Adaptive Waveform Learning: A Framework for Modeling Variability in Neurophysiological Signals

Sebastian Hitziger, Maureen Clerc, Sandrine Saillet, Christian Bénar, Théodore Papadopoulo

Adaptive Waveform Learning: A Framework for Modeling Variability in Neurophysiological Signals

Sebastian Hitziger, Maureen Clerc, Sandrine Saillet, Christian Bénar, and Théodore Papadopoulo

***Abstract*—When analyzing brain activity such as local field potentials (LFP), it is often desired to represent neural events by stereotypic waveforms. Due to the non-deterministic nature of the neural responses, an adequate waveform estimate typically requires to record multiple repetitions of the neural events. It is common practice to segment the recorded signal into event-related *epochs* and calculate their average. This approach suffers from two major drawbacks: (i) epoching can be problematic, especially in the case of overlapping neural events and (ii) variability of the neural events across epochs (such as varying onset latencies) is not accounted for, which may lead to a distorted average.**

In this paper, we propose a novel method called *adaptive waveform learning* (AWL). It is designed to learn multi-component representations of neural events while explicitly capturing and compensating for waveform variability, such as changing latencies or more general shape variations. Thanks to its generality, it can be applied to both epoched (i.e., segmented) and continuous (i.e., non-epoched) signals by making the corresponding specializations to the algorithm. We evaluate AWL's performance and robustness to noise on simulated data and demonstrate its empirical utility on an electrophysiological recording containing intracranial epileptiform discharges (*epileptic spikes*).

***Index Terms*—dictionary learning, epileptiform discharges, local field potential (LFP), sparse representations, signal variability, single-trial analysis**

I. Introduction

When analyzing neurophysiological recordings such as local field potentials (LFP), it is common practice to average over a large number of experimental trials in order to obtain a stereotypic waveform representing the neural activity. This approach, however, does not account for cross-trial variability of the waveforms, such as varying onset latencies or changing shapes, and can thus lead to a distorted representation of the neural responses. In addition, some information about the trial-specific waveform variations might be lost in the average.

Different methods have been proposed to explicitly account for waveform variability. One of the first is Woody's iterative method (1967) [1] which detects different waveform latencies in order to calculate realigned averages. However, this method assumes identical waveform shapes across the different trials, which is often not observed in practice. An alternative consists in modeling the neural events as multi-component waveforms, as done by principal component analysis (PCA) [2] and independent component analysis (ICA) [3]. Especially ICA has proved a valuable tool for separating multi-channel electroencephalogry (EEG) recordings into components representing different active brain sources by assuming their statistical independence [4], [5]. Multilinear techniques, such as parallel factor analysis (PARAFAC), allow to decompose multi-channel EEG recordings into components with multiple dimensions, such as space, time, and frequency [6]. However, as the techniques described above rely on a linear framework, they require isochronicity of the multi-variate input signal. This is a reasonable assumption if the input components are synchronously acquired signals from different recording channels, but it typically does not hold for different experimental trials. The more recent method, differentially variable component analysis (dVCA), combines features of PCA/ICA and Woody's method by extending the linear multi-component framework to include latency variability of each waveform component and has been applied to LFP [7] and EEG recordings [8].

Another approach for analyzing brain signals consists in sparse representations calculated by techniques such as matching pursuit (MP) [9] or least angle regression (LARS) [10]. These methods allow the detection of features taken from a *dictionary*, a predefined and often overcomplete set of *atoms* (i.e., basis waveforms). MP has been applied to EEG data in [11], and extensions have been proposed to specifically address multi-channel [12], [13] and multi-trial [14], [15] data. A drawback of sparse coding techniques is the fact that the *optimal* dictionary is often unknown *a priori* and typical choices such as the symmetric Gabor wavelets [14] may not well represent the neural events. A remedy consists in learning the dictionary directly from the data [16], a popular technique especially in the image processing community [17], [18]. Dictionary learning has furthermore been extended to translation-invariant settings [19], allowing to explicitly account for variable latencies of neural events [20]–[22].

In many neurophysiological applications, it is common practice to segment a long continuous recording into event-related *epochs* in a preprocessing step to facilitate further analysis. However, this epoching step can be problematic if the latencies of the neural events are not exactly known *a priori*. In addition, in the case of overlapping neural events, epoching may lead to significant errors. Optimally, a method should thus be capable of processing the recording as a whole, which requires a model that allows repetitions of the neural events at different latencies.

In this work, we introduce a new framework, called *adaptive waveform learning* (AWL), to learn single- or multi-

S. Hitziger, M. Clerc, and T. Papadopoulo are with the Athena team at Inria Sophia Antipolis, France, e-mail: sebastian.hitziger@gmx.de

S. Saillet and C. Bénar are with the Institut de Neurosciences des Systèmes, UMR 1106 INSERM, Aix-Marseille Université, Faculté de Médecine La Timone, Marseille, France

component neural representations from single-channel recordings. The novelty of AWL is the explicit modeling of signal variability such that each component waveform is subject to variations across the neural events. The AWL model is first presented and analyzed in a very general framework with the possibility to consider arbitrary morphological waveform changes. Then, two concrete algorithms, E-AWL and C-AWL, are derived from this framework to address the processing of both epoched (i.e., segmented) and continuous (i.e., non-epoched) recordings, respectively. For these cases, we limit the waveform variability to amplitude and latency changes, as well as linear temporal scaling (i.e., dilations). Both algorithms are designed to progressively *learn* the different waveforms, and their implementations are based on sparse coding and dictionary learning techniques.

The E-AWL algorithm is evaluated on simulated signal epochs and compared to ICA and the translation-invariant dictionary learning algorithm MoTIF [20]. Finally, E-AWL and C-AWL are applied to an LFP recording containing epileptiform discharges (*spikes*), providing interesting insight into the spikes' variability across the dataset.

II. Modeling the neural events

We start by presenting some commonly used models to represent events in neurological recordings, corresponding to the methods described above. We then show how these models can be generalized in order to cope with different types of signal variability, leading to the *adaptive waveform learning* (AWL) model.

A. Existing models

Let $\{\mathbf{x}_m \equiv \mathbf{x}_m(t) \in \mathbb{R}^T\}_{m=1}^M$ denote a set of one-dimensional signal *epochs* (i.e., event-related signal segments) from a single recording channel. Woody's method [1] assumes an underlying neural event $\mathbf{d} \equiv \mathbf{d}(t) \in \mathbb{R}^T$, which occurs across the epochs with variable latencies δ_m. This leads to

$$\mathbf{x}_m = \mathbf{d}(\cdot - \delta_m) + \epsilon_m, \quad m = 1, \ldots, M, \tag{1}$$

where we use "$\cdot$" to denote the (implicit) time argument for a compact notation and $\epsilon_m \equiv \epsilon_m(t) \in \mathbb{R}^T$ describes noise terms. In contrast, PCA [2] and ICA [3] model the neural event through multiple waveform components $\{\mathbf{d}_k \equiv \mathbf{d}_k(t)\}_{k=1}^K$ in a linear framework,

$$\mathbf{x}_m = \sum_{k=1}^{K} a_{km}\mathbf{d}_k + \epsilon_m, \tag{2}$$

with coefficients $a_{km} \in \mathbb{R}$. While PCA maximizes the explained variance and imposes orthogonality among the $\mathbf{d}_k$, ICA assumes statistical independence of the components $\mathbf{d}_k$. Note that while a full PCA/ICA calculates $K = T$ components, in the applications considered in this paper, we are only interested in the first $K < T$ waveform components. Combination of models (1) and (2) leads to

$$\mathbf{x}_m = \sum_{k=1}^{K} a_{km}\mathbf{d}_k(\cdot - \delta_m) + \epsilon_m, \tag{3}$$

which is the underlying model of dVCA [7]. Note that in the models above, each component waveform $\mathbf{d}_k$ occurs at most once per epoch $\mathbf{x}_m$. In order to include repetitions, we can add another sum over different translations δ_p corresponding to the time samples in each $\mathbf{x}_m$,

$$\mathbf{x}_m = \sum_{k=1}^{K}\sum_{p=1}^{P} a_{kpm}\mathbf{d}_k(\cdot - \delta_p) + \epsilon_m. \tag{4}$$

The set of all KP translated waveforms may be very large and overcomplete if $KP > T$. Hence, the coefficients a_{kpm} should be *sparse*, i.e., $a_{kpm} = 0$ for most triplets (k, p, m). This sparse model underlies translation-invariant dictionary learning techniques, where the $\mathbf{d}_k$ are often called *kernels* or generating functions. The translation-invariant dictionary then contains all shifted versions of these kernels. An example for an application to EEG recordings is reported in [20], where the authors introduce the translation-invariant dictionary learning algorithm MoTIF.

B. AWL model

The idea of the technique presented in this paper is the efficient modeling of the neural events through a small set of kernels $\mathbf{d}_k$ which are sufficiently adaptive to capture the variability across signal epochs. For this purpose, we extend model (4) by including dilations (i.e., linear temporal scaling), which can account, for instance, for changing signal durations and varying frequencies. This yields

$$\mathbf{x}_m = \sum_{k=1}^{K}\sum_{p=1}^{P}\sum_{q=1}^{Q} a_{kpqm}\frac{1}{\sqrt{\gamma_q}}\mathbf{d}_k\left(\frac{1}{\gamma_q}(\cdot - \delta_p)\right) + \epsilon_m. \tag{5}$$

Note that the idea of multi-scale approaches in dictionary learning is not entirely new: in [23], the authors learn dictionaries of image patches with blocks of different sizes, and the technique presented in [24] makes use of predefined scale-invariant wavelets. However, in neither approach is the learned dictionary itself scale-invariant.

In model (5), every neural event is represented as an *instantiation* of a kernel $\mathbf{d}_k$ at a specific temporal location δ_p and with a specific duration γ_q and amplitude a_{kpqm}. This is illustrated in Fig. 1. In addition to capturing the waveform amplitudes, the coefficients a_{kpqm} have a crucial role in *selecting* the relevant waveforms. That is, each non-zero coefficient a_{kpqm} denotes an *occurrence* of a neural event given through $(\delta_p, \gamma_q, \mathbf{d}_k)$ in an epoch $\mathbf{x}_m$. It is therefore essential for the coefficients to be sparse, since we do not expect neural events to occur at every possible time instant (or with every possible dilation parameter). The specific way of imposing this sparsity on the coefficients will be treated in the following sections.

We note that model (5), has many unknown parameters (i.e., the kernels $\mathbf{d}_k$, their parameters δ_p, γ_q, and their variable amplitudes a_{kpqm}), which bears the risk of overfitting the problem and makes interpretation difficult. For the applications considered in this paper, we will therefore never address (5) in its full complexity, but instead specialize it to different settings.

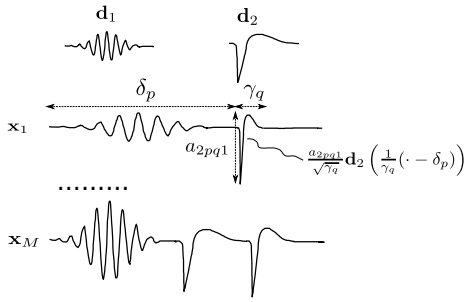

Fig. 1: The AWL framework (5) models each neural event in the signals $\mathbf{x}_m$ as the instantiation of a kernel $\mathbf{d}_k$ through a specific latency δ_p, duration γ_q, and amplitude a_{kpqm}. Note that a kernel may be used multiple times in the same signal $\mathbf{x}_m$ to model repeating events (bottom row).

Model abstraction: Before deriving the concrete algorithms to calculate the AWL model parameters, we formulate model (5) in an abstract form

$$\mathbf{x}_m = \sum_{k=1}^{K}\sum_{l=1}^{L} a_{klm}\phi_l(\mathbf{d}_k) + \epsilon_m, \quad m = 1,\dots,M, \tag{6}$$

where the operators ϕ_l may represent translations, dilations, and their compositions, but can also describe more general morphological deformations. Note that the model parameters in (6) that need to be learned are only the coefficients a_{klm} and the kernels $\mathbf{d}_k$, whereas the finite set $\{\phi_l\}_{l=1}^{L}$ is defined *a priori*. As mentioned above, the non-zero coefficients a_{klm} thus fulfil the role of selecting the relevant operators ϕ_l.

The abstract model (6) has the advantage that the following analysis is not limited to translations and dilations. In fact, we will only require the operators ϕ_l to be (i) linear[1], and (ii) invertible (or at least of high rank). As a result, this analysis will produce an algorithm *template*, which may be used in future work to implement other morphological waveform changes. For example, more general rescaling of the time axis, as addressed in dynamic time warping, may be used to generalize the translations and dilations.[2] In addition to its higher generality, formulation (6) is more compact, which facilitates the following analysis.

Note that there is an indeterminacy in (6) due to scaling ambiguities. In order to capture the waveforms' energies (i.e., their l_2-norms) exclusively by the coefficients a_{klm}, we constrain both the operators ϕ_l and the kernels $\mathbf{d}_k$ to be normalized, i.e.,

$$|||\phi||| = 1 \quad \text{with } |||\phi||| \stackrel{\text{def}}{=} \max_{\|\mathbf{d}\|_2=1} \phi(\mathbf{d}) \quad \text{and} \tag{7}$$

$$\|\mathbf{d}_k\|_2 = 1, \tag{8}$$

where $\|\cdot\|_2$ denotes the l_2-norm. Another indeterminacy consists in the order of the kernels $\mathbf{d}_k$, which will be addressed in the hierarchical learning approach at the end of Section III-A.

[1]Linearity means that $\phi_l\left(\sum_{k=1}^{K} a_k\,\mathbf{d}_k\right) = \sum_{k=1}^{K} a_k\,\phi_l(\mathbf{d}_k)$ holds for any sets of kernels $\{\mathbf{d}_1,\dots,\mathbf{d}_K\}$ and coefficients $\{a_1,\dots,a_K\}$. In particular, translations and dilations are linear operations.

[2]Any temporal rescaling can be represented by a linear operator ϕ.

III. Minimization problem and proposed algorithms

Based on model (6), we formulate a minimization problem in order to learn the kernels $\mathbf{d}_k$, as well as their instantiations in the data, given through the coefficient vector $\mathbf{a} \equiv \{a_{klm}\}$ and the selected operators ϕ_l. For this purpose, we first remain in a general setting, resulting in a *template* algorithm which will then be implemented for an epoched signal setting (E-AWL) and a continuous (i.e., non-epoched) setting (C-AWL).

A. AWL template algorithm

As discussed in the previous section, model (6) is only useful for interpretation if the coefficient vector $\mathbf{a} \equiv \{a_{klm}\}$ is sparse. This sparsity will be induced through the following exclusivity constraint in order to prevent neural events from being detected multiple times by similar instantiated kernels: we impose that an instantiation of a kernel $\mathbf{d}_k$ may exclude certain other instantiations of $\mathbf{d}_k$ (or other similar kernels $\mathbf{d}_{k'}$) in the same signal $\mathbf{x}_m$, which can be expressed as $a_{klm} \neq 0 \;\Rightarrow\; a_{k'l'm} = 0$ for appropriate index tuples (k,k',l,l'). In Sections III-B and III-C, we will give concrete implementations of this constraint, which we denote as $\mathcal{C}(\mathbf{a})$ throughout this section. Since neural events typically occur with the same polarity within the recordings, it is furthermore reasonable to assume non-negativity of the coefficients $\mathbf{a} \geq 0$. This reduces the parameter space of the optimization problem, and the following algorithms can ensure this constraint without an increase in computational complexity.

AWL problem: First note that the finite set of operators $\Phi = \{\phi_l\}_{l=1}^{L}$ is not directly learned, but instead determined *a priori*, e.g., as the set of permitted translations and dilations. The relevant operators corresponding to the neural events are implicitly selected from this set through the non-zero coefficients a_{klm}. Hence, the unknowns in model (6) are only the coefficient vector $\mathbf{a} \equiv \{a_{klm}\}$ and the kernels $\mathbf{d}_k$. Taking into account coefficient sparsity and non-negativity as described above, as well as the normalization of the kernels (8), we formulate the minimization problem

$$\min_{\mathbf{a},\{\mathbf{d}_k\}} \sum_{m=1}^{M} \left\| \mathbf{x}_m - \sum_{k=1}^{K}\sum_{l=1}^{L} a_{klm}\phi_l(\mathbf{d}_k) \right\|_2^2, \tag{9}$$

$$\text{s.t.} \quad \|\mathbf{d}_k\|_2 = 1 \quad \text{for all } k, \tag{10}$$

$$\mathbf{a} \geq 0, \tag{11}$$

$$\mathcal{C}(\mathbf{a}), \tag{12}$$

Note that in the case of $\Phi = \{\text{id}\}$ this problem is similar to the dictionary learning problem [16], where in our case, sparsity is induced through $\mathcal{C}(\mathbf{a})$. This non-convex joint optimization problem is often solved through alternating minimization: starting with an initial set $\{\mathbf{d}_k\}$, the coefficients $\mathbf{a}$ and the kernels $\mathbf{d}_k$ are iteratively updated in separate steps. We adapt this alternating framework to account for the operators ϕ_l and the constraints (11), (12).

Coefficient update: The coefficient update consists in the minimization of (9)–(12) with respect to the coefficients $\mathbf{a}$, while leaving the kernels $\mathbf{d}_k$ and the operators ϕ_l fixed. The

operators ϕ_l can thus be eliminated from the cost function by applying them to the kernels $\mathbf{d}_k$, i.e., by creating the dictionary $\mathbf{D} = \{\mathbf{d}_k^l\}$ with atoms $\mathbf{d}_k^l \stackrel{\text{def}}{=} \phi_l(\mathbf{d}_k)$. Since $\mathbf{D}$ is fixed and there are no dependencies of the coefficients $\mathbf{a}$ across different epochs $\mathbf{x}_m$, the resulting minimization can be performed separately for each $\mathbf{x}_m$. Hence, for each $m = 1, \ldots, M$, we have to solve

$$\underset{\{a_{\cdot\cdot m}\}}{\operatorname{argmin}} \left\| \mathbf{x}_m - \sum_{k=1}^{K} \sum_{l=1}^{L} a_{klm} \mathbf{d}_k^l \right\|_2^2, \tag{13}$$

$$\text{s.t.} \quad \mathbf{a} \geq 0, \tag{14}$$

$$\mathcal{C}(\mathbf{a}). \tag{15}$$

While (13) is an ordinary least squares problem, many choices for the exclusity constraint $\mathcal{C}(\mathbf{a})$ result in a problem (13)–(15) that is non-convex. For its solution, we will make use of sparse coding techniques. In particular, we shall focus on matching pursuit (MP) [9] and least angle regression shrinkage (LARS) [10]. The advantage of MP and LARS is that both algorithms iteratively select *active* atoms (i.e., those with non-zero coefficients) that have maximal dot product with the data. After each selection step, MP subtracts the contribution of the activated atom from the current residual signal and performs the following iteration on the updated residual for the remaining atoms. In contrast, LARS never fully subtracts an atom's contribution. Instead, it keeps track of all activated atoms and can deactivate a selected atom in a later step. While in some cases LARS produces better solutions, it also has a higher computational complexity.

As LARS and MP proceed in successive activation steps, constraints (14), (15) can easily be ensured: First, we impose coefficient non-negativity by selecting only atoms which have positive dot product with the data. For LARS, this variant is also mentioned in [10]. Second, after each activation of some coefficient a_{klm}, we exclude from later selection those coefficients $a_{k'l'm}$ which would violate $\mathcal{C}(\mathbf{a})$.

Note that LARS is typically used to solve the Lasso problem [25] which includes l_1-regularization. In the following Sections III-B and III-B we will provide concrete implementations for (13)–(15) and will describe how LARS can be used without l_1-regularization.

Kernel update: Minimizing (9)–(12) for the kernels $\mathbf{d}_k$ while leaving the coefficients and the operators fixed is a convex problem since the constraints (11), (12) only concern the coefficients $\mathbf{a}$. We can efficiently solve it through block coordinate descent, i.e., by performing loops through the index set $k \in \{1, \ldots, K\}$ and minimizing for each $\mathbf{d}_k$ separately. For each k, we thus have to solve

$$\underset{\mathbf{d}_k}{\operatorname{argmin}} \sum_{m=1}^{M} \left\| \mathbf{x}_m - \sum_{\substack{k'=1 \\ k' \neq k}}^{K} \sum_{l=1}^{L} a_{k'lm} \phi_l(\mathbf{d}_{k'}) - \sum_{l=1}^{L} a_{klm} \phi_l(\mathbf{d}_k) \right\|_2^2, \tag{16}$$

while leaving all $\mathbf{d}_{k'}$ with $k' \neq k$ fixed. Since we assume the operators ϕ_l to be linear (cf. Section II-B), we can define operators

$$\psi_{k'm} \stackrel{\text{def}}{=} \sum_{l=1}^{L} a_{k'lm} \phi_l \tag{17}$$

and rewrite (16) as

$$\underset{\mathbf{d}_k}{\operatorname{argmin}} \sum_{m=1}^{M} \left\| \mathbf{x}_m - \sum_{\substack{k'=1 \\ k' \neq k}}^{K} \psi_{k'm}(\mathbf{d}_{k'}) - \psi_{km}(\mathbf{d}_k) \right\|_2^2.$$

The $\psi_{k'm}$ are fixed and known at this point, so we can differentiate the minimization term with respect to $\mathbf{d}_k$ and write the necessary condition for a minimum. This yields the closed form solution

$$\mathbf{d}_k \leftarrow \left(\sum_{m=1}^{M} \psi_{km}^t \psi_{km} \right)^+ \left(\sum_{m=1}^{M} \psi_{km}^t(\mathbf{r}_{km}) \right), \tag{18}$$

with ψ_{km}^t denoting the adjoint operators, $(\cdot)^+$ the Moore–Penrose pseudoinverse, and

$$\mathbf{r}_{km} = \mathbf{x}_m - \sum_{\substack{k'=1 \\ k' \neq k}}^{K} a_{k'm} \psi_{k'm}(\mathbf{d}_{k'})$$

the residual of the signal $\mathbf{x}_m$ after subtraction of all but kernel $\mathbf{d}_k$'s contribution. The kernel update (18) thus describes a generalized average over these residuals, with the adjoint operations $\psi_{km}^t(\mathbf{r}_{km})$ performing realignments. Invertibility and numerical conditioning of the operator $\sum \psi_{km}^t \psi_{km}$ are discussed later in the concrete applications.

After numerical convergence of the iterative updates (18), which we generally observed already after one cycle through the index set $\{1, \ldots, K\}$, the $\mathbf{d}_k$ are normalized in order to ensure constraint (10). Note that this normalization of the $\mathbf{d}_k$ requires a corresponding adjustment of the coefficients, which is automatically done in the next coefficient update.

Hierarchical learning: Contrary to other dictionary learning applications where dictionaries are typically large, for the applications addressed in this paper, we are interested in learning only a small number K of kernels. This makes it feasible to hierarchically learn representations of growing cardinalities K: First a representation with a single kernel $\mathbf{d}_1$ is learned. Then, a second kernel $\mathbf{d}_2$ is initialized and learning is repeated on the set $\{\mathbf{d}_1, \mathbf{d}_2\}$. This process is repeated until a maximal representation size K_{max} is reached. The advantage of this approach is that we obtain a set of representations with different cardinalities, whose comparison can give interesting insight. In addition, it allows us to determine the optimal representation size K *a posteriori* whose choice is often a difficult task. We only need to ensure that K_{max} is chosen sufficiently large, which may depend on the task and the desired interpretation. For all applications shown in this paper, we found $K_{max} = 5$ to be sufficient.

Note that the hierarchical learning approach also provides an ordering of the kernels, where the last kernels are the ones most recently added to the learning process. This avoids the ordering indeterminacy described in Section II-B.

Implementation: The alternating minimization scheme presented above provides the bricks for an algorithm solving (9)–(12). For a concrete implementation, however, it is necessary to make several additional specifications, including (i) the choice of the operators ϕ_l, (ii) the exact formulation of the exclusivity constraint $\mathcal{C}(\mathbf{a})$, (iii) the choice between MP and LARS in the coefficient update, and (iv) the initialization of the kernels $\mathbf{d}_k$. The choices for (i)–(iv) should be carefully adapted to the specific applications. This is done for two applications in the following sections, leading to the algorithms E-AWL and C-AWL.

Algorithm 1 Hierarchical E-AWL

Input: $\{\mathbf{x}_m\}_{m=1}^M$, $\{\delta_p\}_{p=-P}^P$, $K_{max} \in \mathbb{N}$.

for $K = 1$ to K_{max} **do**
  Initialize $\mathbf{d}_K$ with white Gaussian noise.
  loop
    $\{a_{kpm}\} \leftarrow$ COEFF_UPDATE$(\{\mathbf{x}_m\}, \{\delta_p\}, \{\mathbf{d}_k\}_{k=1}^K)$.
    Drop index p, keeping only the non-zero coefficients a_{km} and their corresponding latencies δ_{km}.
    if stopping criterion reached: **break**.
    $\{\mathbf{d}_k\} \leftarrow$ KERNEL_UPDATE$(\{\mathbf{x}_m\}, \{a_{km}\}, \{\delta_{km}\}, \{\mathbf{d}_k\})$.
  end loop
  Save representation $R_K \leftarrow (\{a_{km}\}, \{\delta_{km}\}, \{\mathbf{d}_k\})_{k=1}^K$.
end for

Output: $R_1, \ldots, R_{K_{max}}$.

procedure COEFF_UPDATE$(\{\mathbf{x}_m\}, \{\delta_p\}, \{\mathbf{d}_k\})$
  Create a dictionary $\mathbf{D} = \{\mathbf{d}_k^p\}$ with $\mathbf{d}_k^p = \mathbf{d}_k(\cdot - \delta_p)$.
  for $m = 1$ to M **do**
    Solve through LARS-0:
    $$\{a_{\cdot\cdot m}\} \leftarrow \underset{\{a_{\cdot\cdot m}\}}{\operatorname{argmin}} \left\| \mathbf{x}_m - \sum_{k=1}^K \sum_{p=-P}^P a_{kpm} \mathbf{d}_k^p \right\|_2^2,$$
    s.t. $\mathbf{a} \geq 0$ and
    $\forall\, k : \|(a_{k(-P)m}, \ldots, a_{kPm})\|_0 \leq 1.$
  end for
end procedure, **return** a

procedure KERNEL_UPDATE$(\{\mathbf{x}_m\}, \{a_{km}\}, \{\delta_{km}\}, \{\mathbf{d}_k\})$
  for $k \in \{1, \ldots, K\}$ **do**
    $$\mathbf{d}_k \leftarrow \sum_{m=1}^M a_{km} \mathbf{r}_{km}(\cdot + \delta_{km}),$$
    with $\mathbf{r}_{km} = \mathbf{x}_m - \sum_{k' \neq k} a_{k'm}(\mathbf{d}_{k'}(\cdot - \delta_{km}))$.
  end for
  for $k = 1$ to K **do**
    $\mathbf{d}_k \leftarrow \mathbf{d}_k(\cdot - \bar{\delta}_k)$, with $\bar{\delta}_k = \frac{\sum_m a_{km}\delta_{km}}{\sum_m a_{km}}$.
    $\mathbf{d}_k \leftarrow \mathbf{d}_k / \|\mathbf{d}_k\|_2$.
  end for
end procedure, **return** $\{\mathbf{d}_k\}$

B. Epoched AWL

The general AWL problem (9)–(12) includes the possibility of repeating neural events within a single $\mathbf{x}_m$ by allowing several instantiations of each kernel $\mathbf{d}_k$. In this section, we assume each neural event to occur at most once per signal epoch $\mathbf{x}_m$. In addition, we limit the variability to translations about $\delta_p \in \{\delta_{-P}, \ldots, \delta_P\}$ of the kernels.[3] The values for δ_{-P} and δ_P determine the maximal shifts to the left and right, respectively, and can be used to control the permitted amount of latency variability. The general AWL problem (9)–(12) can now be specialized to

$$\min_{\mathbf{a}, \{\mathbf{d}_k\}} \left(\sum_{m=1}^M \left\| \mathbf{x}_m - \sum_{k=1}^K \sum_{p=-P}^P a_{kpm} \mathbf{d}_k(\cdot - \delta_p) \right\|_2^2 \right) \quad (19)$$

$$\text{s.t.} \quad \|\mathbf{d}_k\|_2 = 1 \quad \text{for all } k, \quad (20)$$

$$\mathbf{a} \geq 0, \quad (21)$$

$$\|(a_{k(-P)m}, \ldots, a_{kPm})\|_0 \leq 1 \quad \text{for all } k, m. \quad (22)$$

Note that the exclusivity constraint $\mathcal{C}(\mathbf{a})$ is specified by the l_0-constraint (22) which allows at most one instantiation of each kernel $\mathbf{d}_k$ per epoch $\mathbf{x}_m$.

Now we can use the alternating minimization scheme from the previous section to derive the concrete hierarchical E-AWL algorithm. Its pseudocode is given in Algorithm 1, followed by the routines COEFF_UPDATE and KERNEL_UPDATE. In the following, we will discuss this algorithm in detail.

Kernel initialization: In order to learn the kernels $\mathbf{d}_k$ blindly with the least possible bias, we suggest random initialization of the $\mathbf{d}_k$ with white Gaussian noise. Alternatively, inital kernels can be extracted from the data or calculated in a preprocessing step, e.g., by performing a PCA or ICA. However, note that due to the non-convexity of the problem, this bears the risk of converging to a local minimum close to the initialization.

Coefficient update: The minimization of (19)–(22) w.r.t. the coefficients is summarized in the routine COEFF_UPDATE, which we solve using a modification of the LARS algorithm denoted as LARS-0. Standard LARS [10] is designed to solve the Lasso problem

$$\underset{\{a_{\cdot\cdot m}\}}{\operatorname{argmin}} \left\| \mathbf{x}_m - \sum_{k=1}^K \sum_{p=-P}^P a_{kpm} \mathbf{d}_k^p \right\|_2^2 + \lambda \|\mathbf{a}\|_1,$$

with $\lambda \geq 0$ denoting a regularization parameter and the l_1-norm being defined as $\|\mathbf{a}\|_1 \overset{\text{def}}{=} \sum_{k,p,m} |a_{kpm}|$. In fact, a special feature of LARS is its ability to calculate the full regularization path, that is, the solution $\mathbf{a}$ for any parameter $\lambda \geq 0$. For sufficiently large λ', this solution is $\mathbf{a} \equiv 0$. When decreasing λ', certain entries a_{kpm} in the solution vector $\mathbf{a}$ will successively become active, that is, change from zero to non-zero. However, once activated, an entry may become deactivated again on the further regularization path. In our modification LARS-0, we exclude (reinclude) after each activation (deactivation) all entries $a_{kp'm}$, corresponding to translates of the activated (deactivated) kernel, from later activation. This ensures the l_0-constraint (line 7 in COEFF_UPDATE) and is illustrated in Fig. 2. The non-negativity constraint (line 6) is implemented by only activating

[3]This model already proved sufficiently rich for the following epoched applications, and adding dilation invariance did not provide better results. Dilation invariance is therefore explicitly studied only in the continuous setting in Section III-C.

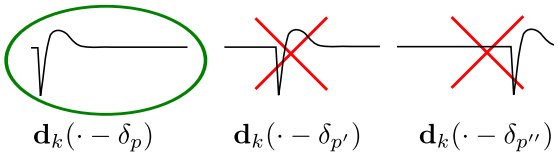

Fig. 2: The l_0-constraint (22) can easily be enforced when performing sparse coding with the LARS algorithm: After each activation of a kernel (green circle) in one LARS step, we exclude all other translations of this kernel from later activation (red crosses). In contrast, if an already active kernel becomes deactivated, its previously excluded translates become available again for activation.

entries a_{kpm} if these become positive, otherwise keeping them zero (cf. Section III-A). Since the problem in lines 5–7 of COEFF_UPDATE does not contain an l_1-regularization term, we calculate the regularization path until $\lambda = 0$. While this use of LARS may seem unconventional, it has two important advantages: (i) While the unconstrained problem in line 5 of COEFF_UPDATE could also be solved by an ordinary least squares solver, ensuring the additional constraints in lines 6,7 is non-trivial, but can be conveniently handled in LARS's regularization path. (ii) When considering only the least squares problem in line 5, following the LARS path until $\lambda = 0$ does in fact provide the exact solution, whereas matching pursuit (MP) would only calculate an approximation.

Kernel update: As ensured by constraint (22), there is maximally one non-zero coefficient a_{kpm} per epoch $\mathbf{x}_m$ and kernel $\mathbf{d}_k$. We can thus drop the index p, denoting by a_{km} only the non-zero coefficients and by δ_{km} the corresponding latencies. The operators $\psi_{k'm}$ defined in the previous section in (17) thus reduce to

$$\psi_{k'm}(\mathbf{d}_{k'}) = a_{k'm}\mathbf{d}_{k'}(\cdot - \delta_{k'm}).$$

Since translations are orthogonal operators, we have $\psi_{k'm}^t\psi_{k'm} = a_{k'm}^2 \cdot \mathrm{id}$, and the update formula (18) furthermore simplifies to

$$\mathbf{d}_k \leftarrow \left(\frac{1}{\sum_{m=1}^M a_{km}^2}\right) \sum_{m=1}^M a_{km}\mathbf{r}_{km}\left(\cdot + \delta_{km}\right), \quad \text{where}$$
$$\mathbf{r}_{km} = \mathbf{x}_m - \sum_{\substack{k'=1 \\ k'\neq k}}^K a_{k'm}(\mathbf{d}_{k'}(\cdot - \delta_{km})),$$

resulting in realigned averages of the residual signals $\mathbf{r}_{km}$.

Note that the absolute temporal positions of the kernels are arbitrary in the sense that we could obtain an equivalent representation by slightly shifting a kernel $\mathbf{d}_k$ and correspondingly adjusting the detected latencies δ_{km}. In order to lift this indeterminacy, we fix the absolute position of each kernel through the alignment

$$\mathbf{d}_k \leftarrow \mathbf{d}_k(\cdot - \bar{\delta}_k), \tag{23}$$

where

$$\bar{\delta}_k \stackrel{\text{def}}{=} \frac{\sum_{m=1}^M a_{km}\delta_{km}}{\sum_{m=1}^M a_{km}}$$

describes the weighted mean of the previously detected shifts δ_{km}. The new position of the kernel $\mathbf{d}_k$ thus represents the mean latency of its instantiations, ensuring that it can make optimal use of the permitted latencies $\{\delta_{-P}, \ldots, \delta_P\}$: Suppose, to the contrary, that a kernel $\mathbf{d}_k$ is shifted to the left in most of its instantiations (resulting, e.g., from random initialization). Then $\mathbf{d}_k$ would not be able to detect events located on the extreme left, i.e., farther than the maximally allowed shift δ_{-P}. This issue can be improved through the realignment (23).

As before, the kernel updates conclude with normalization in order to meet constraint (20). The steps above are summarized in the procedure KERNEL_UPDATE.

Note that the kernel realignment and normalization need to be compensated by making corresponding adjustments to the latencies δ_{km} and the coefficients a_{km}, respectively. This is automatically done in the next coefficient update in Algorithm 1, which is why the stopping criterion is placed below the routine COEFF_UPDATE rather than KERNEL_UPDATE.

C. Continuous AWL

In many applications, the epoched trials addressed in the previous section result from the segmentation of a continuous signal. Such an epoching step can be problematic, especially if the latencies of the neural events are not exactly known or if their waveforms overlap. Therefore, we now present an approach for directly processing a single continuous signal that contains repetitions of neural events.

In the present setting, we will consider both translations and dilations. By making the appropriate specializations to the general AWL problem (9)–(12), we obtain

$$\min_{\mathbf{a},\{\mathbf{d}_k\}} \left\| \mathbf{x} - \sum_{k=1}^K \sum_{p=1}^P \sum_{q=-Q}^Q a_{kpq} \tfrac{1}{\sqrt{\gamma_q}} \mathbf{d}_k\left(\tfrac{1}{\gamma_q}(\cdot - \delta_p)\right) \right\|_2^2 \tag{24}$$
$$\text{s.t.} \quad \|\mathbf{d}_k\|_2 = 1 \quad \text{for all } k, \tag{25}$$
$$a_{kpq} < \rho \Rightarrow a_{kpq} = 0 \quad \text{for all } k, p, q, \tag{26}$$
$$a_{kpq} \neq 0 \Rightarrow a_{kp'q'} = 0 \quad \text{if } |\delta_{p'} - \delta_p| < \Delta. \tag{27}$$

We note that $\mathbf{x}$ now denotes a single long signal, whereas the kernels $\mathbf{d}_k$ are defined on shorter domains. Each translation $\mathbf{d}_k \mapsto \mathbf{d}_k(\cdot - \delta_p)$ is thus implemented by shifting $\mathbf{d}_k$ to the time point δ_p in the signal domain and then zero-padding. Contrary to the previous section where we limited the maximal shifts δ_{-P} and δ_P, we now include translations $\{\delta_1, \ldots, \delta_P\}$ over the entire signal $\mathbf{x}$, allowing kernels to be instantiated at any time sample. We use logarithmically spaced dilations $\gamma_q \in \{\gamma_{-Q}, \ldots, \gamma_Q\}$, with maximal compression and stretch given by γ_{-Q} and γ_Q, respectively.

Note that we replaced coefficient non-negativity by the stronger constraint (26) with a given threshold $\rho > 0$. This ensures that events are only detected if the corresponding kernels have sufficiently large correlation with the data.

The constraint (27) implements the exclusivity constraint $\mathcal{C}(\mathbf{a})$ for this case. It ensures that different instantiations of a kernel do not fully overlap: with l denoting the length of the kernels, the maximally allowed overlap is $(l - \Delta)/l$. This overlap limitation is frequently used in translation-invariant dictionary learning (see for instance [19]) and is due to the fact that waveforms which are slightly shifted are similar to themselves, i.e., have high dot product. Without controlling

Algorithm 2 Hierarchical C-AWL

Input: $\mathbf{x}$, $\{\delta_p\}_{p=1}^{P}$, $\{\gamma_q\}_{q=-Q}^{Q}$, $0<\alpha<1$, $K_{max}\in\mathbb{N}$.

1: **for** $K=1$ to K_{max} **do**
2: Initialize $\mathbf{d}_K$ with data segment in $\mathbf{x}$.
3: **loop**
4: $\mathbf{a}\leftarrow$ COEFF_UPDATE$(\mathbf{x},\{\delta_p\},\{\gamma_q\},\{\mathbf{d}_k\}_{k=1}^{K},\alpha)$.
5: **if** stopping criterion reached: **break**.
6: $\{\mathbf{d}_k\}\leftarrow$ KERNEL_UPDATE$(\mathbf{x},\mathbf{a},\{\delta_p\},\{\gamma_q\},\{\mathbf{d}_k\})$.
7: **end loop**
8: Save representation $R_K\leftarrow(\mathbf{a},\{\delta_p\},\{\gamma_q\},\{\mathbf{d}_k\}_{k=1}^{K})$.
9: **end for**

Output: $R_1,\dots,R_{K_{max}}$.

1: **procedure** COEFF_UPDATE$(\mathbf{x},\{\delta_p\},\{\gamma_q\},\{\mathbf{d}_k\},\alpha)$
2: Create $\mathbf{D}=\{\mathbf{d}_k^{pq}\}$ with $\mathbf{d}_k^{pq}=\frac{1}{\sqrt{\gamma_q}}\mathbf{d}_k\left(\frac{1}{\gamma_q}(\cdot-\delta_p)\right)$.
3: **if** $\max_{k,p,q}\langle\mathbf{d}_k^{pq},\mathbf{x}\rangle<0$: set $\mathbf{d}_k^{pq}=-\mathbf{d}_k^{pq}$, $\mathbf{d}_k=-\mathbf{d}_k,\forall k,p,q$.
4: Set $\lambda=\alpha\cdot\max_{k,p,q}\langle\mathbf{d}_k^{pq},\mathbf{x}\rangle$.
5: Initialize $\mathbf{a}=0$, $I=\{(k,p,q)\}_{k,p,q}$, $\mathbf{r}=\mathbf{x}$.
6: **while** $I\neq\emptyset$ **do**
7: $(\bar{k},\bar{p},\bar{q})\leftarrow\operatorname*{argmax}_{(k,p,q)\in I}\langle\mathbf{d}_k^{pq},\mathbf{r}\rangle$.
8: **if** $\langle\mathbf{d}_{\bar{k}}^{\bar{p}\bar{q}},\mathbf{r}\rangle<\lambda$: **break**.
9: Refine dilation $\gamma_{\bar{q}}$.
10: $a_{\bar{k}\bar{p}\bar{q}}\leftarrow\langle\mathbf{d}_{\bar{k}}^{\bar{p}\bar{q}},\mathbf{r}\rangle$.
11: $\mathbf{r}\leftarrow\mathbf{r}-a_{\bar{k}\bar{p}\bar{q}}\mathbf{d}_{\bar{k}}^{\bar{p}\bar{q}}$.
12: $I\leftarrow I\setminus\{(\bar{k},p,q);\,|\delta_{\bar{p}}-\delta_p|<\Delta,\,-Q\le q\le Q\}$,
13: **end while**
14: **end procedure**, **return** $\mathbf{a}$ (and $\{\mathbf{d}_k\}$ if sign changed, line 3)

1: **procedure** KERNEL_UPDATE$(\mathbf{x},\mathbf{a},\{\delta_p\},\{\gamma_q\},\{\mathbf{d}_k\})$
2: **for** $k=1$ to K **do**
3: $\mathbf{d}_k\leftarrow(\psi_k^t\psi_k)^{+}(\psi_k^t(\mathbf{r}_k))$, where
4: $\mathbf{r}_k=\mathbf{x}-\sum_{k'\neq k}a_{k'}\psi_{k'}(\mathbf{d}_{k'})$, and
5: $\psi_{k'}(\mathbf{d}_{k'})=\sum_{p=1}^{P}\sum_{q=-Q}^{Q}\frac{a_{k'pq}}{\sqrt{\gamma_q}}\mathbf{d}_{k'}\left(\frac{1}{\gamma_q}(\cdot-\delta_p)\right)$.
6: Align $\mathbf{d}_k$ w.r.t. prominent landmark.
7: $\mathbf{d}_k\leftarrow\frac{1}{\sqrt{\bar{\gamma}_k}}\mathbf{d}_k\left(\frac{1}{\bar{\gamma}_k}\cdot\right)$, where
8: $\bar{\gamma}_k=\left(\prod_{p=1}^{P}\prod_{q=-Q}^{Q}\gamma_q^{|a_{kpq}|}\right)^{\frac{1}{\sum_{p,q}|a_{kpq}|}}$.
9: $\mathbf{d}_k\leftarrow\mathbf{d}_k/\left\|\mathbf{d}_k\right\|_2$.
10: **end for**
11: **end procedure**, **return** $\{\mathbf{d}_k\}$

the maximal overlap, a neural event might thus be encoded by several slightly shifted versions of the same kernel. This not only complicates interpretation, it also makes the following algorithm less stable (see kernel updates). In cases where the kernels are very similar, such as the spike classes learned in Section V-C, we also limit the overlap between *different* kernels by replacing the second index k by k' in (27).

Problem (24)–(27) can again be solved by implementing the alternate minimization scheme from Section III-A, The pseudocode is given in Algorithm 2 and discussed in the following paragraphs.

Kernel initialization: In the continuous case, we generally initialize the kernels with predefined templates. This is necessary because the latencies of the neural events are entirely unknown, making their correct detection a more difficult task than in the epoched case. Thus, initializing with Gaussian noise would bear the risk of only detecting random structures in the data. For the processing of the dataset in Section V, we initialized the kernels with epileptiform spikes taken directly from the data.

Coefficient update: For a long signal $\mathbf{x}$ with high sampling rate, the set of possible latencies $\{\delta_1,\dots,\delta_P\}$ is large, making calculation with LARS impractical. In the continuous setting, we thus use MP for the coefficient updates, which we found to yield very good results in the continuous case. This is due to the fact that when processing a long signal, most of the instantiated kernels have mutually non-overlapping support and thus vanishing dot product. In addition, the constraint (27) further limits the overlap between instantiated kernels. Since MP is exact for orthogonal dictionaries, the error commited by MP is thus relatively low in the present setting.

Our MP implementation as described in COEFF_UPDATE successively searches for atoms $\mathbf{d}_k^{pq}$ that have maximal dot product with the current data residual and subtracts their contribution. It stops when the dot product of every remaining atom with the data is less than the threshold ρ from (26). To facilitate the choice of ρ, we define it as a fraction $0<\alpha<1$ of the maximal dot product of all atoms $\mathbf{d}_k^{pq}$ with the signal $\mathbf{x}$. The parameter α should be chosen dependent on the signal-to-noise ratio (SNR), in order to avoid noise fitting. Note that constraint (27) is enforced through the index set I in COEFF_UPDATE, which controls the indices of the permitted atoms $\mathbf{d}_k^{pq}$.

Note that the dilations are more costly to implement than translations (see Section III-D), and directly using a fine resolution γ_{q+1}/γ_q would be computationally infeasible. Hence, we suggest a multi-resolution approach, initially using a coarse resolution of the set $\{\gamma_{-Q},\dots,\gamma_Q\}$. After each activation of an atom $\mathbf{d}_{\bar{k}}^{\bar{p}\bar{q}}$ in line 7 of COEFF_UPDATE, we then refine the corresponding dilation factor $\gamma_{\bar{q}}$ (noted in line 9).

Kernel update: The kernel update is performed by block coordinate descent as described in Section III-A. Now, the operator $\psi_{k'}$ from (17) is given by

$$\psi_{k'}(\mathbf{d}_{k'})=\sum_{p=1}^{P}\sum_{q=-Q}^{Q}\frac{a_{k'pq}}{\sqrt{\gamma_q}}\mathbf{d}_{k'}\left(\frac{1}{\gamma_q}(\cdot-\delta_p)\right),$$

which defines a convolution with a "stretchable" kernel. The update formula (18) reduces to

$$\mathbf{d}_k\leftarrow\left(\psi_k^t\psi_k\right)^{+}(\psi_k^t(\mathbf{r}_k)),\quad\text{where}$$
$$\mathbf{r}_k=\mathbf{x}-\sum_{\substack{k'=1\\k'\neq k}}^{K}a_{k'}\psi_{k'}(\mathbf{d}_{k'}).$$

Hence, the kernel updates are given by a sort of deconvolution of the residual signals $\mathbf{r}_k$. The conditioning of the operators $\psi_{k'}$ strongly depends on the differences between detected latencies, which can be controlled by the parameter Δ above. In case of a poor condition number, regularization should be

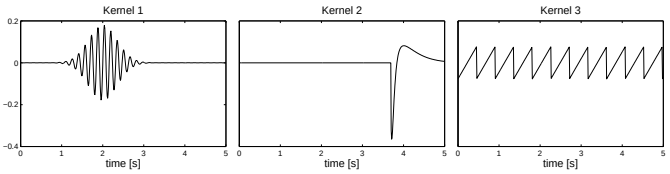

Fig. 3: $K = 3$ kernels were defined in order to generate signal epochs (see Fig. 4). They represent different types of activity of interest in neurological recordings. Note that neural background activity is not represented by these kernels but is modeled through pink noise.

considered. However, this did not occur in our experiments, as we chose Δ sufficiently large.

As in Section III-B, we lift an indeterminacy in the model by realigning the kernels. While we previously used the mean latency across the different epochs for the realignment, this approach is not applicable to the continuous setting where we only have one signal $\mathbf{x}$. Instead, we suggest to align kernels with respect to a prominent landmark, such as the absolute peak of a spike, as done in the following applications.

In addition, the learned kernels should represent the mean duration of their instantiations in the data, in order to make optimal use of the permitted dilations $\{\gamma_{-Q}, \ldots, \gamma_Q\}$, cf. comment after (23). Hence, the following rescaling is applied:

$$\mathbf{d}_k \leftarrow \tfrac{1}{\sqrt{\bar{\gamma}_k}} \mathbf{d}_k \left(\tfrac{1}{\bar{\gamma}_k} \cdot \right),$$

where $\bar{\gamma}_k$ is the geometric mean of the dilations used in the instantiations of $\mathbf{d}_k$.

As before, the kernel update is concluded by normalizing each $\mathbf{d}_k$. The steps above are summarized in the routine KERNEL_UPDATE.

D. Implementation details

Both LARS and MP are based on the dot products between the atoms and the data. In case of translated kernels $\mathbf{d}_k(\cdot - \delta_p)$, this requires the computation of cross-correlations which can be efficiently calculated through the fast Fourier transform. This efficient calculation allows us to use a resolution $\delta_{p+1} - \delta_p$ equal to the sampling resolution of the signals (both for E-AWL and C-AWL).

Dilations were implemented by resampling the discrete signals using linear interpolation; in the downsampling cases, we previously applied an anti-aliasing filter. This implementation means significantly higher computational costs than for translations. In order to still maintain a high resolution between different dilations in Φ, we used the multi-resolution approach described in the coefficient update in Section III-C.

Both E-AWL and C-AWL have been implemented in C++ with MATLAB interface (mex-files). The code for these implementations and the following experiments are freely available at https://github.com/hitziger/AWL.

IV. SYNTHETIC EXPERIMENTS

We use simulated data to evaluate the capability of the E-AWL algorithm to identify three kernels from a set of signals in the presence of amplitude and latency variability as well as noise. The results are compared to those obtained by the translation-invariant dictionary learning algorithm MoTIF and independent component analysis (ICA). The performance of C-AWL is demonstrated on real data in the next section.

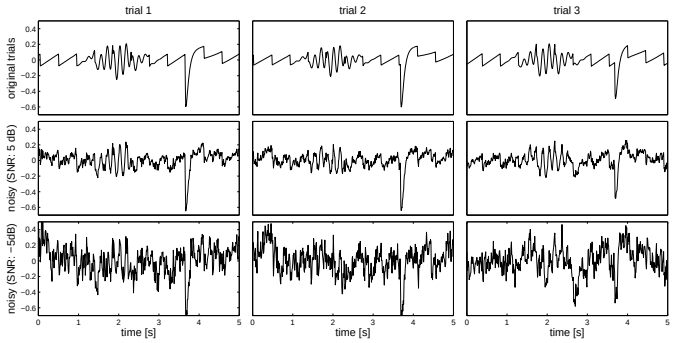

Fig. 4: Three randomly chosen trials, generated from the kernels in Fig. 3 according to the model underlying (19)–(22). **Top row:** noiseless trials. **Middle row:** trials plus pink noise (SNR: 5 dB). **Bottom row:** trials plus pink noise (SNR: -5 dB). See Section IV-A for more details.

A. Data generation

We started by defining $K = 3$ kernels, representing 5-second long signals with 100 Hz sampling rate. They include both transient and oscillatory waveforms (see Fig. 3). These kernels were used to create 200 signal epochs (or *trials*) according to the E-AWL model underlying the minimization problem (19)–(22). Amplitudes and latencies were drawn independently for each kernel from Gaussian distributions with respective means 1 and 0, and respective standard deviations $\sigma_{\mathbf{a}}$ and σ_δ specified in the following paragraphs. Negative amplitudes were discarded to ensure constraint (21). We simulated pink noise with a $1/f$-shaped power spectrum, which is typical for neural background activity. We varied the standard deviation σ_ϵ of the noise throughout the simulations, resulting in different signal-to-noise ratios (SNR) defined as $20 \log(\sigma_{\mathbf{x}}/\sigma_\epsilon)$ [dB], with $\sigma_{\mathbf{x}}$ the standard deviation of the simulated (noiseless) signals. Fig. 4 shows three examples of generated trials with different levels of pink noise. Note that the level of latency jitter σ_δ here is very low and therefore hardly visible.

B. Compared methods

The 200 generated signals were processed with MoTIF, ICA, and E-AWL to recover the underlying kernels. In order to be able to compare to the original kernels, we considered the number K to be known *a priori*.

Like hierarchical E-AWL, the translation-invariant MoTIF algorithm [20] proceeds by incrementally learning the different kernels. In each such step, the new kernel to be learned is constrained to have minimal cross-correlation with all previously learned kernels, to avoid recovering the same kernel multiple times. In contrast to E-AWL, however, the hierarchy in the approach is strict in the sense that after a kernel is calculated, it is not altered anymore while learning the next kernels. This implies a severe drawback of MoTIF: the first learned kernel naturally captures the maximal variance in the data and is therefore susceptible to contain a linear combination of the

original kernels, which cannot be corrected in a later step. For the present comparison, we used the original MATLAB implementation provided to us by the authors of [20]. In order to avoid edge effects, we employed zero-padding at both ends of each trial.

ICA was calculated using the Matlab software package FastICA[4] described in [26]. As suggested by the authors, we performed a PCA prior to the ICA, in order to whiten the data and reduce its dimension.

E-AWL was implemented according to Algorithm 1. We used two different initializations to compare their impact on the learned kernels: (i) random Gaussian noise and (ii) the kernels obtained with ICA. To distinguish these two initializations, approach (ii) is denoted as ICA + E-AWL.

For both MoTIF and E-AWL, we allowed translations $\{\delta_{-P}, \ldots, \delta_P\}$ ranging from -0.1 to 0.1 seconds, with a resolution $\delta_{p+1} - \delta_p$ equal to the sampling period (0.01 s).

C. Kernel distances

In order to quantify the methods' performances, we defined a distance between the original kernels and the calculated ones. For this purpose, let $\tilde{\varepsilon}$ first denote the distance between two normalized kernels $\mathbf{d}$ and $\tilde{\mathbf{d}}$, given by

$$\tilde{\varepsilon}(\mathbf{d}, \tilde{\mathbf{d}}) \stackrel{\text{def}}{=} \sqrt{1 - \max_t \left| \sum_\tau \mathbf{d}(\tau) \tilde{\mathbf{d}}(\tau + t) \right|}.$$

This distance generalizes the one proposed in [27] by replacing the dot product between kernels by the maximal value of their cross-correlation, thus providing for a shift-invariant measure (see also [28]). Note that the use of the absolute value furthermore yields sign-invariance. Both properties are important in the present setting, due to indeterminacies in relative latencies and signs of the calculated kernels. Another indeterminacy consists in the order of the learned kernels, which needs to be accounted for when extending $\tilde{\varepsilon}$ to measure the distance between kernel sets. For this purpose, let $\mathcal{P}(K)$ denote the set of permutations of $\{1, \ldots, K\}$. For two sets of normalized kernels $\{\mathbf{d}_k\}$ and $\{\tilde{\mathbf{d}}_k\}$, we can now define

$$\varepsilon(\{\mathbf{d}_k\}, \{\tilde{\mathbf{d}}_k\}) \stackrel{\text{def}}{=} \min_{\pi \in \mathcal{P}(K)} \frac{1}{K} \sum_{k=1}^{K} \tilde{\varepsilon}(\mathbf{d}_k, \tilde{\mathbf{d}}_{\pi(k)}) \quad \in [0, 1]. \tag{28}$$

The calculation of ε can thus be described as finding a pairing of the kernels $\{\mathbf{d}_k\}$ with the $\{\tilde{\mathbf{d}}_k\}$ such that the average distance between all of these pairs is minimal. Note that in our applications, the number K of kernels is small, such that brute-force minimization over $\mathcal{P}(K)$ is a feasible task.

Due to its invariance properties, ε is only a pseudo-metric since the separability axiom does not hold. For more information on dictionary metrics, see for instance [27] .

D. Quantitative comparisons

We investigate the effects of varying kernel amplitudes and latencies, as well as different noise levels on the performances of MoTIF, ICA, and E-AWL, measured by the distance ε between calculated and original kernels. The results are shown in Fig. 5 and discussed below. Note that, contrary to MoTIF and E-AWL, ICA does not explicitly account for varying latencies and is typically used to separate events across different recording channels, where latency jitter is only a minor concern. However, we think that comparison to ICA is instructive, as it shows ICA's tolerance w.r.t. increasing latency jitter.

1) Increasing number of kernels: We first measured the methods' performances for an increasing number of kernels K. For this purpose, we simulated trials with only the first, the first two, and all three kernels shown in Fig. 3. Amplitude and latency variability were fixed to $\sigma_{\mathbf{a}} = 0.3$ and $\sigma_\delta = 0.01$, respectively (cf. Section IV-A). Pink noise was added, resulting in a signal-to-noise ratio (SNR) of 10 dB. As shown in the upper left plot in Fig. 5, all methods succeed well in recovering a single kernel from the trials (low error ε). However, in the case of two and three kernels, only E-AWL shows good results, both for random and ICA initializations. ICA cannot properly separate the different kernels due to the latency jitter. MoTIF performs even worse at separating several kernels, which is due to the strictly hierarchical learning approach described in Section IV-B. In fact, in the presence of several kernels in the data, the first kernel learned by MoTIF is a linear combination of these kernels (see Fig. 6), which cannot be corrected in a later step.

2) Varying amplitudes: In order to investigate the effect of varying amplitudes, we simulated signals using all $K = 3$ kernels and setting $\sigma_\delta = 0$ (no latency variability). Amplitude variability $\sigma_{\mathbf{a}}$ was increased throughout the simulations from 0 to 1. Again, the SNR was 10 dB. The resulting errors ε are shown in the upper right panel of Fig. 5. ICA yields very good results for increasing $\sigma_{\mathbf{a}}$. In fact, this case is optimal for ICA, as the amplitudes were drawn independently which allows ICA to separate the kernels. In addition, there is no latency variability to cope with. In contrast, MoTIF cannot make use of the amplitude variability to separate the kernels: even for high $\sigma_{\mathbf{a}}$, the first kernel is learned to maximize the data variance and thus contains a mixture of all original waveforms, which cannot be corrected in a later step (cf. Section IV-B). E-AWL initialized with Gaussian noise improves with increasing $\sigma_{\mathbf{a}}$ but does not reach the level of ICA. However, when initialized directly with the ICA components, the E-AWL algorithm converges close to this initialization, as it already provides a very good estimate.

3) Varying latencies: Throughout the next simulations, latency variability σ_δ was increased from 0 to 5 seconds, and SNR was kept at 10 dB. In order to allow for the detection of the largely shifted kernels, the length of each trial was extended from 5 to 15 seconds and the permitted translations for MoTIF and E-AWL were increased to ± 5 seconds (previously ± 0.1 seconds). We maintained some amplitude variability ($\sigma_{\mathbf{a}} = 0.3$) to allow ICA to separate the waveforms. The results in the lower left plot of Fig. 5 show that ICA's initially good performance decreases with latency jitter above $\sigma_\delta = 0.004$ seconds. Around the same value, the kernel error of E-AWL decreases. In fact, E-AWL does not only compensate for the

[4] http://research.ics.aalto.fi/ica/fastica/

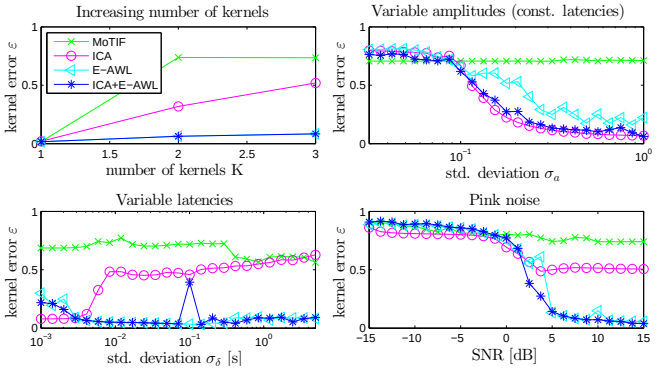

Fig. 5: Performances of MoTIF, ICA, and E-AWL in recovering kernels from trials, measured by the distance ε between original and calculated kernels. Four different settings where studied (see plot titles). For E-AWL, we initialized with both Gaussian noise and ICA components (the latter is denoted by ICA + E-AWL, see legend in first plot). See Section IV-D for a detailed discussion.

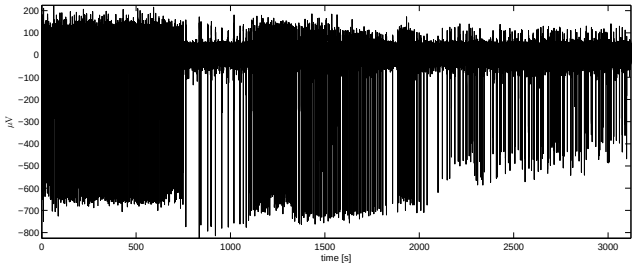

Fig. 7: Recording of local field potentials (LFP) in the cortex of a rat. The vertical lines are epileptiform discharges (spikes), whose density visibly changes throughout the recording. The maximal negative spike amplitudes decrease towards the end of the recording from about -700 μV to -400 μV.

varying latencies, but even makes use of them to properly separate the kernels. When using ICA initialization, E-AWL is susceptible to getting stuck in a local optimum close to this initialization, hence the kink in the ICA+E-AWL curve around $\sigma_\delta = 0.1$. MoTIF slightly improves for large latency jitter, which helps it to separate the kernels. However, compared to E-AWL the error stays large. In fact, we found that even when MoTIF was able to correctly identify the first kernel, the second and third learned kernel did not well represent the originals. This is due to the minimal correlation constraint imposed by MoTIF (cf. Section IV-B), which does not well characterize the original kernels.

4) Varying SNR: Finally, performance for different levels of pink noise was studied. The corresponding errors ε for increasing SNR are shown in the lower right panel of Fig. 5. For low SNRs, ICA shows a slightly more robust performance than E-AWL. This results from E-AWL's greater risk of fitting noise due to its variable latency parameter. Above 4 dB, however, E-AWL steadily improves contrary to ICA, which cannot compensate for the latency variability σ_δ. E-AWL's difficulty to cope with high levels of pink noise will become clearer in the following qualitative comparison. Interestingly, for low SNR, E-AWL's performance does not depend on the initialization, which may be due to the fact that the results from ICA are of similarly low quality. MoTIF improves only minimally for increasing SNR as its error originates mainly from its inability to separate the waveforms and not from the signal quality.

E. Qualitative comparison

For a qualitative comparison, we generated two sets of trials with medium amplitude variability ($\sigma_{\mathbf{a}} = 0.3$), small latency variability ($\sigma_\delta = 0.01$), and pink noise with resulting SNRs of 5 dB and -5 dB, respectively. Three randomly chosen trials are displayed in Fig. 4, the respective rows show original and noisy signals (see caption). The kernels recovered with MoTIF, ICA, and E-AWL are shown in Fig. 6.

The left half of Fig. 6 shows the learned kernels in the case of high SNR. The first kernel calculated with MoTIF (first row) shows a linear mixture of the true kernels (Fig. 3). Since MoTIF learns the kernels strictly hierarchically, this kernel is not corrected when learning the next kernels. The second and third kernels learned by MoTIF do not strongly resemble the original kernels. ICA correctly learns the first original kernel but produces two versions with different phases, resulting from its incapacity to compensate for the latency jitter. In addition, it is not able to correctly separate the second and third original kernels but instead produces a mixture. E-AWL correctly separates all three waveforms. It does so even when initialized with the suboptimal ICA components (last row). However the third learned kernel shows some small baseline change, resulting from fitting low frequencies of the pink noise.

In case of high contamination with pink noise (right half of Fig. 6), MoTIF yields similar results as for high SNR, showing it to be robust against noise. For ICA, the first two kernels are similar to those learned in the high SNR setting. The third kernel, however, seems to capture some low-frequency noise from the pink noise contamination. The kernels learned with E-AWL show strong contaminations with low-frequency noise. E-AWL's ability to compensate for varying latencies makes it susceptible to fitting the low frequency components in the pink noise. Even when initialized with the ICA kernels, E-AWL still strongly picks up this noise.

V. Applications to Neurological Signals

In this section, we demonstrate the usefulness of the AWL framework as a data exploration tool capable of producing compact and insightful representations. For this purpose, we apply AWL to a neuroelectrical recording containing epileptiform discharges or *spikes* using two approaches. The first approach requires a prior segmentation step to produce a set of short signal epochs which are then processed with E-AWL. Such an epoched approach is frequently used in neurological signal processing and allows us to compare to other methods that require multiple input signals, such as ICA and MoTIF. Finally, we demonstrate how the single, continuous (i.e., non-epoched) recording can be directly processed with C-AWL and illustrate the complementary benefits of this second approach.

A. Data acquisition

In an animal model of epilepsy, an electrode was placed in the cortex of a Wistar-Han rat for measuring local field

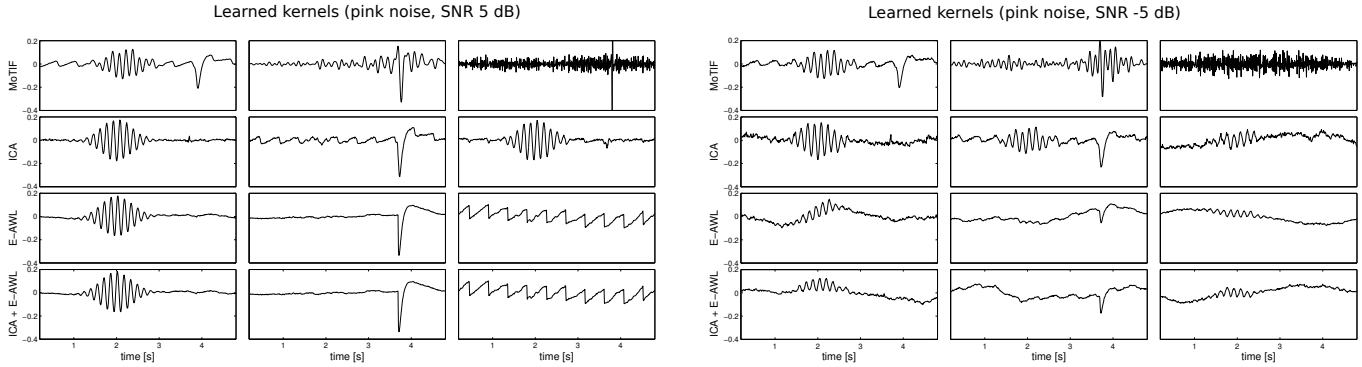

Fig. 6: Kernels recovered from trials contaminated with pink noise (Fig. 4). For high SNR of 5 dB (left), E-AWL shows best performance and correctly identifies the three original kernels (Fig. 3), both for initialization with random noise (third row) and with ICA (bottom row). MoTIF (top row) and ICA (second row) do not separate all kernels correctly. For low SNR of -5 dB (right), the performances of MoTIF and ICA only slightly worsen w.r.t. the high SNR setting. The kernels learned with E-AWL, however, show strong contamination with low-frequency noise.

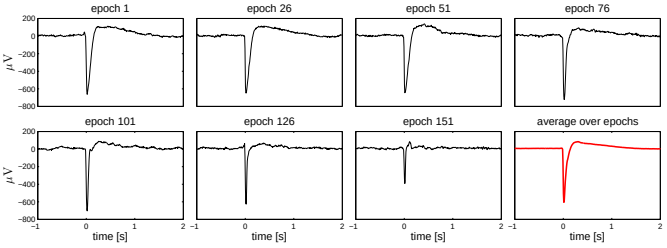

Fig. 8: The first seven plots show sample epochs from the 169 epoched data segments (original order in data maintained). The average over all 169 epochs is plotted in the bottom right. It is apparent that spikes are changing throughout the dataset, decreasing in duration and amplitude. Note that for better visualization only 3 seconds of the 10-second long epochs are shown.

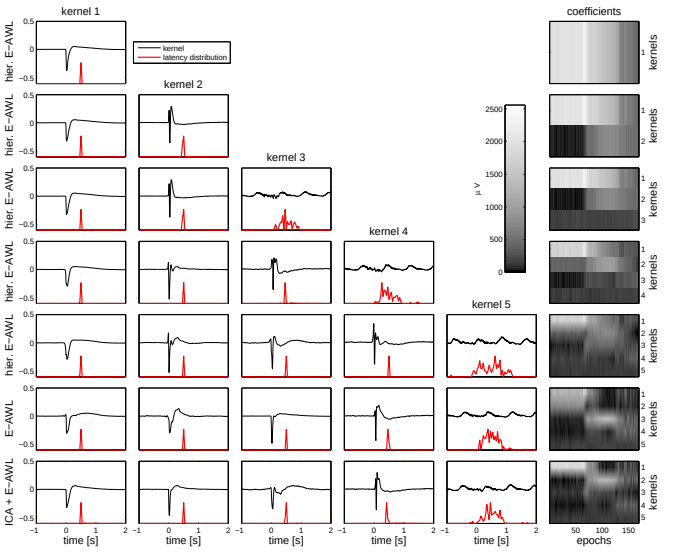

Fig. 9: Kernels learned with E-AWL from 169 spike epochs (Fig. 8). Each of the first five rows corresponds to one set of normalized kernels obtained in the hierarchical learning approach. Note that only the central 3 seconds of the 10-second long kernels are shown. The two bottom rows show the representations learned non-hierarchically, using white Gaussian and ICA initialization, respectively. The last column shows the coefficients of each kernel (row) used in each epoch (column); light colors correspond to large values. The red curve under each kernel shows the distribution of latencies used for this kernel across epochs. Note that these distributions are shifted 0.5 seconds to the right for better visualization, avoiding overlaps with the spikes.

potentials (LFP), i.e., the summed electrical activity of a neural assembly. The recording, sampled at 1250 Hz, lasted approximately one hour, see Fig. 7. Prior to the recording, an inhibition blocker (bicuculline) had been injected into the cortex to provoke epileptiform discharges. This data acquisition was performed simultaneously with other multi-modal recordings and the full experimental protocol can be found in [29].

As can be seen in Fig. 7, the spiking activity changed throughout the recording, with periods of high and low spiking densities and different spike amplitudes. However, the exact spike shapes and their evolution across the dataset cannot be directly seen in this plot. Therefore, the goal of the following analysis was to obtain a compact representation of this LFP dataset, which could provide insight into the spike shapes as well as their variable parameters, such as amplitudes, durations, and spiking rates.

B. Epoched processing

In the first approach to process the LFP signal, 169 spikes with at least 10-second inter-spike intervals (peak-to-peak) were manually selected and segmented into 10-second time windows, centered around the spikes. The time windows were chosen relatively large w.r.t. the duration of the spikes in order to possibly recover other signal structures in their vicinity. In fact, the identification of an oscillatory artefact as shown in the following paragraphs would not have been possible on short epochs. Fig. 8 shows seven sample epochs, as well as the average over all 169 epochs.

Processing with E-AWL: Hierarchical E-AWL (Algorithm 1) was used to learn kernel representations of increasing cardinalities $K = 1, \ldots, 5$. In order to enable blind learning of interesting signal structures, we initialized each newly added kernel with Gaussian noise. We compared this hierarchical approach to the direct learning of the $K = 5$ kernels. In order to allow E-AWL to identify waveforms with large jitter or phase variability, we used translations $\{\delta_{-P}, \ldots, \delta_P\}$ from -2 to 2 seconds.

For each $K = 1, \ldots, 5$, the corresponding kernel representation of hierarchical E-AWL is shown in the first five rows

of Fig. 9 and consists of the K kernels (black signals), their latency distributions (red curves below the kernels), and their coefficients across the epochs (grey values in the last column, see caption). The two bottom rows show the representations learned with non-hierarchical approaches, where the kernels were initialized with Gaussian noise and with ICA, respectively. Note that only 3 seconds of the 10-second long kernels are shown as the remainders of the kernels did not contain any interesting information.

For $K = 1$, the resulting kernel is simply a weighted average across epochs and resembles the average spike shown in Fig. 8. In fact, since the prior manual epoching step accurately aligned spikes across epochs, this kernel's latency changes were negligible, as reflected by the sharply peaked latency distribution. Adding a second kernel results in two spike components (second row). Only after learning the third kernel does an entirely new, oscillatory waveform appear. These oscillations were later identified to be an artefact from the recording device. With the fourth and fifth learned kernels, the spike is further refined into different components. While all kernels representing spike components almost always have zero latency, the oscillatory kernel takes different latencies mainly in a range of about 1 second, corresponding to its period. This dispersed latency distribution indicates the independence of the waveform's phase w.r.t. the positions of the spikes.

Comparing the last three rows of Fig. 9 shows that the kernels produced by E-AWL differ depending on the learning approach (hierarchical vs. non-hierarchical) and the used initialization (random vs. ICA). However, the results are qualitatively similar, each containing four spike components and one periodic waveform.

The learned kernel coefficients (last column of Fig. 9) provide insight into the evolution of the spikes across the recording. For $K = 1$, the coefficient profile shows decreasing energy across the epochs. Interestingly, for $K = 2$, the coefficients of the second learned spike component (second row) only take non-zero values after the first 60 epochs, indicating a sudden change in the spike shape. For $K > 2$, the coefficient profiles reveal even more detailed structural information about the spike's evolution. These profiles may be taken as an indicator for the optimal number of kernels to be learned: while the profiles for $K < 4$ look relatively smooth, we see more frequent changes in the coefficients for $K = 5$, possibly indicating a slight overfitting. Note that the coefficients of the oscillatory kernel remain relatively constant throughout all epochs, indicating a time-independent periodic activity.

Comparison to MoTIF and ICA: The hierarchical E-AWL representation for $K = 5$ was compared to those produced by MoTIF and ICA. For MoTIF the same latency tolerance of ± 2 seconds was used as in E-AWL. Fig. 10 shows the kernels and coefficients learned with MoTIF, ICA, and E-AWL, respectively. All three methods appear to produce spike components, however, only ICA and E-AWL also recover an oscillating waveform. E-AWL produces a more accurate representation of this oscillatory signal component: First, it captures the oscillations in a single kernel, while ICA represent the different phases of the oscillations through linear combinations of the differently shifted sinusoidal kernels 4 and 5. Second, these sinusoidal functions only capture the fundamental frequency of the oscillations and do not show the distinctive pointy shape of the oscillatory waveform clearly visible in the E-AWL representation. Third, only E-AWL clearly separates the oscillatory waveform from the spike components, while the sinusoidal kernels of ICA contain spike artefacts.

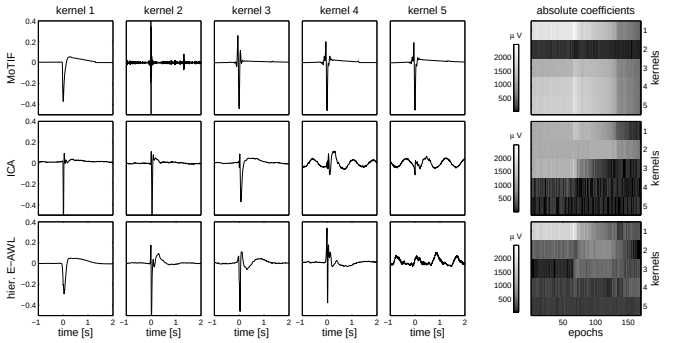

Fig. 10: Five kernels learned with MoTIF, ICA, and hierarchical E-AWL, see respective rows. The last column shows the absolute coefficients of the kernels (rows) used across the epochs (columns). Light colors correspond to large absolute values. While ICA and E-AWL both recover an oscillatory artefact, only E-AWL clearly separates it from the spike components and encodes it in a single kernel.

Note that the original kernels learned with MoTIF contained the spike components at arbitrary temporal locations, due to the translation-invariance in the approach (we only aligned them here for better visualization).

The learned coefficients (last column of Fig. 10) also reveal important differences between the three methods. For MoTIF, we can observe very similar coefficient profiles for most kernels. Only the second kernel shows a very low profile, suggesting that it does not well capture an actual pattern in the data. In fact, the shape of the kernel seems to contain artefacts, possibly resulting from MoTIF's maximal decorrelation constraint (cf. Section IV-B). In the case of ICA, the first three kernels are active together in the first half of the epochs. The coefficient profile of E-AWL appears more contrasted and provides a detailed structuring of the epochs.

For a quantitative comparison, we calculated the distances ε as defined in (28) between the kernel sets learned with MoTIF, ICA, and the three different E-AWL approaches (hierarchical and random vs. ICA initialization). These distances are visualized in the matrix in Fig. 11. The MoTIF kernels have the largest distance to the kernels obtained with the other methods. The smallest distances are found between the different E-AWL approaches.

C. Continuous processing

The epoched approach above suffers from several drawbacks. First, the manual epoching is time-consuming and would not be feasible for a larger set of recordings. Second, this approach requires spikes to be well isolated, which was only the case for a small subset of spikes in the given data. We now demonstrate how the recording can be processed with C-AWL without prior epoching.

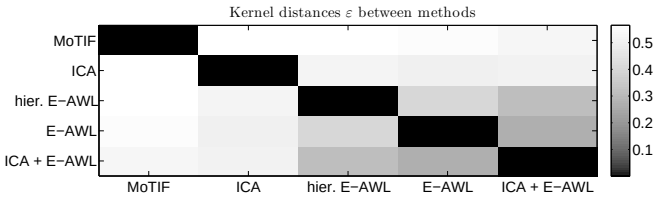

Fig. 11: Distance ε between the kernels learned with MoTIF, ICA, and E-AWL. For the latter, we used three different approaches: hierarchical and non-hierarchical learning with both random and ICA initialization of the kernels.

For the kernels, we used 1.5-second long time windows. We set the parameter $\Delta = 0.2$ seconds (cf. Section III-C), resulting in a maximal overlap between kernel instantiations of 87%. This constraint prevented spikes from being detected multiple times. At the same time, Δ was chosen small enough to still allow the detection of spikes in close succession.

The relative correlation threshold was set to $\alpha = 0.1$. For the presented dataset, which had a high SNR, the small value for α allowed us to even detect low-amplitude spikes. In cases of lower SNR, α should be chosen larger to avoid noise fitting.

We applied C-AWL in two different ways to explain the spike variability: (1) using a multi-class model with different constant kernels $\mathbf{d}_k$ and (2) using a single-class model with one kernel $\mathbf{d}$ of adaptive duration. Note that both are special cases of the C-AWL model. We found that using different kernels *and* variable duration in a single approach provided too many parameters to describe the spike variability and led to redundancies in the representation.

In order to verify the performance of both approaches, the spikes were first detected manually ($n = 520$) and their temporal locations were compared to the ones detected with C-AWL. Note, however, that the main objective of this section is to demonstrate the qualitative advantages of C-AWL compared to the epoched approach from Section V-B. For a more exhaustive quantitative evaluation we refer the reader to Chapter 6 in [30], where C-AWL is compared to template matching in terms of detection performance for different noise levels.

1) Multiple kernels of constant durations: We learned hierarchical representations with $K = 1, \ldots, 5$ kernels using Algorithm 2 without dilation-invariance. Here, we only analyze the representation for $K = 5$, which is illustrated in Fig. 12. The five learned kernels are shown in the upper left plot, where they are scaled with the average coefficients of their respective occurrences in the data. Note that the time window was chosen sufficiently large to learn not only the first negative wave but also the slow positive wave following it. We can see that the spike classes represented by the kernels differ mainly in duration and average amplitude. The plot on the right shows the negative waves of the learned kernels plotted on top of the respective spikes they represent in the recording. Note the sharp kink around 0.03 seconds in the first three kernels, which can also be observed in the real spikes and gives evidence of the good time resolution properties of C-AWL.

The coefficients of the 518 detected spike occurrences are plotted in time across the recording in the middle left of

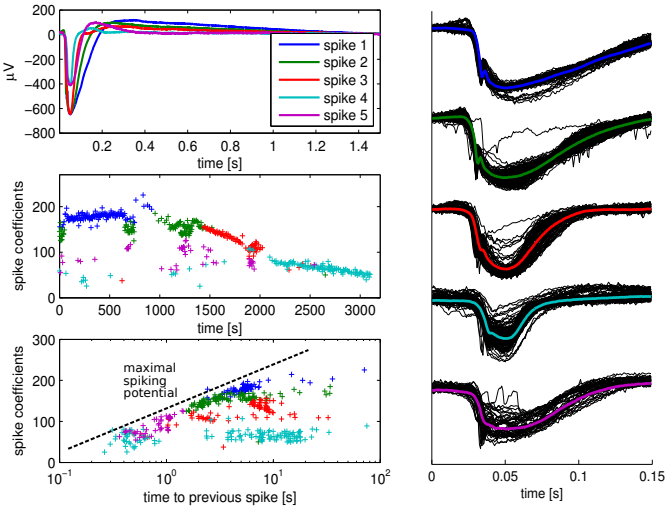

Fig. 12: Spike representation learned with C-AWL using five kernels of constant duration. **Upper left:** the five learned kernels. **Middle left:** spike coefficients plotted in time across the recording. **Lower left:** spike coefficients plotted against the temporal distances to previous spikes, with the dashed line describing the system's maximal spiking potential. **Right:** each learned kernel is plotted on top of the spikes that it represents in the data.

Fig. 12, the colors correspond to the different kernels. We see an overall decrease in spike energy to about one third of the initial energy towards the end of the recording. This is more than the decrease of the spike peaks from about -700 μV to -400 μV, which we can observe in the original recording in Fig. 7. This suggests that the decreasing spike energies result not only from their different amplitudes but also their different durations. Besides the global decrease, the coefficient plot shows well-separated clusters corresponding to the spike classes, which provides an interesting structuring of the dataset. Around 0, 700, 1250, and 1950 seconds, we can see clusters of slightly smaller coefficients. These correspond to the periods of dense spiking activity, which can be directly observed in the original recording in Fig. 7.

The relationship between the spike coefficients (i.e., their l_2-norms) and the spiking density becomes even clearer from the lower left of Figure 12, where we plotted each coefficient against the inter-spike delay w.r.t. the preceding spike (log scale). We can observe that the coefficients are larger for longer inter-spike intervals, suggesting that the system requires some time to regain its full spiking potential after each spike. In fact, the dashed line clearly shows this maximal spiking potential as a function of the inter-spike intervals.

Comparison with the manually detected spikes showed that all 518 spikes were true positives. Only 2 spikes were missed by the C-AWL algorithm, i.e., 100% precision and 99.6% recall.

Note that the processing of the recording with C-AWL using different kernels of constant durations showed to result in a combined spike detection and clustering algorithm. This is similar to an approach recently proposed in [31]. However, the latter algorithm requires the choice of several correlation and feature thresholds for spike detection, whereas C-AWL uses only the correlation threshold α. Another advantage of C-AWL is the possibility to describe the spike variability directly through a dilation parameter, as demonstrated in the following

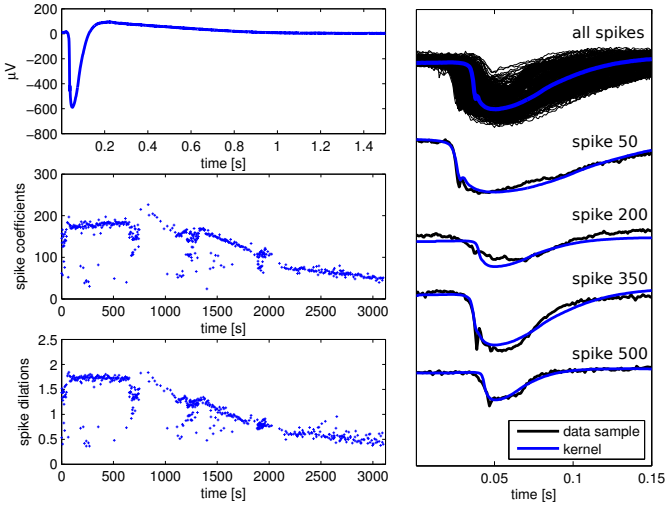

Fig. 13: Spike representation learned with C-AWL using a single kernel with adaptive duration. **Upper left:** the learned kernel, scaled with the mean amplitude of its instantiations. **Middle left:** spike coefficients plotted in time across the recording. **Lower left:** dilation factors plotted in time across the recording. **Right:** the top plot shows the superposed detected spikes and the learned kernel. The four bottom plots illustrate samples of detected spikes with corresponding instantiations of the kernel.

paragraph.

2) Single kernel with adaptive duration: The previous approach produced spike classes which differed mainly in duration and average amplitude. The following approach is therefore designed to capture the changing duration explicitly through the dilation parameters γ_q, using only a single kernel $\mathbf{d}$, i.e., setting $K = 1$ in Algorithm 2. We used the multi-resolution approach (cf. Section III-C) with a total of 761 logarithmically sampled dilation parameters $\gamma_{-Q}, \ldots, \gamma_Q$ and a maximal relative stretch of $\gamma_Q/\gamma_{-Q} = 8$, which resulted in a fine resolution of $\gamma_{q+1}/\gamma_q = 1.0027$.

The resulting spike representation is shown in Fig. 13. The learned kernel $\mathbf{d}$, scaled with the mean duration and amplitude of its instantiations, is plotted in the upper left. The top of the right plot shows the kernel superposing the detected spikes. Below are some spike samples, superposed by the corresponding instantiations of the kernel $\mathbf{d}$. Thanks to its variable duration and amplitude, these instantiations match most of the spikes very well. Note that similarly to the previous multi-class model, a little kink before the negative peak is visible in the learned kernel.

The coefficients and the durations γ_q of the 518 detected spikes are shown in the middle and lower left, respectively. Both profiles look very similar, indicating that the decrease in spike energy can be explained mostly through the decreasing durations, rather than the smaller decrease in spike amplitudes (cf. Fig. 7).

The 518 detected spikes were the same as those detected in the multi-class model, hence the same precision of 100% and recall of 99.6%.

VI. Conclusion

The framework proposed in this paper, *adaptive waveform learning* (AWL), provides a general neurophysiological signal model as well as two concrete algorithms for processing epoched (E-AWL) and continuous single-channel recordings (C-AWL). Through the explicit modeling of waveform variability, AWL is capable of capturing variations across the recorded neural events, such as different amplitudes, latencies, and dilations.

The application to recorded local field potentials (LFP) containing epileptiform discharges showed the capability of both E-AWL and C-AWL to learn interesting data representations. In fact, due to their complementary approaches, the two algorithms provided very different insights into the recording: Using previously epoched spike segments, E-AWL produced detailed decompositions of the spikes into several components. In addition, it revealed a hidden oscillatory artefact in the data. In turn, C-AWL did not require the time-consuming epoching step but was able to automatically detect the spikes in the continuous signal. This gave a more complete representation of the dataset since C-AWL detected even close spike occurrences, which had to be omitted in E-AWL. In summary, E-AWL proved capable of revealing the patterns constituting a signal, while C-AWL is better designed to detect given patterns inside a signal. This suggests a combination of both methods for a fully automatic pattern recognition methodology in future works. For all experiments, we were able to use high resolution across translations and dilations, thanks to an efficient implementation using the fast Fourier transform and a multi-resolution approach.

The general AWL framework furthermore allows to implement other types of waveform variability. For instance, we found that dilations (together with varying amplitudes) could account for the majority of the spike variability, but not for all of it. More general temporal rescaling functions could thus be considered, for example, through the use of dynamic time warping. Besides the processing of epileptiform spikes, E-AWL and C-AWL can also be applied to other neurophysiological signal processing tasks, such as: (i) the identification of event-related brain potentials (ERPs) across a set of experimental trials as well as the description of the inter-trial (or inter-subject) variability with E-AWL and (ii) the automatic detection of spontaneously occurring neural events with C-AWL, such as sleep spindles during stage 2 sleep.

As a drawback, we saw that E-AWL's ability to compensate for latency variability makes it susceptible to fitting low-frequency noise components. In cases of strong low-frequency noise, high-pass filtering, either as a preprocessing step or as a postprocessing of the learned kernels, could therefore be considered. The susceptibility to low-frequency noise furthermore shows that the complexity of the AWL framework (type and amount of permitted variability, number of kernels, additional constraints) should be carefully adapted to each application and the signal-to-noise ratio (SNR).

Currently, AWL is being extended to process multi-channel recordings (e.g., EEG), based on the observation that different channels can be treated similarly to the different trials in E-AWL. However, latencies should only vary across trials, since neural events typically appear across channels without time delay. This is similar to the multi-channel extension of dVCA proposed in [8]. Preliminary work on multi-channel AWL has recently been presented at the International Conference on

Basic and Clinical Multimodal Imaging.

ACKNOWLEDGMENT

This work was supported by a doctoral grant of the region Provence-Alpes-Côte d'Azur and the ANR grants COADAPT (09-EMER-002-01), MULTIMODEL (2010-BLAN-0309-04), and VIBRATIONS (ANR-13-PRTS-0011). The electrophysiological recordings (Section V) were obtained within the MULTIMODEL project.

REFERENCES

[1] C. Woody, "Characterization of an adaptive filter for the analysis of variable latency neuroelectric signals," *Medical and Biological Engineering and Computing*, vol. 5, no. 6, pp. 539–554, 1967.

[2] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[3] P. Comon, "Independent component analysis, a new concept?" *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[4] S. Makeig, A. Bell, T. Jung, T. Sejnowski *et al.*, "Independent component analysis of electroencephalographic data," *Advances in neural information processing systems*, pp. 145–151, 1996.

[5] T. Lagerlund, F. Sharbrough, and N. Busacker, "Spatial filtering of multichannel electroencephalographic recordings through principal component analysis by singular value decomposition," *Journal of Clinical Neurophysiology*, vol. 14, no. 1, pp. 73–82, 1997.

[6] F. Miwakeichi, E. Martínez-Montes, P. A. Valdés-Sosa, N. Nishiyama, H. Mizuhara, and Y. Yamaguchi, "Decomposing eeg data into space–time–frequency components using parallel factor analysis," *NeuroImage*, vol. 22, no. 3, pp. 1035–1045, 2004.

[7] W. Truccolo, K. H. Knuth, A. Shah, S. L. Bressler, C. E. Schroeder, and M. Ding, "Estimation of single-trial multicomponent erps: Differentially variable component analysis (dVCA)," *Biological cybernetics*, vol. 89, no. 6, pp. 426–438, 2003.

[8] K. H. Knuth, A. S. Shah, W. A. Truccolo, M. Ding, S. L. Bressler, and C. E. Schroeder, "Differentially variable component analysis: identifying multiple evoked components using trial-to-trial variability," *Journal of neurophysiology*, vol. 95, no. 5, pp. 3257–3276, 2006.

[9] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on*, vol. 41, no. 12, pp. 3397–3415, 1993.

[10] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[11] P. Durka and K. Blinowska, "Analysis of EEG transients by means of matching pursuit," *Annals of biomedical engineering*, vol. 23, no. 5, pp. 608–611, 1995.

[12] P. Durka, A. Matysiak, E. Martínez-Montes, P. A. Valdés-Sosa, and K. Blinowska, "Multichannel matching pursuit and EEG inverse solutions," *Journal of neuroscience methods*, vol. 148, no. 1, pp. 49–59, 2005.

[13] R. Gribonval, "Piecewise linear source separation," in *Optical Science and Technology, SPIE's 48th Annual Meeting*. International Society for Optics and Photonics, 2003, pp. 297–310.

[14] C. Bénar, T. Papadopoulo, B. Torrésani, and M. Clerc, "Consensus matching pursuit for multi-trial EEG signals," *Journal of neuroscience methods*, vol. 180, no. 1, pp. 161–170, 2009.

[15] C. Sielużycki, R. König, A. Matysiak, R. Kuś, D. Ircha, and P. J. Durka, "Single-trial evoked brain responses modeled by multivariate matching pursuit," *Biomedical Engineering, IEEE Transactions on*, vol. 56, no. 1, pp. 74–82, 2009.

[16] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: a strategy employed by V1?" *Vision research*, vol. 37, no. 23, pp. 3311–25, Dec. 1997.

[17] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *Image Processing, IEEE Transactions on*, vol. 15, no. 12, pp. 3736–3745, 2006.

[18] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, "Supervised dictionary learning," in *Advances in neural information processing systems*, 2009, pp. 1033–1040.

[19] T. Blumensath and M. Davies, "Sparse and shift-invariant representations of music," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 1, pp. 50–57, 2006.

[20] P. Jost, S. Lesage, P. Vandergheynst, and R. Gribonval, "Learning redundant dictionaries with translation invariance property: the MoTIF algorithm," IEEE, Tech. Rep., 2005.

[21] Q. Barthélemy, C. Gouy-Pailler, Y. Isaac, A. Souloumiac, A. Larue, and J. I. Mars, "Multivariate temporal dictionary learning for EEG," *Journal of neuroscience methods*, vol. 215, no. 1, pp. 19–28, 2013.

[22] S. Hitziger, M. Clerc, A. Gramfort, S. Saillet, C. Bénar, and T. Papadopoulo, "Jitter-adaptive dictionary learning–application to multi-trial neuroelectric signals," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[23] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *Multiscale Modeling & Simulation*, vol. 7, no. 1, pp. 214–241, 2008.

[24] B. Ophir, M. Lustig, and M. Elad, "Multi-scale dictionary learning using wavelets," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 5, pp. 1014–1024, 2011.

[25] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[26] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4, pp. 411–430, 2000.

[27] S. Chevallier, Q. Barthélemy, and J. Atif, "On the need for metrics in dictionary learning assessment," in *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*. IEEE, 2014, pp. 1427–1431.

[28] S. Lesage, "Apprentissage de dictionnaires structurés pour la modélisation parcimonieuse des signaux multicanaux," Ph.D. dissertation, Université Rennes 1, 2007.

[29] S. Saillet, A. I. Ivanov, P. Quilichini, A. Ghestem, B. Giusiano, S. Hitziger, I. Vanzetta, C. Bernard, and C. G. Bénar, "Interneurons contribute to the hemodynamic/metabolic response to epileptiform discharges," *submitted*.

[30] S. Hitziger, "Modeling the variability of electrical activity in the brain," Ph.D. dissertation, Université Nice Sophia Antipolis, 2015.

[31] A. Nonclercq, M. Foulon, D. Verheulpen, C. De Cock, M. Buzatu, P. Mathys, and P. Van Bogaert, "Cluster-based spike detection algorithm adapts to interpatient and intrapatient variation in spike morphology," *Journal of neuroscience methods*, vol. 210, no. 2, pp. 259–265, 2012.