Adversarial dictionary learning for a robust analysis and modelling of spontaneous neuronal activity

Eirini Troullinou[a,b], Grigorios Tsagkatakis[b], Ganna Palagina[c,d], Maria Papadopouli[a,b], Stelios Manolis Smirnakis[c,d], Panagiotis Tsakalides[a,b]

[a]*Department of Computer Science, University of Crete, Heraklion, 70013, Greece*
[b]*Institute of Computer Science, Foundation for Research and Technology Hellas, Heraklion, 70013, Greece*
[c]*Department of Neurology, Brigham and Women's Hospital, Harvard Medical School,Boston MA 02115*
[d]*Boston VA Research Institute, Jamaica Plain Veterans Administration Hospital, Harvard Medical School, Boston, United States*

Abstract

The field of neuroscience is experiencing rapid growth in the complexity and quantity of the recorded neural activity allowing us unprecedented access to its dynamics in different brain areas. One of the major goals of neuroscience is to find interpretable descriptions of what the brain represents and computes by trying to explain complex phenomena in simple terms. Considering this task from the perspective of dimensionality reduction provides an entry point into principled mathematical techniques allowing us to discover these representations directly from experimental data, a key step to developing rich yet comprehensible models for brain function. In this work, we employ two real-world binary datasets describing the spontaneous neuronal activity of two laboratory mice over time, and we aim to their efficient low-dimensional representation. We develop an innovative, robust to noise, dictionary learning algorithm for the identification of patterns with synchronous activity and we also extend it to identify patterns within larger time windows. The results on the classification accuracy for the discrimination between the clean and the adversarial-noisy activation patterns obtained by an SVM classifier highlight the efficacy of the proposed scheme, and the visualization of the dictionary's distribution demonstrates the multifarious information that we obtain from it.

Keywords: Dictionary Learning, Supervised Machine Learning, biological neural networks.

1. Introduction

The advances of imaging and monitoring technologies, such as in vivo 2-photon calcium imaging at the mesoscopic regime as well as the massive increases in computational power and algorithmic development have enabled advanced multivariate analyses of neural population activity, recorded either sequentially or simultaneously.

More specifically, high resolution optical imaging methods have recently revealed the dynamic patterns of neural activity across the layers of the primary visual cortex (V1), making it possible to apply network analysis methods to this important question: Neuronal groups that fire in synchrony may be more efficient at relaying shared information and are more likely to belong to networks of neurons subserving the same function.

We have preliminarily investigated this question using 2-photon imaging to monitor the spontaneous population bursts of activity in pyramidal cells and interneurons of L2/3 in mouse V1. We found that the sizes of spontaneous population bursts and the degree of connectivity of the neurons in specific fields of view (FOVs) formed scale-free distributions, suggestive of a hierarchical small-world net architecture [1].

The existence of such groups of "linked" units inevitably shapes the profile of spontaneous events observed in V1 networks [2, 3, 4]. Thus, the analysis of the spontaneous activity patterns provides an opportunity for identifying groups of neurons that fire with increased levels of synchrony (have significant "functional connectivity" between each other).

Many recent studies have adopted dimensionality reduction to analyze these populations and to find features that are not apparent at the level of individual neurons. Dimensionality reduction methods produce low-dimensional representations of high-dimensional data preserving or highlighting features of interest. Such methods are typically applied in settings in which the measured variables covary according to a smaller number of explanatory variables. These methods discover and extract these explanatory variables from the high-dimensional data according to an objective that is specific to each method. Typically, any data variance not captured by the explanatory variables is considered to be noise. Dimensionality reduction algorithms have achieved great success in modelling com-

*Fully documented templates are available in the elsarticle package on CTAN.

plex signals, from images [5] to wireless sensor network data [6] and biological neural networks [7].

We specifically adopt dictionary learning methods, which provide a parsimonious description of statistical features of interest via the produced dictionary, discarding at the same time some aspects of the data as noise. Dictionaries are used in various domains such as face recognition [8, 9] and facial expression recognition [10], object tracking [11] etc., as they are a natural approach for performing exploratory data analysis as well as visualization of the data. Moreover, given the fact that the produced dictionary is the new space of reduced dimensionality, the computational complexity of its management is much smaller comparatively to the initial, raw data. We also combine dictionary learning methods with supervised machine learning techniques, which enables us to discriminate the clean from the adversarial-noisy activation patterns, which are the examples that come out when we add artificial noise to clean data. The proposed methodology could be applied to understand cortical circuit function and malfunction in a number of neurological disorders.

More specifically, in order to capture the synchronicity patterns among neurons, we propose the Adversarial Dictionary Learning Algorithm (ADL). We also extend this to the Relaxed Adversarial Dictionary Learning Algorithm (RADL), which captures activation patterns within bigger time window intervals. We employed two real-world binary datasets that depict the neuronal activity of a 9-day old and a 36-day old C57BL/6 laboratory mouse. Data was collected using two-photon calcium imaging in the V1, L2/3 area of the neocortex of the animals. Fig. 1 illustrates the format of our data, where each column represents an example-activation pattern that consists of $0s$ that are the non-firing events while $1s$ represent the firing events. The main aspects that will be addressed in this work are the following:

- Identification of synchronous activity, which refers to a time window of one time bin ($W = 1$) generated by ensembles of neurons. For example Neurons 2, 4 and 6 (yellow boxes) in Fig. 1 fire simultaneously.
- Identification of temporal correlation of the firing of neurons within a time window $W > 1$. For example Neurons 4 and 5 (green boxes) in Fig. 1 are not activated simultaneously but within a time window interval $W = 2$.
- Discrimination between clean and adversarial-noisy activation patterns

The contributions of this work are summarized as follows:

- Acquisition of an interpretable dictionary, i.e. the dictionary elements are essentially part of the input data and the dictionary construction is not a result of a mathematical transformation, as opposed to other methods, such as K-SVD [12] or PCA [13].

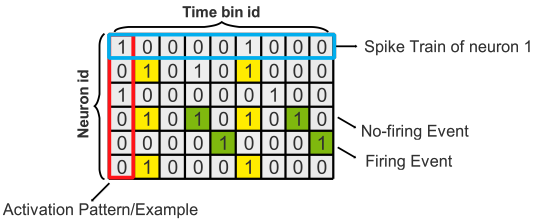

Figure 1: Temporal patterns: Synchronous ($W = 1$) and within larger time windows $W > 1$.

- The dictionary by its construction keeps out patterns, which could be a result of noise. This noise results mainly from calcium fluctuations independent of spiking activity and other sources of imaging noise.
- In contrast to other methods that require a choice of dimensionality K (e.g. the size of the dictionary), here the dictionary size is not a parameter that has to be determined by the user, or be estimated (e.g. based on the choice of arbitrary cutoff values or cross-validation methods [14]).
- Detection of statistically significant synchronous and within a lag temporal patterns of activity, which can be distinguished from shuffled data (i.e. the adversarial-noisy examples) whose temporal correlations are destroyed.

The remainder of the paper is organized as follows: In Section II, we discuss related work on the analysis of neuronal activity in terms of dimensionality reduction. In Section III we describe and analyze the proposed approaches. Evaluation methodology and experimental results are presented in Section IV, while conclusions are drawn in Section V.

2. Related Work

Cell ensembles (or synonymously cell assemblies or cortical patterns-motifs) were originally proposed by Hebb [15] as subsets of synchronously firing neurons to explain brain activity underlying complex behaviors. Multiple studies show evidence of neuronal ensembles and functional subnetworks [1, 16].

Especially in the past two decades where it became possible to record large neuronal populations concurrently [17, 18, 19], methods such as K-SVD [7], Principal Component Analysis (PCA) [20], Independent Component Analysis (ICA) [21] and Non Negative Matrix Factorization (NMF) [22] have been applied to identify neurons repeatedly firing at the same time to find statistically significant ensembles and answer questions about their existence.

The dictionary learning algorithm K-SVD [12], has been used for capturing the behavior of neuronal responses into a small number of representative prototypical signals (i.e.

into the dictionary) and the output dictionary was evaluated with real-world data for its generalization capacity as well as for its sensitivity with respect to noise [7]. Its most severe constraint is that the output dictionary is real-numbered, and when the data consist of binary measurements it is difficult to get insights about the temporal correlations.

PCA, which has been used for the detection of cell ensembles[23], computes the first N_e principal components of the spike matrix and considers those to be the ensembles. Its most severe limitations are that two different ensemble patterns can be merged into a single component and that negative values with no physical meaning are possible in the components.

ICA decomposes a multivariate signal into additive subcomponents assuming that these are non-Gaussian and statistically independent from each other [21]. When used to learn ensembles it overcomes some of the problems of PCA-based methods: Individual neuron-ensemble membership can be recovered easily and neurons belonging to multiple ensembles are also correctly identified [24]. Again negative values are possible in the identified patterns leading to interpretation problems. Santos *et al.* [24] recommend this method for synchronous patterns but temporal correlations with lags or other structures such as synfire chains (synchronous firing chains) cannot be identified by this model.

Diego and Hamprecht [22] use non-negative matrix factorization techniques for the decomposition of binned spike matrix, in order to identify a hierarchical structure of motifs. Again no temporal structure is taken into account and only neurons with synchronous firing activity are considered.

Some common drawbacks that most of the aforementioned approaches have are summarized as follows:

- The number of the neural patterns (i.e. the dimensionality of the new reduced space) needs to be predefined.
- The produced dictionaries are real-valued, which means that in many cases they have no physical meaning.
- Analyses with more complex motifs are missed.

Towards these directions, our approach differs from the related bibliography in the way that was described in Section I, where we summarized the contributions of this work, which surpass the weaknesses of the aforementioned methods. Eventually, given the fact that we use real-world measurements, emphasis is given to the development of robust algorithms.

3. Proposed Dictionary Learning Framework

In this section we present the two proposed dictionary learning methods:

- Adversarial Dictionary Learning Algorithm (ADL) identifies the synchronicity patterns, i.e. patterns where the neurons fire within the same time bin.
- Relaxed Adversarial Dictionary Learning Algorithm (RADL) is the extension of ADL, which gives the potential to detect firing activity within a temporal window of length that is determined by the user.

We also employ a supervised machine learning framework in order to access the learning capacity of the dictionaries that are produced by the two methods as well as their robustness to adversarial noise.

3.1. Adversarial Dictionary Learning Algorithm

The ADL algorithm aims to identify synchronous activation patterns (i.e. patterns whose neurons fire within the same time bin) that exist in the input data and outputs them to a dictionary, which is the new dimensionality reduced space. ADL is an iterative algorithm, which in every iteration selects randomly an example (i.e. an activation pattern) from the data and examines if it will be included in the dictionary or not. Every iteration consists of two stages. In the first stage, the algorithm examines the contribution of the selected example in the representation of the input data, which are the clean examples, and in the second stage it examines the contribution of the example in the representation of noisy data (i.e. data that we have artificially added noise). When these two stages are completed, they are combined in order to determine if the input example will be included in the dictionary or not.

Given a training set $\mathbf{Y}_{clean} \in B^{M\times N}$, where B is the binary set consisting of the values 0 and 1, M is the number of neurons and N the number of clean examples $(y_j)_{j=1}^{N}$, where each one represents an activation pattern (i.e. the activity of all the neurons within one time bin as shown in Fig. 1), we aim to construct a dictionary $\mathbf{D} \in B^{M\times K}$, which at the end of the algorithm will have K dictionary elements that capture the activity among those neurons. Zero columns and those with only one 1-entry (firing of only one neuron within one time bin) have been removed from the training set $\mathbf{Y}_{clean}$, as we are interested only in synchronicity patterns (i.e. when two or more neurons fire simultaneously within the same time bin).

ADL constructs the dictionary $\mathbf{D}$ incrementally, as in every iteration of the algorithm one example $\mathbf{y}_i$ of the set $\mathbf{Y}_{clean}$ is examined as to whether it will be included in the dictionary or not. The algorithm iterates N times (i.e. for each one of the examples $\mathbf{y}_j$ that are in the set $\mathbf{Y}_{clean}$) and stops when all of them are examined. Apart from the dictionary $\mathbf{D}$ that the algorithm will output, it also uses an auxiliary dictionary $\mathbf{D}'$, which in every iteration of the algorithm has all the elements of $\mathbf{D}$ as well as an extra example $\mathbf{y}_i$, which at the current iteration is the example that is examined whether it will be included in the dictionary $\mathbf{D}$ or not. Namely, if at the iteration i, $\mathbf{D} \in B^{M\times k}$

then $\mathbf{D}' \in B^{M\times(k+1)}$. $\mathbf{D}$ is initialized randomly with an example $\mathbf{y}_j$ of the set $\mathbf{Y}_{clean}$ and at the first iteration of the algorithm when the first $\mathbf{y}_i$ is to be examined, dictionaries $\mathbf{D}$ and $\mathbf{D}'$ have the following form:

$$\mathbf{D} = \mathbf{y}_j \quad \text{and} \quad \mathbf{D}' = [\mathbf{D}, \mathbf{y}_i] = [\mathbf{y}_j, \mathbf{y}_i] \tag{1}$$

At the first stage of the algorithm, in order to validate and decide if the example $\mathbf{y}_i$ should be included in the dictionary or not, we also use a set of clean validation examples $\mathbf{V}_{clean} \in B^{M\times(N-1)}$, which consists of all the examples of set $\mathbf{Y}_{clean}$, except the current example $\mathbf{y}_i$ under consideration, namely $\mathbf{V}_{clean} = \left\{(\mathbf{y}_j)_{j=1}^{N-1}, j \neq i\right\}$. According to the sparse representation framework, given the dictionaries $\mathbf{D}$ and $\mathbf{D}'$, where only $\mathbf{D}'$ as it was mentioned before includes the example $\mathbf{y_i}$, we search respectively for the coefficient matrices $\mathbf{X} \in R^{k\times N}$ and $\mathbf{X}' \in R^{(k+1)\times N}$. An approach to this problem is the minimization of the following l_0 norm problems:

$$\min_{\mathbf{X}} ||\mathbf{V}_{clean} - \mathbf{D}\mathbf{X}||_2^2, \quad \text{subject to} \quad ||x_j||_0 \leq T_0 \tag{2}$$

$$\min_{\mathbf{X}'} ||\mathbf{V}_{clean} - \mathbf{D}'\mathbf{X}'||_2^2, \quad \text{subject to} \quad ||x'_j||_0 \leq T_0 \tag{3}$$

where $||\mathbf{x}_j||_0$ and $||\mathbf{x}'_j||_0$ are the l_0 pseudo-norms, which correspond to the number of non-zero elements for every column j of sparse coefficient matrices $\mathbf{X}$ and $\mathbf{X}'$, respectively. The sparsity level T_0 denotes the maximal number of non-zero elements for every column j of sparse coefficient matrices $\mathbf{X}$ and $\mathbf{X}'$, namely each column can have at most T_0 elements. These minimization problems are solved using the OMP Algorithm [25].

Based on the equations (2) and (3), we examine whether $\mathbf{DX}$ or $\mathbf{D}'\mathbf{X}'$, which represent the sets $\mathbf{V}_{clean\_reconstructed}$ and $\mathbf{V}'_{clean\_reconstructed}$ respectively, better approaches the validation set of examples $\mathbf{V}_{clean}$. So, the question that is under discussion is if the example $\mathbf{y}_i$, which is included in $\mathbf{D}'$, helps for the better representation of the set $\mathbf{V}_{clean}$. The metric employed to answer this question is:

$$\mathbf{E}_{clean} = \{\text{RMSE}(\mathbf{V}_{clean}, \mathbf{V}_{clean\_reconstructed})\} \tag{4}$$

$$\mathbf{E}'_{clean} = \{\text{RMSE}(\mathbf{V}_{clean}, \mathbf{V}'_{clean\_reconstructed})\} \tag{5}$$

where RMSE is the root mean squared error. If the representation error $\mathbf{E}'_{clean}$ is smaller than $\mathbf{E}_{clean}$, namely if

$$\mathbf{E}'_{clean} < \mathbf{E}_{clean} \tag{6}$$

this means that the example $\mathbf{y}_i$, which was only included in $\mathbf{D}'$ had indeed an effective result in the representation of the validation set $\mathbf{V}_{clean}$.

We will keep up with the description of the second stage, which is the innovative part of our algorithm and justifies the characterism adversarial that we have given to it. The idea behind this stage of the algorithm is partially inspired from adversarial learning methods [26, 27]. Adversarial training is the process of explicitly training a model on adversarial examples, in order increase its robustness to noisy inputs. Thus, we create an adversarial learning environment by using clean and adversarial-noisy activation patterns aiming to construct a dictionary that will be robust to the measurement noise (i.e. calcium fluctuations) as well as to identifying firing events emerging by chance. The second stage of the algorithm combined with the first stage will determine if the example $\mathbf{y}_i$ will be ultimately added in dictionary $\mathbf{D}$.

More specifically, in order to include the example $\mathbf{y}_i$ in dictionary $\mathbf{D}$, besides its good contribution to the representation of the validation set $\mathbf{V}_{clean}$, it should be simultaneously a non-helpful factor for the representation of an adversarial noisy signal. This aims to the creation of a dictionary that will be robust to noise. In order to achieve this, we create a set of adversarial-noisy examples $\mathbf{Y}_{noisy} \in B^{M\times N}$ by circularly shuffling the spike train of each neuron of the initial set $\mathbf{Y}_{clean}$ by a random number, different for each neuron. Fig. 2 depicts a simple example with five neurons spiking at various time bins showing how the adversarial-noisy signal is created and how the removal of zero columns and those where only one neuron is active (filtering) is performed for both the initial and the noisy signal. Firstly, in order to create the noisy signal, we perform circular shifting to each neuron of the initial signal independently. For example, the spike train of the first neuron is circularly shifted by 2 positions-time units. Accordingly, the spike train of the second neuron is circularly shifted by 5 positions-time units etc. From both the initial and the noisy signal, the zero columns as well as these with a single active neuron are removed (filtering). The advantages of this type of noise over other techniques such as the random flipping of the events is that it preserves the spike distribution of each neuron (firing rate), while it destroys the synchronicity patterns between individual neurons making this type of noise extremely realistic. We also create a validation set of noisy examples $\mathbf{V}_{noisy} \in B^{M\times(N-1)}$, which consists of all the examples included in set $\mathbf{Y}_{noisy}$ except from a random one that is removed so that $\mathbf{V}_{clean}$ and $\mathbf{V}_{noisy}$ have the same number of examples. If $\mathbf{y}_i$ does not aid in the representation of a noisy signal, then the representation error of $\mathbf{V}_{noisy}$ will increase when example $\mathbf{y}_i$ is introduced.

To access the degree to which the example $\mathbf{y}_i$ contributes to the representation of the $\mathbf{V}_{noisy}$ set, the following minimization problems are solved using again the OMP Algorithm:

$$\min_{\mathbf{X}_{noisy}} ||\mathbf{V}_{noisy} - \mathbf{D}\mathbf{X}_{noisy}||_2^2, \quad \text{s.t.} \quad ||x_{j,noisy}||_0 \leq T_0 \tag{7}$$

$$\min_{\mathbf{X}'_{noisy}} ||\mathbf{V}_{noisy} - \mathbf{D}'\mathbf{X}'_{noisy}||_2^2, \quad \text{s.t.} \quad ||x'_{j,noisy}||_0 \leq T_0 \tag{8}$$

Using the same metric as that in Equations (6), we get the

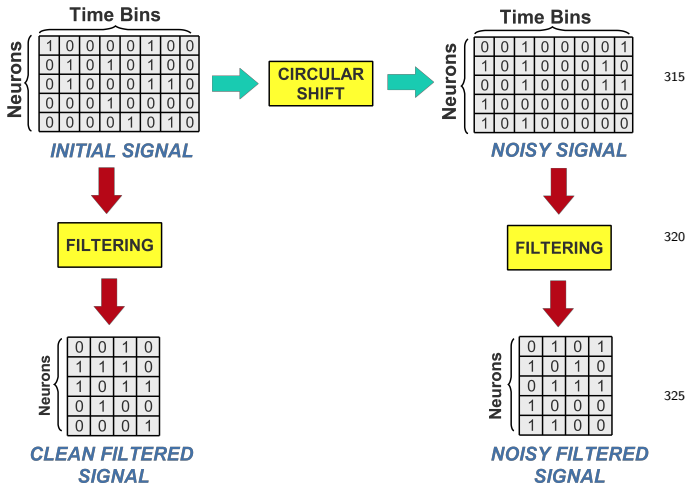

Figure 2: Creation of noisy dataset with circular shift and removal of zero columns and those where only one neuron is active from the initial and the noisy signal (filtering).

following representation errors:

$$\mathbf{E}_{noisy} = \{\text{RMSE}(\mathbf{V}_{noisy}, \mathbf{V}_{noisy\_reconstructed})\} \quad (9)$$

$$\mathbf{E'}_{noisy} = \{\text{RMSE}(\mathbf{V}_{noisy}, \mathbf{V'}_{noisy\_reconstructed})\} \quad (10)$$

This time $\mathbf{E'}_{noisy}$ should be greater than $\mathbf{E}_{noisy}$, namely

$$\mathbf{E'}_{noisy} > \mathbf{E}_{noisy} \quad (11)$$

This would suggest that the example $\mathbf{y}_i$ included in dictionary $\mathbf{D'}$ does not contribute to the good representation of the noisy set of examples $\mathbf{V}_{noisy}$ resulting to a bigger error with its presence in the dictionary $\mathbf{D'}$. That would be exactly the prerequisite for its inclusion in the dictionary $\mathbf{D}$, if we took into account only the second part of our algorithm. Note that the dictionary $\mathbf{D}$ consists of examples only from the set $\mathbf{Y}_{clean}$. The set $\mathbf{V}_{noisy}$, which results from the set $\mathbf{Y}_{noisy}$ is used by the algorithm during the training procedure only in order to determine the appropriateness of the example $\mathbf{y}_i$ in the dictionary $\mathbf{D}$.

Eventually, in order to be determined by the system if the example $\mathbf{y}_i$ will be included in dictionary $\mathbf{D}$, inequalities (6) and (11) are combined in the following way:

$$\frac{\mathbf{E'}_{clean}}{\mathbf{E'}_{noisy} + \epsilon} < \frac{\mathbf{E}_{clean}}{\mathbf{E}_{noisy} + \epsilon} \quad (12)$$

where ϵ is a very small positive quantity so as zero denominators are avoided.

If inequality (12) holds, then $\mathbf{y}_i$ will be also added in dictionary $\mathbf{D}$. Dictionaries $\mathbf{D}$ and $\mathbf{D'}$ would then temporarily be exactly the same, until the next iteration, where another example $\mathbf{y}_i$ would be added in dictionary $\mathbf{D'}$, in order to be examined as to whether it should be eventually included in dictionary $\mathbf{D}$ or not. Otherwise, if

$$\frac{\mathbf{E'}_{clean}}{\mathbf{E'}_{noisy} + \epsilon} \geq \frac{\mathbf{E}_{clean}}{\mathbf{E}_{noisy} + \epsilon} \quad (13)$$

then $\mathbf{y}_i$ is removed from dictionary $\mathbf{D'}$ and it is obviously never included in dictionary $\mathbf{D}$. The algorithm keeps up with selecting randomly the next example $\mathbf{y}_i$ and iterates until all of the examples are examined and a desirable dictionary $\mathbf{D}$ is formed. The procedure that we have described so far is depicted in steps 1-5 of Fig. 3. In step 1 a random example $\mathbf{y}_i$ is selected and the representation errors $\mathbf{E}_{clean}$, $\mathbf{E'}_{clean}$, $\mathbf{E}_{noisy}$ and $\mathbf{E'}_{noisy}$ of stages one and two of the algorithm are computed. Fig. 3 is a snapshot of our algorithm at some iteration j, as $\mathbf{D}$ and $\mathbf{D'}$ are initialized with the example $\mathbf{y}_4$, and the example $\mathbf{y}_2$ was already examined and included in dictionary $\mathbf{D}$, while some other examples may have also been examined but were not included in $\mathbf{D}$. So, at the j^{th} iteration another example $\mathbf{y}_i$ (in blue color) is examined as to whether it will be included in $\mathbf{D}$ or not. Step 2 of Fig. 3 is the combination of stages one and two of our algorithm, i.e. it is the step, where the inclusion of the example y_i in dictionary $\mathbf{D}$ is determined. In step 3, after we have finished with the example $\mathbf{y}_i$ we keep up by selecting randomly the next example $\mathbf{y}_{i+1}$ and the steps 1-2 are repeated again for this example too. Step 4 repeats the steps 1-3 for all the examples that are left and at step 5 we obtain the dictionary $\mathbf{D}$.

In order to report the final dictionary $\mathbf{D}$, the steps 1-5 of Fig. 3 are repeated 4 times-epochs in exactly the same mode that was described previously (we use 4 epochs because as shown and discussed later in Fig. 14, after the third epoch the performance of the algorithm is stabilized). In every epoch of the algorithm the examples in set $\mathbf{Y}_{clean}$ are randomly selected and examined as to whether they will be included in the dictionary or not. Moreover, from the second epoch onward the dictionaries $\mathbf{D}$ and $\mathbf{D'}$ are not initialized with one random example as in the first epoch. Instead, the algorithm initializes both dictionaries $\mathbf{D}$ and $\mathbf{D'}$ with the dictionary $\mathbf{D}$ that was formed in step 5 of the previous epoch, which is essentially used as a baseline for the construction of the next dictionaries.

The reason for introducing the idea of epochs in our algorithm is that in every epoch new examples can be added, which in previous epochs were kept out of the dictionary, because at the time they were selected and examined some other examples with which they could make a good combination were not examined yet, and as a result at that epoch they remained out of the dictionary. Moreover, the use of epochs is a way to examine that the randomness with which the examples are selected, will not change significantly the structure of the dictionary $\mathbf{D}$ in every single epoch. Namely, the examples of set $\mathbf{Y}_{clean}$ that are not included in the dictionary each epoch, only a part of them will be included in the dictionaries of the next epochs and thus, the dictionary formed in the first epoch will have almost the same elements with the dictionary that will be formed in the last epoch (i.e. the 4^{th} epoch). After the completion of these 4 epochs the algorithm terminates and as shown in Fig. 3 we report our final dictionary $\mathbf{D}$. We emphasize once more that the dictionary size does not have to be predefined by the user and the algorithm

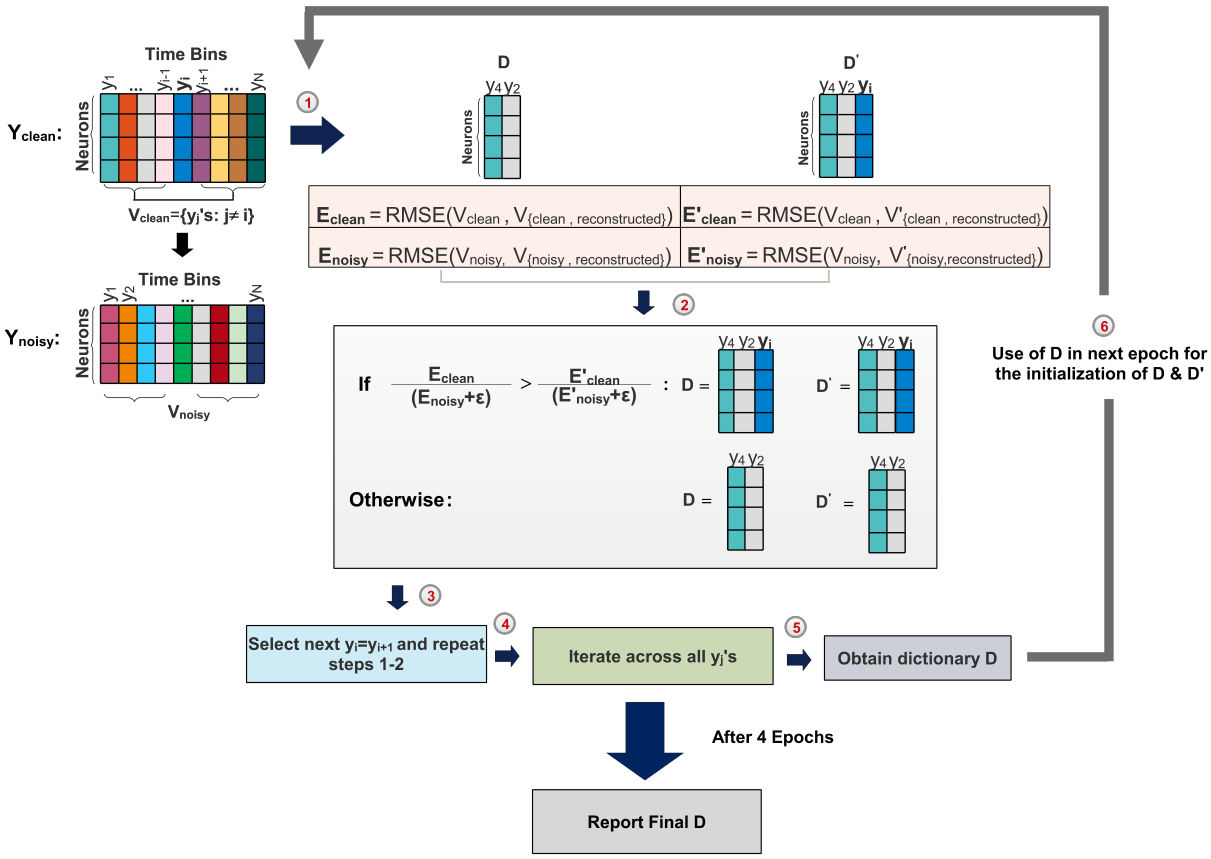

Figure 3: Proposed approach: ADL selects all the appropriate examples of set $\mathbf{Y}_{clean}$ (steps 1-5) and obtains a dictionary $\mathbf{D}$. This procedure (steps 1-5) is repeated 4 times and in every epoch dictionaries $\mathbf{D}$ and $\mathbf{D}'$ are initialized with the dictionary obtained in the previous epoch. After the 4 epochs we report the final $\mathbf{D}$.

decides itself for the number of the dictionary elements-patterns that are sufficient for the effective representation of the data.

3.2. Relaxed Adversarial Dictionary Learning Algorithm

In this section we describe the RADL algorithm, which is the extension of the ADL algorithm that was described in the previous part. In addition to the synchronous activity (i.e. firing activity within the same time bin), RADL can identify temporal patterns within bigger time window intervals and outputs them to a dictionary.

In order to achieve this we define a time-window parameter W, which determines the number of time bins that we will use, in order to search for patterns with some temporal correlation within that interval. Thus, if we define the length of the time-window to be L time bins, we add the content of every L columns-time bins in an overlapping mode. Namely, we sum up the columns $y_1 + y_2 + ... + y_L$, $y_2 + y_3 + ... + y_{L+1}$, $y_3 + y_4 + ... + y_{L+2}$ *etc.* We also normalize all the values that come out from this summation by dividing with the length of the time-window (i.e. by L), so that the values are normalized in the scale $\{0\ 1\}$. The procedure and the idea behind this approach, i.e. the reason why the summing of the columns gives us the possibility to identify temporal patterns within bigger time window intervals is explained with the following example, which is also depicted in Fig. 4. If we define the time window for example to be $W = 2$ time bins, we add the content of every 2 columns-time bins in an overlapping mode as shown in Fig. 4. Namely, we sum up the columns $y_1 + y_2$, $y_2 + y_3$, $y_3 + y_4$ *etc.* and the values that come out from this summation are 0, 1 and 2 (highlighted in blue). The first column of the matrix after the summations indicates that neurons 1, 2 and 3 have some temporal correlation, which is indeed true, as neurons 1, 2 and 3 in the initial signal are activated in consecutive time bins. More specifically, neuron 1 is activated exactly one time bin before neurons 2 and 3, while 2 and 3 are synchronous in the same time bin. In this mode we check temporal correlations among other neurons too. Then, at the normalization step, all values are normalized in the scale $\{0\ 1\}$ by dividing with W so that the and thus, values 0, 0.5, and 1 for $W = 2$ time bins represent:

- 0: Neuron did not fire at all within $W = 2$ time bins
- 0.5: Neuron fired in one of the two time bins
- 1: Neuron fired consecutively at each time bin

Then, at the filtering step, zero columns and those with only one non-zero entry are removed. The same procedure

as it is depicted in Fig. 4 is obviously repeated for the noisy signal too. The summing of the columns in the ini-

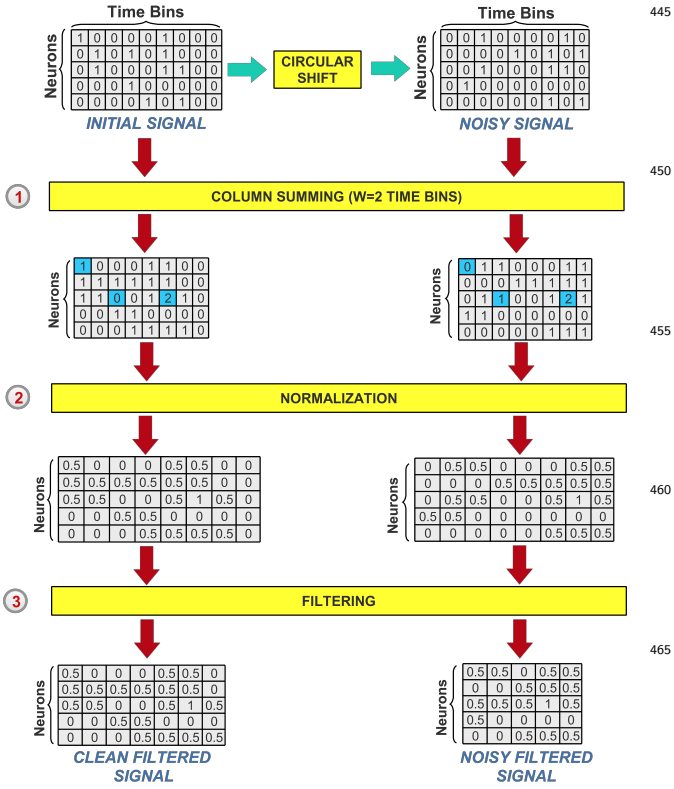

Figure 4: Searching for patterns with temporal correlation within a time window $W = 2$. We sum the signal every 2 columns in an overlapping mode (step 1), we normalize the values (step 2) and we remove zero columns and those where only one neuron is active (step 3), for both initial and noisy signal.

tial signal results to a signal that has less zero columns and columns where only one neuron is active. We can also observe this in Fig. 4, where the initial signal included three zero columns and one column where only the first neuron was active, while after the summing of the columns the signal remained with only one zero column. Thus, during the filtering procedure the amount of columns that are removed is much smaller than before (i.e. when we applied the ADL algorithm and there was no column summing), which results to a training set $\mathbf{Y}_{clean}$ with more examples. Thus, as we increase the time window, the number of columns that have to be removed during the filtering is much smaller, which results to an increase in the number of the examples of each set as shown in Table 1. The increase in the number of the training examples brought also an increase in the size of the dictionary, which RADL outputs and in order to compress it, apart from the training set $\mathbf{Y}_{clean}$, the validation set $\mathbf{V}_{clean}$ and the corresponding noisy sets $\mathbf{Y}_{noisy}$ and $\mathbf{V}_{noisy}$, we also use during the training procedure a testing set $\mathbf{T}_1 \in F^{M \times S}$ of S clean and adversarial-noisy examples, where F is the set of normalized values in scale $\{0\ 1\}$. $\mathbf{T}_1$ is independent from the testing set $\mathbf{T}_2 \in F^{M \times Q}$, where Q is the number of clean and adversarial-noisy examples that will be used in the final step of the algorithm, in order to obtain the final performance of our model.

For the compression of the dictionaries that are produced in every epoch, we use only the clean examples of the set $\mathbf{T}_1$ (the noisy examples of set $\mathbf{T}_1$ are used only after the compression with the clean examples, in order to evaluate the performance of our algorithm in every single epoch). More specifically, in order to compress the dictionary formed in each epoch we remove all the dictionary elements that are not used significantly in the representation of the clean testing examples of the set $\mathbf{T}_1$. So, after the formation of each dictionary $\mathbf{D}$ (step 5 in Fig. 3), and before we use it in the next epoch, we examine how much each dictionary element contributes in the representation of the clean examples of set $\mathbf{T}_1$. The contribution of each dictionary element is measured in the following way: Given the dictionary $\mathbf{D}$ that is formed in the current epoch, we obtain the Coefficient Matrix $\mathbf{X}$, whose columns refer to the clean testing examples of set $\mathbf{T}_1$ described above. For every row-vector i of the Coefficient Matrix $\mathbf{X}$, namely for every x_i that refers to the specific column-vector dictionary element d_i, we calculate its l^2-norm. Then, we sum all the elements of the row-vector x_i and if the summation is smaller or equal with the l^2-norm, then we remove the element d_i from the dictionary. The intuition behind this technique is that we remove all dictionary elements that are used negatively for the representation of most of the examples (i.e. when row-vector x_i has many negative values). Eventually, in the last epoch of the algorithm (i.e. the 4^{th} epoch) we obtain the final dictionary $\mathbf{D}$, which is used with the testing set $\mathbf{T}_2$ that we have available for the testing procedure, in order to evaluate the performance of our model-algorithm.

So, what essentially changes from the ADL algorithm is the input data that we give to the system, where every column-time bin in the new data represents patterns that have a temporal correlation within W time bins. Obviously, this information but in a compressed format is also encoded in the dictionary, providing an insight into temporal correlations. Additionally, during the training procedure of the RADL algorithm, we compress the dictionary of each epoch by removing the dictionary elements that have small contribution in the representation of the clean examples in $\mathbf{T}_1$.

3.3. Evaluation of the dictionary quality

In order to evaluate the quality of the output dictionaries in terms of learning capacity and robustness to noise, we employ a supervised machine learning framework by training an SVM-classifier with the clean and noisy raw data as well as with the reconstructed ones (i.e. the output of DX). We aim to examine the extent to which the classifier can discriminate the clean from the noisy activation patterns, and whether its training with the reconstructed data results to a better classification performance, rather than we use the raw data. Thus, classification performance

is the quantitative metric offering an insight as to extent that the generated dictionary has captured the underlying structure of the data.

4. Performance Analysis

4.1. Dataset Collection

To evalute the merits of the proposed modeling approach, we employed two real-world datasets that were collected using two-photon calcium imaging in the neocortex of a 9-day old mouse and a 36-day old one (C57BL/6). The first dataset of the 9-day old mouse includes 183 neurons of the layer 2/3 of the V1, and neurons were imaged using calcium indicator OGB-1 (imaging depth 130 microns from pia). The dataset of the 36-day old mouse includes 126 neurons of the layer 2/3 of the V1 area. Additionaly, for the 9-day old mouse 29 minutes of spontaneous activity were recorded, comprised of 11970 frames, each of 0.1451 seconds duration, while for the older one the total movie length was 30 minutes comprised of 11972 frames, each of 0.15 seconds duration. The raw fluorescence movie was motion-corrected to remove slow xy-plane drift. After motion correction, we used ImageJ software [28] to draw the ROIs of cells around cell body centers, staying 1-2 pixels from the margin of a cell in the case of the 9-day old mouse, in order to avoid contamination with neuropil signals and 1-2 pixels for the 36-day old mouse. We then averaged the signals of cell ROI pixels and converted them to dF/F [29]. To determine the onsets of spontaneous calcium responses, the dF/F timecourse for each cell was thresholded, using the noise portion of the data, to 3 standard deviations above noise. To make a binary eventogram of the responses, for each cell the frames containing the onsets for this particular cell were assigned the value 1, and all other frames were assigned the value 0. The resulting binary eventogram of all cells was used in subsequent analysis.

4.2. Proposed Approach ADL vs K-SVD

In this section we compare K-SVD, which is an established dictionary learning algorithm with our proposed method ADL for the case of one time bin window interval ($W = 1$). More specifically, we examine which of the two trained dictionaries produced from these two methods is more robust to adversarial noise. In order to quantify this information, we examine the extent to which each trained dictionary can contribute to the discrimination of the clean from the adversarial-noisy activation patterns. Through this analysis the impact of the following parameters is also explored:

- Dictionary size, denoted DS, which is the number of elements considered in the dictionary. While in K-SVD, DS must be defined by the user, in our method, it is automatically inferred.
- Sparsity level, i.e., the maximal number of dictionary elements that are used for representation of the examples.

We also present some more qualitative results of the dictionary that are produced from our proposed method.

4.2.1. Parameter Setup

After the completion of the filtering that is described in Fig. 2, we select 50% of the examples of the clean filtered signal, namely 1138 examples, which will be used by K-SVD for the training of the dictionary. Regarding our proposed method, in order to train the dictionary we select the same 50% examples from the clean filtered signal, as well as 50% of the examples from the noisy filtered signal. Subsequently, the other half of the clean and noisy filtered signal sets will serve as the testing set for each one of the two methods. Namely, they will be used for the training and testing of an SVM-classifier with gaussian kernel and scale $\sigma = 0.01$. The classifier is trained and tested with the:

(i) Raw clean and noisy data
(ii) Reconstructed clean and noisy data, which are binarized by considering all values greater than 0.5 as activations ($1s$), while the rest as zeros.

The number of the testing examples in set $\mathbf{T}_2$ as well as the number of the training examples in set $\mathbf{Y}$, where $\mathbf{Y}$ consists of the clean examples $\mathbf{Y}_{clean}$ and the adversarial-noisy examples $\mathbf{Y}_{noisy}$ for the case of one time bin window interval ($W = 1$) are depicted in Table 1. Note that all sets described in Table 1 ($\mathbf{Y}$, $\mathbf{T}_1$ and $\mathbf{T}_2$) include the number of the clean and the adversarial-noisy examples (i.e. half of the size of each set described in Table 1 refers to the clean examples and the other half refers to the adversarial-noisy examples).

Fig. 5 shows the distribution of the original clean (5 (a)) and of the noisy signal (5 (b)), as it results from the circular shifting procedure. The distributions refer to the activity of the 9-day old mouse before the process of the filtering. Namely, in both figures axis x indicates the size of co-firing neurons (i.e. the number of neurons that co-activate within one time bin) and the log-scaled axis y indicates the number of these patterns that exist in the data. We observe that for the noisy signal, circular shifting has caused a reduction in zero columns-patterns and a simultaneous increase in doublets (i.e. patterns where 2 neurons co-activate within a time bin) as well as in patterns where one neuron is active within a time bin. Finally, more complex patterns with more than seven neurons firing simultaneously are completely destroyed.

4.2.2. Evaluation Results

Fig. 7 illustrates the performance of the SVM-classifier regarding the discrimination between the clean and the noisy signals for the 9-day old mouse, as a function of the sparsity level when the classifier is trained and tested with

Size	$W=1$	$W=2$	$W=3$	$W=4$
Training Set (Y)	2276	2964	3648	4156
Testing Set (T_1)	-	1866	2270	2578
Testing Set (T_2)	2324	2744	3336	3770

Table 1: Sizes of the Sets Y, T_1 and T_2 for all Ws

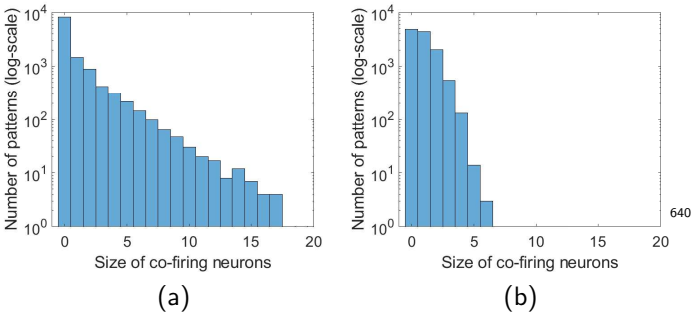

Figure 5: (a) Clean signal distribution (b) Noisy signal distribution

the raw data, the reconstructed data produced by our proposed method ADL and the reconstructed data produced by the K-SVD algorithm. Each point in the errorbar plots corresponds to the mean accuracy of four runs and in every run the examples in the training set are given with a different sequence in terms of the columns (i.e the second column of the training set in the first run may be the fifth column of the training set in the second run). Thus, the K-SVD algorithm is initialized with a different dictionary in every run, as the columns are presented with a different sequence. Regarding our algorithm, the different sequence in the columns of the training set in every run, results to the selection and as a consequence to the examination of the examples with a different sequence as to whether they will be included in the dictionary D or not. The testing set remains the same in all runs. The vertical error bar demonstrates the standard deviation of these for four runs (i.e. how the accuracy of each run differs from the mean accuracy of the four runs). More specifically, as it is illustrated in each subfigure of Fig. 7, we give as input to the K-SVD algorithm a different dictionary size, and we evaluate the performance of the algorithm compared to our proposed method. Fig. 6 depicts the corresponding dictionary sizes that are produced from our method for the case of one time bin window interval ($W=1$). More specifically, for every sparsity level (S.L.), Fig. 6 demonstrates the size of the final dictionary D that is obtained from the 4^{th} epoch for each one of the 4 runs.

We observe in Fig. 7 that when the classifier is trained and tested with the raw data, the accuracy that it achieves is almost 51%. This percentage is quite low and indicates the difficulty of the problem that we are supposed to solve. By using the reconstructed data that are produced by the K-SVD algorithm we observe that the classifier achieves

Run \ S.L.	2	3	4	5	
1	298	422	**50**	68	W=1
2	309	**431**	60	67	
3	315	409	100	91	
4	304	411	68	68	

Figure 6: Size of the final dictionary D for every run and Sparsity Level (S.L).

a better performance with an accuracy of 56% for dictionary size equal to 150 elements and for sparsity level equal to 2. In all of the subfigures we observe that as the sparsity level increases, the accuracy of the classifier decreases, which can be attributed to overfitting of the system. Moreover, the three different dictionary sizes, which were tried as input to the K-SVD algorithm do not affect significantly the performance of the classifier. When we use the reconstructed data that are produced from our method and as depicted in Fig. 7, the classifier achieves better performance results compared to the performance of the K-SVD algorithm. More specifically, we obtain an accuracy of 62% for sparsity level 3 and mean dictionary size (of the 4 runs) equal to 418. We observe that for values of sparsity level greater than 3 the performance deteriorates due to overfitting. Nevertheless, our proposed method gives better performance results for every value of sparsity level. The superiority in the results that are obtained from our method can be put down to the dictionary construction. Namely, as it was described in our proposed method if the candidate dictionary element contributes to the better representation of the noisy activation patterns rather than the clean ones, it is kept out of the dictionary.

As it is already stated, our algorithm executes 4 runs, where in every run the examples of the training set are selected and consequently examined with a different sequence as to whether they will be included in the dictionary or not. These 4 runs are executed in order to examine the sensitivity of our algorithm with respect to the different sequence that the examples are selected. More specifically, we want to examine if the neurons' firing activity captured by the dictionaries of each run is similar or it presents intense variations. To that end, we demonstrate Fig. 8, which depicts the variation in the number of firing events that neurons have across the 4 dictionaries formed in each run, under the consideration of $W=1$ and sparsity level equal to 2. We observe that for most of the neurons (almost 50 neurons) the maximum variation across dictionaries is only 2 firing events, while only one neuron has a variation of 8 firing events. Thus, we end up with 4 dictionaries that have almost the same number of firing events for each neuron, indicating the robustness of our algorithms with respect to the different sequence in the selection of the examples.

Unlike K-SVD, which produces real-numbered dictio-

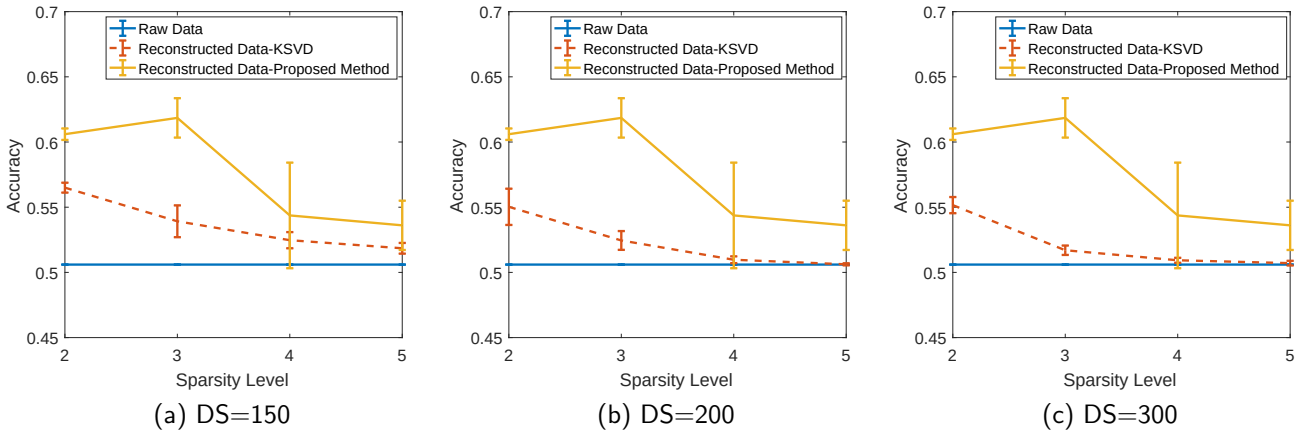

Figure 7: Classification performance when the classifier is trained with the raw data, the reconstructed data produced by our method ADL and the reconstructed data produced by the K-SVD.

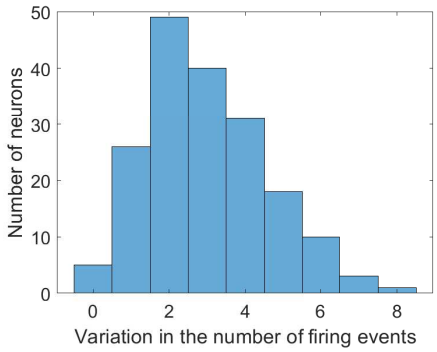

Figure 8: Neurons grouped in the same bin have the same variation in the number of firing events across the 4 dictionaries formed in every run ($W = 1$, Sparsity Level=2).

naries with no physical meaning for our application, our proposed method ADL produces dictionaries that provides us with quantitative as well as qualitative information, giving us an insight about the synchronicity patterns existing in the data. So, Fig. 9 demonstrates the distribution of

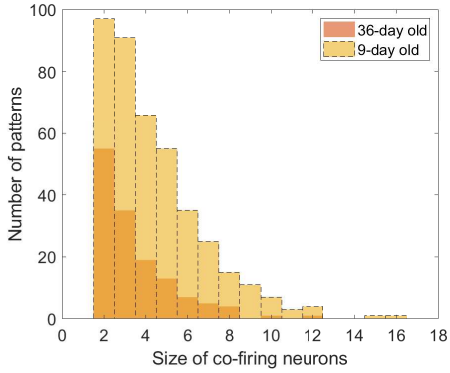

Figure 9: Distribution of the two dictionaries (W=1, Sparsity Level=3).

two dictionaries (we used the dictionaries that were produced from the 4^{th} run of our algorithm, for sparsity level equal to 3) that refer to the spontaneous neuronal activity of a 9-day old and a 36-day old mouse. Namely, axis x indicates the size of the co-firing neurons that exist in the dictionary, i.e. the number of neurons that co-activate within one time bin, such as doublets (when 2 neurons co-activate within one time bin) or triplets (when 3 neurons co-activate within one time bin), etc and axis y indicates the number of these patterns (doublets etc.) that exist in the dictionary. In the original dataset that refers to the 9-day old mouse, firing events occupy the 0.487% of the dataset, while for the 36-day old mouse firing activity occupies only the 0.364% of the dataset. These percentages show the sparseness of our datasets and by extension indicate the low frequency of the neurons' firing activity for both laboratory animals. Moreover, these percentages reveal that the 9-day old mouse has a more intense firing activity, which can be attributed to its young age. All this information is depicted in the distribution of the two trained dictionaries as we observe that the number of the various synchronicity patterns for the 9-day old mouse is greater than the number of patterns for the 36-day old mouse. Additionally, the dictionary that refers to the activity of the 9-day old mouse includes more complex patterns with more than six neurons firing simultaneously, while for the 36-day old mouse such patterns tend to be zero. Eventually, the size of each dictionary also reveals information about the data that we summarize. Namely, the dictionary that refers to the activity of the 9-day old mouse has a size of 411 elements as depicted in Fig. 6, while the dictionary that refers to the older mouse has a size of 51 dictionary elements, which correctly verifies that it fires less.

4.3. RADL

This section demonstrates the analysis for temporal correlation patterns within larger time window intervals

($W > 1$). The analysis assesses the impact of the following parameters:

- Time window interval, denoted W, from which we can extract information about temporal correlations.
- Sparsity level, i.e., the maximal number of dictionary elements that are used for representation.

4.3.1. Parameter Setup

After the completion of the procedure that is described in Fig. 4 we select 40% of the examples of the clean filtered signal, as well as 40% of the examples of the noisy filtered signal for the set $\mathbf{Y}$, which will be used for the training of the dictionary. Then, we select 25% of the examples of the clean filtered signal for the set $\mathbf{T}_1$, which will be used for the compression of the dictionaries that are produced in every epoch as well as 25% of the examples of the noisy filtered signal in order to evaluate the performance of our algorithm at every epoch of each run. Eventually, the other 35% of the clean and noisy filtered examples will be used by the set $\mathbf{T}_2$ and will serve as the testing set, whose half of the examples will be used for the training of an SVM-classifier with gaussian kernel and scale $\sigma = 0.01$ and the other half will be used for the testing of the classifier. The number of the training examples in set $\mathbf{Y}$, as well as the number of the testing examples in sets $\mathbf{T_1}$ and $\mathbf{T}_2$ for all the time window intervals are depicted in Table 1. As it was also stated in the parameter setup section of ADL, all sets described in Table 1 ($\mathbf{Y}$, $\mathbf{T}_1$ and $\mathbf{T}_2$) include the number of the clean and the adversarial-noisy examples (i.e. half of the size of each set described in Table 1 refers to the clean examples and the other half refers to the adversarial-noisy examples). The classifier is trained and tested with the:

(i) Raw clean and noisy data
(ii) Reconstructed clean and noisy data whose values are processed as we describe in the following example

As it was described in section III, for the cases of time window intervals, where $W > 1$, activation patterns are not represented by the values 0 and 1 due to the summing of the columns and the normalization step. For example in the case of $W = 3$, if one neuron has not fired at all within 3 consecutive time bins, we get a 0-event. If it has fired once, we obtain the normalized value of $\frac{1}{3}$, which are the most prevalent values with the 0 value. Additionally, if the neuron has fired twice, we obtain the value $\frac{2}{3}$ and if it has fired consecutively in all of the 3 time bins, we obtain a 1-event, which is not very common due to the refractory period. Because of the fact that we deal with a reconstruction problem, reconstructed values other than those described before may appear. Thus, without loss of generality we make the simplification, which is depicted in Fig. 10. Namely, for $W = 3$ all values which are smaller than $\frac{1}{6}$ are turned into zero. Values in space $\left[\frac{1}{6}, \frac{1}{2}\right)$ are turned into $\frac{1}{3}$ and values in space $\left[\frac{1}{2}, \frac{5}{6}\right)$ are turned into $\frac{2}{3}$. Any other value is turned into 1. Accordingly, we work for any time window W.

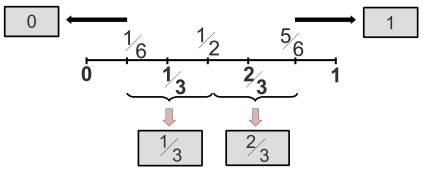

Figure 10: Processing the values of the reconstructed events.

4.3.2. Evaluation Results

Fig. 11 illustrates the performance of the SVM-classifier regarding the discrimination between the clean and the noisy signals for the 9-day old mouse, as a function of the sparsity level when the classifier is trained and tested with the raw data and the reconstructed data produced by our method, i.e. the RADL algorithm. Each point in the errorbar plots corresponds to the mean performance of the four runs of the algorithm, where in every run the examples in the training set are selected and examined with a different sequence as to whether they will be included in the dictionary D or not. The vertical error bar demonstrates the standard deviation of these four runs. More specifically, as it is illustrated in Fig. 11, each subfigure refers to the performance of the classifier for different time window intervals. When the classifier is trained and tested with the raw data, the highest accuracy that it achieves, taking into account all the time windows is 51%, which is a quite low percentage. When we use the reconstructed data that are produced from our proposed method, we observe that as we increase the time window interval, we obtain a better classification performance. More specifically, for sparsity level equal to 5 and for time windows $W = 3$ and $W = 4$ we obtain the highest accuracy performance equal to 65%. Moreover, as opposed to $W = 1$, we notice that for time window intervals $W > 1$, when the sparsity level is increased, the classification performance is increased too. The summing of the columns that we apply to the initial signal data (Fig. 4) for time windows $W > 1$ results to the replacement of zero columns and columns where only one neuron is activated, with columns where two or more neurons co-activate. As a result, during the filtering procedure the number of columns that are removed is much smaller compared to the removal of columns in the case of time window $W = 1$. Thus, as it is also depicted in Table 1 the number of examples-activation patterns are increased, which results to an increase in the size of the trained dictionaries too, as it is also depicted in Fig. 12. As a result, there is also an increase in the complexity of the patterns that appear in the data and by extension in the dictionaries for the cases of $W > 1$. Thus, by increasing the sparsity level, we increase the generalization capacity of our algorithm by allowing it to use more dictionary elements in order to represent the data. On the

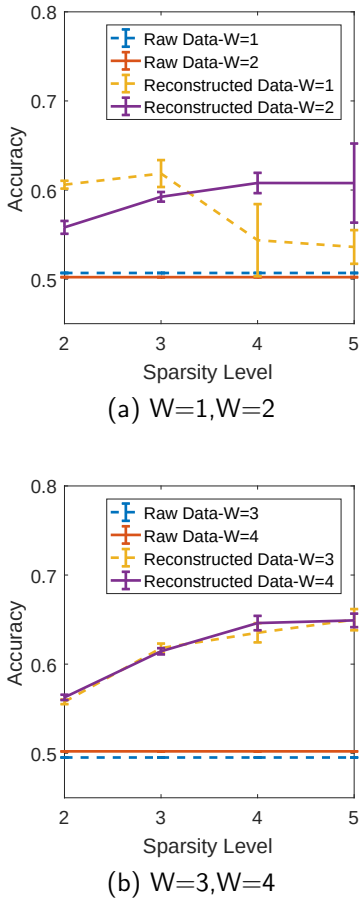

Figure 11: Classification performance when the classifier is trained with the raw data and the reconstructed data produced by RADL with respect to different time window intervals.

Run \ S.L.	2	3	4	5
1	304	386	452	410
2	299	395	437	472
3	**281**	392	455	**482**
4	285	380	430	469

W=2

Run \ S.L.	2	3	4	5
1	**330**	637	528	539
2	352	643	517	544
3	340	636	506	547
4	347	**645**	505	554

W=3

Run \ S.L.	2	3	4	5
1	**349**	506	565	**606**
2	364	508	542	570
3	**349**	494	541	579
4	355	479	523	547

W=4

Figure 12: Size of the final dictionary D for every run and Sparsity Level (S.L)

contrary, for the case of $W = 1$, the dictionary consists of representative patterns, thus increasing the sparsity level leads to worse performance due to overfitting, as shown in Fig. 11.

Fig. 14 illustrates the classification performance that is obtained in every epoch of the algorithm for all the runs and for sparsity level equal to value 3. We observe that for all the cases of time windows the classification performance is either improved or it remains the same in every epoch of the algorithm. Thus, as it is depicted in Fig. 14 the dictionary that is obtained in the 4^{th} epoch of each run, ensures the best possible accuracy performance for the specific run compared to the dictionaries that are formed in the previous epochs.

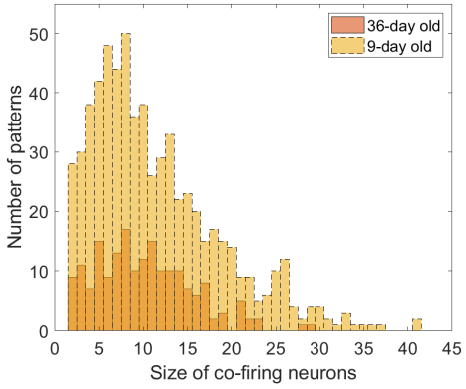

Figure 13: Distribution of the two dictionaries (W=3, Sparsity Level=3).

Fig. 13 demonstrates the distribution of the two dictionaries (we used the dictionaries that were produced from the 4^{th} run of our algorithm) that refer to the spontaneous neuronal activity of the 9-day old and the 36-day old mouse under the consideration of $W = 3$ and sparsity level equal to 3. The figure demonstrates the number of various patterns (doublets, triplets etc.) firing within a temporal window of 3 time bins that exist in each dictionary. As in the case of $W = 1$, we observe that the number of the various synchronicity patterns for the 9-day old mouse is greater than the number of the patterns for the 36-day old mouse. Additionally, the dictionary that refers to the activity of the 9-day old mouse includes more complex patterns with more than 20 neurons having a temporal correlation within 3 time bins, while such patterns appear in much smaller numbers for the 36-day old mouse. Finally, the size of each dictionary also reveals information about the data that we summarize. The dictionary that refers to the activity of the 9-day old mouse has greater size than the dictionary that refers to the activity of the 36-day old mouse, which correctly indicates and verifies that it fires less.

5. Conclusions and Future Work

The world around us, complex as it is, is relatively low-dimensional: the familiar visual scenes made up of textures, faces, buildings, and other objects are highly structured. As it is commonly believed that the developed brain contains an internal model of the environment that it expresses through its structure and activity, it is expected that this model should be similarly highly structured, and that the dimensionality reduction characteriz-

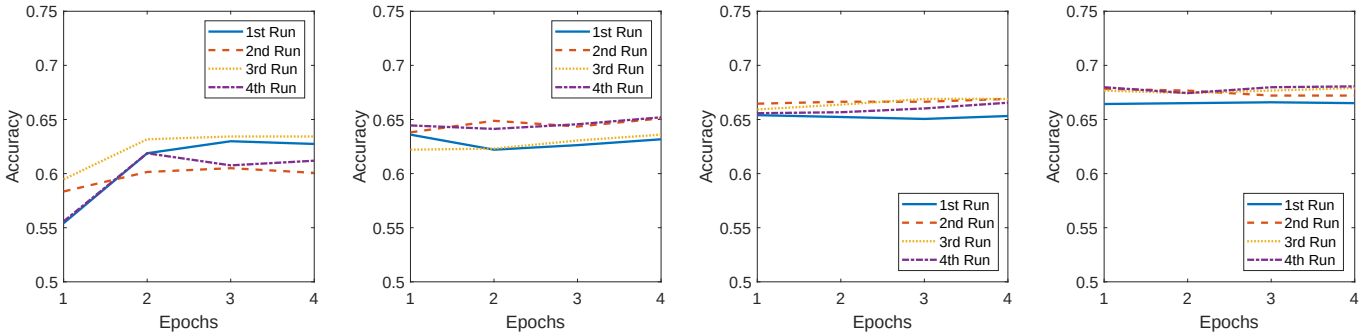

Figure 14: Classification performance with respect to the epochs of the algorithm for each run (Sparsity Level=3).

ing the brain's activity might be related to intrinsic properties of sensory stimuli and motor output.

In this work we employed dictionary learning methods that were applied on real-world data that refer to the spontaneous neuronal activity of a 9-day old and a 36-day old mouse over time. We used the reconstructed signals that were produced by those methods, in order to train and test an SVM-Classifier for the discrimination of the true from the noisy activation patterns. In this work we developed an adversarial dictionary learning framework that is robust to noise and as a consequence it can discriminate the clean from the noisy activation patterns, which in contrast to state-of-the-art K-SVD, produces an interpretable dictionary. Moreover, when the classifier is trained with the reconstructed signals of our proposed method, we obtain a better classification performance. We also extended the idea to a more relaxed approach, the RADL algorithm, which produces a dictionary that captures patterns within bigger time window intervals and is not restricted to the synchronous activity of neurons within the same time bin. Experimental results demonstrate that increasing the activation patterns time window, has a positive effect on the classification performance.

Future work will focus on the extension of our algorithm with graph signal processing methods, which could provide insights related to the temporal dynamics of the network as well as its functional network activities. We also plan to explore the potential of the proposed method in characterizing normal brain organizations as well as alterations due to various brain-disorders, such as schizophrenia, autism, and Alzheimer's disease.

Acknowledgment

This research is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS-5000432), implemented by the State Scholarships Foundation (IKY). S.M.S. received support from the Simons Foundation SFARI Research Award No. 402047, NEI Grant RO1-EY-109272, and NINDS R21 NS096640. This work is also supported from the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat for Research and Technology under grant agreement No. 2285, Erasmus+ International Mobility between University of Crete and Harvard Medical School 2017-1-EL01-KA107-035639, and the Marie Curie RISE NHQWAVE project under grant agreement No. 4500.

References

[1] G. Palagina, J. F. Meyer, S. M. Smirnakis, Inhibitory units: An organizing nidus for feature-selective sub-networks in area v1, Journal of Neuroscience (2019) 2275–18.

[2] J.-e. K. Miller, I. Ayzenshtat, L. Carrillo-Reid, R. Yuste, Visual stimuli recruit intrinsically generated cortical ensembles, Proceedings of the National Academy of Sciences 111 (38) (2014) E4053–E4061.

[3] T. Kenet, D. Bibitchkov, M. Tsodyks, A. Grinvald, A. Arieli, Spontaneously emerging cortical representations of visual attributes, Nature 425 (6961) (2003) 954.

[4] A. Luczak, P. Barthó, K. D. Harris, Spontaneous events outline the realm of possible sensory responses in neocortical populations, Neuron 62 (3) (2009) 413–425.

[5] M. Yang, H. Chang, W. Luo, Discriminative analysis-synthesis dictionary learning for image classification, Neurocomputing 219 (2017) 404–411.

[6] G. Tsagkatakis, P. Tsakalides, Dictionary based reconstruction and classification of randomly sampled sensor network data, in: Sensor Array and Multichannel Signal Processing Workshop (SAM), 2012 IEEE 7th, IEEE, 2012, pp. 117–120.

[7] E. Troullinou, G. Tsagkatakis, G. Palagina, M. Papadopouli, S. M. Smirnakis, P. Tsakalides, Dictionary learning for spontaneous neural activity modeling, in: 2017 25th European Signal Processing Conference (EUSIPCO), IEEE, 2017, pp. 1579–1583.

[8] Z. Lu, L. Zhang, Face recognition algorithm based on discriminative dictionary learning and sparse representation, Neurocomputing 174 (2016) 749–755.

[9] H. Nguyen, W. Yang, B. Sheng, C. Sun, Discriminative low-rank dictionary learning for face recognition, Neurocomputing 173 (2016) 541–551.

[10] K. Yan, W. Zheng, Z. Cui, Y. Zong, T. Zhang, C. Tang, Unsupervised facial expression recognition using domain adaptation based dictionary learning approach, Neurocomputing 319 (2018) 84–91.

[11] T. Zhou, F. Liu, H. Bhaskar, J. Yang, H. Zhang, P. Cai, Online discriminative dictionary learning for robust object tracking, Neurocomputing 275 (2018) 1801–1812.
[12] M. Aharon, M. Elad, A. Bruckstein, K-svd: An algorithm for designing overcomplete dictionaries for sparse representation, IEEE Transactions on signal processing 54 (11) (2006) 4311–4322.
[13] I. T. Jolliffe, Principal component analysis and factor analysis, in: Principal component analysis, Springer, 1986, pp. 115–128.
[14] J. P. Cunningham, M. Y. Byron, Dimensionality reduction for large-scale neural recordings, Nature neuroscience 17 (11) (2014) 1500.
[15] D. O. Hebb, The organization of behavior: A neuropsychological theory, Psychology Press, 2005.
[16] W. Singer, Synchronization of cortical activity and its putative role in information processing and learning, Annual review of physiology 55 (1) (1993) 349–374.
[17] G. Buzsáki, Large-scale recording of neuronal ensembles, Nature neuroscience 7 (5) (2004) 446.
[18] I. H. Stevenson, K. P. Kording, How advances in neural recording affect data analysis, Nature neuroscience 14 (2) (2011) 139.
[19] M. B. Ahrens, M. B. Orger, D. N. Robson, J. M. Li, P. J. Keller, Whole-brain functional imaging at cellular resolution using light-sheet microscopy, Nature methods 10 (5) (2013) 413.
[20] J. K. Chapin, M. A. Nicolelis, Principal component analysis of neuronal ensemble activity reveals multidimensional somatosensory representations, Journal of neuroscience methods 94 (1) (1999) 121–140.
[21] P. Comon, Independent component analysis, a new concept?, Signal processing 36 (3) (1994) 287–314.
[22] F. D. Andilla, F. A. Hamprecht, Learning multi-level sparse representations, in: Advances in Neural Information Processing Systems, 2013, pp. 818–826.
[23] M. A. Nicolelis, L. A. Baccala, R. Lin, J. K. Chapin, Sensorimotor encoding by synchronous neural ensemble activity at multiple levels of the somatosensory system, Science 268 (5215) (1995) 1353–1358.
[24] V. Lopes-dos Santos, S. Ribeiro, A. B. Tort, Detecting cell assemblies in large neuronal populations, Journal of neuroscience methods 220 (2) (2013) 149–166.
[25] J. A. Tropp, A. C. Gilbert, Signal recovery from random measurements via orthogonal matching pursuit, IEEE Transactions on information theory 53 (12) (2007) 4655–4666.
[26] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, J. Tygar, Adversarial machine learning, in: Proceedings of the 4th ACM workshop on Security and artificial intelligence, ACM, 2011, pp. 43–58.
[27] P. Stone, M. Veloso, Towards collaborative and adversarial learning: A case study in robotic soccer, International Journal of Human-Computer Studies 48 (1) (1998) 83–104.
[28] M. Abramoff, P. Magalhaes, S. Ram, Image processing with imagej. biophotonics int. 11: 36–42, Google Scholar.
[29] C. Stosiek, O. Garaschuk, K. Holthoff, A. Konnerth, In vivo two-photon calcium imaging of neuronal networks, Proceedings of the National Academy of Sciences 100 (12) (2003) 7319–7324.