# Sleep Stage Classification with Stochastic Bayesian Inference

L. E. Calvet, J. S. Friedman, D. Querlioz
Centre de Nanosciences et de Nanotechnologies
Université Paris-Sud-CNRS-Université Paris-Saclay
91405 Orsay, France
laurie.calvet@u-psud.fr, damien.querlioz@u-psud.fr

P. Bessière, J. Droulez
Institut des Systèmes Intelligents et de Robotique
Université Pierre et Marie Curie-CNRS,
Paris, France

## ABSTRACT
The design of electronic circuits that can realize Bayesian inference is an important goal for exploiting machine learning in a fast and efficient way. We recently developed a novel architecture based on stochastic computation with Muller C-elements that can realize a circuit level naïve Bayes inference. This technique can be implemented using low power nanodevices exhibiting faults and device variations. Here we show how a more complex classification problem can be transformed into a simple circuit using this framework where an effective classification can be obtained with a minimal amount of information. This suggests that substantially smaller spatial footprints for portable devices could ultimately be achieved.

## CCS Concepts
• **Computer Systems Organization→Embedded and cyber-physical systems→Embedded systems** • **Hardware→Emerging technologies** • **Computing methodologies→Machine Learning → Learning paradigms→Supervised Learning**

## Keywords
Stochastic computing, Bayesian inference, Muller C-element, Sleep classification, Biomedical data

## 1. INTRODUCTION
Stochastic computing uses random binary sequences, called stochastic bitstreams, to represent and calculate probabilities. The data input for such systems is noisy and redundant compared to the more deterministic digital representation in conventional computing; however, the collection of techniques developed for stochastic computations can provide a compact implementation of many complex operations [1]. Taking advantage of such operations, researchers realized that using stochastic neurons would substantially simplify the hardware implementation, based on field programmable gate arrays (FPGAs) of neural networks [2,3]. Another closely related application of stochastic computing is its use in decoding [4] and the subsequent application as a strategy for fault-tolerant nanoscale computing architectures [5,6]. The novel research by the Reidal et al have explored novel architectures to implement stochastic computing using self-assembled arrays of nanowires. [7]


*Nanoarch '16,* July 18-20, 2016, Beijing, China

$15.00
DOI: http://dx.doi.org/10.1145/2950067.2950085

Inspired by this research, we recently showed that Bayesian inference can be accomplished by stochastic computation using Muller C-Elements elements [8]. Other hardware implementations of Bayesian networks include analog CMOS circuits [9] and the use of FPGAs [10], both of which require a substantially larger circuit area. In addition, the stochastic computation uses progressive precision so that the accuracy of the result can be adjusted by choosing the computing time. Our architecture exhibits a high degree of error tolerance so that error-prone nano-devices can be used. Here we show how the C-element architecture can be used to perform and optimize the classification of a multi-state problem, following on our initial paper [8] where just two categories were considered and the details of the features were not discussed.

For this purpose, we consider a model classification problem where electroencephalography (EEG) data is used to classify sleep into three stages [11, 12], and show how the architecture of [8] can be adapted for this task. An important step in classifying this data is the feature selection process, which decides the best set of features exploited for classification. It is especially crucial as the circuit architecture of C-elements changes with the number of selected features. By comparing the results for the exact naïve Bayes classification and that of the circuit of C-elements, we explore how the performance for different numbers of features in terms of both the data represented in these features and the circuit architecture.

Our choice of classification problem is unusual for the Naïve Bayes classification. Naïve Bayes [13] estimations have been used most successfully for classifying data containing many features, such as spam filters. Our results here show that C-element naïve Bayes estimators can provide good classification results for problems with just a few well-chosen features using a simple circuit. The main disadvantages of our technique are that the bitstreams for stochastic computing must be uncorrelated, and that the C-element architecture creates inherent correlations. While there are some solutions to the first problem [1], it represents the greatest challenge for this implementation. With the results here, we show how a judicious choice of features can surmount the second problem of autocorrelations created by C-elements.

## 2. NAÏVE BAYES INFERENCE WITH MULLER C-ELEMENTS
We first review the naïve Bayesian inference classification procedure. The training data $(X,Y)$ consists of data $X$, containing $K$ samples with $L$ features, each one denoted $f_i$, and $Y$, a vector that classifies the data into $N$ categories, each one denoted by $C_n$. The priors $p(C_n)$ and the conditional probabilities $p(f_i|C_n)$ for each $f_i$ and $C_n$ are first determined. In a Gaussian naïve Bayes classification $p(f_i|C_n)$ is determined by assuming that the distribution of features follows a Gaussian distribution. In a multivariate Bernoulli naïve Bayes

classification, the number of features are counted and typically Laplace's rule of succession is employed to determine $p(f_i|C_n)$ for each feature and category. A fundamental assumption in this classification is that the features are conditionally independent of each other. It then follows from Bayes' rule that for a given set of features the probability for a given category is:

$$P(C_n \mid \{f_1, \ldots, f_L\}) = \frac{P(C_n)\prod_{l=1}^{L} P(f_l|C_n)}{\sum_m P(C_m)\prod_{l=1}^{L} P(f_l|C_m)}. \qquad (1)$$

For a test sample $x_{test}$ containing $f_L$ features, the best classification is found by choosing the maximum conditional probability $p(C_n \mid \{f_1, \ldots, f_L\})$ of all the categories.

The canonical example of a data set that can achieve excellent classification with this type of procedure is a spam detector in which the categories are spam and ham (legitimate message) and the features are individual words that make up an email message. There are several variations on the multivariate Bernoulli naïve Bayes classification, in particular the multinomial naïve Bayes model where the frequencies of the words can be taken into account. At present we only consider the multivariate Bernoulli naïve Bayes classification which is easily implemented using C-elements, but other models [13] could easily be demonstrated using slight circuit modifications.

Stochastic computing uses sequences of independent and random bitstreams to encode probabilities. Each bit in this bitstream has a probability *P* of being '1' and a probability of 1-*P* of being 0. For example, the 10 bit bitstream 1010001010 would represent a probability of 0.4. This type of implementation allows a simplified computation of many complex operations; for instance, the product of probabilities is computed by a simple AND gate [1].

Figure 1(a) and 1(b) show the Muller C-element, which can be implemented with 8 transistors [14], as well as its truth table. In this device, the output state is maintained unless the two inputs are equivalent, in which case it switches to the value shared by the inputs. If we consider input bitstreams X and Y as representing probabilities P(X) and P(Y) and assume that the output bitstream *Z* is a stationary signal, it is straightforward to demonstrate that the output represents the probability *P*(*Z*), is [8]:

$$P(Z) = \frac{P(X)P(Y)}{P(X)P(Y) + (1 - P(X)(1 - P(Y))}$$

By associating $P(Z) = P(C_n|f_i)$, $P(Y) = P(C_n)$ and $P(X) = P^*(f_l) = \frac{P(f_l|C_n)}{P(f_l|C_n)+P(f_l|NC_n)}$, where $P(f_i|NC_n)$ is the probability that the feature is not present in category $C_n$, we see that a simple C-element can perform the complete inference of Bayes' rule [8].

In order to perform a more complicated Bayesian inference, C-elements can be cascaded together. To compute an inference with two features, two C-elements can be used: $P^*(f_1)$ and $P^*(f_2)$ are inputs to the first C-element, and this C-element output serves alongside $P(C_n)$ as the inputs for the second C-element. As more features are added, additional stages are included in a cascaded series. However, the output signals in the later stages switch less frequently than those in the earlier stages. This effect, known as autocorrelation, is intrinsic to this architecture and can degrade the quality of the result. For example, a C-element in which both inputs are random signals with a probability of 0.5 outputs a signal that is also random and has probability of 0.5; however, while the inputs are expect to switch in 50% of the clock cycles, the output is expected to switch in only 25% of clock cycles. To mitigate this concern, the architecture can be reorganized into a tree structure, as shown in Figure 1(c), in which the maximum number of C-elements is included at each stage. It also implies that minimizing the number of stages in the tree leads to more accurate classifications. As each input feature requires a C-element, real-world classification problems should contain a maximum amount of information in a minimal amount of features.

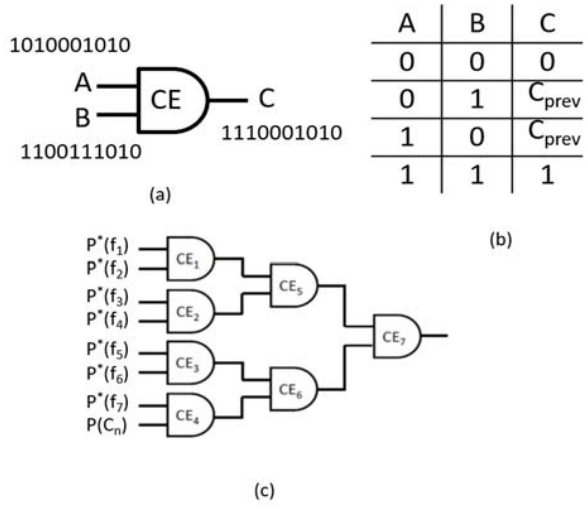


**Figure 1. (a) Schematic of a Muller C-element with sample stochastic bitstreams representing probabilities of 0.4 and 0.6 and output 0.5. (b) The truth table of the Muller C-element. (c) A sample C-element tree for seven features and one prior. Note that one C-element tree is used for each category [8].**

To use this architecture in a multi-category inference problem, a C-element tree is used to compute the probability for each category. The maximum probability of all the trees is the most likely choice.

## 3. SLEEP STAGE CLASSIFICATION USING EEG DATA

In this work we consider a practical application, sleep stage classification based on EEG data [11]. This may be considered eventually in the context of portable biomedical devices. We classify the data into three categories $C_n$, and adopt an 'all or nothing' strategy for computing the probability of inclusion in a given category versus being in any other. Each category is thus considered as a binary variable in the C-element architecture and its probability is obtained. The maximum relative probability is chosen as the classification result. Note that this is different from an exact naïve Bayes solution which considers one variable with all possible states (equation 1). As a result, the posterior distribution of the C-element calculation is not normalized.

Classifying the stages of sleep is an important part of diagnosing sleep disorders, which can result in both a reduction of quality of life and physiological dysfunction. Medical practitioners typically perform a polysomnography which is a quantitative record of different biophysical tests that are taken during sleep including: electrical changes in the brain via EEG, eye movements using electro-oculography (EOG), muscle activity by electromyography (EMG) and heart beat by electrocardiography (ECG). Data sets are divided into 30 second epochs and 6 different sleep stages are assigned manually to each epoch. Previous research has investigated automatic sleep stage classification versus that done by human experts [15]. It was found that while machine learning algorithms were only 60% effective in scoring sleep stages that were viewed as ambiguous by

humans, stages that were unambiguous scored over 90% in the automatic classification.

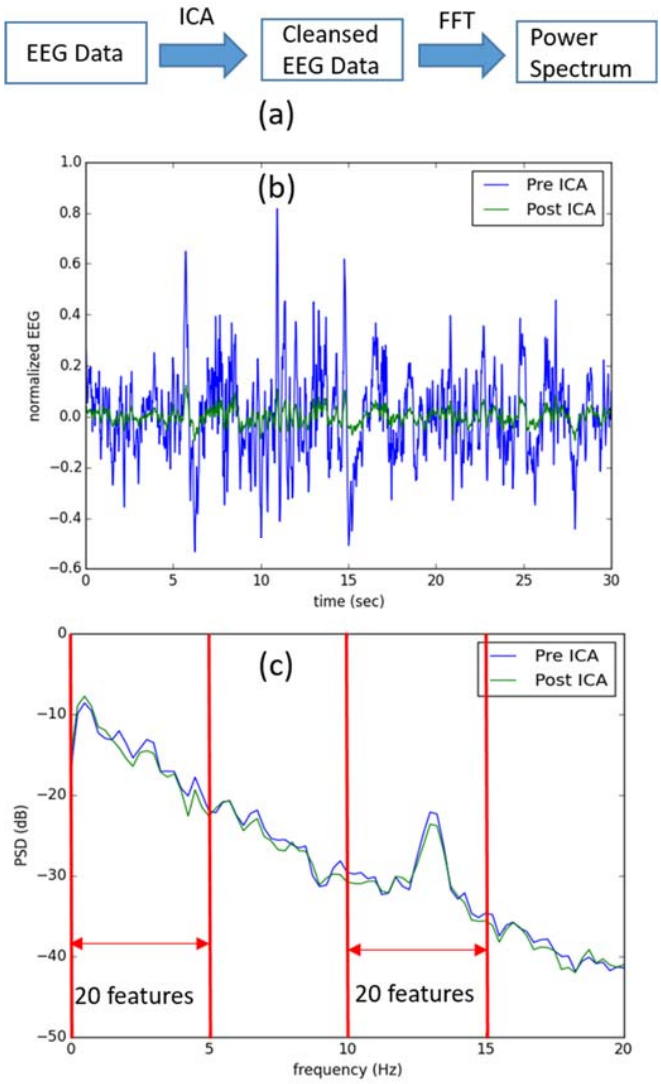


**Figure 2. Data prepration for the sleep data. (a) Schematic of the the procedure. The raw EEG data is cleansed using an independent component analysis (ICA) using the sci-kit learn python module [11] and then a FFT is obtained. (b) Example of the EEG data before and after ICA. (c) Forty frequencies of the resultant EEG data were used for the classification.**

We considered EEG data and classified sleep into the three Non Rapid Eye Movement (NREM) stages: stage 1 'transition sleep', stage 2 'light sleep', and stage 3 'deep sleep'. All the data was from a single night polysomnography. In this data set [12], researchers scored a sleeping individual during a night of sleep independently and four EEG channels were simultaneously measured. We eliminated all epochs that did not fall into these three stages. Note that the proportions of each stage experienced during the night are not equivalent: 11.6% of time was spent in stage 1, 62.4% was spent in stage 2, 25.9% was spent in stage 3. The inclusion of these priors in the C-element circuits is critical for obtaining an accurate classification. The data set was separated into 498 epochs for training and 332 epochs for testing and the ratio between the different sleep stages proportion remained the same in each subset.

For our simulations, we first averaged the four EEG channels. An independent component analysis (ICA), the main accepted solution for obtaining a clean EEG with reduced artifacts [16], was used to reduce the noise using the sci-kit learn python library [17]. A power spectrum of the data was then taken. Figure 2 provides an example and a schematic of how the data was preprocessed.

To efficiently encode the data with the fewest inputs, we used a feature selection process. The use of fewer features minimizes the interdependence of the features and also the number of stages in the calculation, thus reducing the autocorrelation resulting from the use of C-elements. We used a feed-forward wrapper method. To choose the best additional feature to add to an initial set, each remaining feature is added alone to the initial set and the classification is performed. The sets that give the 'best' classification are kept for the next round. The feature selection was performed with a Gaussian naïve Bayes classification from sci-kit learn learn python library [17]and the exact multivariate binomial Bayes classification that the C-element architecture approximates. Here we consider the cleansed data with different sets of optimal features, varying from two to 20.

The type of data used here would more typically be considered in a Gaussian naïve Bayes framework because the features are continuous, as opposed to discrete values (*e.g.*, the presence of absence of a word for a spam detector). We transform the data into the format of a multivariate binomial naïve Bayes classification, as shown in Figure 3. With the cleansed feature selected data (PSD + FS), we calculate the mean and standard deviation for a state $C_n$. We then count the number of times that each feature is within one standard deviation of the mean. Laplace's rule of succession is used to calculate $P(f_L|C_N)$ and $P(f_L|NC_N)$.

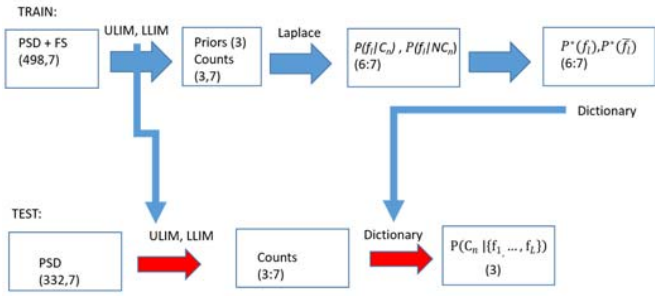


**Figure 3. Schematic of the conversion of the pre-processed data for the multivariate naïve Bayes estimation. The resultant power spectral density (PSD) and the 7 best features from the feature selection (FS) are the starting point for the 498 train data samples. From this data we calculate the mean and standard deviation for each feature and then for each feature of each sample its presence or absence is determined by whether it is within one standard deviation from the mean (upper limit (ULIM) and lower limit (LLIM)). We then obtain the priors and the number of times each feature is present or absent (Counts for each stage) and using Laplace's law of succession to find $P(f_l|C_n)$ and $P(f_l|NC_n)$ for each of the sleep stages. With this information from the train data, we obtain the counts and the probabilities for the test data and can perform the C-element calculation using a cascaded array as in Figure 1.**

We thus create a dictionary of probabilities containing $P^*(f_l) = \frac{P(f_l|C_n)}{P(f_l|C_n)+P(f_l|NC_n)}$ for the presence and $P^*(\bar{f}_l) = \frac{P(\bar{f}_l|C_n)}{P(\bar{f}_l|C_n)+P(\bar{f}_l|NC_n)}$ for the absence of each particular feature in each category. For a given test sample, these probabilities as well as the prior are used as the

inputs into the C-element architecture after being converted into stochastic bitstreams via a pseudo-random number generator. For each of the three sleep stages and for each test sample we construct a C-element tree. The most likely category of the test sample is determined as the highest probability of the three trees and is assigned as the result. For instance, in Figure 1(c) and Figure 3, we consider seven features plus the prior for each of the three stages of sleep. We calculate the probability of each state based on the probability of each feature of the test sample being present or not present. It should be noted that the total probability of all states does not necessarily add to one, because the posterior distribution is not normalized. This multi-state technique broadens the applications far beyond the two categories considered by our previous work [8].

## 4. RESULTS

Using all 40 features, the test error for the Gaussian naïve Bayes classification, performed using the sci-kit learn python library [17], was found to be 74% and using the multivariate binomial naïve Bayes classification the test error was 76%. Concerning the C-element calculation, for all 40 features and 100,000 bits we found a test error of 73.5 ± 0.3%. As a comparison, we used other classification algorithms in the sci-kit learn library and found that the data performance on all pixels using a logistic regression was 82.5% and using the support vector machine algorithm 84.3%.

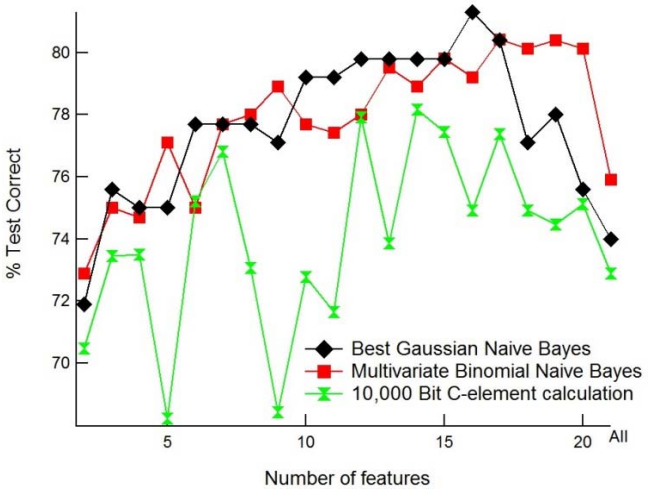


**Figure 4. Test error as a function of the number of features for the three classifications discussed in the paper. The multivariate binomial naïve Bayes (exact solution to the C-element architecture) performs better than the Gaussian naïve Bayes when more features are included. The variation in the C-element calculation is analyzed in section V and in Figure 6.**

As shown in Figure 4, by performing feature selection we are able to significantly improve the test error in the naïve Bayes framework. This figure shows the best results of feature selection for the Gaussian and multivariate naïve Bayes frameworks (exact solution to the C-element tree calculation) discussed in this paper as well as the average of ten trials for the 10,000 bit C-element calculation. The best results were obtained at 16-20 features for the multivariate binomial naïve Bayes (with a test error of 80.4%) and at 16 features for the Gaussian naïve Bayes (with a test error of 81.3%). Note when choosing which sets to use for the C-element architecture we choose the best exact result (the multivariate binomial naïve Bayes feature selection). The C-element architecture showed the best classification at twelve and 14 features with a test error of 77.9%.

Figure 5 shows the evolution of the C-element architecture as the number of bits is increased. In this graph each marker represents the average of ten calculations and the error bars indicate the standard deviation. We observe that the precision at 1,000 bits is sufficient to obtain an excellent classification.

## 5. DISCUSSION

Figures 4 and 5 demonstrate that feature selection is an extremely valuable technique for applications of this C-element stochastic Bayesian inference architecture, and is therefore an important step prior to fabricating such a system. We observe that by reducing the number of features we obtain not only a more energy-efficient and simplified circuit, but also an improved classification. We also observe that with just 1,000 bits, we obtain an excellent classification. We find a trade-off between the best feature selection in the exact calculation, which occurred at higher features, and the calculation obtained by the C-element architecture using fewer features. This difference is attributed to the increased autocorrelation as the number of C-element stages in the circuit is increased. With greater than 15 feature, the improved potential in the classification with additional features does not compensate for the additional autocorrelation observed with the addition of a fourth stage to the C-element architecture.

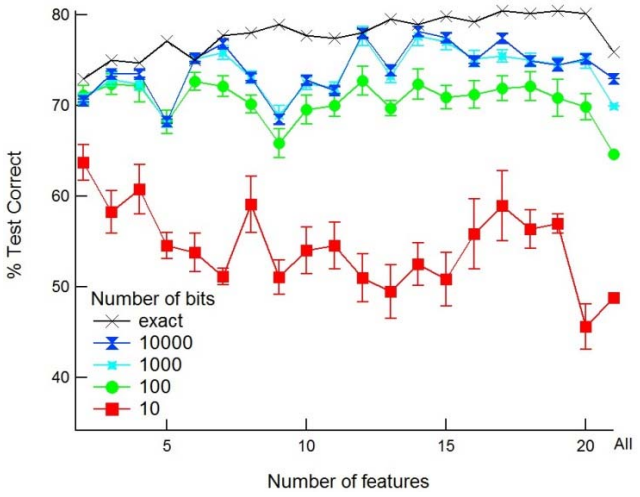


**Figure 5. Evolution of the C-element architecture for different bitstream lengths. We find the best results at 12 and 14 features, and that the use of just 1,000 bits is sufficient**.

The results of the C-element architecture are observed to be highly dependent on the autocorrelation and the specific tree structure that arises from the number of features. To highlight this, Figure 6 shows how the percentage difference between the test error of the 10,000 bit C-element architecture and the test error of the exact multinomial binomial calculation vary as a function of number of inputs in the C-element architecture. Note that the number of inputs includes the prior and is thus one larger than the number of features in Figure 5.

The C-element tree can naturally give rise to intrinsic asymmetries depending on the architecture. In Figure 6, the dashed lines indicate the points at which the number of inputs is a power of two (4,8,16). At these points there are no intrinsic asymmetries in the architecture itself. With each additional power of two, an additional C-element stage is added to the circuit. If the total number of inputs

is not a power of two, then at some point in the architecture, the output of a C-element is not fed directly into the subsequent layer. For instance, with three inputs, the prior is used as the input in the second stage rather than the first stage as with $P^*(f_1)$ and $P^*(f_2)$.

There is also a fundamental asymmetry in our classification scheme which is related to the priors. For this classification task, the priors are: stage 1: 11.6%; stage 2: 62.4%; and stage 3: 25.9%. This prior input will therefore be particularly prone to autocorrelation errors for the stage 1 calculation, especially if the prior is coupled with another C-element input whose probability is either close to 1 or to 0. This observation explains why in Figure 6 there are large percent differences in the architectures depending on the underlying structure. For instance, for seven and 13 inputs the percent difference is low, indicating an excellent classification. Here the prior is used as a C-element input in the second to last stage of the classification, which is particularly beneficial for minimizing the autocorrelation.

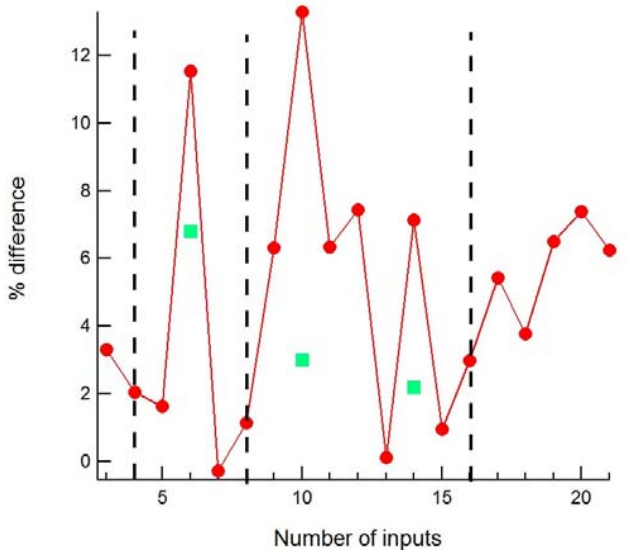


**Figure 6. Percent difference between the test error of 10,000 bit C-element architecture and the test error of the exact multinomial naïve Bayes classification. The square markers indicate the % difference when the prior is placed as the first input instead of as the last input.**

On the other hand, for six and ten inputs in particular (and also to a lesser extent for 14 and 18 inputs), there is a significantly worse classification. Here, the prior is coupled with another feature in stage 1 and its output is then used at a later stage. The feature it is coupled with, however, is the last feature added in the feature selection process, therefore, the one containing the least additional information of all the features. It is thus more likely to contain extreme probabilities. We therefore re simulated the system for 6,10,14 and 18 inputs where the prior was included as the first input and therefore coupled with the feature that provides the most information. In this way we obtained significantly better results, as shown by the additional points in Figure 6. We thus see that by a judicious choice of feature selection and architecture, the autocorrelation can be minimized so that the C-element classification approaches the exact solution.

A C-element circuit permits a much smaller footprint as compared to using a general-purpose processor for a classification/inference problem. Recent research has demonstrated the possibility of obtaining a circuit computation of a fast Fourier transform (FFT) with a significantly reduced footprint [18]. While this paper is a proof of principle, EEG data has many types of biophysiological [19] applications including brain-computer interfaces (tracking brainwaves to move objects), psychological experiments to understand brain function, epilepsy prediction, monitoring brain function (including anesthesia) in hospitals [20], and virtual reality for instance to monitor emotions.

## 6. Conclusion

We have demonstrated that stochastic computation of the naïve Bayes classification can successfully be extended to more complex classifications involving many features and several categories. Our results indicate that the optimization of both feature selection and the C-element architecture can be used advantageously to minimize autocorrelation intrinsic to the C-element inference and thus obtain an inference that can be equivalent to the exact result. These results open the way for compact stochastic computing implementations of biomedical circuits, appropriate for leveraging low-power nanodevices exhibiting faults and device variations.

## 6. Acknowledgment

This work was supported by the FP7 ICT BAMBI (FP7-ICT_2013-C) project and a public grant overseen by the French National Research Agency (ANR) as part of the 'Investissements d'Avenir' program (LabexNanoSaclay, Reference: ANR-10-LABX-0035).

## 7. References

[1] A. Algaghi and J P Hayes, 'Survey of Stochastic Computing' ACM Trans on Embedd Comput Sys vol. 92, no 2s, 2013.

[2] S.L. Bade and B.L. Hutchings, "FPGA-based stochastic neural networks—Implementation", IEEE Workshop on FPGAs for custom Computing Machines, 1994.

[3] Y-C Kim and M.A. Shanblatt, "Architecture and statistical model of a pulse-mode digital multilayer neural network" IEEE Trans on Neur Net vol. 6, n05, pp 1109-1118.

[4] V.C. Gaudet and A.C. Rapley, "Iterative decoding using stochastic computation", Electron Lett vol. 39, no. 3, 2003.

[5] C. Winstead and S. Howard, "A probabilistic LDPC-coded fault compensation technique for reliable nanoscale computing," IEEE Trans. Circuits Syst. II, vol. 56, no. 6, pp. 484–488, Jun. 2009.

[6] C. Winstead, "C-element multiplexing for fault-tolerant logic circuits," Electron. Lett., vol. 45, no. 19, pp. 969–970, 2009.

[7] J.S. Friedman, L.E. Calvet, P. Bessière, J. Droulez and D. Querlioz, "Bayesian Inference with Muller C-Elements", accepted in IEEE Trans Circ. Syst. I.

[8] W. Qian, J. Backes, and M.D. Riedel, "The Synthesis of Stochastic Circuits for Nanoscale Computation", Int. J. Nanotechnology Moleceular Computation vol. 1, no. 4, pp 39-57, 2010.

[9] P. Mroszczyk and P. Dudek, "The accuracy and scalability of continuous-time Bayesian inference in analogue CMOS circuits," in *ISCAS* 2014, pp. 1576–1579.

[10] M. Lin, I. Lebedev, and J. Wawrzynek, "High-throughput Bayesian computing machine with reconfigurable hardware," in *FPGA*, 2010.

[11] : American Academy of Sleep Medicine, *AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and*

*Technical Specifications*. Westchester, IL: American academy of Sleep Medicine, 2007.

[12] Sleep data and EEG signals contributed by the Sleep for Science Research Laboratory of Brown University and E.P. Bradley Hospital, courtesy of Prof. Mary A. Carskadon and Jared Saletin, Ph.D.

[13] V. Metsis, I. Androutsopoulos, G. Paliouras, "SPAM-filtering with naïve Bayes—Which naïve Bayes," CEAS 2006-Third Conference on Email and Anti-Spam, 2006.

[14] M. Shams, J.C. Ebergen, and M.I. Elmasry, "Modeling and comparing CMOS implementations of the C-element," IEEE Trans VLSI, vol. 6, no. 4 pp. 563-567, 1998.

[15] V Helland, A. Gapelyuk, A. Suhrbier, M. Riedl, T. Penzel, J. Kurths and N. Wessel, 'Investigation of an Automatic Sleep stage classification by means of multiscorer Hypnogram', Methods Inf Med., vol. 4, pp. 1-6, 2010

[16] J A. Urigüen and B. Gacia-Zapirain, "EEG artifact removal—state-of-the-art and guidelines", J. Neur Eng,vol. 12, no. 3, pp. 553-559, 2015.

[17] Pedregosa *et al* "Scikit-learn: Machine Learning in Python", JMLR 12, pp. 2825-2830, 2011.

[18] K. Parhi, and M. Ayinala, "Low-complexity Welch Power Spectral Density Computation", IEEE Trans on Circ. and Syst. I, vol 61, no 1, pp. 172-182, 2014.

[19] M. X. Cohen, *Analyzing Neural Time Series Data: Theory and Practice*, Cambridge: MIT Press, pp. 535-548, 2014.

[20] B. J. A. Palanca, G.A. Mashour and M. S. Avidan, "Processed electroencephalogram in depth of anesthesia monitoring", Current Opinion in Anaesthesiology vol 22, pp. 553-559, 2009.