communications biology

Article

https://doi.org/10.1038/s42003-024-05871-w

A machine learning toolbox for the analysis of sharp-wave ripples reveals common waveform features across species

Andrea Navas-Olive [1,4] ✉, Adrian Rubio [1,4], Saman Abbaspoor [2], Kari L. Hoffman [2,3] & Liset M. de la Prida [1] ✉

The study of sharp-wave ripples has advanced our understanding of memory function, and their alteration in neurological conditions such as epilepsy is considered a biomarker of dysfunction. Sharp-wave ripples exhibit diverse waveforms and properties that cannot be fully characterized by spectral methods alone. Here, we describe a toolbox of machine-learning models for automatic detection and analysis of these events. The machine-learning architectures, which resulted from a crowdsourced hackathon, are able to capture a wealth of ripple features recorded in the dorsal hippocampus of mice across awake and sleep conditions. When applied to data from the macaque hippocampus, these models are able to generalize detection and reveal shared properties across species. We hereby provide a user-friendly open-source toolbox for model use and extension, which can help to accelerate and standardize analysis of sharp-wave ripples, lowering the threshold for its adoption in biomedical applications.

The study of brain rhythms has bolstered our understanding of the neural basis of cognition. Because these signals emerge from the coordinated activity of multiple neurons, they can be used as biomarkers of the underlying cognitive process[1]. For example, hippocampal sharp-wave ripples (SWRs) represent the most synchronous pattern in the mammalian brain and are widely considered to contribute to the consolidation of memories[2]. SWRs consist of brief high-frequency oscillations or 'ripples' (100–250 Hz), which can be detected around the hippocampal CA1 cell layer during rest or sleep. An avalanche of excitatory inputs from the CA3 region, typically visible as a slower sharp-wave component, triggers ripples locally in CA1[3,4]. Within the ripple event, neural firing patterns that occurred during exploratory behavior are reactivated outside of the experience[5,6], leading the SWR to be used as an index of consolidation-associated reactivation or replay[7–10].

Although SWRs can be detected across an array of recording methods, subfield locations and species[2,11], their underlying mechanisms and consequent local field potential (LFP) features are understood almost exclusively from measurements in rat and mouse dorsal hippocampal CA1. Even within this region, SWRs exhibit a large diversity of waveforms that presumably reflect the myriad combinations of reactivating ensembles[12–14]. Using spectral methods, their characteristics are shown to vary along the long (septotemporal) CA1 axis within animals[15] and most notably with phylogenetic distance across species, e.g., when measured in rodents vs human and non-human primates[11,16,17]. Furthermore, in diseases affecting hippocampal function, such as Temporal Lobe Epilepsy (TLE), pathological forms of ripples have been reported[18–21], as well as along aging[22,23]. However, spectral properties alone are suboptimal to separate these events from other types of faster oscillations[11,24,25].

To address this challenge, many researchers have developed feature-based strategies for detecting LFP oscillations using machine learning (ML) tools[16,26–31]. These novel strategies have accelerated our understanding of the underlying mechanisms of SWRs and the improvement of closed-loop interventions beyond those using spectral features alone[30,32]. Yet these methods have been focused on a single detection method optimized for a single target application typically in the mouse dorsal CA1, or within lab-specific approaches to detection in the brains of humans with epilepsy. As

[1]Instituto Cajal, CSIC, Madrid 28002, Spain. [2]Psychological Sciences, Vanderbilt Brain Institute, Vanderbilt University, Nashville, TN, USA. [3]Biomedical Engineering, Vanderbilt University, Nashville, TN, USA. [4]These authors contributed equally: Andrea Navas-Olive, Adrian Rubio. ✉e-mail: acnavasolive@gmail.com; lmprida@cajal.csic.es

LFP recordings are increasingly common in the clinic, the need to scale analysis from small laboratory animals to the human brain is pressing[10,33–38]. Developing these new tools will provide the community with straightforward methods to identify SWRs from pathological oscillations across the range of recording technologies, sampled regions, and background pathologies. Therefore, there is a broad demand for a consolidated toolbox of ML methods for LFP feature analysis that can be easily applied across species to aid in understanding brain function, but also advance biomedical applications.

Here, we develop and analyze a set of ML architectures applied to the problem of SWR identification, and compiled in an open toolbox: https://github.com/PridaLab/rippl-AI[39]. To favor an unbiased screening of potential ML solutions, we ran a hackathon with the mission of detecting SWR using algorithms in a supervised manner. Using community-based solutions in neuroscience is gaining traction due to their ability to foster interdisciplinary and diverse perspectives, and to promote collaboration and data sharing[40–43]. We selected the most promising architectures from the hackathon and standardized them for fair comparisons. We show how the different ML models could bias SWR detection and identify conditions for their optimal performance and stability in the mouse hippocampus (*Mus musculus*). We then extend the analysis to SWRs recorded in the macaque hippocampus (*Macaca mulatta*), to demonstrate the generalizability of SWRs detection methods to the primate order. This proof of principle will foster the development of feature-based detection algorithms for future applications to a range of models and approaches, including the human brain.

Results

Community-based proposal of ML models of SWR

To create a diversity of ML-supervised models of SWRs, we organized a hackathon that promoted unbiased community-based solutions from scientists unfamiliar with neuroscience research and SWRs in particular (see Methods). The hackathon challenge was to propose an ML model that successfully identifies SWRs in a dataset of high-density LFP recordings from the CA1 dorsal hippocampus of awake mice, used before for similar purposes[30]. Preparatory courses introduced participants to the main topics required for the challenge (Fig. 1a). To standardize the different ML models, they were given access to Python functions for loading the data, evaluating model performance, and writing results in a common format. Annotated data consisted of raw LFP signals (8 channels) sampled at 30 kHz and containing SWR events manually tagged by an expert (training set: 1794 events, two sessions from 2 mice; validation set: 1275 events; two sessions from 2 mice; Fig. 1b; Supplementary Table 1).

Participants submitted eighteen different solutions (Fig. 1c). The most used architecture was the extreme gradient boosting (XGBoost; 4 proposals), a decision tree-based algorithm very popular for its balance between flexibility, accuracy and speed[44] (Fig. 1c). Some other popular architectures were one and two-dimensional convolutional neural networks (1D-CNN, 2D-CNN; 3 and 3 solutions, respectively), deep neural networks (DNN, 3 solutions)[45], and recurrent neural networks (RNN; 2 solutions)[45] (Fig. 1c). RNN were presented in both their standard feed-forward version, and as the long-short term memory (LSTM) version that includes feedback connections, more suited for processing time series data[46]. Although these NN architectures are typically used for pattern recognition, the way they process and learn from data is remarkably different: for example, whereas CNNs are based on kernels specialized in spotting particular spatially contiguous features of the input, LSTMs use memory cells that look for time-dependent relationships in the data (Fig. 1d). Two other algorithms were also submitted: a support vector machine (SVM; 1 solution; Fig. 1c) and a clustering solution based on dimensionality reduction by principal component analysis (PCA), followed by k-nearest neighbors (kNN) clustering (1 solution; Fig. 1c). From the 18 solutions submitted, 5 were not functional and could not be scored (Fig. 1c, bottom). Analysis of the hackathon experience in relationship to the submitted solutions is summarized in Supplementary Fig. 1 (see Methods for details).

We sought to identify the more promising architectures for a subsequent in-depth analysis. The performance of submitted ML models was measured using the F1-score (see next section). The best performances were achieved by the 2D-CNN, one of the XGBoost models, and the SVM algorithm. Since 1D-CNNs and RNNs were submitted by several groups, and given their previously successful application to SWR detection[27,30], we decided to include them as well, resulting in five different machine learning architectures (Fig. 1c; dark arrowheads).

The goal of the ML models is to identify the presence of a SWR (or part of it) in a given analysis window (Fig. 1d, left). The selected ML architectures covered a range of processing strategies (Fig. 1d, right). XGBoost is a very popular ML algorithm that uses many decision trees in a parallel fashion, making it one of the fastest algorithms[47]. SVM regression lies within the statistical learning framework, and its objective is to find a new space where samples from different categories (SWRs vs no-SWRs) are maximally separated, making it one of the most robust classification methods[48]. LSTMs are especially suited for regression and classification of temporal series, like in natural language processing, using a memory-based strategy to extract relationships between non-continuous time points[46]. CNNs represent a very common approach for many detection and classification tasks applied to different data modalities (1D for signals, 2D for images, and 3D for video or volumetric reconstructions) and can approach human performance on many tasks[49]. While 2D-CNNs process input data by considering adjacency on both dimensions (spatial and temporal, in our case), the 1D-CNN solution treats each channel independently and only considers time adjacency, making them two distinct processing algorithms.

This community-based ML architecture bank that was produced by participants who were unfamiliar with SWR studies can be used to evaluate the problem of SWR automatic detection in experimental contexts. We next focused on standardizing processing and retraining the different models.

Standardization and retraining of selected algorithms

After careful examination of the submitted solutions, we noticed that data pre-processing and training strategies were very different between groups. Data characteristics, like the sampling frequency or the number of channels used for detection, can influence operation. To standardize analysis, we chose to down sample to 1250 Hz and normalize input data using z-scores, which account for differences in mean values and standard deviation across experimental sessions.

We then retrained the submitted ML architectures using the same training set of the hackathon. We randomly divided the training dataset into a set for training (70%) and a set to validate performance (30%) in unseen data prior to a more thorough validation (Fig. 2a, Supplementary Table 1). We explored a wide range of hyper-parameters for each architecture, which included the number of LFP channels (1, 3, or 8), the size of the analysis window (from less than 1 ms up to 50 ms), and model-specific parameters like maximum tree depth for XGBoost, bidirectionality for LSTM or kernel factor for CNNs (Fig. 2a). A trained ML architecture set with a particular combination of its hyper-parameters gives rise to a particular trained model (Fig. 2a). Because each architecture had different numbers of hyper-parameters, we ended up with different numbers of trained models for each architecture (1944 for XGBoost, 72 for SVM, 2160 for LSTM, 60 for 2D-CNN, and 576 for 1D-CNN). We then used the validation set to choose the 50 best models from each architecture and further tested their performance using a new test dataset (7586 SWR events; 21 sessions from 8 mice; Supplementary Table 1), previously used for the 1D-CNN model[30] (Fig. 2a, right).

The goal of training is to make the model output as similar as possible to the ground truth (GT). Because model outputs are continuous numbers between 0 and 1 representing the probability of the presence of the event in the window of analysis, choosing the detection threshold can affect performance (Fig. 2b). Lower thresholds would result in more detections (Fig. 2b, light-gray discontinuous threshold line), normally implying a larger number of both true and false positives, while higher thresholds are more conservative at the expenses of False Negatives (Fig. 2b, dark-gray threshold

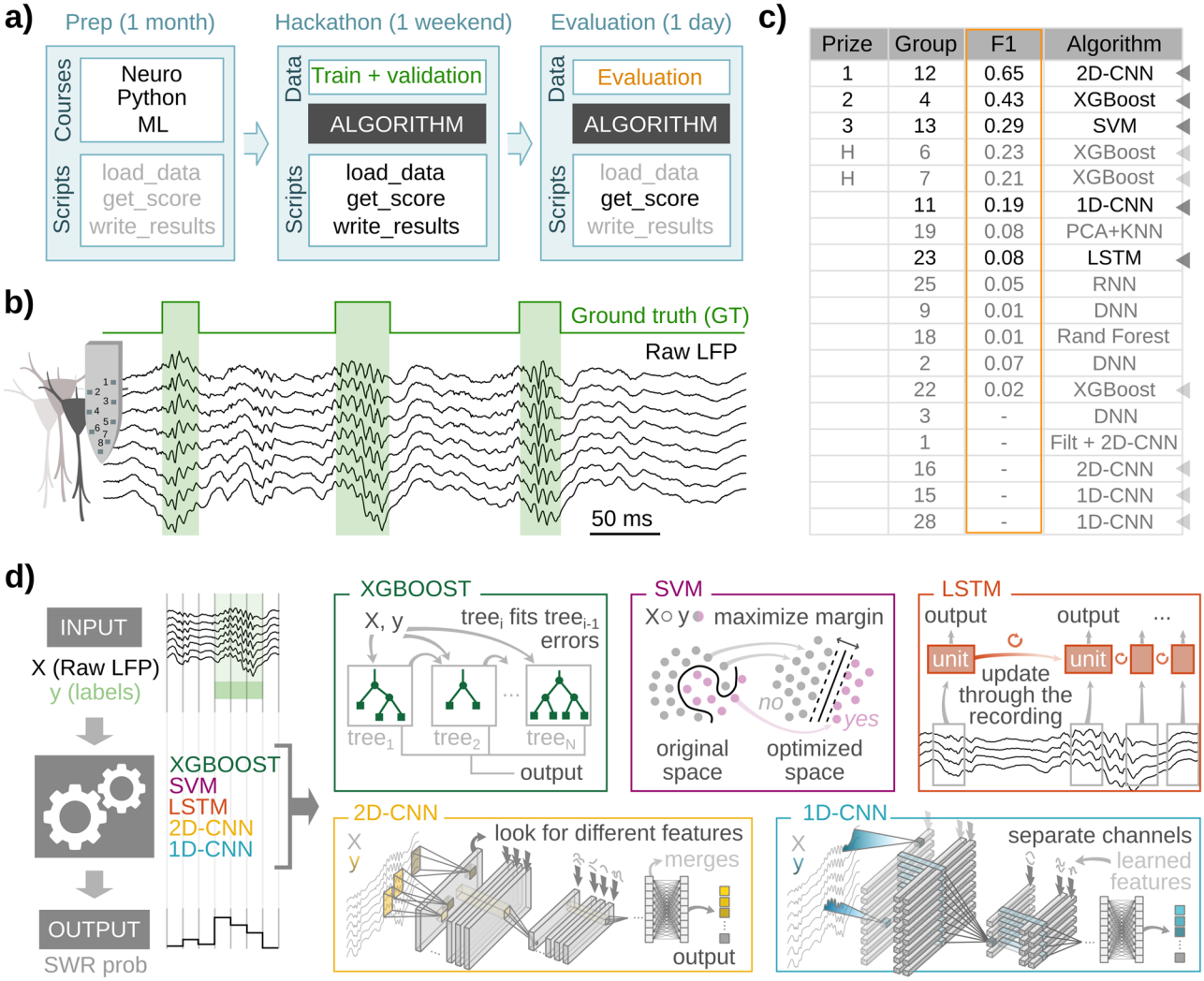

Fig. 1 | Unbiased community-based proposals of ML models for SWR detection. a Organization of the hackathon. A preparatory phase (Prep) established the basic grounds of the challenge in terms of minimal knowledge about SWRs, Python programming, and Machine Learning (ML) models. It also looked to standardize scripts and data management. The second phase consisted of the hackathon, which lasted over 53 h during three days, with participants having access to the annotated training dataset and some Python scripts. During the last evaluation phase, a new validation set was released to participants 3 h before the end of the hackathon. Solutions were ranked using the F1-score (see methods). **b** Example of the training data consisting of 8 channels of raw LFP (black) sampled at 30 kHz, with the manually tagged ground truth (GT), corresponding to SWR events. **c** Results from the hackathon. Solutions were ranked by the F1 score. F1 represents the harmonic mean between Precision (percentage of good detections) and Recall (percentage of detected GT events). Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) with/without Long-Short Term Memory (LSTM); Random Forest decision trees (Rand Forest), Extreme Gradient Boosting (XGBoost), Support Vector Machines (SVM), k-Nearest Neighbors (kNN). Chosen solutions are marked with arrowheads. Darker arrows point to the group that got the highest score of each particular architecture; light arrows point to repeated architectures. **d** Schematic representation of the SWR detection strategy and the 5 ML models used in this work.

line). An ideal model would perform well regardless of the threshold, but in practice, selecting the threshold that optimizes the true positive-false positive trade-off is unavoidable but crucial for experiments. A performance score that takes into account this trade-off is the F1-score, computed as the harmonic mean between Precision (percentage of good detections) and Recall (percentage of detected GT events) (Fig. 2c). F1 values of 1.0 would reflect a perfect match between detections and GT, whereas 0.0 reflects a perfect mismatch. Note this was the same score used to rank models in the hackathon.

After training all architectures by optimizing F1 scores over the validation set, we assessed generalization and performance using the test dataset (Supplementary Table 1). We inspected what parametric combinations gave rise to optimal ML models and found a remarkable variety of distributions (Supplementary Fig. 2a, 50-best models, ranked by best test F1). All architectures showed a great deal of variability, with almost all available parameter combinations covered. However, some parameters showed biases that depended on the ML architectures, pointing to the necessary requirements for a good performance. For example, all of the 50 best XGBoost models used 8 channels, and in general, more than 3 channel was used across successful architectures (Supplementary Fig. 2a). When we focused on the 10 best models (colored lines), we could further see how different architectures had distinct ranges of parameter values. XGBoost models required longer time windows (9/10 models had the highest F1 for 25 ms windows), whereas most SVM models employed shorter windows (<3.2 ms). LSTM, 2D-CNN, and 1D-CNNs with variable window sizes all showed very strong performance for >12.8 ms. LSTM models were allowed to use both uni- and bi-directional input flow, but all of the 10 best models were found to be bidirectional, consistent with previous reports[50]. This supports the idea that the present and future characteristics of an ongoing SWR are interdependent and predetermined by circuit mechanisms (e.g., there is a preceding buildup period followed by replay sequences[51,52]).

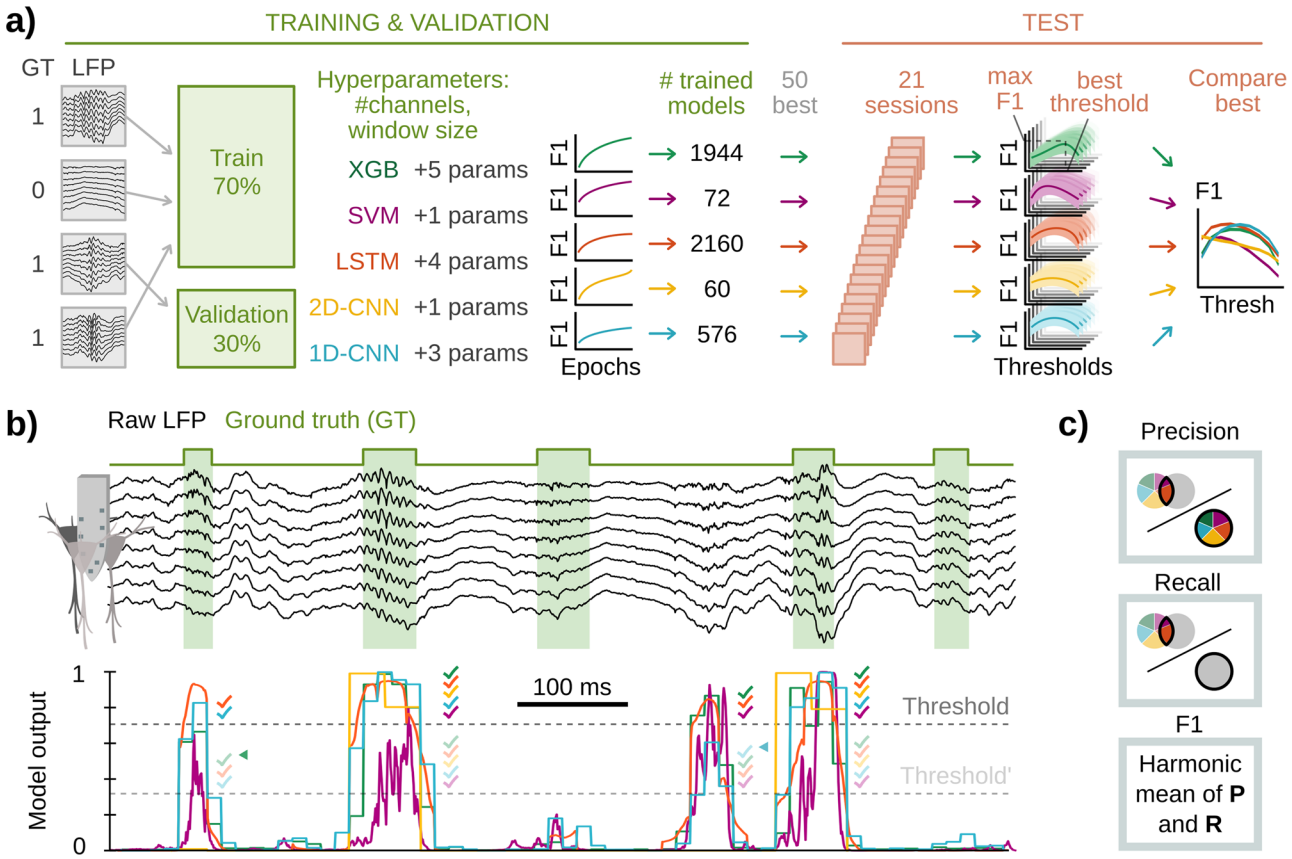

Fig. 2 | Training design and performance of ML models. a Training and selection criteria scheme. The training dataset used in the hackathon was z-scored and down-sampled to 1250 Hz. Training data were shuffled and distributed into train and validation subsets (70–30%, respectively). Each architecture was trained to optimize the F1 of the validation set using several parameters. The 50 best models were tested over a new test data set (7586 events; 21 sessions from 8 animals), generating an F1 vs threshold curve per model/ architecture. Among these 50, the model with the highest mean F1 was selected for between-models comparison (right panel). **b** LFP example of the new test set and the corresponding model outputs per window of analysis. Note different durations of true events. Setting a threshold allows defining the windows containing detected events. Colored ticks represent detections by the different models. Two different thresholds (dark and light gray) can influence what events are detected. Note how detections marked with arrows are dismissed when the threshold increases. Since SWRs constitute about 1–4% of the total recording duration, performance is computed using positive detections; that is, windows without GT or detected events are not computed for performance. **c** Schematic illustration of Precision (percentage of good detections), Recall (percentage of ground truth events that have been detected), and F1-score (harmonic mean between Precision and Recall).

Finally, to assess if there was over-fitting, we evaluated how prediction errors (logarithmic-loss) evolved along training epochs using both training (Supplementary Fig. 2b) and validation datasets (Supplementary Fig. 2c). All of the 10-best models showed decreasing or stabilized evolution of prediction errors in the validation dataset (Supplementary Fig. 2c), except for one LSTM model, which even showing a transient increase in the validation prediction error (indicating over-fitting), it ended up having the 9th best test F1 (indicating good generalization capabilities).

A plug-and-play toolbox to use any of the best five models of each architecture for SWR detection is available: https://github.com/PridaLab/rippl-AI[39].

Influence of the temporal and spatial sampling on training performance

Next, we sought to evaluate the relationship between model performance, parameters, and LFP input characteristics. Given the relevance of the temporal and spatial LFP sampling in the definition of SWRs[30], we started evaluating how the size of the analyzed window and the number of recording channels influenced performance. In order to have as much data as possible, we used the F1-scores of all the trained models over the validation set.

We found that XGBoost and LSTM were very stable, with performances changing very little for any combination of window size and the number of channels used, suggesting that these architectures can capture SWR features that are relatively invariant across temporal/spatial windows in the input data (Fig. 3a, b). Interestingly, the training parameter that most influenced these two architectures was the number of LFP channels, with 3 and 8 channels providing better performances (Fig. 3a).

Spatial information was also important for the SVM and LSTM models, which scored poorly using a single vs several channels (Fig. 3a; magenta and orange, respectively). As mentioned above, temporal resolution was also critical for SVM, which required smaller time windows of <3.2 ms to succeed in detecting SWR (Fig. 3b). For analysis windows >6.4 ms (i.e., the temporal scale of one 150Hz-ripple oscillation) performance dropped significantly, indicating that a single SWR cycle and its particular waveform across channels are optimal input information for the SVM architecture to detect events. This effect could be due to the low number of trainable parameters used for SVM (ranging from 1 to 100; see Methods), which requires less but more informative data to achieve good performances.

Finally, both the 2D- and 1D-CNN models had similar performance for any number of channels, although there was also a trend for higher spatial sampling (Fig. 3b, yellow and aqua). Interestingly, both CNN models presented a large F1 dispersion because their performance was very dependent on the window size (Fig. 3b). The 2D-CNN model exhibited maximal F1-score for 32 ms, while most 1D-CNN models best scored for

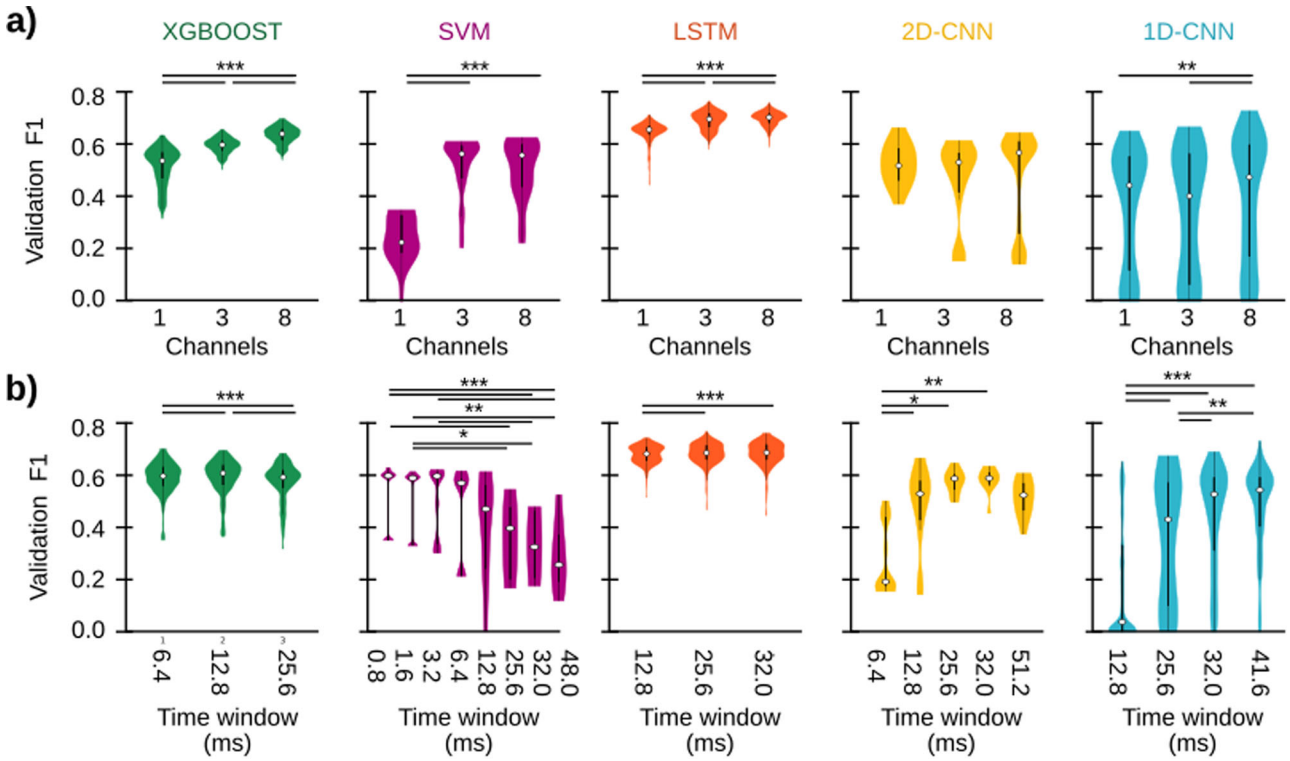

Fig. 3 | Influence of a number of channels and analysis window on training performance. a Final validation F1-score of all trained models depending on the number of input channels: one (pyramidal channel; see methods), three (the pyramidal channel, and extreme channels above and below it), or eight (all channels of the probe). Kruskal–Wallis tests with repeated measures for every architecture: XGBOOST, Chi2(2) = 1282.2, $p < 0.0001$; SVM, Chi2(2) = 33.1, $p < 0.0001$; LSTM, Chi2(2) = 964.4, $p < 0.0001$; 2D-CNN, not significant; 1D-CNN, Chi2(2) = 14.6, $p = 0.0007$. Post hoc tests: *, $p < 0.05$; **, $p < 0.01$, ***, $p < 0.001$. Violin plots represent the median (white dot), thick lines indicate the 25th/75th percentiles and the thin line extends until the most extreme data points are not considered outliers. **b** Same as panel **a**, but depending on the time window used for analysis. Kruskal-Wallis tests with repeated measures for every architecture: XGBOOST, Chi2(2) = 369.5, $p < 0.0001$; SVM, Chi2(7) = 48.8, $p < 0.0001$; LSTM, Chi2(5) = 48.0, $p < 0.0001$; 2D-CNN, Chi2(4) = 16.5, $p = 0.0024$; 1D-CNN, Chi2(3) = 126.5, $p < 0.0001$. Post hoc tests: *, $p < 0.05$; **, $p < 0.01$, ***, $p < 0.001$. Equal meaning of violin plot measures.

>40 ms (Fig. 3b). This may be related to the number of training parameters: the more parameters, the more complex tasks these algorithms can solve, provided the amount of training data is representative enough of the expected variance. This supports accurate detection in longer LFP windows. Examination of the remaining parameters suggested additional differences across architectures (Supplementary Fig. 3a–e). Interestingly, evaluating their impact on F1-scores confirmed the effect of channels and window size on model behavior (Supplementary Fig. 3f). For CNN models, the batch size (1D-CNN) and the number of kernels (2D-CNN) were also critical.

Comparison between optimized models

The analysis above provided insights on how input characteristics and processing parameters can influence detection performance in different ML models. Understanding how each architecture learns to identify ripple-like events can not only aid the development of new tools, but unveil what are the key LFP features used for detection. We thus evaluated conditions for their best performance.

For a fair comparison between architectures, we selected the 10-best models from the test set. Remarkably, our previously published 1D-CNN model[30] was among the 10-best 1D-CNN, outperforming other configurations. In order to compare their performance with state-of-the-art spectral methods, we tested filters with the optimal parametric setting and took the 10-best (see Methods). Plotting F1-scores of all models across a range of thresholds allowed visualization of their performance stability (Fig. 4a). We analyzed ML models along a range of characteristics (performance, robustness, and threshold dependency) to better inform their selection depending on research applications. Five of the 10 best trained models are available at https://github.com/PridaLab/rippl-AI/blob/main/optimized_models/.

The consistency of F1-threshold curves depended on the model architectures (Fig. 4a). Most models reached their maximal F1-score at relatively low threshold values of 0.3-0.4 and remained stable until a probability of around 0.5–0.7. Such behavior indicates robust performance since even low probability (i.e., relatively uncertain) output predictions overlapped with the ground truth. This property is very useful for online experimental applications when choosing different thresholds is not manageable, making detection more robust. Interestingly, we found that XGBoost models exhibited good performance at two threshold ranges (0.2–0.4 and 0.6–0.8), depending on how trained models penalized False negative predictions. Similarly, for both CNN architectures, we found several models operating sharply at low thresholds, while others exhibited a relatively stable operation in the 0.4–0.6 range, especially for 1D-CNN models. We confirmed the variability of different models within a given architecture by looking at their Precision vs Recall curves for the entire threshold range (Supplementary Fig. 4a). The filter showed good stability, but it performed unreliably across sessions, leading to overall lower mean F1 scores (Fig. 4a, gray). This variability suggests that even when arising from the same architecture, algorithmic processes and detection strategies SWR events could differ. This may provide a range of models for different applications.

Next, we selected the model that reached the highest F1 value from each architecture (Fig. 4a, best models, arrowheads), and compared their scores using all test sessions (Fig. 4b). We found that the LSTM and 1D-CNN best models outperformed other architectures, with mean F1-scores over 0.6 (as a reference, the inter-expert F1-score in our lab is ~0.7[30]). Precision-Recall curves from these two models clearly stood out from the other solutions (Supplementary Fig. 4b). In order to further test performance, we tested these models under new contexts. First, we evaluated detection biases by

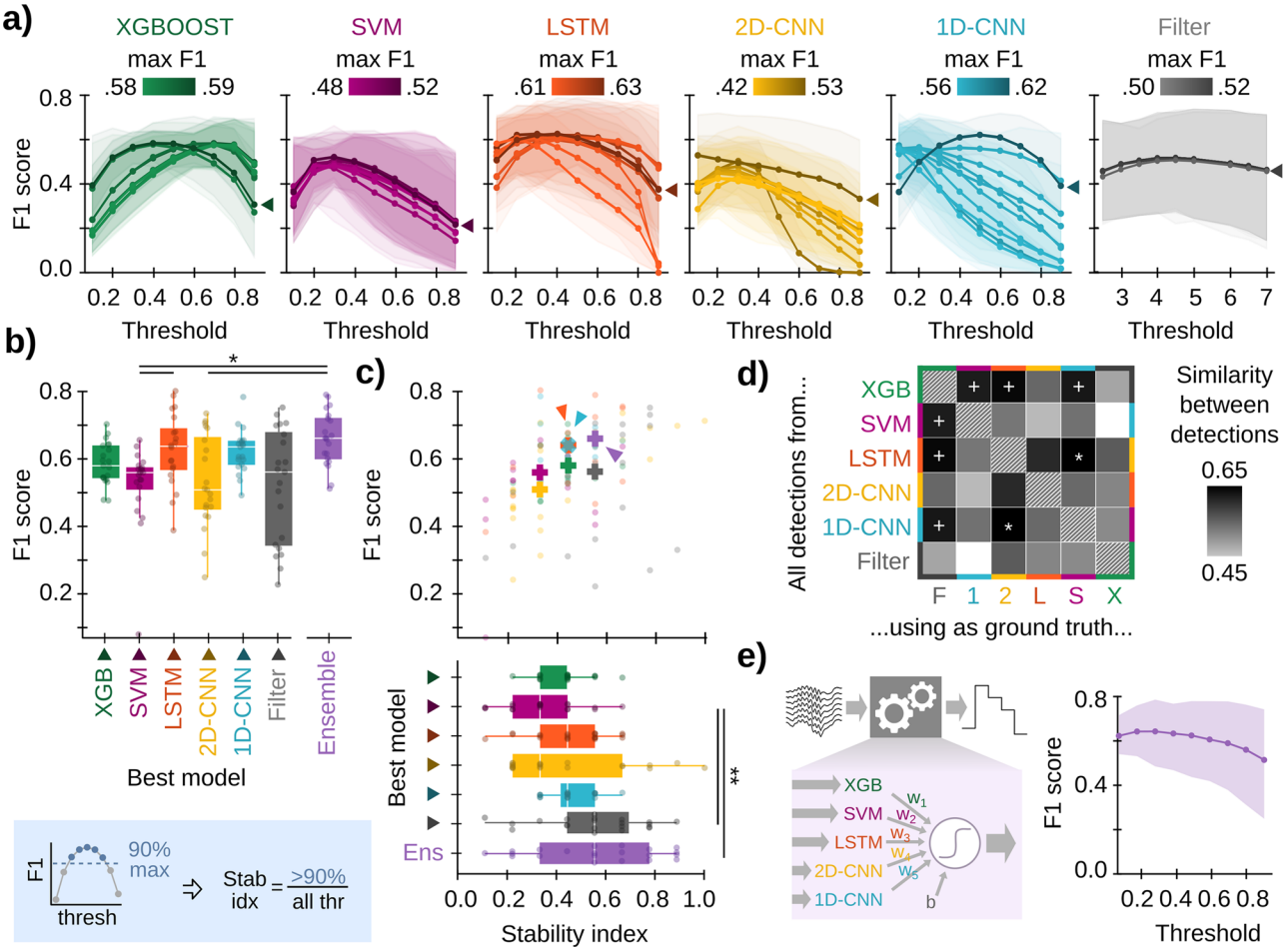

Fig. 4 | Comparison between best-performing ML models. a F1 against threshold from the 10-best models of each architecture as evaluated in the test set and the 10-best filters of all possible parametric combinations. Each line represents the performance of one trained model, colored by its maximal F1. Data was reported as a mean ±95% confidence interval for test sessions. Arrows indicate the best model of each architecture. **b** F1-scores for the best model of panel **a** and ensemble model of panel **e**. The thresholds used are 0.4 for XGBoost, 0.5 for SVM, 0.4 for LSTM, 0.1 for 2D-CNN, 0.5 for 1D-CNN, and 4.5 SD for Filter. Boxplots show the median (white line), percentile 25% and 75% (box size), and SD (error bars); each dot represents a session of the test set ($n = 21$ sessions; 8 mice). Kruskal–Wallis, Chi(6) = 23.03, $p < 0.001$. Post hoc tests: *, $p < 0.05$; **, $p < 0.01$. **c** Stability index for the same models as in panel **b** (bottom), and the stability index vs the F1 (top). Same boxplot representation as in (**b**). Kruskal–Wallis, Chi(6) = 21.29, $p = 0.002$. Post hoc tests. **, $p < 0.01$. **d** Similarity between predicted events of different architectures. Models are the same as in panels (**b**, **c**). To measure the similarity, the mean F1 across test sessions has been computed, using detected events in the *y*-axis as detections and detected events in the *x*-axis as ground truth. Note the similarity between LSTM and 1D-CNN (white *), and that by XGBoost against SVM, LSTM, and 1D-CNN (white +). **e** Ensemble model, trained using the output of the best models of the machine learning architectures. Weights were: w1 = −0.11 (XGBOOST); w2 = −1.56 (SVM); w3 = 5.33 (LSTM); w4 = 2.03 (2D-CNN); w5 = 4.07 (1D-CNN); bias = −4.97. On the right, the mean F1 score (line) ±95% confidence interval (shadow) for test sessions. Maximum F1-score and stability index for test sessions have been included in panels **b** and **c** to facilitate comparison with the rest of the methods.

using a different ground truth, manually tagged by a second expert. Performance did not significantly change for any ML model or filter (Supplementary Fig. 4d), suggesting that detectors were not biased by the first expert's labeling criteria. Second, we evaluated generalization to other behavioral contexts by detecting SWRs in freely moving animals during awake and sleep conditions. We found that the performance of ML models remained consistent, with better performance during sleep recordings (Supplementary Fig. 4e).

Given the importance of consistent threshold performance for practical applications, we also quantified the robustness of F1-threshold curves for the best models using a stability index in the test dataset (see Methods). Models with a stability index of 1.0 provide at least 90% of their maximal performance for any threshold value, a property especially suitable for experimental applications. While the best 2D-CNN model exhibited stability in some test sessions, the best LSTM and especially the best 1D-CNN models exhibited more consistent behavior (Fig. 4c, bottom; Supplementary Fig. 4c). The filter also showed generally high stability, but it was very dependent on the session, leading to a high dispersion (Fig. 4b). We confirmed this result by plotting the stability index vs F1, where both the best LSTM and 1D-CNN best models clearly segregated (Fig. 4c, top; arrowhead).

To evaluate if the different models were targeting similar or different subsets of SWR events, we compared how overlapping their set of detections was. To quantify this similarity, we computed the F1 between both groups of detections, using one of them as the ground truth (Fig. 4d). Interestingly, the 1D-CNN and LSTM showed a high level of similarity, in line with their consistent and accurate behavior (Fig. 4d, white *), while 1D-CNN and the filter showed the least overlapping. XGBoost scored a high similarity with all other architectures except for the 2D-CNN (Fig. 4d, white +). Possibly, this reflects the fact that very few of the XGBoost detections were also predicted by 2D-CNN, leading to a very low Precision (Supplementary Fig. 4f). In general, high similarities did not seem to be caused by a particularly high Precision or Recall (model A detects so few events

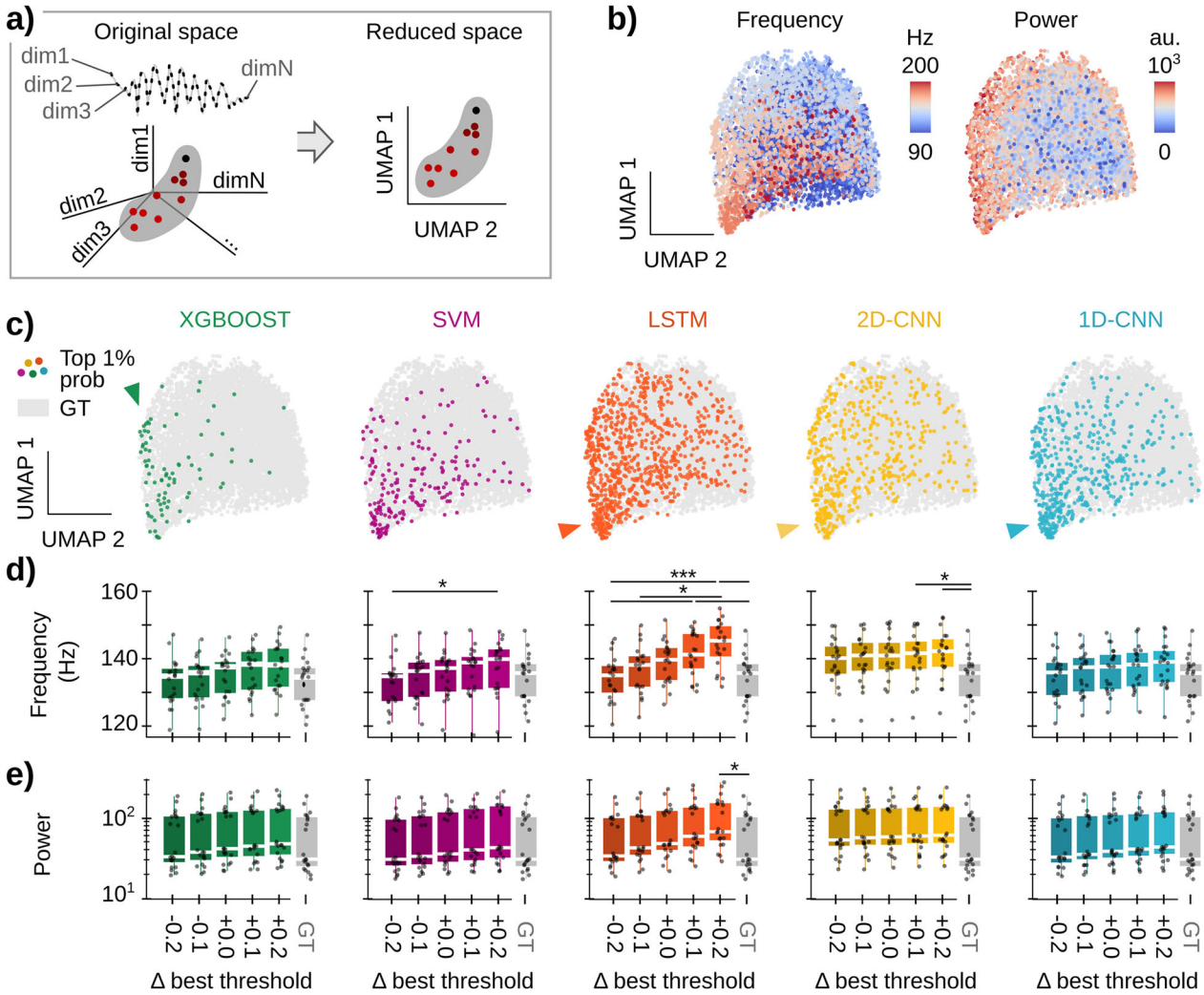

Fig. 5 | Effect of ML models and thresholds on the type of detected SWR. a Low-dimensional analysis of SWR features[14]. GT ripples are represented in a high-dimensional space by mapping each timestamp to a particular dimension. Since the sampling rate is 1250 Hz, and windows around SWRs were cut to 50 ms, there are 63 timestamps per event, and so the original space has 63 dimensions. The SWR cloud is embedded in a low-dimensional space using UMAP. **b** UMAP embedding projected into the two first axes. Each dot represents a GT ripple, and its color reflects its frequency (left) and power (right). Note how ripples in the cloud are distributed according to frequency and power, meaning that in the original space, ripples with similar features are close together. **c** Colored dots superimposed over gray GT data represent the top 1% of detected events for every given architecture, i.e., True Positive events with an output SWR probability above 99% of the maximum probability for that given model. Note that different distributions of events in the cloud reflect biases of the ML model used for detection. **d** Frequency of True Positive SWR detected by each architecture for different thresholds. In gray, frequency of all GT events. Boxplots show the median (white line), percentile 25% and 75% (box size), and SD (error bars); each dot represents the mean frequency of detected ripples of one test session (21 sessions from 8 animals). Kruskal–Wallis tests for every architecture: XGBOOST, not significant; SVM, Chi2(5) = 11.1, $p = 0.049$; LSTM, Chi2(5) = 29.9, $p < 0.0001$; 2D-CNN, Chi2(5) = 13.8, $p = 0.017$; 1D-CNN, not significant. Post hoc tests: *, $p < 0.05$; ***, $p < 0.001$. **e** Spectral power of True Positive events detected by each architecture. Same boxplot representation as in (**d**). Kruskal–Wallis tests for every architecture: XGBOOST, SVM, 2D-CNN, and 1D-CNN are not significant; LSTM, Chi2(5) = 14.0, $p = 0.016$. Post hoc tests: *, $p < 0.05$.

that all coincide with detections of model B), but by a good balance between both (events of model A and B highly overlap) (Supplementary Fig. 4f).

Such a variety of detections could be the result of internal data processing particularities of individual ML models. We wondered if we could take advantage of this heterogeneity by building an ensemble model. The ensemble model was designed to use the output of the best ML model of each architecture as an input (Fig. 4e, left; see Methods). We trained the ensemble model with the training set and tested it using the test set (Fig. 4e, right). We found an increment of both performance and stability (Fig. 4b, c; purple arrow). As expected, the higher weighted models were the ones with higher performance: LSTM ($w_3 = +5.33$) and 1D-CNN ($w_5 = +4.07$). Interestingly, XGBoost and SVM were weighted negatively ($w_1 = -0.11$, $w_2 = -1.56$, respectively).

Effect of different ML models on the features of detected SWRs

The results above suggest that different ML models may be relying on different strategies for recognizing SWRs. We thus wondered whether models could be biased towards SWRs with different features (frequency, amplitude, etc…) and whether these biases could also be reflected over different ranges of output probabilities.

In order to evaluate these issues, we resorted to a low-dimensional analysis of SWRs, which allows for their unbiased topological characterization[14]. In this strategy, SWR events are considered points in an N-dimensional space, where each dimension X (dimX) represents the LFP value sampled at a given timestamp X (Fig. 5a). In our case, as events were GT ripples of 50 ms sampled at 1250 Hz (i.e., 63 timestamps), the original space was 63 dimensions. To align SWR waveforms, we centered the 50 ms window on the SWR trough closest to the highest SWR spectral power.

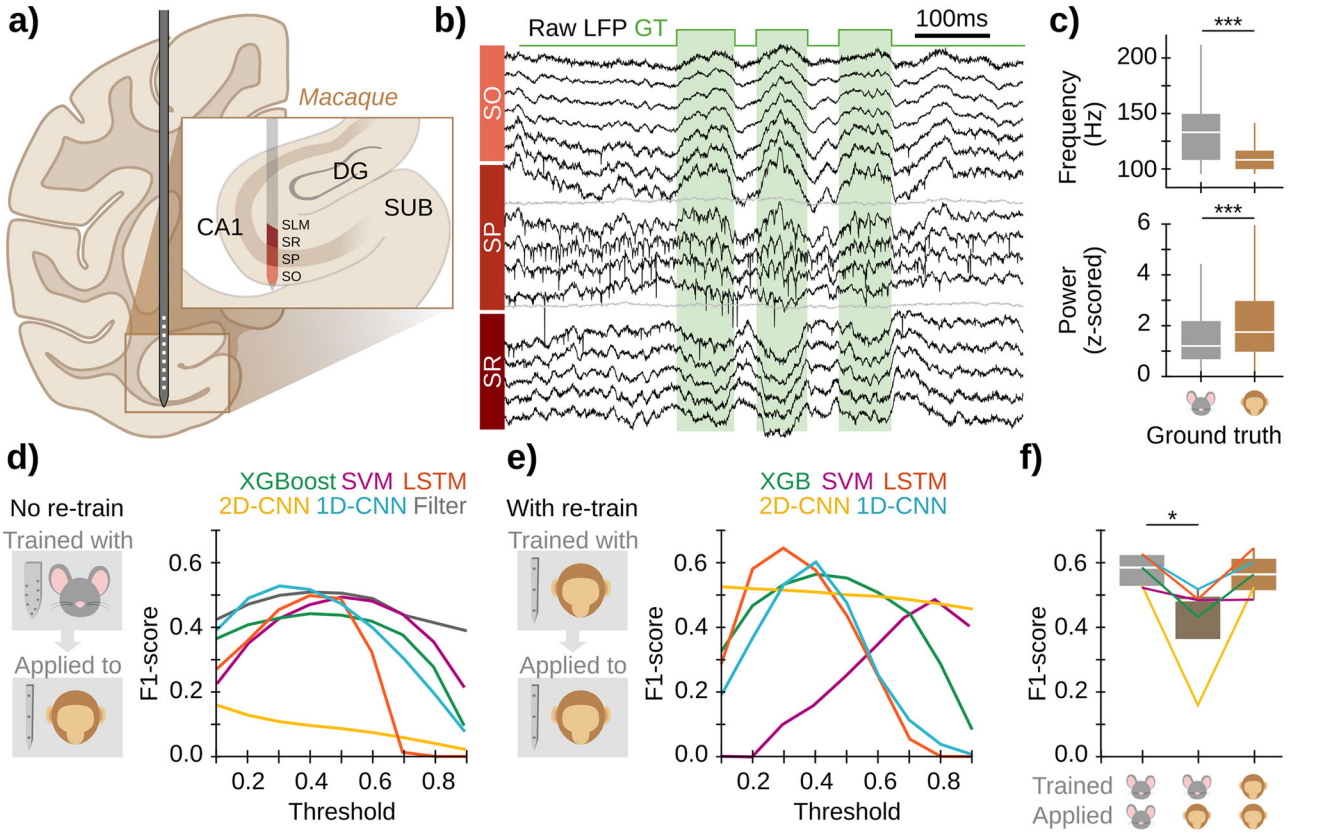

Fig. 6 | Extending sharp-wave ripple detection to non-human primates. a Linear multichannel probes were used to obtain LFP recordings from the anterior hippocampus of a freely moving monkey. **b** SWR events were manually tagged (4133 events) as in mouse data. **c** Significant differences between SWR recorded in mice and monkeys. Boxplots show the median (white line), percentile 25% and 75% (box size), and SD (error bars). Kruskal–Wallis Chi2 = 1649, $p < 0.0001$ for frequency; Kruskal–Wallis Chi2 = 407, $p < 0.0001$ for power. Post hoc tests: ***, $p < 0.001$. Data from the GT in both cases. **d** The best model of each architecture trained in mouse data and the best filter configuration for mouse data were applied to detect SWRs on the macaque data. Input data consisted of 5 LFP channels of SO, SP, and SR, and 3 interpolated channels (see Methods for details). We evaluated all models by computing the F1-score against the ground truth (GT). Note relatively good results from non-retrained ML models and filters. **e** Results of model re-training using macaque data. Data were split into a training and validation dataset (50% and 20%, respectively), used to train the ML models; and a test set (30%), used to compute the F1 (left panel). The filter was not re-trained. **f** F1-scores for the maximal performance of each model before and after re-training. Same boxplot representation as in (**c**). Kruskal-Wallis test, Chi2(2) = 8.06, $p = 0.018$. Post hoc tests: *, $p < 0.05$.

Plotting all SWR events will result in a point cloud, with events sharing similar LFP features lying close to each other, while those of different characteristics distribute separately (Fig. 5a). To ease visualization, the SWRs were embedded in a low-dimensional representation using uniform manifold approximation and projection (UMAP)[14,53].

First, we analyzed how ripple frequency and power were distributed in the UMAP embedding by coloring each dot (i.e., each SWR) based on their frequency (Fig. 5b, left) and power (Fig. 5b, right). As expected from our previous work[14], these features followed different distributions, segregating high-frequencies towards the bottom of the cloud and high-power events radially out (Fig. 5b). We then inspected events detected by the best model of each architecture by plotting the top 1% detections, defined as True Positive events for which the model output probability was >99% of its maximum probability (Fig. 5c). Interestingly, each model showed different distributions of preferred SWRs. For example, XGBoost was biased towards a subset of high-power and fast SWR events (Fig. 5c, green arrowhead), whereas the SVM model exhibited a more heterogeneous distribution. In turn, LSTM and both CNNs assigned higher probabilities to events that had a good frequency-power balance (Fig. 5c, orange, yellow, and blue arrowheads). Note how these models have more colored events, consistent with their higher stability indices reported above (Fig. 4c).

To quantify detection biases in each ML model, we analyzed the frequency and power of their True Positive events and compared them against those in the GT. Consistent with the UMAP distributions, SWR frequency was highly dependent on the threshold for SVM, LSTM, and 2D-CNN algorithms (Fig. 5d). The case of LSTM was particularly striking, with differences accumulating for all thresholds. Instead, for the SVM and 2D-CNN biases were significant only when thresholds differed ±0.2 from the optimal value (Fig. 5d). As previously reported[30], the 1D-CNN exhibited roughly consistent behavior with SWR features not statistically different from GT events. SWR power exhibited no major dependency on the threshold in any of the models but the LSTM, especially at higher detection thresholds (Fig. 5e).

Altogether, this analysis suggests that the different ML models can be exploited to detect a wide range of SWRs with different characteristics.

Using the toolbox to identify SWRs in non-human primates

A major motivation of our study is to develop methods that can be generalizable for a wider range of detection contexts, including a greater range of species and biomedical applications. Thus, we applied our ML models to LFP recordings from the hippocampus of the macaque, which shares a high level of genetic, morphological, and physiological characteristics with that of its fellow primate, the human, while enabling precise localization of signals roughly comparable to those used for the algorithm development. To accomplish this, we recorded hippocampal LFP signals from a freely moving macaque using a multichannel linear probe[54] (Fig. 6a). Unlike the original high-density probes (20 µm), recordings were obtained every 90/60 µm and spanned CA1 layers (Fig. 6a). As in mice, SWRs were manually identified (4133 events) to generate the annotated ground truth (Fig. 6b). Consistent

Table 1 | Re-training computational times

	Re-training time (s)	Detection time (s)
XGBoost	36.5	0.8
SVM	14.9	4.9
LSTM	130.6	37.4
2D-CNN	29.8	12.9
1D-CNN	24.4	10.7

The first column shows how long the re-training lasted with each model for 5 epochs, using a training batch of 32 samples (in seconds). The second column shows how long the re-trained models took to process 1 h 55 min of session (in seconds).

with the literature[16,17], macaque SWRs had lower frequencies and higher power as compared to mouse ripples (Fig. 6c).

We applied the best model of each architecture trained in head-fixed mice to macaque recordings and evaluated their performance. For a fair comparison, we flipped laminar LFP signals upside down and sampled the channel combination that best matched the characteristic mouse LFP profile (see Methods and layer orientation in Fig. 6a). Strikingly, 4/5 models reached a maximum F1 of ~0.5 (Fig. 6d), close to their maximal performance on mice data (~0.6). SVM, 1D-CNN, and LSTM exhibited the best performance, as compared to XGBoost and 2D-CNN (Fig. 6d). Importantly, the fact that both LSTM and 1D-CNN have relatively good generalization capability in raw data, even when compared to a ground truth from a different expert (Supplementary Fig. 5a), suggests that they successfully capture shared features of SWRs from mice and macaques. The filter also proved to generalize to non-human primate data using both ground truths (Fig. 6d, Supplementary Fig. 5a; gray trace). However, it was very sensitive to noise, decreasing its F1-score from 0.51 (removing noise and artifacts) to 0.16 (raw signal). This did not happen in the ML models, where changes in the F1-score were all ≤0.06 (XGBoost: 0.02; SVM: 0.01; LSTM: 0.05; 2D-CNN: 0.01; 1D-CNN: 0.06).

We next chose to re-train the 5 ML models with the macaque dataset, using 50% for training and 20% for validation. The remaining 30% was used as the test dataset to compute the final F1. We used the best model of each architecture as the starting point of the re-training and let the internal weights evolve to adjust to the new dataset features (a process commonly known as transfer learning; Supplementary Fig. 5b), which made the re-training very fast (<2 min, Table 1). Performances improved after retraining for 4/5 models, reaching an F1 increase of +0.3 for 2D-CNN (Fig. 6e). The best model was LSTM, followed by 1D-CNN and XGBoost. Furthermore, the performance of macaque SWR detection after re-training reached the mouse level (Fig. 6e), suggesting that these models identified similar key features in both species and could readily be trained to similar levels of accuracy across mice and monkeys. Similar results were observed when using the new ground truth to re-train the models (Supplementary Fig. 5c), where even SVM improved. A user-friendly open Python notebook to re-train any of the 5 models and use it for event detection is available at https://github.com/PridaLab/rippl-AI/blob/main/examples_retraining.ipynb.

Discussion

Here, we provide a pool of models for automatic SWR detection based on different ML architectures. These include some of the most used ML solutions, such as XGBoost, SVM, 1D- and 2D-CNN and LSTM. The models, which resulted from unbiased community-based proposals, are able to capture a wealth of SWR features recorded in the dorsal hippocampus of head-fixed mice. When applied to LFP recordings from a freely moving macaque, these models were able to generalize detection. Moreover, an ML ensemble model exhibited more stable performance and accuracy even when compared to standard filter approaches, further supporting AI-based solutions to the analysis of LFP events.

The need for detecting and classifying high-frequency oscillations such as SWRs has accelerated over recent years for advanced biomedical applications[27,32,34,41,55]. Identification of these events can help to delineate normal from pathological epileptogenic territories[18,56,57], and to develop closed-loop intervention strategies for boosting memory function[32,34]. However, spectral-based methods have proved suboptimal, especially to differentiate between normal and pathological oscillations[58,59]. Moreover, while using spectral filters online can provide reasonable good detection levels, experimental artifacts and noise compromise reliability. Therefore, the community is actively seeking novel LFP feature-based strategies. Recently, solutions based in ML methods have started to emerge[25,27,30,56]. Although their performance is not increasing beyond the expert's limit or spectral methods, they provide opportunities for advanced analysis, especially when dealing with high-density recording channels. Using these tools will drive advances not only in the online detection of SWRs but also in their unbiased categorization for better mechanistic understanding[11,13,21,30,60], including their functional links to visuospatial and episodic memory[10,11,16,33,37,38].

Amongst the 5 ML architectures examined here, we found the LSTM and 1D-CNN to provide the best performance and reliability using rodent data. The other models exhibited roughly similar behavior depending on the input parameter selection (recording channels and analysis windows). While, in general, we found that all of them performed better with high-density multi-channel recordings (8 channels), some of them (e.g., 2D-CNN) exhibited similar results while operating over data sampled with 1–3 channels. This suggests they may be able to identify characteristic features with reduced spatial information, which could facilitate applications to human recordings[19,36].

Detection of SWR candidates with ML models is based on using a probability threshold. We found that the different models exhibited a degree of sensitivity to threshold selection, with LSTM, XGBoost, and 1D-CNN providing a wider range of operational stability. Thus, there is a large range of thresholds in these models, which provide relatively similar performance. This is very important for online applications when threshold selection can affect experimental results in real time[25].

The different ML models are biased toward SWRs with slightly different properties, probably reflecting their internal representations of the characteristic LFP features[30]. During training, each model learns to identify specific LFP features, making ripples distinguishable from background LFP signals so that during SWR detection, the presence of those features raises their output probability. The fact that the properties of detected SWR depend on the probability threshold for SVM, LSTM, and 2D-CNN suggests they rely more on frequency and relative power features. On the contrary, XGBoost and 1D-CNN models showed less bias. Such behavior is critical to make detection more generalizable across physiological states (e.g., sleep vs awake), species (e.g., macaque), and cognitive demands (e.g., learning), known to significantly influence SWR features[14,38].

We found that this behavior of ML models is also consistent with their reliable performance using ground truth data from different experts. Previously, we found about 0.7 inter-expert F1 for SWR in mice and suggested the importance of considering community-tagging strategies[30]. By aggregating data from multiple laboratories and using the ensemble ML model presented here, we hope our work permits advancing in more inclusive and sharable solutions for detecting sharp wave ripples from diverse datasets.

When applied to data from the macaque anterior hippocampus, we found that models trained with LFP signals from the dorsal hippocampus of mice can perform relatively well, especially considering established differences in frequency and in LFP shape in monkeys and human[10,16,17]. After re-training, their operation improved significantly, reaching the inter-experts' performance levels at 0.7[30]. This demonstrates the strong capability of the ML models to generalize and suggests the existence of shared features across species. This is of particular importance because many human applications may not have the exact spatial localization or the same electrode types, in some cases even within studies, and so any effective ML applications will need a high degree of generalizability, potentially boosted by the use of transfer learning. It also demonstrates the proof of principle for applying to a

wider range of measurements, including other animal models and ripple-adjacent pathologies such as TLE seizures[20,56].

More testing along these lines will identify the extent of generalizability across different permutations of species, locations, electrode sampling, and types to find the limits of these ML models. To enable such developments, we made several of the 10 best trained models and our coding strategies for detection and retraining openly available to the research community at https://github.com/PridaLab/rippl-AI[39]. They can be tested through open-source notebooks that are ready to use, with enough examples to illustrate their operation capability. Although the notebooks provide easily readable code, they may not be optimal for further code development. That is why the core functions are written as separate Python modules. Users can test these models for SWR detection by loading their own data and defining the channels. The rippl_AI repository has a wide variety of SWR detection tools that include optional supervised detection curation and a graphical user interface for a quick visual exploration of detected events depending on the threshold chosen, as well as the option of retraining a model with the user's own data.

This collection of resources joins the many other community-based approaches for model benchmarking[29,41,55]. Crowdsourced strategies are becoming a tool to advance solutions to particularly difficult problems that require knowledge integration[40,43]. This provides the field with a set of platforms for detecting SWR from diverse datasets using traditional and state-of-the-art algorithms (e.g., our own rippl-AI toolbox and https://www.sharpwaveripples.org/). Our toolbox goes beyond SWR detection, easing the development of personalized ML models to detect other electrophysiological events of interest[31]. This may be critical in experimental and/or clinical cases, where other detection criteria than those maximizing performance, may be more important. For instance, different experiments may call for avoiding either type I or type II errors, and hence the balance between Precision and Recall. Such versatility of our toolbox may be further exploited to accelerate our understanding of hippocampal function and to support the development of biomedical applications.

Methods

The hackathon

In order to explore a wide variety of ML models to the problem of SWR detection, we organized a hackathon (https://thebraincodegames.github.io/index_en.html). We specifically targeted people unfamiliar with SWR studies who could provide unbiased solutions to the challenge. A secondary goal of the hackathon was to promote their interest and engagement at the interface between Neuroscience and Artificial Intelligence, especially for future young scientists. The event was held in Madrid in October 2021, using remote web platforms. Some of us (ANO) coordinated the event. Consent to participate and to share relevant personal data was obtained prior to the event. All participants were informed of the goal of the hackathon and agreed that their solutions were subject to subsequent investigation and modification.

The hackathon comprised 36 teams of 2–5 people (71% males, 29% female), for 116 participants in total. They represent 45% of undergraduate students, 38% of master students, 15% of Ph.D. students, and 3% of non-academic workers (Supplementary Fig. 1a). On average, they were young in their professional careers, with 77% of participants being research-oriented (Supplementary Fig. 1a). Previous to the hackathon, we monitored the participants' self-declared knowledge level on Neuroscience, Python programming, and ML, in general, using a survey (Supplementary Fig. 1b). To provide a homogeneous floor to address the challenge, we organized three online seminars to cover each of the three topics one month before the activity. Seminars were recorded and made available for review along with the experience.

The hackathon was held over one weekend (Friday to Sunday), during which groups had to design and train an ML algorithm to detect SWRs. To standardize the different algorithms for future comparison, they were given Python functions to load the data, compute a performance score, and write results in a common format. Data sets were available from a public research-oriented repository at Figshare (https://figshare.com/projects/cnn-ripple-data/117897). Participants were given a training set to train their algorithms and a validation set to run tests.

Data consisted of raw 8-channel LFP signals from the hippocampal CA1 region, recorded with high-density probes, which were used before for similar purposes[30]. SWR was manually tagged to be used as ground truth (training set: 1794 events, two sessions from two mice; validation set: 1275 events; two sessions from two animals). Since participants had two days to design and train solutions, groups were allowed to interact with us to ask for technical questions and clarification.

We monitored participant's engagement throughout the hackathon using short questionnaires. This allowed us to check their motivation and other emotional states (i.e., frustration, interest, etc…). Some people dropped out during the days of the hackathon (Supplementary Fig. 1d). We found many participants felt confused and frustrated with the challenge, and this correlated with their performance, as a posterior analysis suggested (Supplementary Fig. 1e).

Training, validation, and test datasets, and ground truth

Participants of the hackathon were provided with an annotated dataset consisting of raw LFP signals recorded from head-fixed mice using high-density probes (8 channels)[30]. Awake SWR events were manually tagged by an expert who identified the start and the end of each event. The start of the SWR was defined near the first ripple of the sharp-wave onset. The end of the event was defined at the latest ripple or when the sharp wave resumed. The training set consisted of two recording sessions from 2 mice[30]. They contained 1794 manually tagged SWRs. The validation set consisted of two recording sessions from another 2 mice and contained 1275 SWR events (Supplementary Table 1).

For posterior analysis of the results of the hackathon, we used an additional test dataset consisting of the 2 validation sessions mentioned before plus another 19 sessions for a total of 21 sessions from 8 different mice. They all contained a total of 7423 manually tagged SWRs (Supplementary Table 1). In addition, to evaluate the effect of different expert's definitions of SWRs, we used the ground-truth dataset tagged by a new expert (nGT) to compare against the original GT (oGT) used for training.

To test the ability of the trained models to detect SWR in different physiological conditions, we used data from freely moving mice recorded during awake and sleep conditions, as reported recently[14]. This data consisted of LFP signals obtained with linear silicon probes (16 channels) from 2 mice (Supplementary Table 1). Signals were sampled around the CA1 cell body layer and expanded by interpolation to meet the 8-channel input of the ML models (Supplementary Fig. 4e). SWR was tagged by the original expert.

ML models specifications

Five architectures were selected out of the 18 solutions submitted to the hackathon: XGBoost, SVM, LSTM, 2D-CNN, and 1D-CNN. For the purpose of fair comparisons, they were retrained and tested using homogenized pre-processing steps and data management strategies (see below).

We used Python 3.9.13 with libraries Numpy 1.19.5, Pickle 4.0, and H5Py 3.1.0. To build the different neural networks, we used the Tensorflow 2.5.3 library, with Keras 2.5.0 as the application programming interface. XGBoost 1.6.1 was used to train and test the boosted decision tree classifiers. Scikit-learn 1.1.2 and Imbalanced-learn 0.9.1 were used to train support vector machine classifiers. Analysis and training of the models were conducted on a personal computer (i7-11800H Intel processor with 16 GB RAM and Windows 10).

Data preparation

For subsequent training and analysis of the architectures selected from the hackathon, all data was pre-processed. From each recording session, two matrices were extracted: X, with the raw LFP data, shaped (# of timestamps, # of channels), and Y, the ground truth generated from the expert tagging (# of timestamps). A timestamp of Y is 1 if a SWR event is present.

Values for matrix X were subsampled at 1250 Hz, taking into consideration that SWRs are events that have frequencies in the range of up to 250 Hz. Before retraining the algorithms, data was z-scored with the mean and standard deviation of the whole session.

Training and validation split

For retraining the architectures, the same training dataset provided in the hackathon was used (2 sessions from 2 mice; 1794 SWR events). For initial testing, these two sessions were split according to a 70/30 train/validation design. To evaluate the generalization capabilities of the models when presented with unseen data, we used several test sessions, which provide the necessary animal-to-animal, as well as within-animal (sessions) variability. Test sessions included the 2 sessions from the validation dataset provided in the hackathon and 19 additional sessions (21 sessions from 8 mice, 7423 SWR events).

For re-training, the two training sessions were concatenated and divided into 60 s epochs. Each epoch was assigned randomly to the train or validation set, following the desired split proportion. The data was reshaped to be compatible with the required input dimensionality of each architecture (see below). In order to evaluate model performance, two different datasets were used: the validation set described above (used for an initial screening of the 50 best models for each architecture) and the test set (used for generalization purposes).

Identification of SWR events in the data was implemented using analysis windows of different sizes. To identify SWR events detected by the ML models, we set a probability threshold to identify windows with positive and negative predictions. GT was annotated in the different analysis windows of each session. Accordingly, predictions were classified into four categories: True Positive (TP), when the prediction was positive and the GT window did contain an SWR event; False Positive (FP), when the prediction was positive in a window that did not contain any SWR; False Negative (FN), when the prediction was negative in a window with an SWR; and True Negative (TN) when the prediction was negative and the window did not contain any SWR event.

If a positive prediction had a match with any window containing a SWR, it was considered a TP, or it was classified as FP otherwise. All true events that did not have any matching positive prediction were considered FN. Negative predictions with no matching true events windows were TN.

With predicted and true events classified into those four categories, there are three measures that can be used to evaluate the performance of the model. Precision (P), which was computed as the total number of TPs divided by TPs and FPs, represents the percentage of predictions that were correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + FP}$$

Recall (R), which was calculated as TPs divided by TPs and FNs, represents the percentage of true events that were correctly predicted.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + FN}$$

Finally, the F1-score, calculated as the harmonic mean of Precision and Recall, represents the network performance, penalizing imbalanced models.

$$F1 = \frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + Recall}$$

To ease subsequent evaluation of ML models for SWR analysis, we provide open access to codes for retraining strategies[39]: https://github.com/PridaLab/rippl-AI.

Parameter fitting

Different combinations of parameters and hyper-parameters were tested for each architecture during the training phase (1944 for XGBoost, 72 for SVM, 2160 for LSTM, 60 for 2D-CNN, and 576 for 1D-CNN).

Two parameters were shared across all architectures: the number of channels and the number of timestamps in the analysis window (referred to as the window size). These parameters define the dimensionality of the input data (# timesteps × # channels), i.e., the number of input features.

The number of channels to be used was set at 1, 3, or 8. When 1 channel was chosen, it was that corresponding to the CA1 pyramidal layer channel, defined as the channel with the most power in the ripple bandwidth (150–250 Hz). The superficial, pyramidal, and deep channels were used as 3 channels. All the channels in the shank were used for the 8-channel input configuration.

The number of timestamps defines the window size. The tested values depended on each architecture and ranged between windows of 0.8–51.2 milliseconds. The rest of the parameters were specific for each architecture (see below).

The F1-score metric for the training and validation set was calculated to compare the performance of the models, with the validation F1 serving as a priori metric of the generalization of the models, allowing for a selection of models without performing a complete test.

For each model, a test-F1 array was calculated with different thresholds (generally, from 0.1 to 0.9 with 0.1 increments), and the highest value for each model was used for comparison among models of the same architecture. As a result, the 50-best performing models were selected after the initial retrained test.

Validation process

The aim of validation is to find the model that generalizes best to unseen data for each architecture. With that in mind, defining a metric that takes this into account is not a straightforward task.

To weigh each validation session (21) independently, an F1 array was calculated for each individual session, resulting in a matrix of 21 per number of threshold values (#th). The mean of sessions gives us a #th array that quantifies the performance/generalization of the model as a function of the chosen threshold. The maximum value of this array will represent the best performance that could be achieved with this model if the threshold is correctly selected. This single value is what will be compared. Using this strategy, we narrowed down available models to the 10 best of each architecture before selecting the best model.

XGBoost

Based in the Gradient Boosting Decision Trees algorithm, this architecture trains a tree with a subset of samples and then calculates its output[44]. The misclassified samples are used to train a new tree. The process is repeated until a predefined number of classifiers are trained. The final model output is the weighted combination of individual outputs.

In the training process, we worked with quantitative features (LFP values per channel), and a threshold value for a specific feature was considered in each training step. If this division correctly classifies some samples of the subset, two new nodes are generated in the next tree level, where the operation is repeated until the maximum tree depth is achieved, and a new tree with the misclassified samples is generated. The input is one dimensional (# of channels × # of timesteps) and produces a single output.

Specific parameters of XGBOOST are the Maximum depth and the maximum levels for each tree, which may lead to overfitting. Learning rate, which controls the influence of each individual model in the ensemble of trees. Gamma is the minimum loss reduction required to make a further partition on a leaf node, with larger values leading to conservative models. Parameter λ contributes to the regularization, with larger values preventing overfitting. Scale is used in imbalanced problems; the larger the more penalized false negatives are during training.

Trained models had a number of trainable parameters ranging from 1500 to 17,900.

SVM

A support vector machine is a classical classifier that searches for a hyperplane in the input dimensionality that maximizes the separation between different classes. This is only possible in lineal separation problems, so some misclassifications are permitted in real tasks. Usually, SVM performs a transformation on the original data using a kernel (linear or otherwise) that increases the data dimensionality but facilitates classification.

During training, the parameters that define the separation hyperplane are updated until the maximum number of iterations is achieved or the rate of change in the parameters go below a threshold. The input is one-dimensional (# of channels × # of timesteps) and produces a single output.

Specific parameters of SVM are the kernel type. Using nonlinear kernels resulted in an explosive growth in training and predicting times due to the enormous number of training data points. Only the linear kernel produced manageable times. The under-sample proportion rules out negative samples (windows without ripple) until the desired balance is achieved: 1 indicates the same number of positives and negatives.

Trained models had a number of trainable parameters ranging from 1 to 480.

LSTM

Recurrent neural networks (RNNs) are a subtype of NNs especially suited to work with temporal series of data, extracting the hidden relations and tendencies between non-contiguous instants. Long short-term memory (LSTMs) are RNNs with modifications that prevent some associated problems[46].

During training, three sets of weights and biases are updated in each LSTM unit, associated with different gates (Forget, input, and output). To prevent overfitting, two layers of dropout (DP) and batch normalization (batchNorm) were inserted between LSTM layers. DP randomly prevents some outputs from propagating to the next layer. BatchNorm normalizes the output of the previous layer. The final layer is a dense layer that outputs the event probability. The input is two-dimensional (# of timesteps, # of channels) and produces a probability for each timestep. After each window, the internal weights are reset.

Specific LSTM parameters: bidirectional if the model processes the windows forwards and backward simultaneously; # of layers is the number of LSTM layers; # of units is the number of LSTM units in each layer, and # of epochs, which is the number of times the training data is used to perform training.

Trained models had a number of trainable parameters ranging from 156 to 52851.

2D-CNN

Convolutional neural networks use convolutional layers consisting of kernels (spatial filters) to extract the relevant features of an image[49]. Successive layers use this as inputs to compute general features of the image. This 2D-CNN moves the kernels along the two axes, temporal (timesteps) and spatial (channels). The first half of the architecture includes MaxPooling layers that reduce the dimensionality and prevent overfitting. A batchNorm layer follows every convolutional layer. Finally, a dense layer produces the event probability of the window.

During training, the weights and biases of every kernel are updated to minimize the loss function, which was taken as the binary cross entropy:

$$H_p(q) = \frac{-1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p(y_i)\right) + (1 - y_i) \cdot \log\left(1 - p(y_i)\right)$$

N is the number of windows in the training set, y_i is the label of the i window and $p(y_i)$ is the probability of ripple that the model predicts. The input is # of timesteps and # of channels; and produces a single probability for each window.

The 2D-CNN was tested with a fixed number of layers and kernel dimensions. The kernel factor parameter determined the number of kernels in this structure: $32 \times kf\,(2 \times 2)$, $16 \times kf\,(2 \times 2)$, $8 \times kf\,(3 \times 2)$, $16 \times kf\,(4 \times 1)$, $16 \times kf\,(6 \times 1)$, and $8 \times kf\,(8 \times 1)$. In parenthesis, the size of the kernels in each layer.

Trained models had a number of trainable parameters ranging from 1713 to 24,513.

1D-CNN

This model is also a convolutional neural network, but the kernels only move along the temporal axis while processing spatial information. The number of layers and the kernel size were fixed. The tested models had 7 sets of 1D convolutional layer, batchNorm, and LeakyRelu layer, followed by a dense sigmoid activation unit. This model is similar to our previous CNN solution[30].

During training, the weights and biases of the layers were also updated with the objective of minimizing the binary cross entropy. The input is # of timesteps and # of channels and produces a single probability for each window.

The specific parameters for 1D-CNN included the kernel factor, which defined the number of kernels in each conv layer. The size and stride for each layer were equal and fixed. The size of the kernels in the first layer was defined as the length of the input window divided by 8. Structure: $4 \times kf$ (# timesteps//8 × # timesteps//8), $2 \times kf$ (1×1), $8 \times kf$ (2×2), 4×1 (1×1), $16 \times kf\,(2 \times 2)$, $8 \times kf\,(1 \times 1)$, and $32 \times kf\,2 \times 2)$. Parameters also include # of epochs, the number of times the training data is used to perform training, and # of training batch samples, which is the number of windows that are processed before parameter updating.

Trained models had a number of trainable parameters ranging from 342 to 4253.

Filter

We used a Butterworth filter, which is considered the gold standard for SWR detection[25]. The parameters that we varied were the high-cut frequency, the low-cut frequency, and the filter order. No training was run, but instead, all combinations between parameters were tested, and the best 10 models were kept. The 10 best models had the following set of parameters: (1) 100–250 Hz and 5th order, (2) 100–250 Hz 4th order, (3) 100–250 Hz 8th order, (4) 100–250 Hz 7th order, (5) 100–250 Hz 6th order, (6) 100–250 Hz 3rd order, (7) 100–250 Hz 9th order, (8) 100–250 Hz 10th order, (9) 100–250 Hz 2nd order, (10) 90–250 Hz 5th order.

In order to extract the event times using the filter output, the envelope of the filtered signal is computed. The standard deviation of this signal is multiplied by a factor used to define a threshold. The intervals where the filtered signal surpasses said threshold are the detected events.

Threshold alignment to compare F1 curves in Supplementary Fig. 4 was done by selecting 9 standard deviation multiplication factors that resulted in a similar F1 curve as those in the ML models: 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, and 7.

Ensemble model

This model consists of a single-layer perceptron, with 5 inputs and 1 output, computed using a sigmoid as the activation function. It takes the predicted output of the previously trained ML models and combines them in a weighted probability.

During training, the weights and biases of the layer were updated with the objective of minimizing the binary cross entropy. The input shape is 5 (the output of the models) and generates a probability for each timestamp.

Parameters tested during training were the number of epochs and the number of samples per training batch. This trained model had 6 trainable parameters: 5 weights and 1 bias.

Stability index

This metric, shown in Fig. 4c, quantifies the consistency of the performance of a model across all possible thresholds. It is calculated as the number of thresholds whose F1 is above the 90% of the best F1 value of the model divided by the total number of thresholds.

Possible metric values range from 1, from a very consistent model, and 0, from a completely inconsistent model.

Characterization of SWR features

SWR properties (ripple frequency and power) were computed using a 100 ms window around the center of the event, measured at the pyramidal channel of the raw LFP. Preferred frequency was computed first by calculating the power spectrum of the 100 ms interval using the enlarged band-pass filter 70 and 400 Hz, and then looking for the frequency of the maximum power. In order to account for the exponential power decay in higher frequencies, we subtracted a fitted exponential curve ('fitnlm' from MATLAB toolbox) before looking for the preferred frequency. To estimate the ripple power, the spectral contribution was computed as the sum of the power values for all frequencies lower than 100 Hz normalized by the sum of all power values for all frequencies (of note, no subtraction was applied to this power spectrum).

Dimensionality reduction using UMAP

To classify SWR, we used topological approaches[14]. The UMAP version 0.5.1 (https://umap-learn.readthedocs.io/en/latest/) in Python 3.8.10 Anaconda was used, which is known to properly preserve local and global distances while embedding data in a lower dimensional space. In all cases, we used default values for reconstruction parameters. Algorithms were initialized randomly. UMAP provided robust results independent of initialization. Events were GT ripples sampled at 1250 Hz, centered around the SWR trough closest to the highest SWR spectral power, and taking a 50 ms window around that point. As a result, events were points in a 63 ($1 + 0.025*1250$) dimensional cloud. The parameters chosen to fit the cloud were: the metric (metric) was Euclidean; the number of neighbors (n_neighbors), which controls how UMAP balances local vs global structure in the data, was set to 20; minimum distance (min_dist), which controls how tightly UMAP is allowed to pack points together by setting the minimum distance apart that points are allowed to be in the low dimensional representation, was set to 0.1; and the number of components (n_components), that sets the dimensionality of the reduced dimension space we will be embedding the data into, was set to 4. This goes in accordance with previous studies that had shown the intrinsic dimension of SWRs is 4D[14].

Prediction and re-training of non-human primate data set

To study the generalization capabilities of the different architectures, we used data from a freely moving macaque targeting similar CA1, as completed in our mouse data (methods are described in ref. [54]). Recordings were obtained with a 64-ch linear polymer probe (custom 'deep array probe', Diagnostic Biochips) that recorded across the CA1 layers of the anterior hippocampus (Fig. 6a) where layers were identifiable relative to the main pyramidal layer, which contains the greatest unit activity and SWP power. LFP signals were sampled at 30 kHz using a Freelynx wireless acquisition system (Neuralynx, Inc.). Data corresponds to periods of immobility for a duration of almost 2 h and 40 min, predominantly comprised of sleep in overnight housing.

Similar to the procedures used in mice, SWR beginning and ending times were manually tagged (ground truth). First, the best model of each architecture, already trained with the mouse data, was used to predict the output of the primate data with no retraining. For this purpose, we used recordings of different channels around the CA1 pyramidal channel, matched to meet the laminar organization of the dorsal mouse hippocampus. Specifically, we used one CA1 radiatum channel, 720 µm from the pyramidal layer, three channels in the pyramidal layer, at +90 µm, +0 µm and −90 µm from the pyramidal channel, and a stratum oriens channel 720 µm from the pyramidal channel. The pyramidal channel was defined at the site with the maximal ripple power. We complemented these 5 recordings with 3 more interpolated signals, making a total of 8 input channels [oriens, interpolated, pyramidal, pyramidal, pyramidal, interpolated, interpolated, radiatum] using a linear interpolation script available at Github: https://github.com/PridaLab/rippl-AI/blob/main/aux_fcn.py. The applied preprocessing was the same as with the mice data: subsampling to 1250 Hz and *z*-score normalization.

With the aim of studying the effect of retraining with completely different data, we retrained the models. Data was split in three sets (50% training, 20% validation, 30% test), and used to retrain and validate the models. For re-training, we reset all trainable parameters (internal weights) but kept all architectural hyper-parameters fixed (input number of channels, input window length, number of layers, etc…) as with the mouse data, making the re-training process much faster than the original training that required a deep hyper-parametric search (per model re-train: 2 min for XGBoost, 10–30 min for SVM, 3–20 min for LSTM, 1–10 min for 2D-CNN and 1–15 min for 1D-CNN). We used a second expert tagging to evaluate the generalization capability of retrained models.

Statistics and reproducibility

Statistical analysis was performed with Python and/or MATLAB. Kruskal–Wallis tests were applied for group analysis. Post hoc comparisons were evaluated with Tukey–Kramer two-tailed tests with appropriate adjustments for multiple comparisons. In most cases, values were *z*-scored (subtract the mean from each value and divide the result by the s.d.) to make data comparable between experimental sessions and across layers. Reproducibility was tested in several experimental sessions, with the number of replications specified.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

Data used in this study are publicly available. In particular, the training and validation datasets are available at https://figshare.com/projects/cnn-ripple-data/117897 and listed independently as follows: M de la Prida, Liset (2021): Amigo2_2019-07-11_11-57-07. figshare. Dataset. https://doi.org/10.6084/m9.figshare.16847521.v2 M de la Prida, Liset (2021): Som2_2019-07-24_12-01-49. figshare. Dataset. https://doi.org/10.6084/m9.figshare.16856137.v2 M de la Prida, Liset (2021): Dlx1_2021-02-12_12-46-54. figshare. Dataset. https://doi.org/10.6084/m9.figshare.14959449.v4 M de la Prida, Liset (2021): Thy7_2020-11-11_16-05-00. figshare. Dataset. https://doi.org/10.6084/m9.figshare.14960085.v1 Source data in Figs. 2b, 3, 4a–c, e, 5d, e, 6c–f are available https://figshare.com/projects/rippl-IA/191988.

Code availability

Codes for some of the best-trained models of all architectures are available in an open-source repository https://github.com/PridaLab/rippl-AI and documented in open-source notebooks for model retraining https://github.com/PridaLab/rippl-AI/blob/main/examples_retraining.ipynb and for SWR detection https://github.com/PridaLab/rippl-AI/blob/main/examples_detection.ipynb. Additionally, the current version of the code is available at https://zenodo.org/records/10513183.

Received: 2 July 2023; Accepted: 29 January 2024;
Published online: 04 March 2024

References

1. da Silva, F. L. EEG and MEG: relevance to neuroscience. *Neuron* **80**, 1112–1128 (2013).
2. Buzsáki, G. Hippocampal sharp wave-ripple: a cognitive biomarker for episodic memory and planning. *Hippocampus* **25**, 1073–1188 (2015).
3. Csicsvari, J., Hirase, H., Mamiya, A. & Buzsáki, G. Ensemble patterns of hippocampal CA3-CA1 neurons during sharp wave-associated population events. *Neuron* **28**, 585–594 (2000).
4. Stark, E. et al. Pyramidal cell-interneuron interactions underlie hippocampal ripple oscillations. *Neuron* **83**, 467–480 (2014).
5. Buzsáki, G., Leung, L. W. & Vanderwolf, C. H. Cellular bases of hippocampal EEG in the behaving rat. *Brain Res. Rev.* **6**, 139–171 (1983).

6. Kudrimoti, H. S., Barnes, C. A. & McNaughton, B. L. Reactivation of hippocampal cell assemblies: effects of behavioral state, experience, and EEG dynamics. *J. Neurosci.* **19**, 4090–4101 (1999).
7. Genzel, L. et al. A consensus statement: defining terms for reactivation analysis. *Philos. Trans. R. Soc. Lond. B. Biol. Sci*. **375**, 20200001 (2020).
8. Joo, H. R. & Frank, L. M. The hippocampal sharp wave–ripple in memory retrieval for immediate use and consolidation. *Nat. Rev. Neurosci.* **19**, 744–757 (2018).
9. Pfeiffer, B. E. The content of hippocampal 'replay'. *Hippocampus* https://doi.org/10.1002/hipo.22824 (2017).
10. Mil, A. et al. Replay of cortical spiking sequences during human memory retrieval. *Science* **367**, 1128–1130 (2020).
11. Liu, A. A. et al. A consensus statement on detection of hippocampal sharp wave ripples and differentiation from other fast oscillations. *Nat. Commun.* **13**, 1–14 (2022).
12. Reichinnek, S., Künsting, T., Draguhn, A. & Both, M. Field potential signature of distinct multicellular activity patterns in the mouse hippocampus. *J. Neurosci.* **30**, 15441–15449 (2010).
13. Ramirez-Villegas, J. F., Logothetis, N. K. & Besserve, M. Diversity of sharp-wave-ripple LFP signatures reveals differentiated brain-wide dynamical events. *Proc. Natl Acad. Sci. USA* **112**, E6379–E6387 (2015).
14. Sebastian, E. R. et al. Topological analysis reveals input mechanisms behind feature variations of sharp-wave ripples. *Nat. Neurosci.* **26**, 2171–2181 (2023).
15. Patel, J., Schomburg, E. W., Berényi, A., Fujisawa, S. & Buzsáki, G. Local generation and propagation of ripples along the septotemporal axis of the hippocampus. *J. Neurosci.* **33**, 17029–17041 (2013).
16. Leonard, T. K. et al. Sharp wave ripples during visual exploration in the primate hippocampus. *J. Neurosci.* **35**, 14771–14782 (2015).
17. Skaggs, W. E. et al. EEG sharp waves and sparse ensemble unit activity in the macaque hippocampus. *J. Neurophysiol.* **98**, 898–910 (2007).
18. Bragin, A., Engel, J., Wilson, C. L., Fried, I. & Mathern, G. W. Hippocampal and entorhinal cortex high-frequency oscillations (100-500 Hz) in human epileptic brain and in kainic acid-treated rats with chronic seizures. *Epilepsia* **40**, 127–137 (1999).
19. Worrell, G. A. et al. High-frequency oscillations in human temporal lobe: Simultaneous microwire and clinical macroelectrode recordings. *Brain* **131**, 928–937 (2008).
20. Alvarado-Rojas, C. et al. Different mechanisms of ripple-like oscillations in the human epileptic subiculum. *Ann. Neurol.* **77**, 281–290 (2015).
21. Valero, M. et al. Mechanisms for selective single-cell reactivation during offline sharp-wave ripples and their distortion by fast ripples. *Neuron* **94**, 1234-1247.e7 (2017).
22. Cowen, S. L., Gray, D. T., Wiegand, J. P. L., Schimanski, L. A. & Barnes, C. A. Age-associated changes in waking hippocampal sharp-wave ripples. *Hippocampus* **30**, 28–38 (2020).
23. Born, H. A. et al. Genetic suppression of transgenic APP rescues hypersynchronous network activity in a mouse model of Alzeimer's disease. *J. Neurosci.* **34**, 3826–3840 (2014).
24. Engel, J., Bragin, A., Staba, R. & Mody, I. High-frequency oscillations: what is normal and what is not? *Epilepsia* **50**, 598–604 (2009).
25. Sethi, A. & Kemere, C. Real time algorithms for sharp wave ripple detection. *Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.* **2014**, 2637–2640 (2014).
26. Kulkarni, P. M. et al. A deep learning approach for real-time detection of sleep spindles. *J. Neural Eng.* **16**, 36004 (2019).
27. Hagen, E. et al. RippleNet: a recurrent neural network for sharp wave ripple (SPW-R) detection. *Neuroinformatics* **19** (2021).
28. Nadalin, J. K. et al. Application of a convolutional neural network for fully-automated detection of spike ripples in the scalp electroencephalogram. *J. Neurosci. Methods* **360**, 109239 (2021).
29. Valenchon, N. et al. The Portiloop: a deep learning-based open science tool for closed-loop brain stimulation. *PLoS ONE* **17**, e0270696 (2022).
30. Navas-Olive, A., Amaducci, R., Jurado-Parras, M.-T., Sebastian, E. R. & de la Prida, L. M. Deep learning based feature extraction for prediction and interpretation of sharp-wave ripples in the rodent hippocampus. *Elife* **11**, e77772 (2022).
31. Frey, M. et al. Interpreting wide-band neural activity using convolutional neural networks. *Elife* **10**, 66551 (2021).
32. Talakoub, O., Gomez Palacio Schjetnan, A., Valiante, T. A., Popovic, M. R. & Hoffman, K. L. Closed-loop interruption of hippocampal ripples through fornix stimulation in the non-human primate. *Brain Stimul.* **9**, 911–918 (2016).
33. Norman, Y. et al. Hippocampal sharp-wave ripples linked to visual episodic recollection in humans. *Science* **365**, eaax1030 (2019).
34. Geva-Sagiv, M. et al. Augmenting hippocampal-prefrontal neuronal synchrony during sleep enhances memory consolidation in humans. *Nat. Neurosci.* **26**, 1100–1110 (2023).
35. Tong, A. P. S., Vaz, A. P., Wittig, J. H., Inati, S. K. & Zaghloul, K. A. Ripples reflect a spectrum of synchronous spiking activity in human anterior temporal lobe. *Elife* **10**, e68401 (2021).
36. Curot, J. et al. Local neuronal excitation and global inhibition during epileptic fast ripples in humans. *Brain* **146**, 561–575 (2023).
37. Leonard, T. K. & Hoffman, K. L. Sharp-wave ripples in primates are enhanced near remembered visual objects. *Curr. Biol.* **27**, 257–262 (2017).
38. Hussin, A. T., Leonard, T. K. & Hoffman, K. L. Sharp-wave ripple features in macaques depend on behavioral state and cell-type specific firing. *Hippocampus* **30**, 50–59 (2020).
39. Navas-Olive, A., Rubio, A. & de la Prida, L. M. Machine learning toolbox codes for the analysis of sharp-wave ripples (v1.0). *Zenodo* https://doi.org/10.5281/zenodo.10513183 (2024).
40. Berens, P. et al. Community-based benchmarking improves spike rate inference from two-photon calcium imaging data. *PLoS Comput. Biol*. **14**, e1006157 (2018).
41. Kuhlmann, L. et al. Epilepsyecosystem.org: crowd-sourcing reproducible seizure prediction with long-term human intracranial EEG. *Brain* **141**, 2619–2630 (2018).
42. Wheeler, D. W. et al. Hippocampome.org: a knowledge base of neuron types in the rodent hippocampus. *Elife* **4**, e09960 (2015).
43. de la Prida, L. M. & Ascoli, G. A. Explorers of the cells: Toward cross-platform knowledge integration to evaluate neuronal function. *Neuron* **109**, 3535–3537 (2021).
44. Chen, T. & Guestrin, C. XGBoost: A Scalable Tree Boosting System. *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min*. **13–17-Aug**, 785–794 (2016).
45. Schmidhuber, J. Deep learning in neural networks: an overview. *Neural Netw.* **61**, 85–117 (2015).
46. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **9**, 1735–1780 (1997).
47. Friedman, J. H. Greedy function approximation: a gradient boosting machine on JSTOR. *Ann. Stat.* **29**, 1189–1232 (2001).
48. Cortes, C., Vapnik, V. & Saitta, L. Support-vector networks. *Mach. Learn.* **20**, 273–297 (1995).
49. Cun, L. et al. Handwritten digit recognition with a back-propagation network. *Adv. Neural Inf. Process. Syst.* **2**, 396–404 (1990).
50. Graves, A. & Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Netw.* **18**, 602–610 (2005).
51. de la Prida, L. M. et al. Threshold behavior in the initiation of hippocampal population bursts. *Neuron* **49**, 131–142 (2006).
52. de la Prida, L. M. Potential factors influencing replay across CA1 during sharp-wave ripples. *Philos. Trans. R. Soc. B* **375**, 20190236 (2020).

53. McInnes, L., Healy, J. & Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. (2018).
54. Abbaspoor, S., Hussin, A. T. & Hoffman, K. L. Theta- and gamma-band oscillatory uncoupling in the macaque hippocampus. *Elife* **12**, e86548 (2023).
55. Dutta, S., Ackermann, E. & Kemere, C. Analysis of an open source, closed-loop, realtime system for hippocampal sharp-wave ripple disruption. *J. Neural Eng*. **16**, 016009 (2019).
56. Blanco, J. A. et al. Unsupervised classification of high-frequency oscillations in human neocortical epilepsy and control patients. *J. Neurophysiol.* **104**, 2900–2912 (2010).
57. Kucewicz, M. T. et al. High frequency oscillations are associated with cognitive processing in human recognition memory. *Brain* **137**, 2231–2244 (2014).
58. Ibarz, J. M., Foffani, G., Cid, E. & Inostroza, M. & Menendez de la Prida, L. Emergent dynamics of fast ripples in the epileptic hippocampus. *J. Neurosci.* **30**, 16249–16261 (2010).
59. Menendez De La Prida, L., Staba, R. J. & Dian, J. A. Conundrums of high-frequency oscillations (80-800 Hz) in the epileptic brain. *J. Clin. Neurophysiol*. **32**, 207–219 (2015).
60. Liu, X. et al. E-Cannula reveals anatomical diversity in sharp-wave ripples as a driver for the recruitment of distinct hippocampal assemblies. *Cell Rep*. **41** (2022).

Acknowledgements

This work was supported by the Fundación La Caixa (LCF/PR/HR21/52410030 and LCF/PR/HR22/52420005) to L.M.P. and by the Whitehall Foundation and BRAIN Initiative NINDS (R01NS127128) to K.L.H. We thank the Spanish Society of Neuroscience (SENC) and the Universidad Autónoma de Madrid Doctorate for partially supporting the hackathon. A.N.O. was supported by PhD fellowships from the Spanish Ministry of Education (FPU17/03268). AR was supported by a JAE-Intro Fellowship of the AI-HUB CSIC program (JAE Intro AI HUB21) and by the CSIC Interdisciplinary Thematic Platform Neuro-Aging (PTI+Neuro-Aging). We thank all participants of the hackathon. Thanks to Rodrigo Amaducci, Enrique R Sebastian, Daniel García-Rincón, and Adrián Gollerizo for co-organizing the hackathon.

Author contributions

A.N.O. and L.M.P. designed the project. A.N.O. and A.R. developed the toolbox and performed the analysis. S.A. and K.L.H. designed and performed macaque experiments. A.N.O. and L.M.P. wrote the paper. All the authors have read and agreed to the published version of the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42003-024-05871-w.

Correspondence and requests for materials should be addressed to Andrea Navas-Olive or Liset M. de la Prida.

Peer review information *Communications Biology* thanks Yusuke Watanabe, Antonio Fernandez-Ruiz and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Primary Handling Editors: Joao Valente.