An MCU Implementation of PCA/PSA Streaming Algorithms for EEG Features Extraction

This is the author's version of the article that has been presented at IEEE BioCAS2021
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS49922.2021.9645035

An MCU Implementation of PCA/PSA Streaming Algorithms for EEG Features Extraction

Luciano Prono[‡], Alex Marchioni[*], Mauro Mangia[*], Fabio Pareschi[‡†], Riccardo Rovatti[*†] and Gianluca Setti[‡†]
[*]DEI, [†]ARCES, University of Bologna, Italy - Email: {alex.marchioni, mauro.mangia2, riccardo.rovatti}@unibo.it
[‡]DET, Politecnico di Torino, Italy - Email: {luciano.prono,fabio.pareschi,gianluca.setti}@polito.it

***Abstract*—Patient monitoring requires the acquisition of increasingly larger amounts of biosignal data that needs to be managed and transferred with minimum energy consumption.**

As huge quantities of data are often very redundant, it is possible to reduce their size directly on the *edge* of the system, right after the acquisition. To do so, subspace analysis can be considered a fundamental tool that can be used to significantly reduce the size of high-dimensional data, thus minimizing the requirements for data transfer. The problem of these methods is that they often come with big memory and computation requirements, as they are ultimately equivalent to the expensive eigenspace evaluation.

In order to use subspace analysis methods with the minimum requirements in terms of cost and energy consumption, we here rely on two specialized *streaming* algorithms for the estimation of the subspace of electroencephalogram (EEG) signals directly after the acquisition on edge devices.

The implementation of these state-of-the-art algorithms is tested on a common low-end microcontroller unit (MCU), which is an ideal candidate as edge computing digital hardware platform. The functional performance of these methods is evaluated along with the requirements in term of computational time, energy consumption and memory footprint.

I. Introduction

Nowadays, the acquisition and processing of bio-signals by means of low-cost and low-power edge devices has gained an increasing importance, due to the growing need to record large amounts of data from patients.

In order to efficiently transfer large amounts of data, it is often needed to extract only its most important features, an operation that can be performed using a plethora of methods, of which principal component/subspace analysis (PCA/PSA) is certainly one of the most widespread. In this way it is possible to pre-elaborate the acquired data right after its acquisition and before its transmission to cloud storage, and this can be generally performed on dedicated edge devices. In fact, large amounts of the collected data are often redundant during the extraction of the actual information and fully transmitting them would be wasteful.

The idea behind PCA/PSA is to find a subspace of the original signal space such that the corresponding data sets projection has some special properties and maintains most of the energy of the original data. Some of the many applications of PCA/PSA consist in pattern identification in computer network traffic analysis [1], anomaly detection [2], blind source separation [3], surveillance [4], [5], and of course biomedical applications [6], [7], tough this list is far from being complete.

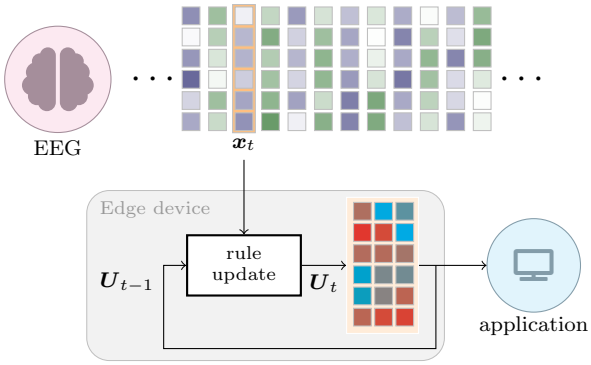

Fig. 1. General scheme of a streaming algorithm for subspace analysis in case of EEG feature extraction.

The most straightforward way of performing PCA/PSA is by means of *batch* methods, such as the eigenvalue decomposition (EVD) or the singular-value decomposition (SVD) techniques [8]. These methods work on big chunks of data at the same time, thus having high memory requirements and being computationally expensive. Because of this, they are not suited for cheap and low-power edge devices.

A good alternative to batch methods can be found in PCA/PSA streaming methods. Here, data windows are not stored but immediately processed as they are available, thus cutting down the memory requirements and the computational complexity. There is a large number of streaming algorithms and some of them have been presented recently (see, e.g., [9]–[11]).

In this paper we explore the use of PCA/PSA streaming methods for the extraction of features from EEG signals, with the dataset presented in [12]. The setup taken into account is a simplified version of [13], where EEG signals are sent to the streaming algorithm in mini-batches of multiple spatial vectors consisting of subsequent time samples and then tested on ASIC hardware. Unlike [13], our focus is on the study of two different streaming methods (the well-known Oja method [14], [15], and the more recent GROUSE [16]) and on their implementation on a low-end microcontroller that could be used as edge device. In addition, we consider a batch-size of only one spatial vector for further reduction of the memory footprint.

The paper is organized as follows: Section II describes

This is the author's version of the article that has been presented at IEEE BioCAS2021
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS49922.2021.9645035

PCA/PSA streaming algorithms and the two employed methods. Section III reports the performance of the two methods implemented in Python language with maximum precision while in Section IV the implementation on a low-end microcontroller edge device is described along with the performance results and the cost in terms of memory footprint and energy consumption. Finally the conclusion is drawn.

II. Methods description

We define a data stream as a sequence of vectors $\boldsymbol{x}_t \in \mathbb{R}^n$, with $t = 0, 1, \ldots$ containing one reading for each sensor for the same time instant t. We model $\boldsymbol{x}_t$ as an instance of a discrete-time and ergodic stochastic process characterized by a covariance matrix $\boldsymbol{\Sigma} = \mathbf{E}\left[\boldsymbol{x}_t \boldsymbol{x}_t^\top\right] \forall t$, whereas $\mathbf{E}[\cdot]$ refers to the expectation operator and $\cdot^\top$ indicates transposition.

Principal Subspace Analysis (PSA) aims at identifying the subspace spanned by the $m < n$ principal eigenvectors of $\boldsymbol{\Sigma}$, i.e., the one corresponding to the m largest eigenvalues. Principal Component Analysis (PCA) refers to the identification of the actual principal eigenvectors. The principal subspace can be expressed as the span of the columns of a matrix $\boldsymbol{U} \in \mathbb{R}^{n \times m}$, that is column orthonormal, i.e., $\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_m$, where $\boldsymbol{I}_m$ is the $m \times m$ identity matrix.

Streaming algorithms work on an estimation $\boldsymbol{U}_t$ at time t that is iteratively updated each time a new vector $\boldsymbol{x}_t$ is acquired, so that $\boldsymbol{U}_t \to \boldsymbol{U}$ when t grows (see Figure 1).

In this section we describe the two methods we want to implement and test. For each method we give the *update step*, i.e., the sequence of operations necessary to update matrix $\boldsymbol{U}_{t-1}$ to matrix $\boldsymbol{U}_t$ considering the current occurrence $\boldsymbol{x}_t$.

A. Oja

The first streaming algorithm is Oja, originally proposed in [14] for the principal subspace estimation with $m = 1$, and then extended to the rank-m cases in [15]. The algorithm starts from a random column-orthonormal matrix $\boldsymbol{U}_0 \in \mathbb{R}^{n \times m}$. The estimation of matrix $\boldsymbol{U}$ at time t is updated every time an input sample $\boldsymbol{x}_t$ is received with the following rule

$$\boldsymbol{U}_t = \Omega\left(\boldsymbol{U}_{t-1} + \gamma_t \boldsymbol{x}_t \boldsymbol{x}_t^\top \boldsymbol{U}_{t-1}\right) \tag{1}$$

where $\Omega(\cdot)$ is an operator that makes the columns of its argument orthogonal. This can be, for example, the extraction of $\boldsymbol{Q}$ matrix in the QR decomposition of the input matrix [17, Chapter 2]. Finally, γ_t is the *learning rate* or step size that in general may change with time.

Interestingly, the works in [18]–[20] regard Oja as an extension of the well-know power method described in [21] that, at the same time, is the same of solving a maximization problem where the cost function is

$$J_{\mathrm{Var}}(\boldsymbol{U}_t) = \mathbf{E}\left[\left\|\boldsymbol{U}_t^\top \boldsymbol{x}_t\right\|^2\right] = \mathrm{tr}\left(\boldsymbol{U}_t^\top \boldsymbol{\Sigma} \boldsymbol{U}_t\right) \tag{2}$$

where $\|\cdot\|$ indicates the l_2 norm, $\mathrm{tr}(\cdot)$ evaluates the trace of its argument, and $\boldsymbol{U}_t$ has a column orthonormality constraint. Additionally, $\boldsymbol{\Sigma}\boldsymbol{U}_t$ is the gradient of (2) with respect to $\boldsymbol{U}_t$ and because of this (1) is equivalent to the update of a stochastic gradient descent method where $\boldsymbol{x}_t \boldsymbol{x}_t^\top$ is the approximation of the correlation matrix $\boldsymbol{\Sigma}$, γ_t is the learning rate, and $\Omega(\cdot)$ induces the update to output a matrix that is column-orthonormal.

Furthermore, in order to reduce the overall computation, Ω operation could be applied only after a certain number of updates [18]. Anyways, the actual number of steps to run without the orthonormalize operation depends on the application.

Finally, since the convergence of the algorithms is related to the choice of the starting matrix $\boldsymbol{U}_0$, [22] suggests a way of warm start of the method that excludes the use of random matrices.

B. GROUSE

The second method, Grassmaniann Rank-One Update Subspace Estimation (GROUSE), has been proposed in [16] as a streaming subspace tracking algorithm. Even tough it has been designed to work on the cases where some elements of $\boldsymbol{x}_t$ are not known, we consider here the variation of the algorithm in the case of complete data. The operations consist in the application of the stochastic gradient descent to minimize an objective function based on the so-called *spiked model* [23] where the observable $\boldsymbol{x}$ is assumed as an expansion of an m-dimensional vector $\boldsymbol{s}$ such that $\boldsymbol{s}_t = U_t^\top \boldsymbol{x}_t$ [16], [24]. The objective function is

$$J_{\mathrm{SM}}(\boldsymbol{U}_t) = \mathbf{E}\left[\|\boldsymbol{x}_t - \boldsymbol{U}_t \boldsymbol{s}_t\|^2\right] \tag{3}$$

that is optimized while moving strictly in the set of all possible column-orthonormal matrices, i.e., the Grassmaniann manifold of the m-dimensional subspaces of $\mathbb{R}^n$.

Starting from one of these column-orthonormal matrices, the update operations for the case in which all the elements of $\boldsymbol{x}_t$ are known is

$$\begin{aligned}
\boldsymbol{p}_t &= \boldsymbol{U}_{t-1} \boldsymbol{y}_t \\
\theta_t &= \arctan\left[(1 - \alpha_t)\frac{\|\boldsymbol{r}_t\|}{\|\boldsymbol{p}_t\|}\right] \\
\boldsymbol{z}_t &= \cos(\theta_t)\frac{\boldsymbol{p}_t}{\|\boldsymbol{p}_t\|} + \sin(\theta_t)\frac{\boldsymbol{r}_t}{\|\boldsymbol{r}_t\|} \\
\boldsymbol{U}_t &= \boldsymbol{U}_{t-1} + \left(\frac{\boldsymbol{z}_t}{\|\boldsymbol{z}_t\|} - \frac{\boldsymbol{p}_t}{\|\boldsymbol{p}_t\|}\right)\frac{\boldsymbol{y}_t^\top}{\|\boldsymbol{y}_t\|}
\end{aligned} \tag{4}$$

where the expression for the evaluation θ_t comes from Eq. (3)–(4) in [25] and where α_t, that may or may not be constant with time, is employed to alleviate the effect of noise. The convergence of GROUSE is analyzed in [25], [26].

III. Performance of the methods

In this section we test Oja and GROUSE methods on an EEG dataset by detecting a subspace that characterize it. In particular, we focus on Evoked Potentials (EPs) recordings of brain electrical activity evoked by an external stimulus that is delivered repeatedly to the subject. Data consists in actual EP recordings taken from a normal-hearing subject who

This is the author's version of the article that has been presented at IEEE BioCAS2021
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS49922.2021.9645035

performed a simple auditory task that consisted in listening to one second spaced speech syllables.

The dataset used is the same of [12]. The signals were acquired through 32 brain channels, with a sampling frequency of 512 Hz and then filtered with offline denoising algorithms as described in [12]. Even tough the offline filtering contrasts with the purpose of this work, we use it to obtain a clean dataset useful as a starting point for the exploration of the algorithms applied to real-world biosignals. Time-Shift PCA (TSPCA) [27] is first applied in order to remove environmental noise. This filtering algorithm consists in delaying a pair of extra channels specifically used as noise reference that are then orthogonalized and used as a basis to project the brain channels. Channels are finally cleaned by removing the signal projected on this base. After this, Sensor Noise Suppression (SNS) [28] is used to filter the resulting signals. This technique uses the assumption that all the relevant information is acquired by more than one channel. Because of this, noise reduction can be performed by projecting each signal on the subspace spanned by its neighbor signals and then replaced by the result. During this process, glitches and wide-band noise not present in the neighbor signals is removed, while shared features are kept. Finally, a spatial filter is used to remove unwanted physiological sources. A blind source separation technique called Denoising Source Separation (DSS) [29] is used to partition the recorded activity into components that are stimulus-related and stimulus-unrelated, based on the reproducibility of stimulus-evoked signals.

Having 32 EEG tracks, data can be represented so that, for each time sample, we have a vector $x_t \in \mathbb{R}^n$, with $n = 32$. The total number of time samples is 357888 and each sample is sent to the streaming algorithms one after another.

We set as target the subspace matrix $\boldsymbol{U} \in \mathbb{R}^{n \times m}$ evaluated with EVD batch PCA. The columns of $\boldsymbol{U}$ are the m eigenvectors corresponding to the m greatest eigenvalues of the correlation matrix $\boldsymbol{\Sigma}$.

To quantify the effectiveness in subspace analysis, we use a Python framework running on a double precision Intel CPU architecture and for both methods we monitor the sequence of reconstruction errors

$$e_t = \|\boldsymbol{U} - \boldsymbol{U}_t \boldsymbol{U}_t^\top \boldsymbol{U}\|_F \tag{5}$$

where $\|\cdot\|_F$ indicates the Frobenius norm of its argument.

In case of correct estimation we have $\boldsymbol{U}_t = \boldsymbol{U}$ and thus $e_t = 0$ while the error is maximum when $\boldsymbol{U}_t$ is orthogonal to $\boldsymbol{U}$, yielding $e_t = m$.

Tests are performed to obtain a subspace with $m = 4$ and results in terms of e_t are shown in Figure 2. The trend of e_t with growing t is highly dependent on the parameters γ_t for Oja and α_t for GROUSE. In fact, a bigger γ_t (or a lower α_t) results in a faster convergence, but the algorithm produces larger e_t values. Because of this we show the curves for two different constant values of the parameters for each method ($\gamma_t = 5 \cdot 10^{-3}$ and $\gamma_t = 5 \cdot 10^{-5}$ for Oja, $\alpha_t = 0.985$ and $\alpha_t = 0.9995$ for GROUSE), so that it is possible to appreciate

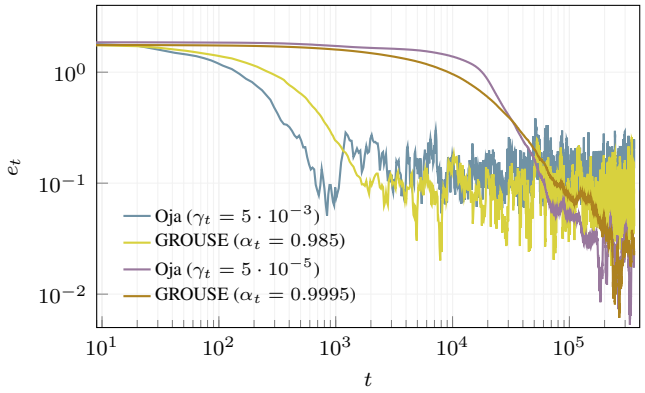

Fig. 2. Performance comparison for subspace identification on Python framework with 357888 EEG samples with $n = 32$, $m = 4$. For each method are presented two different curves, with different parameters γ_t and α_t.

the trend variation. By having a varying γ_t and α_t, it would be possible to maximize the convergence speed and stability of the algorithms.

As expected, the two methods are able to deal with the subspace identification task with an $e_t < \frac{1}{10}$. GROUSE is capable to keep the same performance of Oja even without the orthonormalize operation.

IV. Implementation on Edge Devices

We propose the implementation of the two streaming PCA/PSA methods on edge devices. In particular, we choose for the tests a low-end device such as STM32H743ZIT (Rev. V), based on ARM Cortex M7 architecture.

In order to perform the task, we developed a C library[1] targeting the ARM Cortex microcontroller (MCU) family of devices. The library mainly uses the well know ARM CMSIS-DSP library[2] for algebraic computation.

The methods are implemented with the maximum effort in minimizing the memory requirements and the computation time by means of techniques such as loop unrolling and register blocking operations.

While the operations for GROUSE only need some minor care in order to minimize the overall cost, Oja requires extra attention due to the orthonormalization operator $\Omega(\cdot)$ used in (1). $\Omega(\cdot)$ is implemented as QR decomposition, which comes to be the main bottleneck of the method.

As the QR decomposition is to be applied to matrix U_t, which is tall ($n > m$), a good choice for its implementation is to employ Cholesky decomposition (Chol-QR) [30], one of the cheapest algorithms for this task. While this algorithm is generally unstable for many applications, it is revealed to be good enough in this case as shown in the numerical evidences.

Chol-QR algorithm is described in Algorithm 1.

As previously mentioned, we employ a STM32H743ZIT (rev. V) MCU based on ARM Cortex M7 family with a 32-bit floating point unit, $f_{\text{CLK}} = 480\,\text{MHz}$ and both instruction

[1] online repository https://github.com/SSIGPRO/streaming_pca
[2] online repository https://github.com/ARM-software/CMSIS_5

This is the author's version of the article that has been presented at IEEE BioCAS2021
The editorial version of the paper is available at http://dx.doi.org/10.1109/BioCAS49922.2021.9645035

Algorithm 1 Cholesky-based QR decomposition

1: $cholesky(\boldsymbol{A}^\top \boldsymbol{A}) \rightarrow \boldsymbol{L}\boldsymbol{L}^\top$ where $\boldsymbol{L}$ is lower triangular
2: $\boldsymbol{R} \leftarrow \boldsymbol{L}^\top$
3: $\boldsymbol{R}^{-1} \leftarrow BS(\boldsymbol{R})$ (BS: backward substitution)
4: $\boldsymbol{Q} \leftarrow \boldsymbol{A}\boldsymbol{R}^{-1}$

TABLE I
PERFORMANCE ON STM32H743ZIT (REV. V) @1.8 V, 480 MHz, CACHE ON ($n = 32, m = 4$)

Method	Clock cycles per update	Time per update	Energy per update (no peripherals)	Energy per update (all peripherals)
Oja	5240	10.92 µs	2.91 µJ	4.44 µJ
GROUSE	2394	4.99 µs	1.33 µJ	2.03 µJ

cache and data cache enabled[3]. The case study used is the same described in Section III.

For each iteration, energy consumption is estimated as $E_{\text{update}} = V_{\text{DD}} \times I_{\text{DD}} \times t_{\text{update}}$, where V_{DD} is the supply voltage, I_{DD} is the absorbed current and t_{update} is the time per update, evaluated as the measured number of clock cycles divided by the clock frequency. In particular, the values of I_{DD} are obtained from datasheet of STM32H743ZIT (rev. V) considering $V_{\text{DD}} = 1.8\,\text{V}$ and the two cases where either no peripherals or all the peripherals are enabled.

Table I shows time per update and energy consumption for both methods.

Neglecting the stack memory, which is very small, we evaluate the memory footprint as the sum of the memory necessary for the storage of the matrix $\boldsymbol{U}_t$, the vector $\boldsymbol{x}_t$ and the extra memory buffers required by each method.

For each method, the size of the extra buffers and the overall memory rule, along with the actual memory footprint for the case study ($n = 32, m = 4$) are reported in Table II.

Finally, the performance on the subspace identification of the two methods running on MCU is reported in Figure 3. The variation compared to Figure 2 is minimal as the MCU keeps a good subspace extraction performance on the dataset.

V. CONCLUSION

We have reviewed and tested two different streaming methods for subspace identification, with the aim of reducing the dimensionality of the acquired data directly at the edge, before it is transferred to the cloud storage.

In order to assess the performance of these algorithms, we have extracted the principal subspace of a group of recorded real EEG signals. The functional performance has been tested with a high-precision architecture running a Python framework.

As the main target of the work is to make these algorithm work on low-end edge devices, further tests have been performed directly on an MCU device running a dedicated framework written in C. Special care has been dedicated to the minimization of computational time and memory footprint.

The algorithms show good performance in the extraction of the principal subspace of EEG data, even being lightweight both in terms of computational complexity, memory footprint and working on a low-end, energy efficient device.

[3]code has been compiled with fast target gcc option (-Ofast) in order to maximize the speed performance.

TABLE II
MEMORY REQUIREMENTS RULE FOR EACH METHOD (MEMORY EXAMPLE WITH $n = 32$,M = 4, 32-BIT SCALARS)

Method	Size of extra buffers	Overall memory rule	Memory example
Oja	$k \times k$	$n(k+1)+k^2$	704 B
GROUSE	$n,\ k$	$n(k+2)+k$	784 B

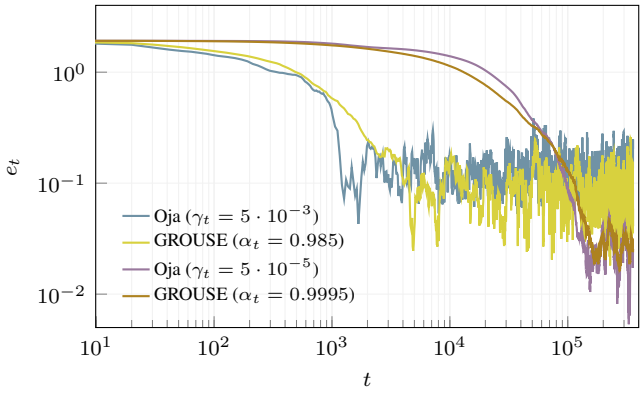

Fig. 3. Performance comparison for subspace identification on STM32H743ZIT (rev. V) with 357888 EEG instances with $n = 32$, $m = 4$. For each method are presented two different curves, with different parameters γ_t and α_t.

REFERENCES

[1] P. Zhang, X. Huang, X. Sun, H. Wang, and Y. Ma, "Privacy-preserving anomaly detection across multi-domain networks," in *9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2012, pp. 1066–1070. doi: 10.1109/FSKD.2012.6234272

[2] A. Marchioni, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "Subspace energy monitoring for anomaly detection @Sensor or @Edge," *IEEE Internet of Things Journal*, pp. 1–1, 2020. doi: 10.1109/jiot.2020.2985912

[3] Xiao-Long Zhu, Xian-Da Zhang, Zi-Zhe Ding, and Ying Jia, "Adaptive nonlinear PCA algorithms for blind source separation without prewhitening," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 3, pp. 745–753, Mar. 2006. doi: 10.1109/TCSI.2005.858489

[4] L. A. Thomaz, E. Jardim, A. F. da Silva, E. A. B. da Silva, S. L. Netto, and H. Krim, "Anomaly Detection in Moving-Camera Video Sequences Using Principal Subspace Analysis," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 3, pp. 1003–1015, Mar. 2018. doi: 10.1109/TCSI.2017.2758379

[5] C. Ding *et al.*, "Non-contact human motion recognition based on UWB radar," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 2, pp. 306–315, Jun. 2018. doi: 10.1109/JETCAS.2018.2797313

[6] V. Senger and R. Tetzlaff, "New signal processing methods for the development of seizure warning devices in epilepsy," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 5, pp. 609–616, May 2016. doi: 10.1109/TCSI.2016.2553278

[7] B. Yu, T. Mak, X. Li, F. Xia, A. Yakovlev, Y. Sun, and C. S. Poon, "Real-time FPGA-based multichannel spike sorting using hebbian eigenfilters," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 4, pp. 502–515, Dec. 2011. doi: 10.1109/JETCAS.2012.2183430

[8] I. T. Jolliffe, *Principal components analysis*. Elsevier, 1986. ISBN 978-0-08-044894-7

[9] L. Balzano, Y. Chi, and Y. M. Lu, "Streaming PCA and Subspace Tracking: The Missing Data Case," *Proceedings of the IEEE*, vol. 106, no. 8, pp. 1293–1310, Aug. 2018. doi: 10.1109/JPROC.2018.2847041

[10] T. V. Marinov, P. Mianjy, and R. Arora, "Streaming principal component analysis in noisy setting," in *Proceedings of the 35th international conference on machine learning*, J. Dy and A. Krause, Eds., vol. 80. PMLR, Jul. 2018, pp. 3413–3422.

[11] N. Lassami, A. Aïssa-El-Bey, and K. Abed-Meraim, "Low cost sparse subspace tracking algorithms," *Signal Processing*, vol. 173, p. 107522, Aug. 2020. doi: 10.1016/j.sigpro.2020.107522

[12] N. Bertoni *et al.*, "Low-power eeg monitor based on compressed sensing with compressed domain noise rejection," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 522–525. doi: 10.1109/ISCAS.2016.7527292

[13] T. Wu, W. Zhao, H. Guo, H. H. Lim, and Z. Yang, "A streaming pca vlsi chip for neural data compression," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 6, pp. 1290–1302, 2017. doi: 10.1109/TBCAS.2017.2717281

[14] E. Oja, "Simplified neuron model as a principal component analyzer," *J. Math. Biol.*, vol. 15, no. 3, pp. 267–273, 1982. doi: 10.1007/BF00275687

[15] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. Appl.*, vol. 106, no. 1, pp. 69–84, Feb. 1985. doi: 10.1016/0022-247X(85)90131-3

[16] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *48th annu. Allert. Conf. Commun. Control. Comput. Allert. 2010*, 2010, pp. 704–711. doi: 10.1109/ALLERTON.2010.5706976

[17] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[18] R. Arora, A. Cotter, K. Livescu, and N. Srebro, "Stochastic optimization for PCA and PLS," in *50th annual allerton conference on communication, control, and computing (allerton)*, Oct. 2012, pp. 861–868. doi: 10.1109/Allerton.2012.6483308

[19] I. Mitliagkas, C. Caramanis, and P. Jain, "Memory limited, streaming PCA," in *Adv. Neural inf. Process. Syst. 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2886–2894.

[20] M. Hardt and E. Price, "The noisy power method: A meta algorithm with applications," in *Adv. Neural inf. Process. Syst. 27*, G. Z., W. M., C. C., L. N. D., and W. K. Q., Eds. Curran Associates, Inc., 2014, pp. 2861–2869.

[21] G. H. Golub and F. V. L. Charles, *Matrix computations*. Johns Hopkins University Press, 2012. ISBN 978-1-4214-0794-4

[22] Z. Allen-Zhu and Y. Li, "First efficient convergence for streaming k-PCA: A global, gap-free, and near-optimal rate," in *58th annual symposium on foundations of computer science (FOCS)*, vol. 2017-10. IEEE Computer Society, Nov. 2017, pp. 487–492. doi: 10.1109/FOCS.2017.51

[23] I. M. Johnstone, "On the distribution of the largest eigenvalue in principal components analysis," *Ann. Statist.*, vol. 29, no. 2, pp. 295–327, Apr. 2001. doi: 10.1214/aos/1009210544

[24] Y. Chi, Y. C. Eldar, and R. Calderbank, "PETRELS: Subspace estimation and tracking from partial observations," in *IEEE int. Conf. Acoust. Speech signal process. (ICASSP)*, Mar. 2012, pp. 3301–3304. doi: 10.1109/ICASSP.2012.6288621

[25] D. Zhang and L. Balzano, "Global convergence of a Grassmannian gradient descent algorithm for subspace estimation," in *Proc. 19th int. Conf. Artif. Intell. Stat.*, A. Gretton and C. C. Robert, Eds., 2016, pp. 1460–1468.

[26] L. Balzano and S. J. Wright, "Local convergence of an algorithm for subspace identification from partial data," *Found. Comput. Math.*, vol. 15, no. 5, pp. 1279–1314, Oct. 2015. doi: 10.1007/s10208-014-9227-7

[27] A. de Cheveigné and J. Z. Simon, "Denoising based on time-shift pca," *Journal of Neuroscience Methods*, vol. 165, no. 2, pp. 297–305, 2007. doi: https://doi.org/10.1016/j.jneumeth.2007.06.003

[28] A. de Cheveigné and J. Z. Simon, "Sensor noise suppression," *Journal of Neuroscience Methods*, vol. 168, no. 1, pp. 195–202, 2008. doi: https://doi.org/10.1016/j.jneumeth.2007.09.012

[29] A. de Cheveigné and J. Z. Simon, "Denoising based on spatial filtering," *Journal of Neuroscience Methods*, vol. 171, no. 2, pp. 331–339, 2008. doi: https://doi.org/10.1016/j.jneumeth.2008.03.015

[30] T. Terao, K. Ozaki, and T. Ogita, "LU-Cholesky QR algorithms for thin QR decomposition," *Parallel Computing*, vol. 92, p. 102571, Apr. 2020. doi: 10.1016/j.parco.2019.102571