorded episode. The continuation checks establish this behavior on the tested browser versions. A later

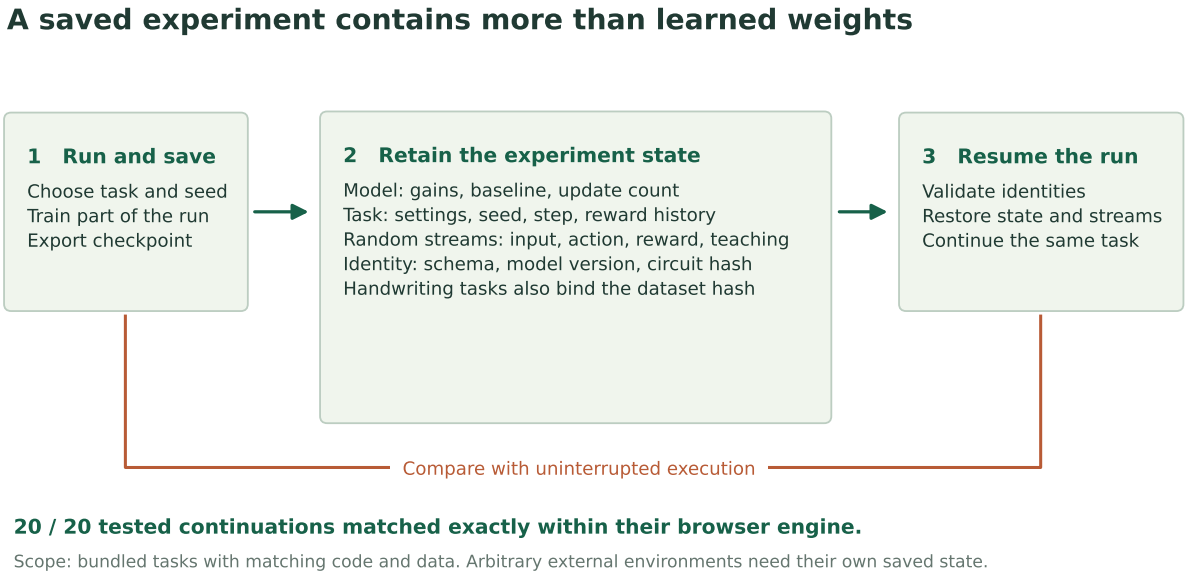


Figure 7: The implemented save and continuation workflow. The checkpoint carries the model and data identifiers, configuration, learned state, progress and random streams required by the supplied experiments. Import validates that state before continuation. The verification count summarizes the 20 browser runs reported in the text, all of which reproduce uninterrupted continuation within the same engine. The diagram describes the software contract and tested paths; it does not add a new experimental result or guarantee equivalence on untested platforms.

implementation can reuse the format and verification procedure, while reporting whether its own model, environment and browser satisfy the same conditions. This is a tested use of established checkpoint practice. Exporting learned weights alone does not preserve an experiment's future observations or rewards.

The public model interface also allows a researcher to supply a new environment with 314 input values, two possible actions and a scalar reward. The executable example calls prediction before the environment reveals the reward, then updates the gains from that feedback. It demonstrates the interface rather than a further held out learning result. Custom environments must save their own state and random streams if continuation is required. They also need a separate evaluation protocol; changing the reward does not guarantee that the circuit can solve the task.

A subsequent study can use the released task results as comparison data or adapt the implementation to a new question. It should identify the circuit, model version and components it retains, and report changes to the encoding, reward or training budget. This makes the reused baseline distinguishable from the new experiment. Work using the original connectome or handwriting observations should also cite the relevant data source [1] or [8].

Several limitations constrain the publication claim. The study concerns a fixed anatomical source, a selected feedforward circuit and engineered sensory encodings. The output gains are tied and the decoder is structurally close to a constrained linear head. Both cell dynamics and the learning rule omit most biological detail. The task suite is small. The baseline set does not include optimized multilayer image classifiers, and the paper does not claim to outperform them. Rewiring samples are produced by a finite swap procedure rather than a proven uniformly mixed graph sampler.

The measurements establish that the artifact runs and that its controls distinguish several explanations of success. They do not yet establish that researchers save time, that students learn more, or that the browser implementation is preferable to existing platforms. Those benefits require independent use and comparative evaluation. Experiments combining several circuits, recurrent memory, language prediction, physical devices and animal data are outside the evidence presented here.

7 Availability and reproducibility

The companion site provides the interactive lab, an accessible manuscript and a downloadable reproduction bundle. The bundle contains the circuit and handwriting artifacts, extraction scripts, experiment protocols, complete run records, analysis code, software checks and LaTeX source. Node executes the primary experiments without third party runtime dependencies. Python with NumPy and Matplotlib generates the analyses and figures; rebuilding the circuit from the original files also requires PyArrow. The original source file hashes and the frozen implementation hashes are recorded with the results.

Interactive artifact and reproduction bundle: https://gfly.site/research/. Research paper and supporting materials: https://hios.app/research/gfly/. The software artifact is distributed under the MIT license. MaleCNS data and the UCI handwriting data retain their respective CC BY 4.0 terms and attribution. The author is Jiahe Tian, HiOS Labs Inc., with correspondence at ipo@stanford.edu. AI coding and writing tools assisted implementation and preparation of the manuscript. The reported measurements come from the released execution records; responsibility for their interpretation remains with the author.

References

[1] MaleCNS Collaboration. MaleCNS v1.0 connectome data release. Janelia Research Campus and Google Research, 2026. CC BY 4.0. https://male-cns.janelia.org/

[2] Yoshinori Aso et al. The neuronal architecture of the mushroom body provides a logic for associative learning. eLife 3:e04577, 2014. https://doi.org/10.7554/eLife.04577

[3] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8:229-256, 1992. https://doi.org/10.1007/BF00992696

[4] Katherine Xie and Gabriel Koch Ocker. The Impact of Structural Changes on Learning Capacity in the Fly Olfactory Neural Circuit. arXiv:2509.19351, 2025. https://arxiv.org/abs/2509.19351

[5] Jacob Morra, Andrew Flynn, Andreas Amann, and Mark Daley. Multifunctionality in a Connectome-Based Reservoir Computer. arXiv:2306.01885, 2023. https://arxiv.org/abs/2306.01885

[6] Connectome Lab contributors. Connectome Lab training interface and implementation. Software documentation, accessed 29 October 2026. https://github.com/berkeyc/connectome-lab

[7] Bingni W. Brunton, Elliott T. T. Abe, Lawrence Jianqiao Hu, and John C. Tuthill. The digital sphinx: Can a worm brain control a fly body? eLife reviewed preprint 111516v1, 2026. https://doi.org/10.7554/eLife.111516.1

[8] E. Alpaydin and C. Kaynak. Optical Recognition of Handwritten Digits. UCI Machine Learning Repository, 1998. https://doi.org/10.24432/C50P49

[9] Shiu et al. A Drosophila computational brain model reveals sensorimotor processing. Nature, 2024. https://doi.org/10.1038/s41586-024-07763-9