

Neural Activity Classification with Machine Learning Models Trained on Interspike Interval Series Data

Ivan Lazarevich

Group for Neural Theory,

Laboratoire de Neurosciences Cognitives,

École Normale Supérieure,

Paris, France

Lobachevsky State University,

Nizhny Novgorod, Russia

vanya.lzr@gmail.com

Ilya Prokin

Dataswati,

Orsay, France

isprokin@gmail.com

Boris Gutkin

Group for Neural Theory,

Laboratoire de Neurosciences Cognitives,

École Normale Supérieure,

Paris, France

Center for Cognition and Decision Making,

NRU Higher School of Economics,

Moscow, Russia

boris.gutkin@ens.fr

Abstract—The flow of information through the brain is reflected by the activity patterns of neural cells. Indeed, these firing patterns are widely used as input data to predictive models that relate stimuli and animal behavior to the activity of neural populations. However, relatively little attention in machine learning based neural decoding was paid to single neuron spike trains as predictors of cell or network properties in the brain. In this work, we introduce an approach to neuronal spike train data mining which enables effective classification and clustering of neuron types and network activity states based on single-cell spiking patterns. This approach is centered around applying state-of-the-art time series classification/clustering methods to sequences of interspike intervals recorded from single neurons. These methods significantly outperform simple benchmarks in both tasks involving classification of neuron type (e.g. excitatory vs. inhibitory cells) and neural circuit activity state (e.g. awake or sleep state of the animal) on an open-access cortical spiking activity dataset. Furthermore, the achieved high accuracy of classification indicates that individual neurons carry a substantial amount of information about the global neural network state.

Index Terms—spike train, neural decoding, spiking activity classification

INTRODUCTION

Modern advances in multineuronal recording technologies such as calcium imaging [1] and extracellular recordings with multielectrode arrays [2] allow producing single-neuron resolution brain activity data with remarkable magnitude and precision. In addition to experimental technique development, various data analysis methods were introduced over the years which enable better processing as well as understanding of neural activity data. Recent developments range from accurate inference of spiking events from calcium fluorescence traces based on a variety of machine learning approaches (e.g. [3]) to a myriad of spike sorting techniques for identification of originating neurons in multi-electrode recordings [4], [5].

Modern machine learning techniques have been successfully applied both to neural activity decoding (predicting stimulus/action from spiking activity) [6] as well as neural encoding

(predicting neural activity from stimuli) [7]. These neural decoding/encoding approaches typically focus on firing rate signals of multiple neurons from a population upon stimulus presentation/action execution. In this context, the fine temporal structure of neuronal firing patterns is not usually considered as a predictor of cell or network properties in question. However, it is known that the temporal structure of neuronal spike trains may significantly vary across cell types and also across particular activity states within a single neuron type [8]. Thus, it may be hypothesized that certain features of neuronal spike trains carry information about cell type or network activity state that can be, in principle, decoded from these activity patterns. In this study, we demonstrate that effective feature representation of neuronal spike trains enables good performance in supervised classification tasks which involve identifying a particular neuron type or activity state of a neural circuit (for instance, pyramidal cell vs. interneuron classification, awake vs. sleep circuit state classification, etc.).

A number of previous studies on feature vector representations of spike trains usually focused on defining a spike train distance metric [9] for identification of neuronal assemblies [10]. Several different definitions of the spike train distance exist such as van Rossum distance [11], Victor-Purpura distance [12], SPIKE- and ISI- synchronization distances [13] (for a thorough list of existing spike train distance metrics see [9]). These distance metrics were used to perform spike train clustering and classification based on the k-Nearest-Neighbors approach [14]. In a recent study, Jouty et al. [15] employed ISI and SPIKE distance measures to perform clustering of retinal ganglion cells based on their firing responses.

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. [16] 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. [17] 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 features are defined and hand-crafted to address specific problems and might not provide an optimal feature representation of spiking activity data at hand. Finally, not only spike timing information was used to characterize neurons in a supervised classification task. Jia et al. [18] used waveform features of extracellularly recorded action potentials to classify them by brain region of origin.

In this work, we propose feature vector representations for neuronal spike-train time series. These representations combined with general time series classification/clustering methods enable good classification results on single-cell spike trains in both cell type classification and network activity state classification tasks significantly outperforming simple benchmarks. For both of these tasks, we used an open-access cortical spiking activity dataset. For the cell type classification task, we classified putative excitatory vs. inhibitory neurons, whereas for the neural network activity state classification task, we classified awake vs. sleep state of the animal from single-neuron spiking activity. For both of these tasks, using individual neural spike trains, we obtained baseline performance estimates for a range of supervised machine learning algorithms trained on our feature vector representations. We thus demonstrated that single neurons carry a sufficient amount of predictive information that allows discriminating not only between cell types, but also between global states of the neural network.

I. METHODS

A. Overview of time series classification methods

We applied general time series feature representation methods [19] 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.

1) Neighbor-based models with time series distance measures: A good baseline algorithm for time series classification is k-nearest-neighbors (kNN). Although raw time series are used, the distance metric used by kNN implicitly determines feature representation, that is, it defines which properties of time series are compared. To obtain good results with kNN, it is important to choose the right distance metric. The most common metrics used in kNN for time series classification are the Minkowski L_p distance [19] and the Dynamic Time Warping (DTW) distance [20]. Conversion to interspike-interval (ISI) series representation of the spike train can be done prior to calculating the inter-train distance. Moreover, the inter-train distance can be defined based on differences of ISI

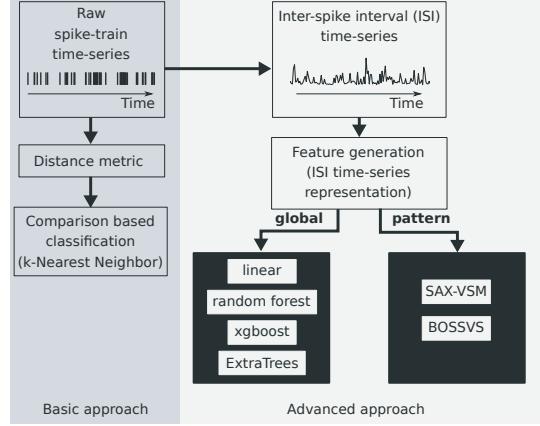


Fig. 1. Schematic representation of our approaches to spike-train series classification: baseline kNN-based approach (dark gray shading, left), advanced feature-based approach (light gray shading, right).

value distributions within the trains. For this one can utilize distribution similarity measures (e.g. Kolmogorov-Smirnov distance, KullbackLeibler divergence, Wasserstein distance) and compute its values between ISI distributions of given 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 [9].

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 various feature space representations available for time series data.

2) Manual time series feature engineering: 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 *htsfa* MATLAB package [21] and automated feature extraction in the *tsfresh* Python package [22]. Here we utilize the *tsfresh* package which enables calculation of 794 descriptive time series features for each spike train, ranging from Fourier and wavelet expansion coefficients to

coefficients of a fitted autoregressive process.

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 kNN, random forests, gradient boosting machines [23]. We found this approach to yield good benchmark classification results in both cell type identification and neural activity state classification tasks.

3) Quantization/bag-of-patterns transforms: Some state-of-the-art algorithms in general time series classification use text mining techniques and thus transform time series into bags-of-words (patterns). This is typically done the following way. First, a time series of real numbers is transformed into a sequence of letters. One of the methods to perform this transform is Symbolic Aggregate approXimation (SAX) [24]. In SAX, bins are computed for each time series using gaussian or empirical quantiles. After that, each datapoint in the series is replaced by the bin it is in (a letter). Another algorithm commonly used for this task is Multiple Coefficient Binning (MCB). The idea is very similar to SAX and the difference is that the quantization is applied at each timestamp. The third algorithm for the series-letter transform is Symbolic Fourier Approximation (SFA) [25]. It performs a discrete Fourier transform (DFT) and then applies MCB, i.e. MCB is applied to the selected Fourier coefficients of each time series. Once the time series is transformed into a sequence of letters, a sliding window of fixed size can be applied to define and detect words (letter patterns) in the sequence. After that, the bag-of-words (BOW) representation can be constructed whereby each "sentence" (time series) turns into a vector of word occurrence frequencies.

Several feature generation approaches were developed utilizing the BOW representation of time series data. One such method is Bag-of-SFA Symbols (BOSS) [26]. According to the BOSS algorithm, each time series is first transformed into a bag of words using SFA and BOW. Features that are created after this transformation are determined by word occurrence frequencies.

Some classification algorithms which use this bag-of-patterns approach represent whole classes of samples with a set of features. One example of such a method is an algorithm called SAX-VSM [27]. The outline of this algorithm is to first transform raw time series into bags of words using SAX and BOW, then merge, for each class label, all bags of words for this class label into a single class-wise bag of words, and finally compute term-frequency-inverse-document-frequency statistic (tf-idf) [28] for each bag of words. This leads to a tf-idf vector for each class label. To predict an unlabeled time series, this time series is first transformed into a term frequency vector, then the predicted label is the one giving the highest cosine similarity among the tf-idf vectors learned in the training phase (nearest neighbor classification with tf-idf features). A very similar approach is Bag-of-SFA Symbols in Vector Space (BOSSVS) [29] which is equivalent

to SAX-VSM, but words are created using SFA rather than SAX. The choice of SAX/MCB or SFA representation of the time series depends on the task at hand – in particular, SFA would work best if spectral characteristics of the time series are important for classification, while SAX/MCB would be efficient for describing the temporal structure (e.g. reoccurring patterns) of the series.

These time series representation methods are implemented in the *pyts* Python package [30], which was used in the present work. Whenever we apply BOSSVS and SAX-VSM algorithms to classify neural activity, we make use of the ISI representation of the corresponding spike trains.

4) Other approaches: Lastly, there are deep learning based approaches [31] such as Long Short-Term Memory (LSTM) nets [32] and one-dimensional convolutional neural networks (CNNs) [33] that enable good classification quality on general time series datasets [34]. Finally, to attain better prediction performance, all the discussed models may be effectively combined in an ensemble of models using model stacking/blending [35] to improve classification results. These approaches are to be explored in further work.

5) Image representation of time series: Several methods to represent time series as images (matrices with spatial structure) were developed and utilized for classification as well. One such image representation method is called the recurrence plot [36]. It transforms a time series into a matrix where each value corresponds to the distance between two trajectories (a trajectory is a sub time series, i.e. a subsequence of back-to-back values of a time series). The obtained matrix can then be binarized using some threshold value.

Another method of time series image representation is called Gramian Angular Field (GAF) [37]. According to GAF, a time series is first represented as polar coordinates. Then the time series can be transformed into a Gramian Angular Summation Field (GASF) when the cosine of the sum of the angular coordinates is computed or a Gramian Angular Difference Field (GADF) when the sine of the difference of the angular coordinates is computed.

Yet another image representation method is the Markov Transition Field (MTF). The outline of the algorithm is to first quantize a time series using SAX, then to compute the Markov transition matrix (the quantized time series is treated as a Markov chain) and finally to compute the Markov transition field from the transition matrix.

6) Deep learning approaches: Lastly, one can make use of modern deep learning approaches [31] to perform algorithm training on raw time series data. Recurrent neural networks such as Long Short-Term Memory (LSTM) nets [32] and their variations based on one-dimensional convolutional neural networks (CNNs) [33] were shown to enable good classification quality on general time series datasets [34]. Some of the frequently used neural network architectures are plain LSTMs [38], Convolutional Neural Network LSTMs (CNN-LSTMs) [39] and Convolutional LSTMs (ConvLSTMs) [40]. These approaches (together with image-based representations

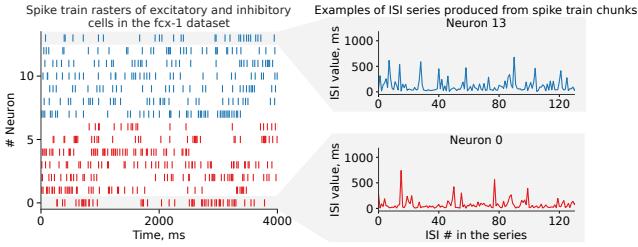


Fig. 2. Examples of spiking activity recordings in the CRCNS fcx-1 dataset. Left: spike train raster of a random subset of excitatory cells (red) and inhibitory cells (blue). Right: examples of ISI series produced from spike train chunks of inhibitory (excitatory) cells in the fcx-1 dataset shown on top (bottom) panel.

of spike trains as described above) are to be explored in further work.

7) Ensembling: the best of all worlds: All method classes listed in the above subsections (e.g. neighbor-based models, models based on engineered features, bag-of-patterns classifiers) are fundamentally different in their underlying feature representation of the time series and the kind of information they extract from these features. To attain better prediction performance, all these models may be effectively combined in an ensemble of models using model stacking/blending [35] to improve classification results.

B. Data

For the tasks of spike train analysis and classification, high-quality datasets with recordings of neural firing activity are of foremost importance. Here we used an open-access neural activity dataset from the Collaborative Research in Computational Neuroscience (CRCNS) repository (<http://crcns.org/>) [41]. Specifically, the following dataset was used to define classification benchmarks:

- **fcx-1** dataset [42], [43]: 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 2 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. Different classification problems might be addressed with this dataset: (a) excitatory vs. inhibitory cell classification from spike train data with similar mean firing rate, and (b) WAKE vs. SLEEP activity state classification.

C. Cross-validation scheme and data preprocessing

Suppose we are given a dataset containing data from several mice 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 spike extraction from fluorescence traces were performed beforehand). The number of spikes within each train is going to be variable from train to train, so vectors of spike times for each neuron would have different length. A natural way to standardize the length of spike-train-vectors would be dividing the full spike train into chunks of N_{size} spike times, where N_{size} is fixed for each chunk. The chunks can be produced by moving a sliding window across the spike-timing-vector. Thus, each neuron would contribute a different number of spike-timing-chunks depending on its average firing rate. To remove the trend component from each chunk, differencing can be applied to get the ISI-series representation of the spike train chunk. Here we used a classical cross-validation strategy [44] whereby the data is split into two parts: i) training data that is used to fit the parameters of a machine learning model and ii) testing (validation) data used to evaluate the model fitted on training data. A simple approach is to take the whole dataset of shape $[N_{\text{chunks}}, N_{\text{size}}]$ with corresponding class labels and perform the train-test subset split for classification quality assessment. However, it would lead to an overly-optimistic estimate of the algorithm's performance, because similar ISI-series chunks coming from the same neuron/experiment/animal can become a part of both train and test datasets. A more accurate validation scheme would be to first split animal IDs/experiments/neurons into train and test subsets, so that test prediction is performed on a neuron/experiment/animal not present at all in the train set. After this splitting by spike-train ID is done, one could generate fixed length chunks of spiking activity within each subset. When splitting the data into training and testing subsets we made sure that neuron IDs do not overlap between data subsets.

D. Algorithm quality assessment

Metrics we used to assess classification performance are accuracy (when class distribution balancing was performed in the dataset beforehand) and AUC-ROC [44] (when probabilistic estimates for a sample to belong to a certain class are available). In general, choice of a particular metric of interest should be determined according to the underlying classification task. For instance, in the classification of diseased states of a neural circuit, more emphasis can be drawn to values of recall (true positive rate) – due to foremost importance of diseased state recognition and not vice versa.

If the classification task is to determine certain activity states of the neural circuit, one could collect activity of several neurons at a time (e.g. by sorting spikes from MEA recordings or from calcium imaging) and correspondingly perform classification for each measured neuron independently. If the

final classification is done by majority voting from all single-neuron predictions and we assume that recorded neurons are randomly sampled from the whole ensemble, the optimistic estimate for accuracy increase with the number of neurons N_{cells} would be

$$\mu = \sum_{i=m}^{N_{\text{cells}}} C_{N_{\text{cells}}}^i p^i (1-p)^{N_{\text{cells}}-i}$$

where μ is the probability that the majority vote prediction is correct, p is the probability of a single classifier prediction being correct, N_{cells} is the number of predictions made, $m = [N/2] + 1$ is the minimal majority of votes. If single neurons are to be classified (e.g. into subtypes), one can split a large spike train into several chunks with N_{size} spikes and perform predictions on each of those independently (with final output being the major vote from every chunk). In this case the above estimate is likely overly optimistic, since ISI sequences within a single spike train are probably distributed differently than ISI sequences across different neurons.

II. RESULTS

As we have discussed above, one could consider posing different types of classification problems utilizing available spiking data – for instance, cell type classification or network activity state classification. Here, while considering the fcx-1 dataset, we tested if it is possible to infer both cell type (e.g. excitatory or inhibitory) and network activity state (e.g. WAKE or SLEEP) from spiking activity of individual neurons.

A. Excitatory vs. inhibitory cell spike train classification

We started with a basic excitatory vs. inhibitory cell classification task for the fcx-1 dataset. We added spike train samples from both cell types to the dataset regardless of the underlying network state (sleep/wake). We ended up with 995 excitatory cells and 126 inhibitory cells recorded in total. For our validation dataset, we took cells corresponding to ~40% of the total spike count from both classes. The rest of the data was used for the training set. Every spike train in the data is then represented in an ISI-series form. For both train and validation sets, we extract ISI series chunks of size $N_{\text{size}} = 100$ ISIs by applying a rolling window with a step = 50 ISIs. We found that ISI values across spike train chunks follow a heavy-tail distribution which is different depending on the cell type. We then only left spike train chunks with similar ISI value distributions for both cell types to make the classification task challenging. To do that, we chose a interval in ISI values (here it is taken to be [50, 200] ms) and only left the spike train chunks which contained a certain percentage (here 70%) of ISI values within this chosen interval. This means we only kept excitatory-cell spike trains with sufficiently high firing rates (comparable to inhibitory cells) in the dataset. In this setting, the classification problem had a slight class imbalance, since we obtained ~ 26000 excitatory spike train chunks and ~ 75000 inhibitory spike train chunks in the train set after preprocessing. The validation (holdout) set consists of ~ 22000 and excitatory and ~ 48000 inhibitory spike

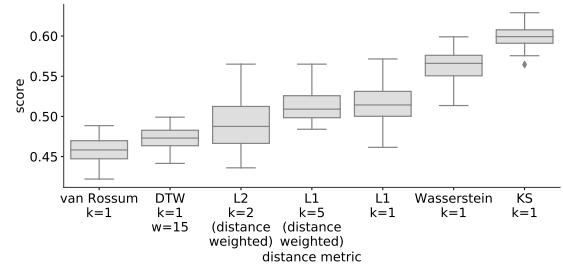


Fig. 3. Accuracy score distributions of kNN classifiers trained and validated on random subsamples of the full excitatory vs. inhibitory dataset (6000 samples in the training set, 6000 samples in the validation set, balanced) depending on the distance metric used.

train chunks. Spike train chunks were extracted regardless of neural activity state (i.e. sleep or awake) during particular time intervals in the recording.

1) *Neighbor-based models on raw ISI series:* To obtain the baseline classification quality for our task, we evaluated the performance of nearest-neighbor models with several general distance metrics trained on raw spiking time series. We trained the kNN model with the Minkowski metric ($p = 1$ and $p = 2$), DTW distance (window size equal to 15 time steps), Kolmogorov-Smirnov and Wasserstein distance for ISI value distributions within the train and, finally, spike-train-specific similarity metric, van Rossum distance. Fig. 3 shows achieved accuracy values on the validation dataset for each distance metric trained and validated on several random subsamples of the full dataset (3000 samples of both classes in both train and validation sets). Mean accuracy scores were generally found to be in the range of 45-60% depending on the distance metric used, and the Kolmogorov-Smirnov and Wasserstein distance metrics were found to perform best compared to the rest of the metrics even for the single nearest-neighbor-based predictions in this task. Since the time series samples in our dataset were taken at different starting times across different neurons, pattern-matching-based metrics (e.g. Euclidean and spike-train-specific metrics) are not expected to yield excellent performance in this setting. In some cases they perform well because they indirectly capture the differences in the ISI value distribution across classes. Naturally, the metrics which directly compare these distributions (e.g. the Kolmogorov-Smirnov and Wasserstein distances) perform better in such tasks. Therefore, we propose using our approach that combines kNN with distribution comparison metrics for spike-train classification in the general setting when differences between ISI-distributions are essential. In cases when the activity classes have very similar ISI distributions, we expect all of aforementioned kNN methods to perform poorly and thus more advanced methods which capture the temporal structure of the spike trains are needed to yield satisfying classification results. We further focused on development and application of such approaches.

2) *Manual feature extraction and baseline classifier models:* Next, we used the manual feature extraction approach.

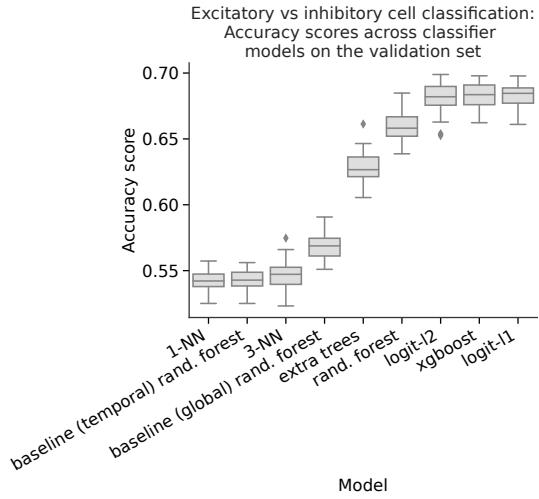


Fig. 4. Classification accuracy values of the classifiers trained on manually engineered (*tsfresh*) features of spike trains (full set of 794 features) achieved on the validation set in an excitatory vs. inhibitory spike train classification task.

That is, instead of feeding the raw temporal ISI sequence data directly to the classification model, we used the *tsfresh* Python package to calculate a representative set of 794 features for each spike train chunk. This can be done independently for the training and validation sets as features are calculated for each ISI-series independently. For each feature obtained from the training set, mean and variance were calculated to apply standard scaling to both training and testing sets. Low-variance features, which are not informative, were removed using the condition $std/(mean + \varepsilon) < \theta$, where θ is set to 0.2 and ε to 1e-9. Some of the features extracted from ISI series data by *tsfresh* were highly correlated with each other, as measured by the Pearson correlation coefficient between feature values. Removal of correlated features can be implemented by leaving a single feature out of each pair of features which have Pearson correlation (R) greater than a threshold value R_{thr} . Exact value of R_{thr} is generally a free parameter, however, we found that even slight removal of correlated features ($R_{thr} < 1$, but close to unity) had a negative effect on classification performance, and, a negative general trend for classification accuracy was observed as R_{thr} was decreased. Hence, in most of the tests presented here, we did not perform preliminary removal of correlated features for classification. It was, however, performed prior to further applying dimensionality reduction algorithms, as these methods are known to be sensitive to feature correlations. Once ISI series samples are converted to feature-based vector representation, standard classifier models such as random forests and gradient boosting machines can be used to evaluate baseline accuracy scores for the particular task. To compare accuracy scores across different classification model classes, we used (a) a kNN model with Euclidean distance metric (with $k = 1$ and $k = 3$ nearest neighbors), (b) a linear logistic regression model (with either L_1 or L_2 regularization terms, $C = 0.1$ for both models) and (c) several nonlinear tree-based models: random forest classifier (with $n = 200$ trees and

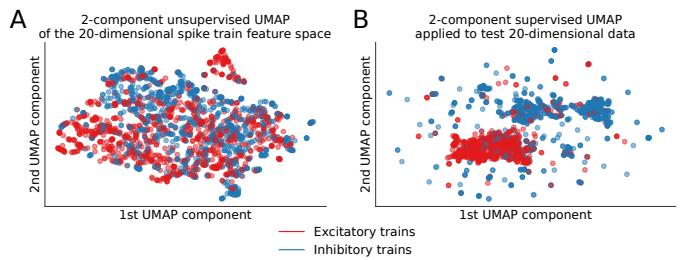


Fig. 5. Two-dimensional embeddings of the (20-dimensional) selected-*tsfresh*-feature space using (A) unsupervised UMAP (B) supervised UMAP embedding algorithms for inhibitory (points marked blue) vs. excitatory (points marked red) spike trains. For supervised UMAP, the transformation was fitted on a training set of the data (7000 samples) and applied a validation set (3000 samples, shown on the figure). In both cases, a separation of excitatory vs. inhibitory spike train classes can be observed even in the two-dimensional embedding space.

no regularization constraints), randomized decision trees (extra trees, with $n = 200$ trees) classifier and an implementation of decision tree gradient boosting from the *xgboost* library. Hyperparameter values used for the gradient boosting machine were: $n_{trees} = 250$, $learning_rate = 0.05$, $max_depth = 5$, $gamma = 0.322$, $colsample_bytree = 0.466$. We also introduced two simple benchmark models based on reduced sets of time series features: (a) a model where each ISI series is represented by 6 basic statistical features – the mean, median, minimum and maximum ISI values, the standard deviation and the absolute energy of the series (mean of squared values), and (b) a model based on autocorrelation coefficients of the series at different lags, partial autocorrelation coefficients and statistics of autocorrelation coefficients (mean, median and variance) across different lags. We trained random forest classifiers on these reduced feature representations and referred to the resulting models as the global and temporal baseline models, respectively.

Fig. 4 shows classification accuracy scores achieved with the models trained and tested on a balanced dataset obtained by undersampling the inhibitory ISI series class (~ 53000 samples in the training set total and ~ 44000 samples in the validation set). We were generally able to achieve mean accuracy higher than 65% even for a linear model (logistic regression) trained on samples from the full 794-dimensional feature space. Simple global and temporal random forest benchmarks (giving $< 60\%$ accuracy) were significantly outperformed by the models trained on the full feature space. Note that the ranking of models shown in Fig. 4 is robust to the choice of metric (in addition to accuracy, we tried AUC-ROC, not shown here).

Being able to estimate feature importance ranks allows us to detect the most discriminating features of ISI series for a particular classification problem. In order to select the important groups of discriminative features, we applied the following procedure to the full set of *tsfresh* features: first, we trained 10 logistic regression models with L_1 regularization penalty (with different random seeds, $C = 0.01$) on the full feature set; we then selected the features which had non-zero values of corresponding coefficients in all trained models. After that,

we identified highly correlated pairs of features ($|R| > 0.98$), which represent almost equivalent quantities, and removed one randomly selected feature out of each such pair. Further, we trained several random forest classifier models (with different random seeds) and calculated the aggregated feature importance ranks across models to select groups of features relevant for the excitatory vs. inhibitory cell classification task. After removal of the features corresponding to the first 50th percentile of the feature importance distribution, the following groups of *tsfresh* features are selected:

- *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 DFT expansion coefficients.
- *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 of excitatory vs. inhibitory cell spike trains 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 excitatory/inhibitory labels of the series. Results are shown in Fig. 5 for two Uniform Manifold Approximation and Projection, UMAP [45] 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 excitatory-cell and inhibitory-cell classes. Furthermore, we trained a supervised version of UMAP on a subset of train data samples (by providing class label information to the algorithm along with feature vectors) and applied the trained UMAP transformation to another test subset of data samples. Results of this transformation are shown in Fig. 5B. One can see that supervised UMAP learned a two-dimensional embedding of the data for excitatory vs. inhibitory class separation that was quite effective (compared to unsupervised methods). It is to be expected that adding these nonlinear UMAP features to the dataset can result in improved performance of the classification algorithms. Results presented in Fig. 5 show transformations

done by dimensionality reduction algorithms trained and applied to a subset of the full training dataset (consisting of 3000 samples total; except for supervised UMAP, for which, additional 7000 train samples were used to fit UMAP before applying it to transform 3000 samples).

We conclude that on our dataset manual feature engineering combined with classical machine learning models outperforms the straightforward kNN approach to the classification of excitatory vs inhibitory neurons. Next, we explored the performance of other state-of-the-art machine learning approaches to time series representation and classification.

3) BOW representation time series classification algorithms: We examined performance of two bag-of-patterns time series classification methods: BOSSVS and SAX-VSM (implementation from *pyts* Python package [30]). We found that performance quality of both these algorithms strongly depends on the choice of their hyperparameter values, with accuracy scores ranging from the chance level to values comparable to some other classification methods, depending on the exact choice of hyperparameters. We thus performed global hyperparameter optimization using the Parzen Estimator Tree method available in the *hyperopt* Python package. We collected hyperparameter values and corresponding accuracy scores on each iteration of the global search in order to evaluate how validation accuracy is distributed across the hyperparameter space. All possible hyperparameter values were considered during the initial global search, however results of the initial search revealed that, for some hyperparameters, particular value choices delivered consistently better accuracy scores. Therefore, we fixed some hyperparameter values for both algorithms: for SAX-VSM, we set *quantiles* = "empirical", *numerosity_reduction* = *False*, *use_idf* = *True*; for BOSSVS we set *quantiles* = "empirical", *norm_mean* = *False*, *norm_std* = *False*, *smooth_idf* = *True*, *sublinear_tf* = *False*. The remaining free hyperparameters that we searched over are: for SAX-VSM, *n_bins*, *window_size*, *smooth_idf*, *sublinear_tf*; for BOSSVS, *window_size*, *n_bins*, *variance_selection*, *variance_threshold*, *numerosity_reduction*. Fig. 6 shows distributions of validation accuracy scores for different hyperparameter values of BOSSVS and SAX-VSM algorithms evaluated on a balanced dataset (a random subsample of the full inhibitory vs. excitatory dataset was used, consisting of 3000 samples of both classes in both train and validation sets). BOSSVS algorithm was generally found to perform similarly to SAX-VSM on this task, with validation accuracy scores being in the range of 50-57%. Overall, both BOSSVS and SAX-VSM classification methods produced accuracy scores higher than the chance level implying that symbolic representation of the ISI series in both temporal and frequency domain reflects differences in the two classes of spike trains.

B. WAKE/SLEEP network state classification from single neuron spiking patterns

In the previous section, we evaluated how time series classification methods perform in a cell type classification task

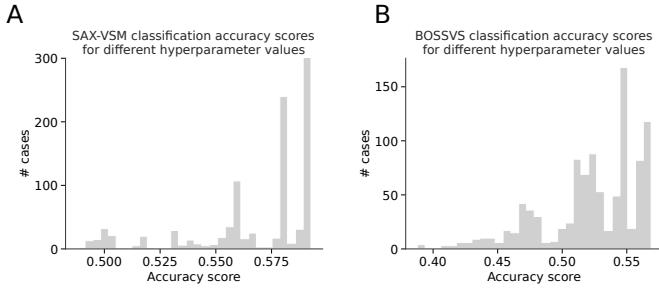


Fig. 6. Distribution of accuracy scores achieved by SAX-VSM (left) and BOSSVS (right) classifiers on the excitatory vs. inhibitory trains dataset during iterations of the global hyperparameter search. Classifiers were trained and accuracy was calculated on a subsample of the full dataset consisting of 3000 samples from both classes in both training and validation subsets.

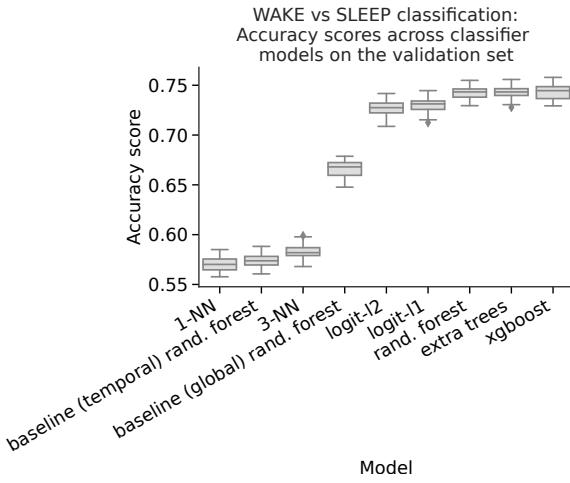


Fig. 7. Validation accuracy scores achieved on neural circuit activity classification tasks WAKE vs. SLEEP state classification.

(in particular, classification of principal cells vs. interneurons). However, our approach is not limited to classification of cell types. In order to demonstrate this, we evaluated classification performance between two distinct activity states of the neural circuit. This is a challenging problem since we aim to detect changes in the global network state by examining samples of spike trains from individual neurons in the network. The assumption here is that the change in spiking pattern properties due to different activity states can be decoded from single cells. We make use of the CRCNS fcx-1 dataset, which contains information about the time periods when mice were in an awake or sleep states. Moreover, particular sleep phase intervals (e.g. REM sleep vs. nonREM/SWS sleep) are also labelled. Here, we approached the problem of WAKE vs. SLEEP (REM+nonREM) state classification. For this, we extracted spike train data from interneurons at time intervals corresponding to WAKE and SLEEP phases of the recording.

The resulting dataset contained spiking patterns of 118 cells, from which we took 70 cells for the training set and the rest for the validation set, so that 60% of the total ISI count is used for training. We found that ISI distributions of inhibitory spike trains during the episodes of WAKE and SLEEP activity states

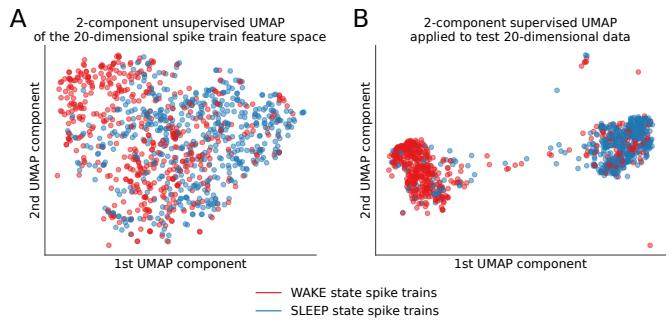


Fig. 8. Spike train feature embeddings for WAKE (points marked red) vs. SLEEP (points marked blue) activity states of the neural circuit. Two-dimensional embeddings of the (20-dimensional) selected *tsfresh*-feature space using (A) unsupervised UMAP and (C) supervised UMAP embedding algorithms for spike trains corresponding to WAKE vs. SLEEP activity states.

are quite similar, so that limiting a specific ISI value interval during data preprocessing is not necessary. We only limited the tail of the distribution by not considering the trains with mean ISI values > 400 ms. We repeated the same procedure for spike train extraction, with fixed number of ISIs $N_{\text{size}} = 100$ in each chunk generated by a rolling window of size 50 ISIs. We ended up with ~ 7900 spike train chunks in the WAKE state and ~ 12200 chunks in the SLEEP state for the training dataset. Validation set contained ~ 4300 spike train chunks in the WAKE state and ~ 9100 chunks in the SLEEP state.

We then applied the same classification pipeline as was done for cell type classification: we calculated the full set of *tsfresh* features for each spike train chunk and performed standard scaling and removal of low-variance features. As it was done before, dataset undersampling was performed which resulted in the training set consisting of ~ 7000 samples from both WAKE and SLEEP classes.

We then trained several different classification models on these feature vectors to estimate the baseline accuracy scores that can be obtained with this approach. Achieved validation accuracy scores are shown in Fig. 7. We found that performance of kNN models on *tsfresh* feature vectors is significantly worse than for linear and decision-tree-based models. Furthermore, models trained on the full feature space (accuracy scores $> 70\%$) once again significantly outperformed the global and temporal baseline models (accuracy scores $< 70\%$).

To be consistent with analysis in II-A2, we looked at the feature importance values for the trained random forest classifier and left the 15 features with largest importance ranks. As it was done for the cell type dataset, we applied UMAP embedding algorithm to visualize class separation in two dimensions. Indeed, good class separation can be observed for the WAKE vs. SLEEP activity state classes, as it can be seen in Fig. 8.

Similarly to excitatory vs. inhibitory cell classification, we generated a list of classification-relevant feature groups to determine which feature types are the most discriminative for WAKE vs. SLEEP activity state classification in our case. In

addition to the list of feature types relevant for excitatory vs. inhibitory cell type classification, the following feature types appeared to be specific for WAKE vs. SLEEP activity state classification: i) additional statistics of the ISI series e.g. kurtosis of the DFT spectrum of the series ii) *sample_entropy*, *approximate_entropy* – sample and approximate entropy values, iii) *agg_autocorrelation*, *autocorrelation* – variance and mean of autocorrelation value distribution over different lag values, autocorrelation values for specific lags. The importance of these feature groups hints at the importance of information about the temporal structure of the spike trains in the WAKE vs. SLEEP classification tasks.

III. DISCUSSION

In summary, we have demonstrated good performance of a range of time series analysis models applied to spiking pattern activity classification. The methods described here are very general and can be applied to various tasks from cell type classification to functional state classification of the neural circuit. The latter can cover both different functional states of the same circuit (e.g. WAKE/SLEEP state of the cortex) or disease-induced activity states vs. healthy controls. Detection of disease-driven neural activity might be of high importance in this context, and usage of an ensemble of various predictive models might enable precise detection of such activity patterns. In general, we expect that the approaches discussed here could be applied to a range of classification/clustering tasks involving spiking activity in a straightforward way.

In this study, we were able to achieve good classification results in both cell-type (excitatory vs. inhibitory cells) and circuit-state classification (awake activity vs. activity in different sleep phases) tasks using open-access spiking activity data obtained from frontal cortices of rats. Further work will be focused on incorporating more data from different neuron types and different brain areas. A particularly interesting problem in this context is investigating how the structure of the spike train data in the feature vector space depends on the brain area/cell type, and which spike train representation (embedding) is best at encoding the crucial features which differentiate spiking activity recorded across the brain. These are the topics which are going to be tackled in our future work.

This work was supported by The Russian Science Foundation, agreement No. 18-11-00294.

REFERENCES

- [1] M. Pachitariu, C. Stringer, S. Schröder, M. Dipoppa, L. F. Rossi, M. Carandini, and K. D. Harris, “Suite2p: beyond 10,000 neurons with standard two-photon microscopy,” *Biorxiv*, p. 061507, 2016.
- [2] D. Tsai, E. John, T. Chari, R. Yuste, and K. Shepard, “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, pp. 7510–7513.
- [3] P. Berens, J. Freeman, T. Deneux, N. Chenkov, T. McColgan, A. Speiser, J. H. Macke, S. C. Turaga, P. Mineault, P. Rupprecht *et al.*, “Community-based benchmarking improves spike rate inference from two-photon calcium imaging data,” *PLoS computational biology*, vol. 14, no. 5, p. e1006157, 2018.
- [4] J. J. Jun, C. Mitelut, C. Lai, S. Gratiy, C. Anastassiou, and T. D. Harris, “Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction,” *bioRxiv*, p. 101030, 2017.
- [5] P. Yger, G. L. Spampinato, E. Esposito, B. Lefebvre, S. Deny, C. Gardella, M. Stimberg, F. Jetter, G. Zeck, S. Picaud *et al.*, “Fast and accurate spike sorting in vitro and in vivo for up to thousands of electrodes,” *BioRxiv*, p. 067843, 2016.
- [6] J. I. Glaser, R. H. Chowdhury, M. G. Perich, L. E. Miller, and K. P. Kording, “Machine learning for neural decoding,” *arXiv preprint arXiv:1708.00909*, 2017.
- [7] A. S. Benjamin, H. L. Fernandes, T. Tomlinson, P. Ramkumar, C. VerSteeg, R. H. Chowdhury, L. E. Miller, and K. P. Kording, “Modern machine learning as a benchmark for fitting neural responses,” *Frontiers in computational neuroscience*, vol. 12, 2018.
- [8] B. W. Connors and W. G. Regehr, “Neuronal firing: Does function follow form?” *Current Biology*, vol. 6, no. 12, pp. 1560–1562, 1996.
- [9] T. Tezuka, “Multineuron spike train analysis with r-convolution linear combination kernel,” *Neural Networks*, vol. 102, pp. 67–77, 2018.
- [10] M. D. Humphries, “Spike-train communities: finding groups of similar spike trains,” *Journal of Neuroscience*, vol. 31, no. 6, pp. 2321–2336, 2011.
- [11] M. v. Rossum, “A novel spike distance,” *Neural computation*, vol. 13, no. 4, pp. 751–763, 2001.
- [12] J. D. Victor and K. P. Purpura, “Metric-space analysis of spike trains: theory, algorithms and application,” *Network: computation in neural systems*, vol. 8, no. 2, pp. 127–164, 1997.
- [13] M. Mulansky and T. Kreuz, “Pyspikea python library for analyzing spike train synchrony,” *SoftwareX*, vol. 5, pp. 183–189, 2016.
- [14] T. Tezuka, “Spike train pattern discovery using interval structure alignment,” in *International Conference on Neural Information Processing*. Springer, 2015, pp. 241–249.
- [15] J. Jouty, G. Hilgen, E. Sernagor, and M. Hennig, “Non-parametric physiological classification of retinal ganglion cells,” *bioRxiv*, p. 407635, 2018.
- [16] P. Charlesworth, E. Cotterill, A. Morton, S. G. Grant, and S. J. Eglen, “Quantitative differences in developmental profiles of spontaneous activity in cortical and hippocampal cultures,” *Neural development*, vol. 10, no. 1, p. 1, 2015.
- [17] M. Li, F. Zhao, J. Lee, D. Wang, H. Kuang, and J. Z. Tsien, “Computational classification approach to profile neuron subtypes from brain activity mapping data,” *Scientific reports*, vol. 5, p. 12474, 2015.
- [18] X. Jia, J. Siegle, C. Bennett, S. Gale, D. Denman, C. Koch, and S. Olsen, “High-density extracellular probes reveal dendritic backpropagation and facilitate neuron classification,” *bioRxiv*, p. 376863, 2018.
- [19] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, “The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances,” *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, 2017.
- [20] Y.-S. Jeong, M. K. Jeong, and O. A. Omataomu, “Weighted dynamic time warping for time series classification,” *Pattern Recognition*, vol. 44, no. 9, pp. 2231–2240, 2011.
- [21] B. D. Fulcher and N. S. Jones, “htcsa: A computational framework for automated time-series phenotyping using massive feature extraction,” *Cell systems*, vol. 5, no. 5, pp. 527–531, 2017.
- [22] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, “Time series feature extraction on basis of scalable hypothesis tests (tsfresh—a python package),” *Neurocomputing*, 2018.
- [23] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001.
- [24] J. Lin, E. Keogh, L. Wei, and S. Lonardi, “Experiencing sax: a novel symbolic representation of time series,” *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.
- [25] P. Schäfer and M. Höglqvist, “Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets,” in *Proceedings of the 15th International Conference on Extending Database Technology*. ACM, 2012, pp. 516–527.
- [26] P. Schäfer, “The boss is concerned with time series classification in the presence of noise,” *Data Mining and Knowledge Discovery*, vol. 29, no. 6, pp. 1505–1530, 2015.
- [27] P. Senin and S. Malinchik, “Sax-vsm: Interpretable time series classification using sax and vector space model,” in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 1175–1180.

- [28] K. Sparck Jones, “A statistical interpretation of term specificity and its application in retrieval,” *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.
- [29] P. Schäfer, “Scalable time series classification,” *Data Mining and Knowledge Discovery*, vol. 30, no. 5, pp. 1273–1298, 2016.
- [30] J. Faouzi, “pyts: a Python package for time series transformation and classification,” May 2018. [Online]. Available: <https://doi.org/10.5281/zenodo.1244152>
- [31] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.
- [32] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [33] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.
- [34] F. Karim, S. Majumdar, H. Darabi, and S. Chen, “Lstm fully convolutional networks for time series classification,” *IEEE Access*, vol. 6, pp. 1662–1669, 2018.
- [35] S. Džeroski and B. Ženko, “Is combining classifiers with stacking better than selecting the best one?” *Machine learning*, vol. 54, no. 3, pp. 255–273, 2004.
- [36] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle, “Recurrence plots of dynamical systems,” *EPL (Europhysics Letters)*, vol. 4, no. 9, p. 973, 1987.
- [37] Z. Wang and T. Oates, “Imaging time-series to improve classification and imputation,” *arXiv preprint arXiv:1506.00327*, 2015.
- [38] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel, “Learning to diagnose with lstm recurrent neural networks,” *arXiv preprint arXiv:1511.03677*, 2015.
- [39] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, long short-term memory, fully connected deep neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4580–4584.
- [40] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, “Convolutional lstm network: A machine learning approach for precipitation nowcasting,” in *Advances in neural information processing systems*, 2015, pp. 802–810.
- [41] J. L. Teeters and F. T. Sommer, “Crcns. org: a repository of high-quality data sets and tools for computational neuroscience,” *BMC Neuroscience*, vol. 10, no. S1, p. S6, 2009.
- [42] B. Watson, D. Levenstein, J. Greene, J. Gelinas, and G. Buzsaki, “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.
- [43] B. O. Watson, D. Levenstein, J. P. Greene, J. N. Gelinas, and G. Buzsáki, “Network homeostasis and state dynamics of neocortical sleep,” *Neuron*, vol. 90, no. 4, pp. 839–852, 2016.
- [44] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, NY, USA:, 2001, vol. 1, no. 10.
- [45] L. McInnes, J. Healy, N. Saul, and L. Groberger, “Umap: Uniform manifold approximation and projection,” *The Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.