Reinforcement-trained recurrent networks reproduce human beat-synchronization dynamics: Supplementary Information

Hyperparameter Search Procedure

To maximize training success, we used a guided hyperparameter search strategy inspired by the Tree-structured Parzen Estimator (TPE) approach[1]. For each agent, the following hyperparameters were explored:

- **Learning Rate:** Ranged from 10^{-6} to 10^{-3}, with ReduceLROnPlateau dynamically reducing the rate when performance plateaued.
- **Batch Size:** Tested values of 16, 32, 64, and 128. Selected based on stability and speed of learning.
- **Entropy Coefficient:** Ranged from 0 to 10^{-2} to balance exploration vs. convergence.
- **Episode Horizon:** Varied between 64 to 512 timesteps. 300 timesteps was found optimal across agents.

Hyperparameter tuning was repeated for each reward scheme. Below is a summary of the final values used in the reported experiments.

Agent	Learning Rate	Batch Size	Entropy Coef.	Episode Horizon
Symmetric	3×10^{-5}	64	0.005	300
Symmetric + Interval	1×10^{-4}	32	0.008	300
Asymmetric	5×10^{-5}	64	0.002	300
Asymmetric + Interval	1×10^{-4}	32	0.001	300

Table 1. Final per-agent hyperparameter settings. Values were selected independently for each agent by Optuna/TPE search using an identical search space, budget, and selection criterion. For the fairness control described in the main text, every agent was additionally retrained with the Asymmetric + Interval settings; this did not change the qualitative outcomes.

Training Configuration

Training was implemented using the Gymnasium environment[2] with stable-baselines3 contrib PPO[3]. All experiments were conducted on a MacBook Air 2022 (Apple M2, 16GB RAM), using Python 3.9.13 and PyTorch 2.2.1. Each agent trained for up to 1.4×10^6 simulated seconds (388 hours) or until performance plateaued. Ten seeds were used to ensure reproducibility.

PPO Algorithm Details

PPO minimizes a clipped surrogate objective:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t) \right] \tag{1}$$

where $r_t(\theta)$ is the probability ratio between new and old policy, $\hat{A}_t$ is the estimated advantage, and $\varepsilon = 0.2$. An entropy bonus term $\beta S(\pi_\theta)$ was added to encourage exploration.

The actor network was the LSTM agent, while the critic was a feedforward network estimating value for each state-action pair. Gradient updates were computed using backpropagation through time[4].

References

1. Watanabe, S. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance (2023). 2304.11127.
2. Towers, M. *et al.* Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032* (2024).
3. Raffin, A. *et al.* Stable baselines3. https://github.com/DLR-RM/stable-baselines3 (2019).
4. Werbos, P. J. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks* **1**, 339–356, DOI: https://doi.org/10.1016/0893-6080(88)90007-X (1988).