Spikebench: an open benchmark for spike train time-series classification

Ivan Lazarevich[1*], Ilya Prokin, Boris Gutkin[1,2◕], Victor Kazantsev[3◕],

1 École Normale Supérieure, Laboratoire de Neurosciences Cognitives, Group for Neural Theory, Paris, France
2 Center for Cognition and Decision Making, National Research University Higher School of Economics, Moscow, Russia
3 Lobachevsky State University, Nizhny Novgorod, Russia

◕These authors contributed equally to this work.
* ivan@lazarevi.ch

Abstract

Modern well-performing approaches to neural decoding are based on machine learning models such as decision tree ensembles and deep neural networks. The wide range of algorithms that can be utilized to learn from neural spike trains, which are essentially time-series data, results in the need for diverse and challenging benchmarks for neural decoding, similar to the ones in the fields of computer vision and natural language processing. In this work, we propose a spike train classification benchmark, based on open-access neural activity datasets and consisting of several learning tasks such as stimulus type classification, animal's behavioral state prediction and neuron type identification. We demonstrate that an approach based on hand-crafted time-series feature engineering establishes a strong baseline performing on par with state-of-the-art deep learning based models for neural decoding. We release the code allowing to reproduce the reported results.

Author summary

Machine learning-based neural decoding has been shown to outperform the traditional approaches like Wiener and Kalman filters on certain key tasks [1]. To further the advancement of neural decoding models, such as improvements in deep neural network architectures and better feature engineering for classical ML models, there need to exist common evaluation benchmarks similar to the ones in the fields of computer vision or natural language processing. In this work, we propose a benchmark consisting of several *individual neuron* spike train classification tasks based on open-access data from a range of animals and brain regions. We demonstrate that it is possible to achieve meaningful results in such a challenging benchmark using the massive time-series feature extraction approach, which is found to perform similarly to state-of-the-art deep learning approaches.

Introduction

The latest advances in multi-neuronal recording technologies such as two-photon calcium imaging [2], extracellular recordings with multi-electrode arrays [3], Neuropixels probes [4] allow producing large-scale single-neuron resolution brain activity data with remarkable magnitude and precision. Some of the neural spiking data recorded in animals has been released to the public in the scope of data repositories such as CRCNS.org [5]. In addition to increasing experimental data access, various neural data analysis tools have been developed, in particular for the task of neural decoding, which is often posed as a supervised learning problem [1]: given firing activity of a population of neurons at each time point, one has to predict the value of a certain quantity pertaining to animal's behaviour such as its velocity at a given point in time.

Such a formulation of the neural decoding task implies that it is a multivariate time-series regression or classification problem. An array of supervised learning methods focused specifically on general time-series data has been developed over the years, ranging from classical approaches [6] to deep neural networks for sequential data [7]. It is not fully clear, however, how useful these methods are for the specific tasks of learning from neural spiking data. In order to establish a sensible ranking of these algorithms for neural decoding, there is a need for a common spiking activity recognition benchmark. In this work, we propose a diverse and challenging spike train classification benchmark based on several open-access neuronal activity datasets. This benchmark incorporates firing activity from different brain regions of different animals

(retina, prefrontal cortex, motor and visual cortices) and comprises distinct task types such as visual stimulus type classification, animal's behavioral state prediction from individual spike trains and interneuron subtype recognition from firing patterns. All of these tasks are formulated as univariate time-series classification problems, that is, one needs to predict the target category based on an individual spike train chunk recorded from a single neuron. The formulation of the classification problems implies that the predicted category is stationary across the duration of the given spike train sample.

Our main contributions can be summarized as follows:

- We propose a diverse spike train classification benchmark based on open-access data.
- We show that global information such as the animal's behavioral state or stimulus type can be decoded (with high accuracy) from *single-neuron* spike trains containing several tens of interspike intervals.
- We establish a strong baseline for spike train classification based on hand-crafted time-series feature engineering that performs on par with state-of-the-art with deep learning models.

Well-established machine learning techniques such as gradient boosted decision tree ensembles and recurrent neural networks have been successfully applied both to neural activity decoding (predicting stimuli/action from spiking activity) [1, 8] as well as neural encoding (predicting neural activity from stimuli) [9]. Neural decoding tasks are often formulated as regression problems, wherein binned spiking count time-series of a single fixed neural population are used to predict the animal's position or velocity in time.

A number of previous studies on feature vector representations of spike trains also focused on defining a spike train distance metric [10] for identification of neuronal assemblies [11]. Several different definitions of the spike train distance exist such as van Rossum distance [12], Victor-Purpura distance [13], SPIKE- and ISI- synchronization distances [14] (for a thorough list of existing spike train distance metrics see [10]). These distance metrics were used to perform spike train clustering and classification based on the k-Nearest-Neighbors approach [15]. Jouty et al. [16] employed ISI and SPIKE distance measures to perform clustering of retinal ganglion cells based on their firing responses to a given stimulus.

In addition to characterization with spike train distance metrics, some previous works relied on certain statistics of spike trains to differentiate between cell types. Charlesworth et al. [17] calculated basic statistics of multi-neuronal activity from cortical and hippocampal cultures and were able to perform clustering and classification of activity between these culture types. Li et al. [18] used two general features of the interspike interval (ISI) distribution to perform clustering analysis to identify neuron subtypes. Such approaches represent neural activity (single or multi-neuron spiking patterns) in a low-dimensional feature space where the hand-crafted features are defined to address specific problems and might not provide an optimal feature representation of spiking activity data for a general decoding problem. Finally, not only spike timing information can be used to characterize neurons in a supervised classification task. Jia et al. [19] used waveform features of extracellularly recorded action potentials to classify them by brain region of origin.

The aforementioned works were aimed at, to some extent or another, trying to decode properties of neurons or stimuli given recorded spiking data. In some of the cases the datasets used were not released to be openly available, in some of the cases the predictive models used constituted quite simple baselines for the underlying decoding/cell identification tasks. In this work, we aim to propose a benchmark base on open-access datasets that is diverse and challenging enough to robustly demonstrate

gains of advanced time-series machine learning approaches as compared to some of the simple baselines used in previous works. We release the code allowing to reproduce the reported results.

Materials and methods

Overview of time series classification methods

We applied general time series feature representation methods [6] for classification of neuronal spike train data. Most approaches in time series classification are focused on transforming the raw time series data into an effective feature space representation before training and applying a machine learning classification model. Here we give a brief overview of state-of-the-art approaches one could utilize in order to transform time series data into a feature vector representation for efficient neural activity classification.

Neighbor-based models with time series distance measures

A strong baseline algorithm for time series classification is k-nearest-neighbors (kNN) with a suitable time series distance metric such as the Dynamic Time Warping (DTW) distance or the edit distance (ED) [6]. In this work, we evaluated performance of nearest-neighbor models for generic distance measures such as l_p and DTW distance, converting spike trains to the interspike-interval (ISI) time-series representation prior to calculating the spike-train distances. Some of the distance metrics we also used for evaluation are essentially distribution similarity measures (e.g. Kolmogorov-Smirnov distance, Earth mover's distance) which allow comparing ISI value distributions within spike trains. Such a spike train distance definition would only use the information about the ISI distribution in the spike train, but not about its temporal structure. Alternatively, one can keep the original event-based representation of the spike train and compute the spike train similarity metrics such as van Rossum or Victor-Purpura distances or ISI/SPIKE distances [10].

The choice of the distance metric determines which features of the time series are considered as important. Instead of defining a complex distance metric, one can explicitly transform time series into a feature space by calculating various properties of the series that might be important (e.g. mean, variance). After assigning appropriate weights to each feature one can use kNN with any standard distance metric. Moreover, such a representation allows the application of any state-of-the-art machine learning classification algorithm beyond kNN to obtain better classification results. In the following, we discuss approaches using various feature space representations available for time series data.

Models using hand-crafted time series features

One of the useful and intuitive approaches in time series classification is focused on manually calculating a set of descriptive features for each time series (e.g. their basic statistics, spectral properties, other measures used in signal processing and so on) and using these feature sets as vectors describing each sample series. There exist approaches which enable automated calculation of a large number of time series features which may be typically considered in different application domains. Such approaches include automated time series phenotyping implemented in the *hctsa* MATLAB package [20] and automated feature extraction in the *tsfresh* Python package [21]. Here we utilize the *tsfresh* package which enables calculation of 779 descriptive time series features for each spike train, ranging from Fourier and wavelet expansion coefficients to coefficients of a fitted autoregressive process.

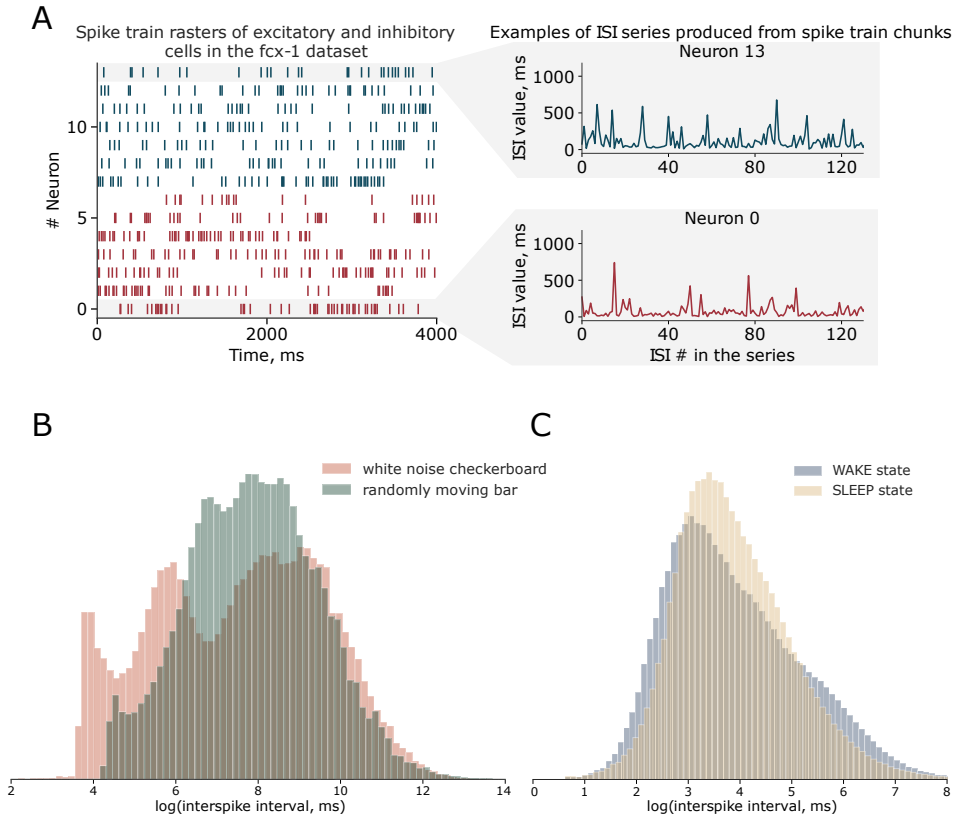

Fig 1. (A) Examples of spiking activity recordings in the CRCNS fcx-1 dataset in the WAKE state. Left: spike train raster of a random subset of excitatory cells (red) and inhibitory cells (blue). Right: examples of ISI series produced from spike train chunks of inhibitory/excitatory cells in the fcx-1 dataset. (B) Interspike interval value distribution histograms generated from the aggregated spike trains of retinal ganglion cells in response to a "white noise checkerboard" visual stimulus (blue) and a "randomly moving bar" stimulus (red). (C) Interspike interval value distribution histograms generated from the aggregated PFC spike trains (fcx-1 dataset) corresponding to the WAKE (blue) or SLEEP (red) state of the rat.

Once each time series (spike train) is represented as a feature vector, the spiking activity dataset has the standard form of a matrix with size $[n_{\text{samples}}, n_{\text{features}}]$ rather than the raw dataset with shape $[n_{\text{samples}}, n_{\text{timestamps}}]$. This standardized dataset can be then used as an input to any machine learning algorithm such as logistic regression or gradient boosted trees [22]. We found this approach to set a strong baseline for all of the classification tasks we considered.

Deep learning models

Lastly, there are deep learning based approaches working well for time-series classification [7] such as deep recurrent networks like LSTMs and GRUs [23] and 1D convolutional neural networks (1D-CNNs) [24, 25]. While rather generic model architectures have been typically applied to neural decoding tasks [1, 8], there exist models specifically designed for time-series classification and regression tasks, like InceptionTime [25], achieving state-of-the-art results on benchmarks like the UCR Time Series Classification Archive [26]. Recent developments in deep learning model for time series also include the Time Series Transformer [27] and convolutional architectures like the Omniscale-CNN [28]. Perhaps surprisingly, we found that deep learning models could not significantly outperform the baseline with hand-crafted time series features on the spike train classification tasks, oftentimes performing worse than the baseline.

The proposed spike train classification benchmark.

We propose a spike train classification benchmark comprising several different open-access datasets and distinct classification tasks. The datasets used for the benchmark are as follows:

- **Retinal ganglion cell stimulus type classification** based on the published dataset [29, 30]: Spike time data from multi-electrode array recordings of salamander retinal ganglion cells under four stimulus conditions: a white noise checkerboard, a repeated natural movie, a non-repeated natural movie, and a bar exhibiting random one-dimensional motion. We define the 4-class classification task to predict the stimulus type given the spike train chunk, also considering binary classification tasks for pairs of stimuli types (e.g. "white noise checkerboard" vs. "randomly moving bar").

- **WAKE/SLEEP classification** based on fcx-1 dataset [31, 32] from CRCNS.org [5]: Spiking activity and Local-Field Potential (LFP) signals recorded extracellularly from frontal cortices of male Long Evans rats during wake and sleep states without any particular behavior, task or stimulus. Around 1100 units (neurons) were recorded, 120 of which are putative inhibitory cells and the rest is putative excitatory cells. Figure 1 shows several examples of spiking activity recordings that can be extracted from the fcx-1 dataset. The authors classified cells into an inhibitory or excitatory class based on the action potential waveform (action potential width and peak time). Sleep states (SLEEP activity class) were labelled semi-automatically based on extracted LFP and electromyogram features, and the non-sleep state was labelled as the WAKE activity class. We define the binary classification task as the prediction of WAKE or SLEEP animal state given a spike train chunk recorded from a putative excitatory cell.

- **Interneuron subtype classification task** based on the Allen Cell Types dataset [33]: Whole cell patch clamp recordings of membrane potential in neurons of different types. We selected the PV, VIP and SST interneurons from the whole dataset, as these interneuron groups comprise the majority of inhibitory cells in

the prefrontal cortex [34]. We selected the spike trains recorded under the naturalistic noise stimulation protocol (as a proxy for the *in vivo* spontaneous activity in these cells). The non-trivial prediction task is defined for VIP vs. SST spike train classification, since the PV interneuron spike trains can be easily distinguished from the other interneuron types. The latter is due to a significantly higher firing frequency in PV interneurons that we found in the Allen Cell Types dataset.

- **Unsupervised temporal structure recognition task.** We defined a set of spike train classification tasks constructed in a self-supervised manner [35]. In such tasks, we take any set of (unlabelled) neuronal spike train recordings and generate a additional set of spike trains by applying a given transformation to the original data. The target classification task is to determine whether a given spike train chunk belongs to the original dataset or to the transformed one. Note that this task can be constructed for any spiking dataset without the need for the ground truth labels, i.e. in an unsupervised way. The spike train transformations we consider here are (i) adding spike timing jitter via, in particular, timing noise following a truncated normal distribution, (ii) random shuffling of the interspike intervals in the spike train, (iii) reversing the spike train. The models trained in such tasks learn to detect the temporal structure of the original spike trains, since the order/precise values of interspike intervals have been disrupted by the transformation (e.g. by ISI shuffling), while the ISI value distribution is preserved by some of the transformations (e.g. by the shuffling and reversal operations). The final trained model accuracy in a shuffled vs. non-shuffled spike train classification task can thus be thought of as a measure of temporal structure in the original spiking dataset (test set accuracy would be on the chance level if the ISI values in the original spike trains were independently sampled from a fixed value distribution, i.e. the exact ordering of the ISIs did not contain any predictive information). We consider the the fcx-1 and retinal ganglion cell datasets described above to construct the temporal structure recognition tasks.

Validation scheme and data preprocessing

Suppose we are given a dataset containing data from several animals each recorded multiple times with a large number of neurons captured in each recording. For each recorded neuron, we have a corresponding spike train captured over a certain period of time (assuming that the preprocessing steps like spike sorting or spiking time inference from fluorescence traces were performed beforehand). The number of spikes within each spike-train is going to be variable. A natural way to standardize the length of spike-train sequences would be dividing the full spike train into chunks of N spike times, where N is fixed for each chunk. We produce these spike train chunks for all datasets by moving a fixed-size sliding window across each single-neuron spike train. Hence, the pre-processing pipeline for all dataset is as follows:

- Encode all of the spike trains in the interspike interval format (i.e. time-series $[ISI_1, ISI_2, ISI_3, ...]$)
- Apply a rolling window of size N with step S to each single-neuron spike train separately, producing several spike train chunks (of N interspike intervals) for each neuron.
- Pool all the chunks from all neurons together to form a data matrix of size $[M, N]$ where M is the total number of samples (chunks) in the training/test dataset and N is the chunk size.

- (Optional) Apply a logarithmic transform to each sample (due to the originally heavy-tail distribution of ISI values in the data) and standard scaling to the dataset. Alternatively, encode the spike train chunks using a binned spike count representation.

We train a decoder (classifier) using the training set data matrix, so a single decoder is trained for all the neurons in the training set from single-neuron spike train chunks.

The validation strategy we use in this work is based on group splits, which means we determine the split into the training and the test datasets based on animal identifiers available in the original data. The motivation is that, in cases recordings are performed in several animals and corresponding animal identifiers are available, the set of animals used to construct the training dataset and the set of animals for the test dataset should not overlap in order to test whether the trained decoding models could generalize across different animals. In case animal identifiers are not available, we split the dataset into training and testing based on non-overlapping neuron identifiers in train and test.

Most of the datasets in the benchmark have an imbalanced class distribution. For performance evaluation, we consider two different setups in this work: (i) keeping all the available data in the training/test datasets, and measuring performance with metrics that are robust to class imbalance, such as the Cohen's kappa score [36], and the geometric mean score [37] and (ii) balancing the class distribution in the training and testing datasets by data undersampling and measuring standard classification metrics such as accuracy and AUC-ROC. The second approach allows to rank models without the effect of class imbalance on training and evaluation [38], however some data points are being lost in the undersampling process. Although AUC-ROC is generally considered to be a performance metric less affected by the class imbalance than e.g. accuracy, we do not report AUC-ROC values for the full (imbalanced) data setting because of its possible skeweness [39].

The base task we consider for all benchmark datasets is classification given an individual spike train chunk (part of a single-neuron recording). However, prediction performance can be improved by aggregating predictions from spike trains of several neurons (in case such data is available) or from several chunks of a large single-neuron spike train. If the final classification in such a setting is done by majority voting from all single spike-train chunk predictions and we assume that recorded spike-trains chunks are randomly sampled from the whole set, the optimistic estimate for accuracy improvement with the number of spike train chunks N_{chunks} would be

$$\mu = \sum_{i=m}^{N_{\text{chunks}}} C^{i}_{N_{\text{chunks}}} p^{i} (1-p)^{N_{\text{chunks}}-i} \tag{1}$$

where μ is the probability that the majority vote prediction is correct, p is the probability of a single classifier prediction being correct (single spike train prediction accuracy), N_{chunks} is the number of predictions made, $m = [N_{chunks}/2] + 1$ is the minimal majority of votes. We found that the empirical values of accuracy improvement are close to the optimistic analytical estimate (1) in both cases when the spike train chunks are sampled from different neurons and from a large spike train of a single neuron (see Fig. S1).

Results

Visual stimulus type classification from retinal spike trains

We first start with looking the at the retinal ganglion cell spike train classification task. Recorded spike trains in the dataset are associated with one of the four categories

	Retina dataset (Cohen's kappa score)	Retina dataset (geometric mean score)
kNN, $k = 1$, l_1	0.31796	0.53368
kNN, $k = 5$, l_1	0.28188	0.470469
kNN, $k = 5$, l_2	0.043361	0.232801
kNN, $k = 1$, DTW ($r = 50$)	0.47593	0.76079
kNN, $k = 1$, Victor-Purpura	-0.018385	0.451142
kNN, $k = 1$, ISI	0.24427	0.43871
kNN, $k = 1$, KS	**0.546388**	**0.831646**

Table 1. Spike train classification results for the retinal neuron activity dataset for nearest-neighbor models with different distance metrics. The task is defined as binary classification of the stimulus type ("white noise checkerboard" or "randomly moving bar").

corresponding to different visual stimulus types, labelled with "white noise checkerboard", "randomly moving bar", "repeated natural movie" and "unique natural movie". The classification task is, given a chunk of the spike train recording, predict the corresponding stimulus type category. The number of neurons in the dataset belonging to each category is 155, 140, 178, 152, respectively. The number of interspike intervals is quite variable among individual cells (due to firing rate variability) ranging from 100 ISIs per recording to as much as 60000 ISIs per recording.

We focus on the binary classification task aimed at predicting one of the two types of stimuli: "randomly moving bar" (corresponding to a label of 0) or "white noise checkerboard" (corresponding to a label of 1). We select recorded spike trains corresponding to those stimuli types and split 70% of recorded neurons (108 neurons) for the training part of the dataset and the remaining 30% (46 neurons) for the testing dataset. We encode spike trains using the ISI representation and apply a rolling window of size equal to 200 ISIs with a stride (step) of 100 ISIs to each recorded neuron. This results in 8006 training samples and 3650 testing samples, each sample containing 200 ISIs. The average target value in the training set is 0.7660 and 0.7882 in the testing set, hence class imbalance is present in the retinal dataset.

Nearest-neighbor models for spike train classification

We evaluated performance of nearest-neighbor models with different distance metrics on the retinal stimulus classification task, results are shown in Table 1. The results in presented in Table 1 were obtained without performing undersampling to balance the class distribution, hence we looked at imbalance-robust metrics such as Cohen's kappa and geometric mean scores. We found that the nearest neighbor model with the DTW distance is amongst the best performing ones, but is still outperformed by the 1-NN model with the Kolmogorov-Smirnov (KS) distance, suggesting that differences in ISI distributions contain significant discriminative information helpful for the classification task at hand. We further include the results obtained with the 1-NN KS-distance model as a baseline to compare against other methods.

Hand-crafted feature extraction + classification models

The kNN results clearly suggest that characteristics of the interspike-interval distribution of the given spike train are predictive of the category label in our classification task. At the same time, one would expect the the temporal (sequential) information contained in the spike train also has certain predictive power. A straightforward way to incorporate both types of features in the model is to build a corresponding vector embedding of the spike train time-series. An efficient way to do so is to use a set of hand-crafted time-series features, like for example the set of 779

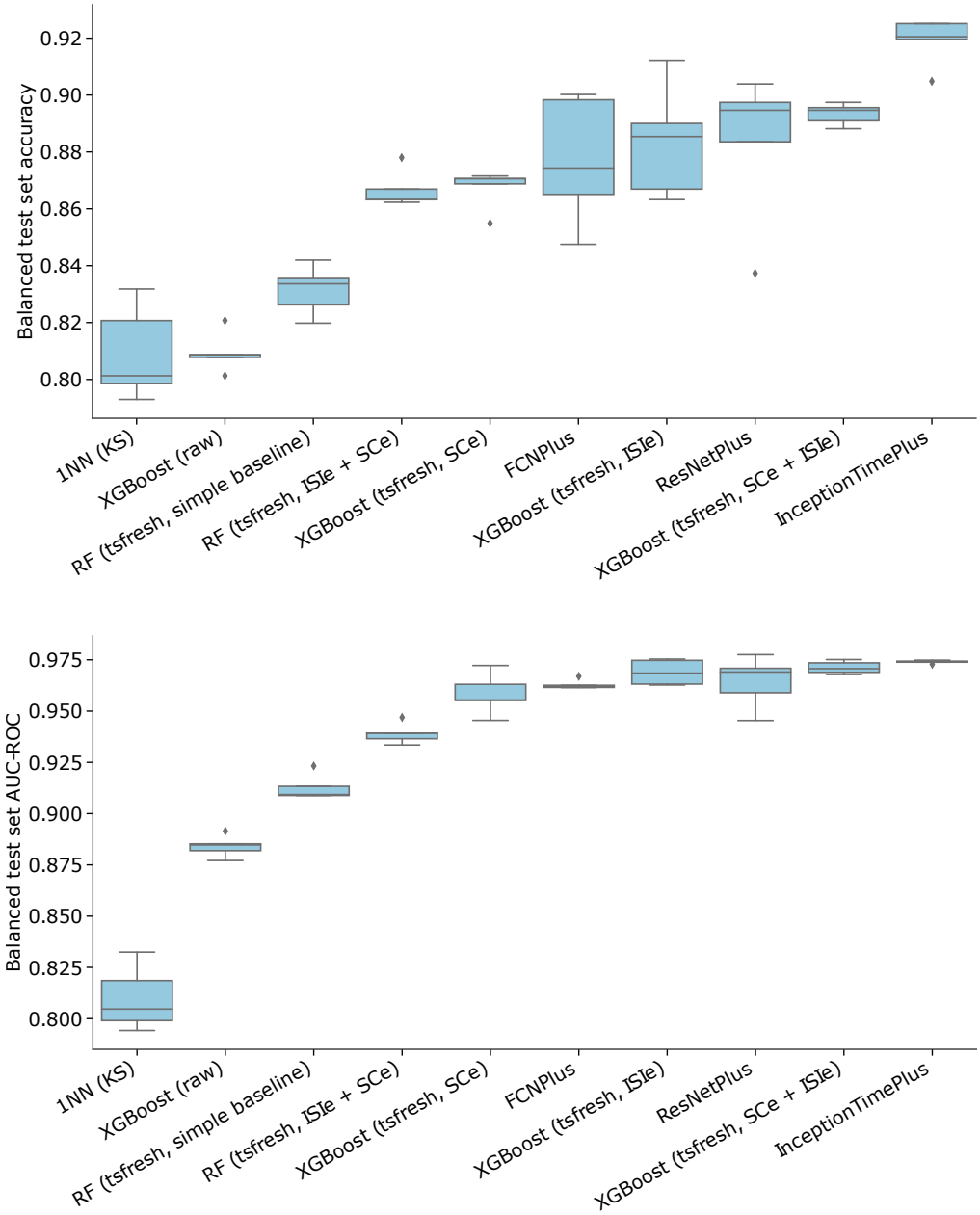

Fig 2. Spike train classification metric values (top panel - accuracy, bottom panel - AUC-ROC) for the retinal neuron activity dataset on a range of models. The task is defined as binary classification of the stimulus type ("white noise checkerboard" or "randomly moving bar"), with the test set balanced in class distribution by undersampling (that is, accuracy=0.5 corresponds to chance level). Models are ranked in ascending order of the median metric value. The "simple baseline" model tag corresponds to spike trains encoded with 6 basic distribution statistics, the "raw" tag implies that the model has been directly trained on ISI time-series data without feature extraction. The "tsfresh" tag corresponds to encoding with the full set of time-series features. "ISIe" stands for interspike-interval encoding of the spike train, "SCe" stands for spike-count encoding. "ISIe + SPe" means that feature vectors corresponding to both types of encoding are concatenated. InceptionTimePlus, FCNPlus and ResNetPlus refer to implementations in the PyTorch-based *tsai* package.

	Cohen's kappa score	Geometric mean score
InceptionTimePlus	**0.827437**	0.905105
ResNetPlus	0.772328	0.894081
FCNPlus	0.711119	0.885681
XceptionTimePlus	0.795861	**0.927107**
XGBoost (tsfresh, ISIe + SCe)	0.779920	0.908246
Random Forest (tsfresh, ISIe + SCe)	0.701796	0.875086
XGBoost (tsfresh, ISIe)	0.752612	0.900478
Random Forest (tsfresh, ISIe)	0.729350	0.885312
XGBoost (tsfresh, SCe)	0.682994	0.830571
Random Forest (tsfresh, SCe)	0.609953	0.772864
XGBoost (tsfresh, simple baseline)	0.540849	0.828692
Random Forest (tsfresh, simple baseline)	0.515981	0.816995
XGBoost (raw)	0.542582	0.784036
1NN (KS distance)	0.546388	0.831647

Table 2. Spike train classification metric values (for imbalance-robust metrics) for the retinal neuron activity dataset on a range of models. The "simple baseline" model tag corresponds to spike trains encoded with 6 basic distribution statistics, the "raw" tag implies that the model has been directly trained on ISI time-series data without feature extraction. The "tsfresh" tag corresponds to encoding with the full set of time-series features. "ISIe" stands for interspike-interval encoding of the spike train, "SCe" stands for spike-count encoding. "ISIe + SPe" means that feature vectors corresponding to both types of encoding are concatenated. InceptionTimePlus, FCNPlus, ResNetPlus and XceptionTimePlus and refer to implementations in the PyTorch-based *tsai* package.

	Retinal stimulus (white noise vs. moving bar)	fcx-1 WAKE/SLEEP state prediction	Allen Cell Types SST/VIP INs
InceptionTimePlus	**92.0518 ± 0.835891**	76.8249 ± 1.066148	75.9843 ± 1.760683
ResNetPlus	89.4640 ± 2.675756	77.0132 ± 0.785901	77.1654 ± 2.605574
FCNPlus	87.4307 ± 2.243006	**78.0172 ± 0.576124**	**77.9528 ± 2.847194**
XceptionTimePlus	86.4140 ± 3.126660	77.2223 ± 0.826571	**77.9528 ± 1.791230**
XGBoost (tsfresh, ISIe + SCe)	89.4640 ± 0.373135	75.4863 ± 0.214632	68.5039 ± 3.443476
Random Forest (tsfresh, ISIe + SCe)	86.3216 ± 0.654824	76.4903 ± 0.291478	66.9291 ± 1.825515
XGBoost (tsfresh, ISIe)	88.5397 ± 1.972500	76.0301 ± 0.725762	69.6850 ± 4.909435
Random Forest (tsfresh, ISIe)	86.1368 ± 0.766597	76.6158 ± 0.462620	66.1417 ± 6.628921
XGBoost (tsfresh, SCe)	87.0610 ± 0.699600	73.7921 ± 0.571588	61.4173 ± 2.335811
Random Forest (tsfresh, SCe)	85.8595 ± 0.801460	75.2144 ± 0.459060	64.9606 ± 0.980307
Random Forest (tsfresh, simple baseline)	83.3641 ± 0.858575	74.0013 ± 0.355759	63.3858 ± 3.450221
XGBoost (tsfresh, simple baseline)	82.9945 ± 1.312580	71.3867 ± 0.602334	59.4488 ± 3.156980
XGBoost (raw)	80.8688 ± 0.702038	71.9724 ± 0.323558	63.7795 ± 2.602597
1NN (KS distance)	80.1294 ± 1.644998	67.3081 ± 1.439905	61.0236 ± 2.401252

Table 3. Spike train classification accuracy values for different datasets on a range of models. The reported accuracy is measured on balanced test sets to mitigate class imbalance, median value and standard deviation in percent is shown. Model names correspond to the same ones from Table 2.

features provided in the *tsfresh* Python package. In order to compute vector embeddings for the spike trains in the training and testing datasets, one has to convert spike times into a time-series, which in principle could be done using either an interspike-interval encoding (the time-series is the sequence of ISIs) or a spike-count encoding (time is binned and spike counts in each time bin comprise the time series). The latter type of encoding depends on an additional hyperparameter which is the size

	Interspike interval encoding	Spike count encoding	Interspike interval + spike count encoding
Logistic Regression	**84.1035 ± 1.5484**	75.0462 ± 1.1661	84.0111 ± 0.3543
Random Forest	86.1368 ± 0.7666	85.8595 ± 0.8015	**86.3216 ± 0.6548**
Extra Trees	85.5823 ± 0.3556	86.1368 ± 1.0187	**86.2292 ± 1.0052**
XGBoost	87.0610 ± 1.9725	88.5397 ± 0.6996	**89.464 ± 0.3731**

Table 4. Balanced test set accuracy values on the retina dataset with different classifiers trained on tsfresh feature representations obtained from the (i) interspike interval encoding of the spike trains, (ii) the spike count encoding of the spike trains, (iii) combined interspike interval + spike count encoding.

of the time bin while ISI-encoding is parameter-free. We tested both types of spike-train encoding for our task and observed that models trained using the ISI-encoding of spikes generally perform better than the ones using binned spike counts. Furthermore, we found that combining features corresponding to both encoding types leads to better performance compared to using a single encoding scheme (see Table 4 and Figure 2).

For each spike-train encoding type, we computed the 779-dimensional *tsfresh* time-series embeddings independently for each sample in the training and testing datasets (no statistic aggregation across samples is performed). We then performed simple pre-processing steps by (i) removing low-variance features from the embedding (features f satisfying $\mathrm{std}(f)/(\mathrm{mean}(f)+\varepsilon) < \theta$ with $\theta = 0.2$ and $\varepsilon = 10^{-9}$ were removed) and (ii) performing standard scaling for each feature using mean and variance statistics collected over the training dataset. Note that since the number of spikes in each data sample is fixed and interspike intervals are highly variable, the number of time bins also becomes variable from sample to sample. The fixed-size *tsfresh* embeddings can nevertheless be computed since they are applicable to variable-length time series.

We then trained classification models on the resulting spike-train vector embeddings. We chose a representative set of classification models comprising (i) a linear model, namely logistic regression with an l_2 regularization penalty and (ii) several types of tree-based ensembles: a random forest classifier (via the scikit-learn's RandomForestClassifier implementation), randomized decision trees (via the scikit-learn's ExtraTreesClassifier), and a gradient boosted decision tree (GBDT) ensemble (via the XGBoost implementation). The classifier hyperpameter values we used are specified in the Supplementary Materials. We have not included validation metric values of the logistic regression models in figures and tables of the main text because we found that linear models always perform worse compared to decision tree ensembles; results for linear models can be found in Supplementary Materials.

Classification results obtained with the described *tsfresh*-based approach are presented in Figure 2 and Tables 2, 3, 4. Note that in cases where we report balanced test set metrics (accuracy and AUC-ROC) we have performed class balancing via undersampling on the dataset and also have randomly sampled 70% of the training set data over several trials to estimate the variance in validation metrics. In cases when we don't perform any undersampling, we report the values of Cohen's kappa and geometric mean score metrics.

We were able to reach significant performance levels (> 0.88 accuracy, > 0.95 AUC-ROC) with our best *tsfresh*-based models on the binary retinal stimulus classification task ("randomly moving bar" vs. "white noise checkerboard"). To make better sense of these metric values, we compared our *tsfresh*-based models against two simple baselines: (a) a logistic regression model on ISI-encoded spike-trains represented by 6 basic statistical features – the mean, median, minimum and maximum ISI values, the standard deviation and the absolute energy of the ISI-sequence (the mean of squared ISI values) and (ii) an XGBoost model trained directly on "raw" ISI-encoded

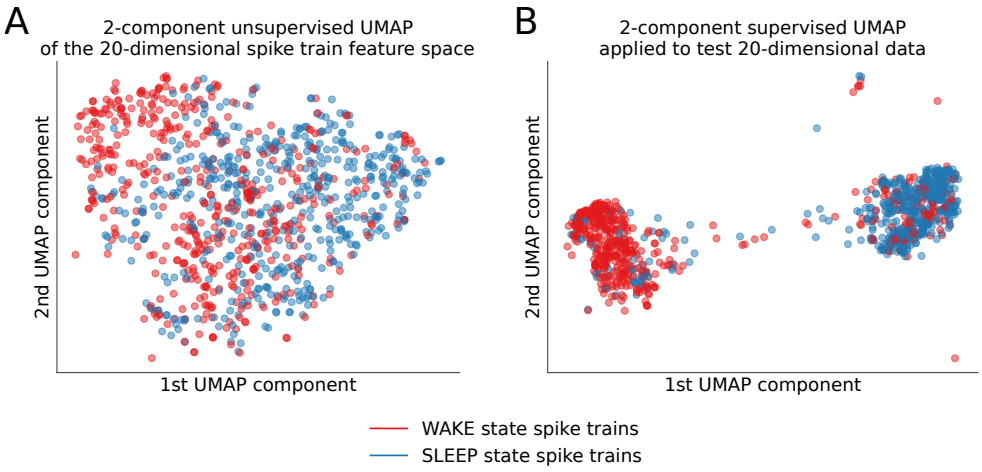

Fig 3. Spike train feature embeddings for WAKE (points marked red) vs. SLEEP (points marked blue) activity states of the neural circuit. Two-dimensional embeddings of the (20-dimensional) selected-*tsfresh*-feature space using (A) unsupervised UMAP and (B) supervised UMAP embedding algorithms for spike trains corresponding to WAKE vs. SLEEP activity states.

spike trains.

The best performing model using the 6 basic features of the ISI time-series got a median balanced test set accuracy of 83.36, while training an XGBoost model directly on the ISI time-series gave an accuracy of 80.86. Using full *tsfresh* embeddings on ISI time-series boost the accuracy to 88.53 with the best model (XGBoost). We generally found that using the binned spike count encoding of spike trains works worse with *tsfresh* compared to the ISI encoding (see Table 4), while combining feature vectors obtained from both encodings results in improved accuracy (89.46 for the XGBoost model on the retina dataset).

We also evaluated the performance of state-of-the-art deep learning models in our retinal stimulus classification tasks, using implementations from the *tsai* package [40]. Accuracy results for a range of convolutional architectures (FCN, InceptionTime, XceptionTime, ResNet) are shown in Tables 2, 3. We found that convolutional neural nets generally outperform manual-feature-based methods, although by a relatively small margin both in cases of class-balanced and imbalanced datasets.

We then performed the same pre-processing steps for the WAKE/SLEEP and VIP/SST datasets as for the retinal stimulus classification dataset. The rolling window of size equal to 200 ISIs and a stride of 100 ISIs for the WAKE/SLEEP data produced a dataset of 24363 training samples (from 78 neurons) and 10634 testing samples (from 35 neurons) with average target values of 0.3796 and 0.3211 in the training and testing sets, correspondingly. A rolling window of size equal to 50 ISIs and a stride of 20 ISIs for VIP/SST interneuron data produced a dataset of 2690 training samples and 1217 testing samples, with mean target values of 0.6026 and 0.8504, correspondingly.

We observed similar trends both for the WAKE/SLEEP state and VIP/SST interneuron classification tasks, shown in Table 3. The best performing non-deep-learning models were found to be *tsfresh*-based ones using the combined ISI and spike count encodings of the underlying spike trains. Convolutional neural networks were found to outperform the classical models, sometimes by a considerable margin (i.e. the Allen cell types dataset).

Unsupervised spike train temporal structure recognition

The spike train temporal structure recognition task is defined as follows: for a set of spike train activity data, we generate a binary classification task by producing an additional category of spiking data consisting of spike trains from the original dataset with a certain transformation applied to them. We consider the following spike train transformations: (i) ISI shuffling inside the spike train (random shuffling applied to the ISI time series), (ii) reversing the ISI time series and (iii) adding spike timing jitter sampled from the truncated normal distribution to the time series. Note that the first two transformation types do not change the value distribution of the time series, only its temporal structure (the exact ordering of the interspike intervals of the spike train). Hence, if it is possible to construct a classification model capable of distinguishing between the two activity classes (original spiking activity versus the transformed one), then one could say that the model has learned to detect the temporal structure in the time series. In the case of classifiers trained on *tsfresh* feature vectors, the classification accuracy metrics obtained can be thought of as measures of the amount of temporal structure contained in the spike trains (to the extent encoded in *tsfresh* features). The classification results (AUC-ROC values) for different base spiking data and different transforms is shown in Table 5. Notably, one could observe higher AUC-ROC values for the retinal ganglion cell spiking data in case of the randomly moving bar stimulus as compared to the white noise checkerboard stimulus for all of the three transforms considered. The same difference in accuracy values is observed for the fcx-1 dataset whereby classification AUC-ROC value is higher for all of the three transforms when the SLEEP state is used as the base spiking dataset as opposed to the WAKE state. The accuracy values that we observe for the temporal structure recognition tasks are above the chance level in most cases, with low values for fcx-1 WAKE-state data in case of reverse and noise transforms.

	Reverse transform	Shuffling transform	Noise transform
Retinal ganglion cells (randomly moving bar input)	0.81777	0.90519	0.702568
Retinal ganglion cells (white noise checkerboard input)	0.869284	0.927350	0.936033
fcx-1 WAKE spike trains	0.588104	0.936633	0.541783
fcx-1 SLEEP spike trains	0.7611201	0.952244	0.596265

Table 5. Test set AUC-ROC values for the unsupervised temporal structure recognition tasks for different base spiking datasets and different transforms.

The set of most discriminative spike train features

Being able to estimate feature importance ranks from trained decision tree ensembles allows us to detect the most discriminating features of ISI time-series for a particular classification problem. In order to select the important groups of discriminative features, we trained several random forest classification models (with different random seeds) on each dataset and used the feature importance scores extracted from the classifiers to rank the *tsfresh* features. We have used the mean scores averaged over different datasets to identify the features that are discriminative for spiking data across different scenarios. According to this feature ranking procedure, the following groups of *tsfresh* features are selected (see also Fig. S2):

- *median*, *kurtosis*, *quantile_q* – simple statistics of the ISI value distribution in the series like the median ISI value, q quantiles and kurtosis of the ISI value

distribution

- *change_quantiles* – this feature is calculated by fixing a corridor of the time series values (defined by lower and higher quantile bounds, q_l and q_h, which are hyperparameters), then calculating a set of consecutive change values in the series (differencing) and then applying an aggregation function (mean or variance). Another boolean hyperparameter *is_abs* determines whether absolute change values should be taken or not.

- $fft_coefficient$ – absolute values of the fast Fourier transform coefficients (individual coefficient values and aggregates).

- *entropy* – values of the sample entropy, the approximate entropy and the binned entropy of the power spectral density of the time series.

- *agg_linear_trend* – features from linear least-squares regression (standard error in particular) for the values of the time series that were aggregated over chunks of a certain size (with different aggregation functions like min, max, mean and variance). Chunk sizes vary from 5 to 50 points in the series.

To visualize class separation for the WAKE vs. SLEEP state spike trains from the fcx-1 dataset as point clouds in two dimensions, we took the top-20 importance *tsfresh* features identified during the above feature selection procedure. We then used dimensionality reduction techniques on this reduced 20-dimensional dataset to visualize the structure of the data with respect to the WAKE/SLEEP state labels of the series. Results are shown in Fig. 3 for two Uniform Manifold Approximation and Projection, UMAP [41] low-dimension embedding algorithms. We also applied other methods such as PCA and t-SNE (t-distributed Stochastic Neighbor Embedding) which gave essentially the same results (not shown). In all cases, classes cannot be linearly separated in two-dimensional embedding spaces, however, there is a separation of large fraction of the points of the WAKE and SLEEP state classes.

We conclude that both the convolutional neural networks and the hand-crafted feature engineering approach combined with strong tree-based learning models set a strong baseline for spike train classification for all of the tree studied tasks.

Discussion

In this work, we have introduced a diverse neuronal spike train classification benchmark to evaluate neural decoding algorithms. The benchmark consists of several single-neuron spike train prediction tasks spanning stimulus type prediction, neuron type identification and animal behavioural state prediction. The interneuron type classification task (SST vs. VIP interneurons) is shown to be non-trivial due to the firing patterns of these two neuron types being quite similar in properties (as opposed to, for instance, PV interneurons, which typically have significantly higher firing rates compared to SST/VIP neurons as well as pyramidal cells). The other two tasks relate spiking activity of individual neurons to the global state of the underlying neural circuit, which is in one of the cases stationary during a considerable time period (fcx-1 dataset) and the other one could be viewed as a transient stimulus-driven one (retinal dataset). In both cases, we have demonstrated that individual neuronal spike trains contain information related to the global state of the neural circuit and this information can be decoded (from a relatively small spike train) if appropriate time-series learning models are used. Extensive experiments on several datasets that we have conducted imply that not only ISI value distribution is important for global state identification but also the temporal information contained in the spike trains, that is, features related to the exact

sequences of interspike intervals in neural firing. We have identified groups of features highly informative for neural decoding tasks and established that this feature encoding combined with strong supervised learning algorithms such as gradient boosted tree ensembles establishes a strong baseline on the proposed benchmark that performs on par with state-of-the-art deep learning approaches. It was shown that significantly large accuracy values can be obtained on all of the proposed tasks using the hand-crafted feature encoding approach on single-neuron spike train chunks containing as low as 50 interspike intervals. We suggest that accuracy values can further be improved by hyperparameter search, model ensembling and test-time data augmentation. We propose that neural decoding models be evaluated on diverse and challenging tasks including the proposed benchmark (as well as regression tasks used to evaluate decoding models previously [1,8]) in order to establish a sensible model performance ranking similarly to what is done for computer vision and natural language understanding problems. We believe that this would drive further development of highly accurate neural decoding/neural activity mining approaches enabling their application in precision-critical tasks such as identifying pathological disease-related firing activity patterns in the brain.

Conclusion

To summarize our contributions, we have proposed a challenging and diverse benchmark for *individual cell* spike train classification to evaluate neural decoding models.

We have shown that a classical machine learning baseline comprised of massive time-series feature extraction from different spike train encodings coupled with well-performing classification approaches such as gradient boosting produces results on par with deep learning models, although with deep neural nets still slightly outperforming the classical methods. Furthermore, we have shown that the firing of individual neurons contains information about the global state of the organism as well as the information about the neuron type that can be decoded with machine learning approaches. This approach was further generalized to the unsupervised (self-supervised) setting, which helped reveal interesting structural properties of the spiking data we considered, in particular the WAKE-state time-reversal invariance and spiking jitter robustness of the cortical activity in the fcx-1 dataset. The massive time-series feature engineering approach helped detect groups of time-series features that have discriminative power over a set of different tasks in our benchmark and might thus be useful in general neural decoding tasks.

References

1. Glaser JI, Chowdhury RH, Perich MG, Miller LE, Kording KP. Machine learning for neural decoding. arXiv preprint arXiv:170800909. 2017;.

2. Pachitariu M, Stringer C, Schröder S, Dipoppa M, Rossi LF, Carandini M, et al. Suite2p: beyond 10,000 neurons with standard two-photon microscopy. Biorxiv. 2016; p. 061507.

3. Tsai D, John E, Chari T, Yuste R, Shepard K. High-channel-count, high-density microelectrode array for closed-loop investigation of neuronal networks. In: Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE. IEEE; 2015. p. 7510–7513.

4. Steinmetz NA, Koch C, Harris KD, Carandini M. Challenges and opportunities for large-scale electrophysiology with Neuropixels probes. Current opinion in neurobiology. 2018;50:92–100.

5. Teeters JL, Sommer FT. Crcns. org: a repository of high-quality data sets and tools for computational neuroscience. BMC Neuroscience. 2009;10(S1):S6.

6. Bagnall A, Lines J, Bostrom A, Large J, Keogh E. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery. 2017;31(3):606–660.

7. Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller PA. Deep learning for time series classification: a review. Data Mining and Knowledge Discovery. 2019;33(4):917–963.

8. Livezey JA, Glaser JI. Deep learning approaches for neural decoding across architectures and recording modalities. Briefings in bioinformatics. 2021;22(2):1577–1591.

9. Benjamin AS, Fernandes HL, Tomlinson T, Ramkumar P, VerSteeg C, Chowdhury RH, et al. Modern Machine Learning as a Benchmark for Fitting Neural Responses. Frontiers in computational neuroscience. 2018;12.

10. Tezuka T. Multineuron spike train analysis with R-convolution linear combination kernel. Neural Networks. 2018;102:67–77.

11. Humphries MD. Spike-train communities: finding groups of similar spike trains. Journal of Neuroscience. 2011;31(6):2321–2336.

12. Rossum Mv. A novel spike distance. Neural computation. 2001;13(4):751–763.

13. Victor JD, Purpura KP. Metric-space analysis of spike trains: theory, algorithms and application. Network: computation in neural systems. 1997;8(2):127–164.

14. Mulansky M, Kreuz T. PySpike—A Python library for analyzing spike train synchrony. SoftwareX. 2016;5:183–189.

15. Tezuka T. Spike Train Pattern Discovery Using Interval Structure Alignment. In: International Conference on Neural Information Processing. Springer; 2015. p. 241–249.

16. Jouty J, Hilgen G, Sernagor E, Hennig M. Non-parametric physiological classification of retinal ganglion cells. bioRxiv. 2018; p. 407635.

17. Charlesworth P, Cotterill E, Morton A, Grant SG, Eglen SJ. Quantitative differences in developmental profiles of spontaneous activity in cortical and hippocampal cultures. Neural development. 2015;10(1):1.

18. Li M, Zhao F, Lee J, Wang D, Kuang H, Tsien JZ. Computational classification approach to profile neuron subtypes from brain activity mapping data. Scientific reports. 2015;5:12474.

19. Jia X, Siegle J, Bennett C, Gale S, Denman D, Koch C, et al. High-density extracellular probes reveal dendritic backpropagation and facilitate neuron classification. bioRxiv. 2018; p. 376863.

20. Fulcher BD, Jones NS. hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. Cell systems. 2017;5(5):527–531.

21. Christ M, Braun N, Neuffer J, Kempa-Liehr AW. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh–A Python package). Neurocomputing. 2018;.

22. Friedman JH. Greedy function approximation: a gradient boosting machine. Annals of statistics. 2001; p. 1189–1232.

23. Karim F, Majumdar S, Darabi H, Harford S. Multivariate LSTM-FCNs for time series classification. Neural Networks. 2019;116:237–245.

24. Zhao B, Lu H, Chen S, Liu J, Wu D. Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics. 2017;28(1):162–169.

25. Fawaz HI, Lucas B, Forestier G, Pelletier C, Schmidt DF, Weber J, et al. Inceptiontime: Finding alexnet for time series classification. Data Mining and Knowledge Discovery. 2020;34(6):1936–1962.

26. Dau HA, Bagnall A, Kamgar K, Yeh CCM, Zhu Y, Gharghabi S, et al. The UCR time series archive. IEEE/CAA Journal of Automatica Sinica. 2019;6(6):1293–1305.

27. Zerveas G, Jayaraman S, Patel D, Bhamidipaty A, Eickhoff C. A Transformer-based Framework for Multivariate Time Series Representation Learning. arXiv preprint arXiv:201002803. 2020;.

28. Tang W, Long G, Liu L, Zhou T, Jiang J, Blumenstein M. Rethinking 1d-cnn for time series classification: A stronger baseline. arXiv preprint arXiv:200210061. 2020;.

29. Prentice JS, Marre O, Ioffe ML, Loback AR, Tkačik G, Berry MJ. Error-robust modes of the retinal population code. PLoS computational biology. 2016;12(11):e1005148.

30. Loback GPJSIMLBIMJMO Adrianna R ; Tkačik. Multi-electrode retinal ganglion cell population spiking data. Dryad Dataset. 2016;.

31. Watson B, Levenstein D, Greene J, Gelinas J, Buzsaki G. Multi-unit spiking activity recorded from rat frontal cortex (brain regions mPFC, OFC, ACC, and M2) during wake-sleep episode wherein at least 7 minutes of wake are followed by 20 minutes of sleep. CRCNS.org. 2016;doi:http://dx.doi.org/10.6080/K02N506Q.

32. Watson BO, Levenstein D, Greene JP, Gelinas JN, Buzsáki G. Network homeostasis and state dynamics of neocortical sleep. Neuron. 2016;90(4):839–852.

33. Allen Cell Types dataset;.

34. Rudy B, Fishell G, Lee S, Hjerling-Leffler J. Three groups of interneurons account for nearly 100% of neocortical GABAergic neurons. Developmental neurobiology. 2011;71(1):45–61.

35. Jing L, Tian Y. Self-supervised visual feature learning with deep neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence. 2020;.

36. Jeni LA, Cohn JF, De La Torre F. Facing imbalanced data–recommendations for the use of performance metrics. In: 2013 Humaine association conference on affective computing and intelligent interaction. IEEE; 2013. p. 245–251.

37. Barandela R, Sánchez JS, Garcıa V, Rangel E. Strategies for learning in class imbalance problems. Pattern Recognition. 2003;36(3):849–851.

38. He H, Garcia EA. Learning from imbalanced data. IEEE Transactions on knowledge and data engineering. 2009;21(9):1263–1284.

39. Saito T, Rehmsmeier M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PloS one. 2015;10(3):e0118432.

40. Oguiza I. tsai - A state-of-the-art deep learning library for time series and sequential data; 2020. Github. Available from: https://github.com/timeseriesAI/tsai.

41. McInnes L, Healy J, Saul N, Großberger L. UMAP: Uniform Manifold Approximation and Projection. The Journal of Open Source Software. 2018;3(29):861.

Supplementary materials.

S1 Appendix. Classifier hyperparameter values.

Listed below are hyperparameter values and implementation references for all of the classifier types we used. For more reference, see the example script.

- Random Forest: sklearn implementation, $n_estimators = 500$, $max_depth = 10$.
- Extra Trees Classifier: sklearn implementation, $n_estimators = 500$, $max_depth = None$ (no limit on depth).
- Logistic Regression: sklearn implementation, l_2 penalty, $C = 0.001$
- XGBoost: xgboost implementation
  - $max_depth = 8$
  - $learning_rate = 0.1$
  - $n_estimators = 500$
  - $objective = binary : logistic$
  - $booster = gbtree$
  - $gamma = 0$
  - $min_child_weight = 1$
  - $max_delta_step = 0$
  - $subsample = 0.7$
  - $colsample_bytree = 1$
  - $colsample_bylevel = 1$
  - $colsample_bynode = 1$
  - $reg_alpha = 0$
  - $reg_lambda = 1$
  - $scale_pos_weight = 1$
  - $base_score = 0.5$
- FCN, InceptionTime, XceptionTime, ResNet: tsai implementation
  - $epochs = 200$
  - $max_lr = 0.1$
  - $optimizer = sgd$
  - $weight_decay = 1e-4$
  - $batch_size = 128$
  - $lr_schedule = cosine$
  - $best_model =$ with the largest *cohen_kappa* or *accuracy* on the validation set

We have looked at how the performance of a CNN is robust to the above training hyperparameters using the XceptionTime architecture on the retinal stimulus classification dataset (see Table S1).

We found that generally the CNN performance is not significantly affected by the changes in the main hyperparameters, being relatively robust with respect to these changes. We do, however, expect that a thorough hyperparameter search (including search over the architecture's parameters) could result in a significant improvement in performance.

Varied hyperparameters w.r.t. default setting	Geometric mean score
Default setting	0.927028
Peak lr = 0.01	0.89778
Adam optimizer, peak lr = 0.01	0.904811
One cycle lr schedule	0.909487
100 training epochs	0.92039
Batch size = 256	0.9198

Table S1. Geometric mean score obtained for the XceptionTime architecture trained on the retinal stimulus classification dataset with alterations in training hyperparameters.

Supplementary figures.

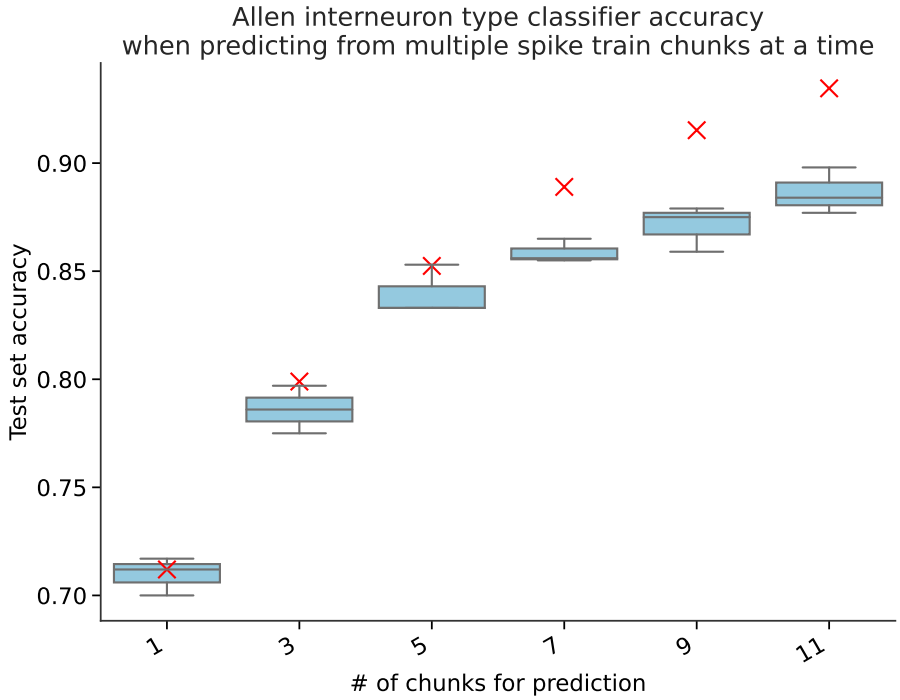

Fig S1. Classification accuracy for the Allen cell types VIP/SST interneuron classification task in the case of multiple randomly sampled same-class spike train chunks per prediction (with prediction done via majority voting). The model trained in these trials is a random forest classifier on the full set of *tsfresh* features. The boxplots reflect the median accuracy and the variance between different train/test splits as done in the main text for the fcx-1 data set. The red crosses correspond to the theoretical estimate under the assumption of independently sampled spike train chunks.

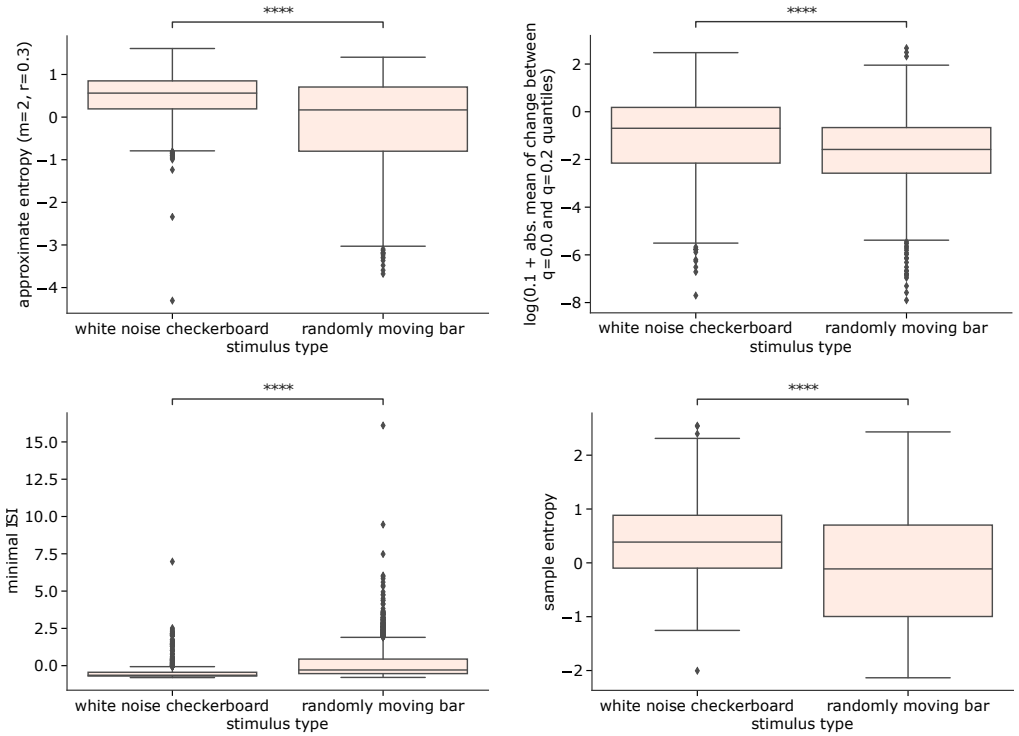

Fig S2. Boxplots of *tsfresh*-extracted feature distributions for features with high discriminative power as detected by the trained decision tree ensemble classifiers in the retinal stimulus type prediction task. Two-sided Mann-Whitney-Wilcoxon test with Bonferroni correction is performed to assess statistical significance; **** denotes $p < 1e-4$.