IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151166

The Connectome Lab: Connectome-Constrained Neural Chess

T Milan Prabhu[1], Gokul J[2], Sudarshan J[3], Dr. Beenarani Manoj[4]

Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, India[1-3]

Mentor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, India[4]

Abstract: We present The Connectome Lab, an interactive system that connects browser chess positions to connectome-constrained neural models through an authenticated model API. The interface combines a fixed, untrained leaky integrate-and-fire model with MFly-1.0, a larger chess-trained recurrent rate model, and displays action selection, model identity, and recorded activity. The deployed trained path uses a pretrained checkpoint; a separate project training workflow creates and exports project checkpoints. A 2,000-neuron CPU experiment completed imitation learning and short self-play, reaching checkpoint step 68. Its final logged imitation check at step 60 produced 1.56% top-one and 3.12% top-three move agreement on 64 held-out positions, with value mean squared error of 1.0007. The exported checkpoint recorded four draws against a random opponent and two draws and two losses against a material opponent in eight bounded games. Saved API observations and six browser fixtures verify connected decisions and action handling. The work demonstrates an integrated experimental application and a training pipeline. The brief experiment and unmatched model configurations do not establish competitive chess strength or an advantage from biological wiring.

Keywords: connectome-constrained networks, neural chess, leaky integrate-and-fire, recurrent neural networks, imitation learning, self-play, model API, recorded activity.

1. INTRODUCTION

Chess provides a structured setting for examining neural decisions. A position has an explicit state, legal actions can be checked by a rules engine, and a predicted move can be compared with a recorded target or tested in a game. This makes it possible to study input encoding, neural computation, action selection, and application behaviour within the same task. A useful demonstration should expose these stages rather than present a final move without its computational context.

Connectomics introduces an anatomical constraint on network structure. The FlyWire reconstruction describes the neuronal wiring of an adult fruit-fly brain [1], while associated annotations identify neuronal classes and cell types [2]. These resources can define the connections and populations of a computational network. They do not, by themselves, specify a chess-playing system. Board inputs, neural dynamics, output projections, learning objectives, and inference rules remain explicit modelling decisions.

We developed The Connectome Lab to connect these modelling choices to an inspectable browser application. The system presents a common chess workspace for two inference paths, identifies the graph and checkpoint used, and replays a sample of activity recorded during a decision. We also completed a separate training and checkpoint experiment. The present study reports the implemented system and the evidence retained for that small experiment, with separate interpretation of browser fixtures, live inference observations, and learning measurements.

Project Objectives

Our objectives are to provide a playable chess interface, validate the browser-to-model request path, preserve the raw output of an untrained comparison, expose the action-selection procedure of a trained model, and display labelled recorded activity. The training objective is to verify data preparation, optimisation, interruption and resume, checkpoint export, and bounded evaluation. The study focuses on a reproducible application and experimental workflow; a controlled comparison of anatomical and alternative wiring remains future work.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151166

2. RELATED WORKS

Dorkenwald et al. describe a dense adult *Drosophila* wiring reconstruction [1]. Schlegel et al. complement this resource with whole-brain annotations and cell typing [2]. Together, these studies support the use of an observed directed graph and biologically informed populations in computational models. Counts from a retained or filtered model graph must be distinguished from those of the complete anatomical reconstruction.

Shiu et al. use a connectivity-based leaky integrate-and-fire model to investigate sensorimotor processing in *Drosophila* [3]. This provides relevant modelling context for explicitly specifying state dynamics and biological assumptions. Our fixed LIF comparison is a separate simplified implementation. Its chess input projection and action decoder are artificial interfaces, and the displayed simulation is not a measurement from a living brain.

Neural chess systems can combine a policy over actions, a value estimate for a position, and a search procedure. AlphaZero demonstrates policy and value learning through self-play with tree search [4]. These components must be distinguished when interpreting an output: a selected move can reflect rule filtering or search as well as the root policy. The deployed chooser in our application uses a much narrower one-ply value check, while the small project training configuration includes limited Monte Carlo tree search during self-play.

MFly-1.0 (labelled 'Trainable' in the interface) is the project name used here for the rate-network architecture, 20-plane board encoder, training code, and 134,209-neuron checkpoint. The implementation is sourced from [5] under the MIT licence, and the pretrained weights from [6] under CC BY-NC 4.0. Our contributions are the laboratory interface, service integration, fixed comparison, response validation, recorded-activity presentation, and separate project training workflow. The project-trained 2,000-neuron checkpoint has a separate identity.

3. SYSTEM ARCHITECTURE

The application separates chess interaction from neural execution. A React and TypeScript frontend maintains the board, game history, selected phase, and request state. The browser uses chess.js for move validation [7] and Three.js for visual scenes [8]. An authenticated HTTPS model API connects the interface to a Python FastAPI service [9]. Figure 1 shows the request path and the two deployed providers.

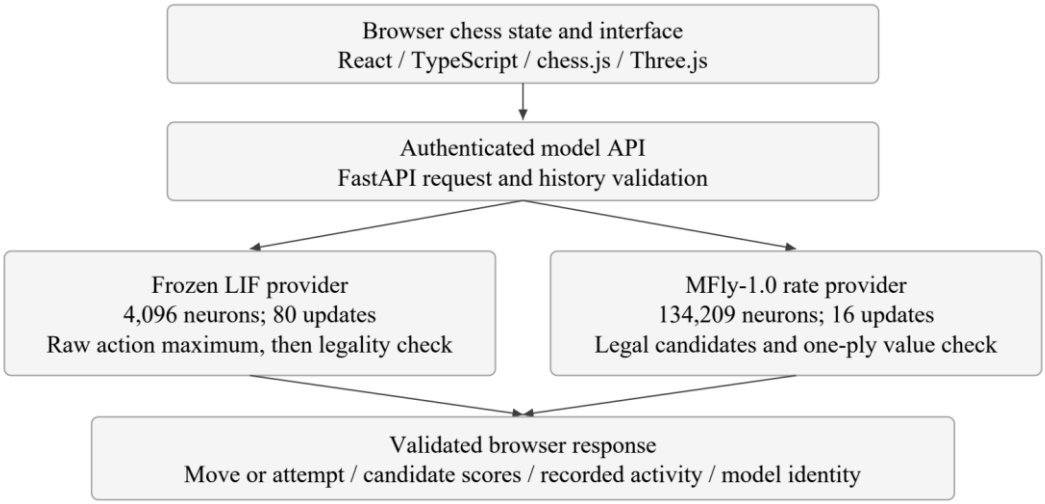


Fig. 1. Application architecture and the two deployed inference paths. Training is managed separately from public inference requests.

System Workflow

The browser submits the current Forsyth-Edwards Notation (FEN), initial FEN, move history, and selected phase. The service reconstructs the game and checks that the history agrees with the submitted position. It selects a provider, encodes the board, and runs inference with fixed weights. The response includes a move or attempted action, candidate scores, model and graph identity, timing, and sampled recurrent activity. The browser applies a move only after checking the response against the active request and its own chess rules.

DOI: 10.17148/IJARCCE.2026.151166

Frozen uses a fixed LIF network and ranks all 20,480 output slots before checking legality. A legal maximum is applied; an illegal maximum is shown as an attempt while the board remains unchanged. MFly-1.0 uses the prepared rate network, a legal policy shortlist, and a one-ply value comparison. Neither provider updates its parameters during a website request. Training is started from the separate experiment notebook.

Property	Frozen	MFly-1.0
Neural dynamics	Fixed LIF comparison	Sparse recurrent rate network
Neurons / directed edges	4,096 / 138,615	134,209 / 2,700,513
Board representation	19 planes; 1,216 values	20 planes; 1,280 values
Updates per position	80 simulation steps	16 recurrent steps
Action output	20,480 raw slots	4,168 move logits
Selection rule	Raw maximum, then legality check	Legal shortlist and one-ply value check
Weight identity	Fixed and untrained	Pretrained deployed checkpoint

Table 1. Deployed model configurations. Sources: provider configuration and model specifications [5, 6].

Implementation Tools and Key Features

The interface provides model selection, move history, a connected-state indicator, candidate readouts, and activity replay. The service validates requests and returns structured responses. Source revisions and graph/checkpoint hashes identify the loaded computation. Resetting the board, changing phase, or disconnecting invalidates stale responses. A disconnected local demonstration is explicitly labelled. The Neural Brain page supplies a separate procedural visual exploration, whose geometry is distinguished from recorded inference activity.

4. METHODOLOGY

Dataset Description and Preparation

The study uses an anatomical graph and chess position data. The graph is derived from FlyWire resources under their stated attribution and usage terms [10]. Lichess game records provide positions and recorded moves for the project imitation experiment [11]. In the PGN preparation path, value targets represent the game outcome from the side-to-move perspective. Positions retain their board representation, move index, value target, and supporting game information.

For deployment, graph and checkpoint files are checked against pinned sizes and SHA-256 digests before loading. The project training experiment uses a smaller graph with 2,000 neurons and 43,170 directed edges. We execute the learning workflow with PyTorch [12] and retain the configuration, graph fingerprint, metrics, exported checkpoint, and summary in the project archive [13]. The data-build record reports 12,534 training positions from 180 games and 407 validation positions from five separately held-out games.

Board Preprocessing and Neural Dynamics

The service reconstructs the board from the initial position and UCI move history. A current FEN alone cannot encode repetition history, so consistency is checked before neural execution. Frozen uses an absolute-coordinate 19-plane representation containing pieces, side to move, castling rights, en passant, and the normalised halfmove clock. MFly-1.0 uses the original mover-relative 20-plane encoding, including rule and history features [5].

Frozen projects 1,216 board values into the fixed LIF graph. Membrane voltage leaks, threshold crossings produce spikes, and recurrent synaptic state propagates activity. Mean spike activity reaches the fixed action decoder. MFly-1.0 maps 1,280 features through sensory and retina pathways into a sparse recurrent rate network. Its core update can be expressed as:

$$h_{t+1} = (1 - a)h_t + a\,f(z_t) \quad (1)$$

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151166

Here h_t is neuronal state, a is a per-neuron leak parameter, f is the activation function, and z_t combines recurrent drive, projected input, bias, and neuromodulatory effects. Policy and value heads read selected output activity [5]. These numerical rate activations are dimensionless. The deployed model uses different dynamics and settings from the small training experiment, as described below.

Training Configuration and Learning Objectives

We start the project experiment from fresh initialisation on the 2,000-neuron graph. The saved configuration uses four recurrent steps, ReLU activation, value-head hidden size 32, and seed 2026. Imitation uses AdamW, batch size 32, learning rate 0.001, weight decay 0.0001 on the configured dense parameter group, and gradient clipping at 1.0 [5, 13]. For imitation training and validation, policy loss is negative log-likelihood over all 4,168 move slots without legal masking. MFly-1.0 inference separately masks illegal moves. Value loss is mean squared error against the supplied target, with equal weighting:

$$L = L_{policy} + L_{value} \quad (2)$$

The run completes 60 imitation steps, then two short self-play iterations. The self-play configuration uses four generated games per iteration, eight MCTS simulations, batch size 16, four training updates per iteration, and learning rate 0.0001. The selected best.pt weights at step 68 are exported as m-smoke-selfplay-step68.pt, preserving the stage and step in the filename [13].

Item	Recorded configuration
Execution environment	CPU; Windows 11; Python 3.12.14; PyTorch 2.14.0+cpu
Graph / recurrent steps	2,000 neurons; 43,170 edges; 4 steps
Imitation data	12,534 training positions; 407 validation positions
Validation check size	At most 64 positions per logged check
Imitation / self-play updates	60 imitation steps; exported checkpoint at step 68
Seed / batch sizes	2026; imitation 32; self-play 16
Opponent evaluation	Four random and four material-opponent games; maximum 40 plies

Table 2. Recorded project experiment settings and data. Source: the saved m-smoke archive [13].

Evaluation Protocol and Activity Recording

Held-out move agreement is measured as the proportion of evaluated targets ranked first or within the three highest-scoring moves. The denominator for the last logged imitation check is 64, rather than the complete 407-position validation shard. Value MSE measures squared prediction error. We distinguish the step-60 held-out check from the later step-68 exported weights. Opponent evaluation uses the exported checkpoint and records wins, draws, and losses under the 40-ply limit [13].

System evaluation uses separate evidence. Six browser fixtures cover illegal, same-square, legal, legacy-service, MFly-1.0, and inconsistent-response scenarios [14]. A saved live API session contains two MFly-1.0 observations [15]. Model-side duration and browser request time are kept separate. Connected decisions record sampled neural values; the interface replays up to 96 selected neurons with schematic positions. A sample is a display aid, rather than a representative population estimate or a causal explanation of a move.

5. RESULTS AND DISCUSSION

Data Preparation and Training Results

The retained archive contains the prepared shards, graph, step-68 checkpoint, configuration, and metric files. The interruption-and-resume workflow completed a real optimisation and export cycle. Figure 2 is drawn from the logged imitation and validation measurements. The training and validation losses decrease over the brief run, but move agreement remains low. Self-play losses are excluded from this comparison because they use different targets and should not be interpreted as the same objective continuing on the same examples.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151166

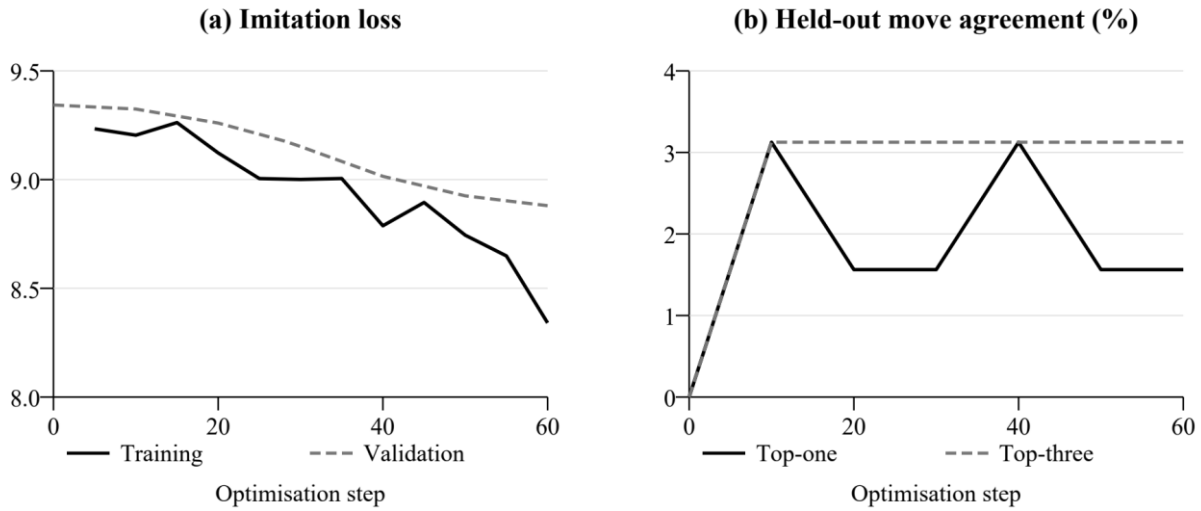


Fig. 2. Imitation training loss and held-out move agreement from the project experiment [13]. Validation points precede the later self-play updates.

Measurement	Recorded result
Total validation loss	8.8800
Policy negative log-likelihood	7.8793
Value MSE	1.0007
Top-one move agreement	1.56% (1 of 64)
Top-three move agreement	3.12% (2 of 64)

Table 3. Final logged imitation validation check: step 60, 64 evaluated positions [13].

The last logged check produces total loss 8.8800, policy loss 7.8793, and value MSE 1.0007. Only one evaluated target move ranks first and two appear in the top three. These measurements support a working learning pipeline, not strong chess play. The small evaluation batch and single seed also limit the precision and generality of the observed percentages. No held-out evaluation of all 407 positions after self-play is asserted.

Bounded Opponent Evaluation

Opponent	Games	Wins	Draws	Losses
Random	4	0	4	0
Material	4	0	2	2

Table 4. Opponent results for the exported step-68 checkpoint; all games use the 40-ply cap [13].

The checkpoint records four draws against the random opponent, and two draws and two losses against the material opponent. A short cap can terminate games before a decisive result, so the draw totals should not be treated as evidence of sustained competitive strength. Eight bounded games are also insufficient for a stable Elo estimate. Larger independent test sets, longer games, and repeated seeds would be required for a stronger performance claim.

Connected Inference and Application Behaviour

The live record dated 27 October 2026 contains MFly-1.0 moves Nf3 and Nf6 from GPU device cuda:1 [15]. Model-side times are 57.60 ms and 56.66 ms; browser request times are 667 ms and 657 ms. Both responses identify 134,209 graph neurons, 96 sampled neurons, and 16 recorded frames. These are two observations from one session, rather than

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151166

a latency benchmark. In the second response, Nc6 has the highest root policy probability while the value-based chooser selects Nf6, confirming that the displayed shortlist and final selection have different roles.

All six retained browser fixture scenarios pass [14]. Figures 3 and 4 show the implemented interface and its visual distinctions. The fixture values are controlled test inputs and are not included in the performance measurements. The archived API session also contains an older filtered Frozen response; it is excluded as evidence of the later unfiltered Frozen behaviour. The current illegal-attempt path is supported by the separate browser verification.

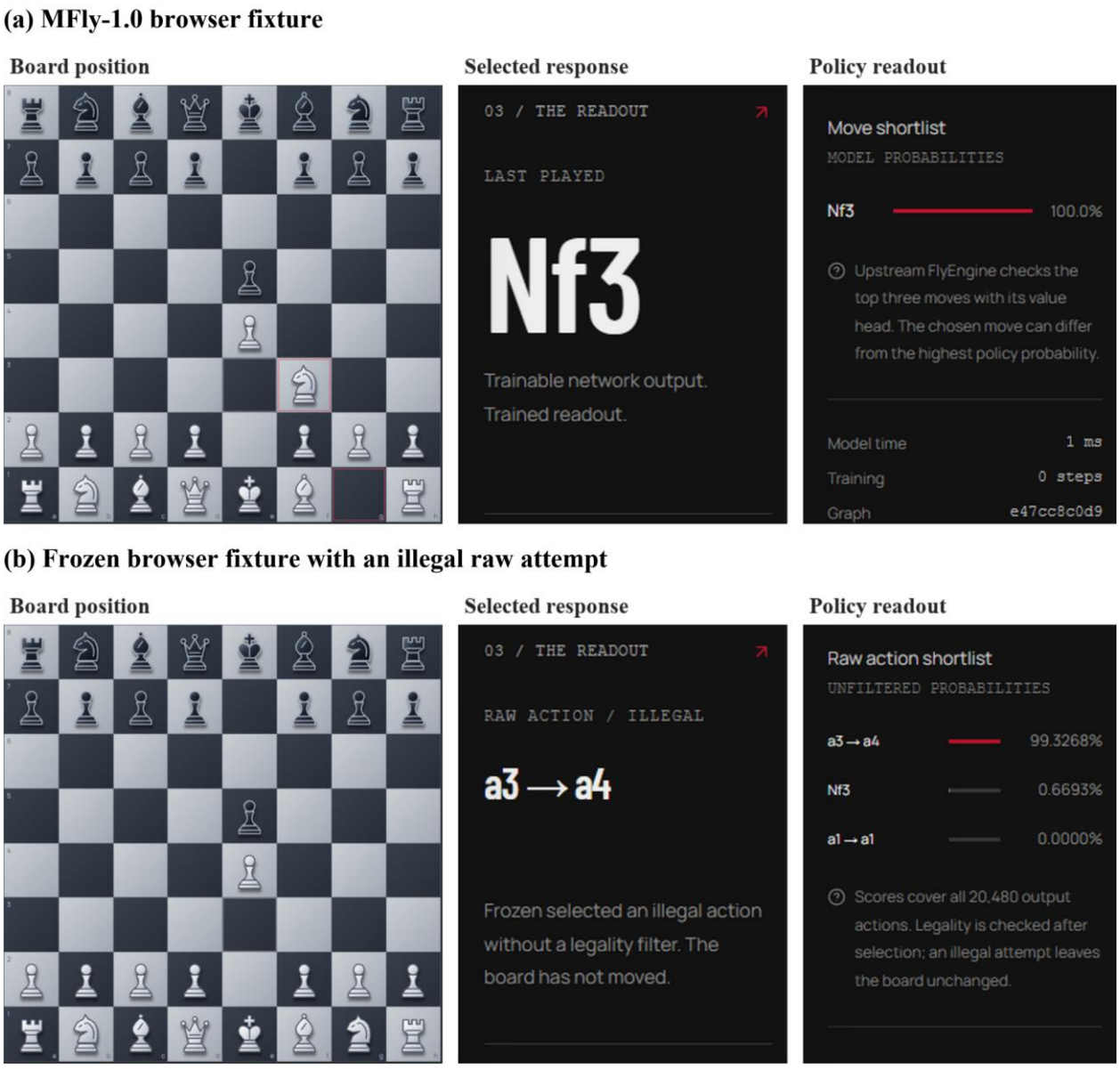


Fig. 3. MFly-1.0 and Frozen browser fixtures [14]. 'Training 0 steps' is test metadata, not the deployed checkpoint's training history. An illegal attempt leaves the board unchanged.

The activity panel distinguishes cumulative simulated spike activity in Frozen from dimensionless rate activity in MFly-1.0. Frames correspond to the input position before the selected move, even when the board already shows the resulting position. Schematic coordinates and slowed replay support inspection. The separate Neural Brain page is a procedural visual exploration and does not represent a reconstruction of the measured graph or its current activation state.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151166

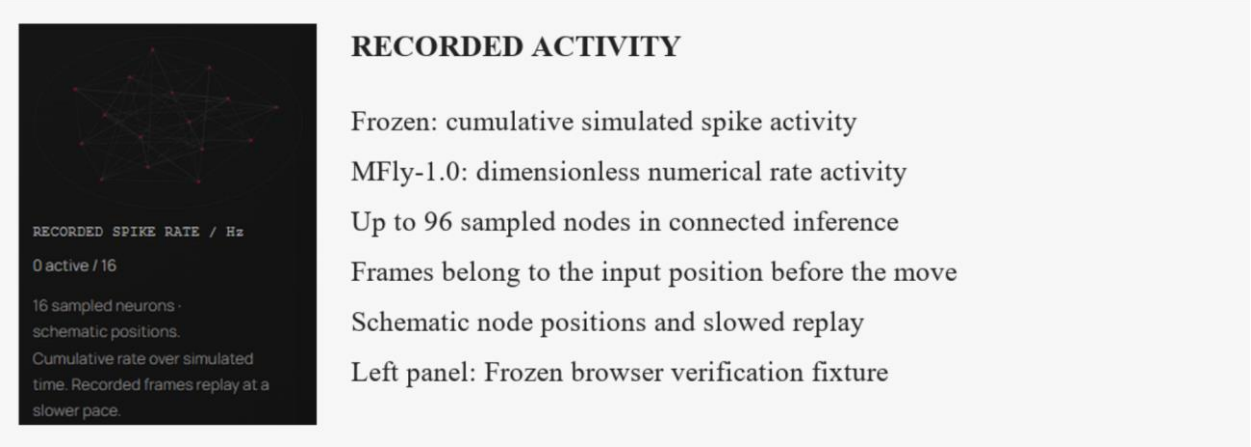


Fig. 4. Browser activity fixture with 16 sampled neurons; live inference may display up to 96. Schematic replay and test inputs are distinct from measured anatomy and live results.

Limitations and Interpretation

The project experiment uses one small graph, one seed, a short optimisation schedule, limited validation batches, and eight capped opponent games. It verifies learning, checkpoint persistence, resume, evaluation, and export, but cannot establish general chess strength. The two deployed providers also differ in neuron count, dynamics, encoding, output representation, and legality assistance. Their behaviour is therefore not a controlled estimate of the effect of training.

Anatomical connectivity is one modelling constraint among several. Demonstrating an advantage from biological wiring would require matched trained and untrained networks, rewired-graph controls, comparable parameter counts and compute budgets, and sufficient independent evaluation. Recorded activity alone does not identify a causal mechanism for a move. The current evidence supports an inspectable software laboratory and a completed bounded training experiment.

6. CONCLUSION

We developed The Connectome Lab as an integrated application connecting browser chess interaction, a model API, neural providers, validated action handling, and recorded-activity replay. Frozen preserves its raw maximum and exposes illegal attempts, while MFly-1.0 combines legal policy candidates with a one-ply value check.

The separate 2,000-neuron CPU run exports a step-68 checkpoint. Its last step-60 imitation check yields 1.56% top-one and 3.12% top-three agreement on 64 held-out positions. The opponent games and API observations verify the reported workflow within their limits. Further work should extend training and independent evaluation, and compare matched and rewired graphs.

REFERENCES

[1]. Dorkenwald, S., et al. (2024). "Neuronal wiring diagram of an adult brain." Nature, 634, 124-138. https://doi.org/10.1038/s41586-024-07558-y

[2]. Schlegel, P., et al. (2024). "Whole-brain annotation and multi-connectome cell typing of Drosophila." Nature, 634, 139-152. https://doi.org/10.1038/s41586-024-07686-5

[3]. Shiu, P. K., et al. (2024). "A Drosophila computational brain model reveals sensorimotor processing." Nature, 634, 210-219. https://doi.org/10.1038/s41586-024-07763-9

[4]. Silver, D., et al. (2017). "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm." arXiv, 1712.01815. https://arxiv.org/abs/1712.01815

[5]. Esposito, C. (2026). "fly-chess." Source repository, MIT licence. Pinned revision: 3c83567dfb6fa71cecc1556c75f17c8bb38fd5cf. https://github.com/cesp99/fly-chess (accessed 5 November 2026).

[6]. Esposito, C. (2026). "fly-chess brains." Model card and fly3-selfplay checkpoint, CC BY-NC 4.0. Pinned revision: cad733198a14d76a35149ecbe5f68bc6efa97732.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151166

https://huggingface.co/cesp99/fly-chess (accessed 5 November 2026).
[7]. chess.js contributors. "chess.js." Official source repository and documentation. https://github.com/jhlywa/chess.js (accessed 5 November 2026).
[8]. Three.js contributors. "three.js documentation." https://threejs.org/docs/ (accessed 5 November 2026).
[9]. FastAPI contributors. "FastAPI documentation." https://fastapi.tiangolo.com/ (accessed 5 November 2026).
[10]. FlyWire Consortium. "Citing guidelines for publishing with FlyWire." https://flywire.ai/guidelines (accessed 5 November 2026).
[11]. Lichess. "lichess.org open database." Chess games and engine evaluations. https://database.lichess.org/ (accessed 5 November 2026).
[12]. PyTorch contributors. "PyTorch documentation." https://docs.pytorch.org/docs/stable/ (accessed 5 November 2026).
[13]. The Connectome Lab (2026). "Project experiment archive m-smoke-20261009." Summary, metrics, training report, graph, and checkpoint manifest, 9 October 2026. Unpublished project records.
[14]. The Connectome Lab (2026). "Frozen unfiltered browser verification." Six fixture scenarios in browser-report.json. Unpublished project verification record.
[15]. The Connectome Lab (2026). "Saved live API verification." Report checked 27 October 2026. Unpublished project verification record.

 This work is licensed under a Creative Commons Attribution 4.0 International License