archives-ouvertes.fr

Fault Tolerance of Self Organizing Maps

Cesar Torres-Huitzil, Oleksandr Popovych, Bernard Girau

To cite this version:

Cesar Torres-Huitzil, Oleksandr Popovych, Bernard Girau. Fault Tolerance of Self Organizing Maps . WSOM+17 - 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization, Jun 2017, Nancy, France. hal-01574212

HAL Id: hal-01574212
https://hal.inria.fr/hal-01574212

Submitted on 11 Aug 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fault Tolerance of Self Organizing Maps

Cesar Torres-Huitzil
Cinvestav Tamaulipas
Mexico
Email: ctorres@tamps.cinvestav.mx

Oleksandr Popovych
Igor Sikorsky KPI - Ukraine
Université de Lorraine / LORIA - France
Email: oleksandr.popovych@inria.fr

Bernard Girau
Université de Lorraine / LORIA
France
Email: bernard.girau@loria.fr

***Abstract*—As the quest for performance confronts resource constraints, major breakthroughs in computing efficiency are expected to benefit from unconventional approaches and new models of computation such as brain-inspired computing. Beyond energy, the growing number of defects in physical substrates is becoming another major constraint that affects the design of computing devices and systems. Neural computing principles remain elusive, yet they are considered as the source of a promising paradigm to achieve fault-tolerant computation. Since the quest for fault tolerance can be translated into scalable and reliable computing systems, hardware design itself and the potential use of faulty circuits have motivated further the investigation on neural networks, which are potentially capable of absorbing some degrees of vulnerability based on their natural properties. In this paper, the fault tolerance properties of Self Organizing Maps (SOMs) are investigated. To asses the intrinsic fault tolerance and considering a general fully parallel digital implementations of SOM, we use the bit-flip fault model to inject faults in registers holding SOM weights. The distortion measure is used to evaluate performance on synthetic datasets and under different fault ratios. Additionally, we evaluate three passive techniques intended to enhance fault tolerance of SOM during training/learning under different scenarios.**

I. Introduction

Artificial neural networks are computational models that have attracted intensive research interest and enjoyed significant renewed growth in artificial intelligence related applications. In neural networks research, one of the main problems is the architecture optimization, which aims at appropriately choosing the neural architecture and its parameters for high generalization/performance at solving a given task. However, the fact that performance maximization is of primary concern does not necessarily imply that it is the only goal that has been or should be pursued [1]. Artificial neural networks are generally assumed to acquire some other desirable features of biological systems such as their tolerance against imprecision, uncertainty and faults, which make them even harder to study or design. Fault tolerance is the attribute of a system that allows it to preserve its expected behavior after faults have manifested themselves within the system [2]. More precisely, a fault-tolerant system might be defined as one that has provisions to avoid failure after faults have caused errors within the system. When a statement about neural networks fault tolerance is made, it should be implicitly assumed a failure condition or criterion related to its intended computational task. That is, the threshold below which it cannot longer perform its function according to the specification. As such, fault tolerance in neural networks depends on the definition of the acceptable degree of performance and its intended application [3].

According to neurobiological studies, the human brain is able to tolerate a small amount of synapse or neuron faults or even use noise as a source of computation [4]. Nervous systems are complex, highly parallel information processing architectures made of seemingly imperfect and slow, but exceptionally adaptive and power-efficient components that carry out information processing functions [5]. Moreover, brains have the capability to relearn by growth of new neurons and/or neural connections and/or retraining of the existing neural architecture. Derived from these observations, it is commonly claimed that the majority of neural network models, abstracted from biological ones, have built-in or intrinsic fault tolerance properties due to their parallel and distributed structure, and the fact that they contain more neurons or processing elements than the necessary to solve a given problem, i.e., some natural redundancy due to overprovisioning. Claiming an equivalent fault tolerance only on the basis of rough architectural similarities therefore cannot hold true in general, especially for small size neural networks [6]. Studies have shown that an artificial neural network has a very limited fault tolerance capability and, as a matter of fact, most neural networks cannot be considered intrinsically fault tolerant, without a proper design. Obtaining truly fault tolerant neural networks is still a very attractive and important issue both for i) artificial intelligence based solutions, where, for instance, pervasive embedded systems will require smart objects fully merged with the environment in which they are deployed to cope with unforeseeable conditions [7], and ii) as a source for reliable computing systems built from unreliable components, as suggested in [8]. Rooted on neural computing, a new paradigm is needed to take advantage of new emerging devices at nanoscale dimensions and deal with both manufacturing defects and transient faults. This trend might lead to even consider faults/errors as an essential/intrinsic part of a system design. In this last direction, the robustness and the potential fault-tolerant properties of neural models call for attention as permanent and transient faults, device variation, thermal issues, and aging will force designers to abandon current assumptions that transistors, wires, and other circuit elements will function perfectly over the entire lifetime of a computing system, relying mainly on digital integrated circuits [9].

In the literature, several experimental and less analytic works have been carried out to study issues related to neural networks fault tolerance, including the analysis of the effect of noise on the output sensitivity [10], the weight error sensitivity [11], and the relationship among fault tolerance, generalization and model complexity [1], [6], [12]. Such works have been carried out at different levels of abstraction, from very specific low level physical implementations, but mostly, to the high level intrinsic fault masking capacity of neural paradigms. It is worth to point out that most works have been focused on feedforward neural networks and few attempts have been made to study and improve fault tolerance of other neural models, such as self-organizing maps (SOMs). Yet, we consider that the principle of self-organization itself might have an influence onto the fault tolerance of computing systems. Therefore in this paper, we are not only interested in the way pre-learned SOMs react to faults, but also in the way fault tolerance evolves during the self-organization of the neural population. This is why we chose to study SOMs because they stand as one of the most well-recognized models of self-organizing computing systems. In this effort, the aim of this work is twofold: i) to study the intrinsic fault tolerance capabilities of SOMs under bit-flip faults on weights by analyzing their performance degradation with variable fault ratios using synthetic datasets, and 2) to apply/adapt some reported techniques for enhancing fault tolerance during training/learning in neural networks to SOMs and compare their performance under different testing scenarios. We are interested in faults in weights, since when a SOM is intended to be implemented onto a digital hardware substrate, the value of each weight vector can be disturbed by various causes such as electromagnetic field or radiation, which can be modeled by a bit-flip fault model.

The paper is organized as follows. Section II presents some relevant related work on fault tolerance in SOMs. Section III presents the fundamentals of the Kohonen SOM model, and the on-line learning mechanism for self-organization. Section IV first describes the general approach for fault tolerance assessment in neural networks, and then the specificities for SOMs when targeted to a fully parallel digital hardware implementation are presented. Section V presents three techniques that have been used to improve the intrinsic fault tolerance of SOMs, which involves a modification of the learning/training mechanism. Section VI describes the datasets used for testing and the obtained experimental results for different scenarios.

II. Related work

Despite of their importance, the studies and results on fault tolerance reported in the literature are mostly concerned with feedforward neural networks. However, such results are difficult to generalize across different neural models. Among other neural models, the Kohonen SOM is a neural network which main property is to perform data quantization while preserving some topological relations among the set of learned prototypes. This model is a recognized tool for data visualization, and it has also been used for many practical applications such as pattern classification, image processing, and robotics [13]. The SOM model is inspired by the organizational structure of feature maps in the brain. In a SOM, each neuron iteratively learns to extract a feature vector from the input data. Simultaneously this learning maintains neighborhood relations (close neurons learn close prototypes). This learning, property of effectively creating spatially organized internal representations, is performed autonomously in the SOM and is considered as a self-organization process. Such networks self-organize, learn through associations, and scale well to very large systems due to their regularity and modularity, potentially leading to easy hardware implementation in existing technologies: either dedicated very large scale integrated (VLSI) chips or field programmable gate array (FPGA) structures.

However, the self-organizing behavior of SOMs in the presence of faults and its fault-tolerant capability have not yet been fully explored, in contrast to the more in-depth existing studies for feedforward networks. As some previous works have shown, SOMs are expected to tolerate defective neuron weights thanks to their self-organizing mechanisms. For instance, in [14] authors studied the fault tolerance capability of SOM in the presence of defective neurons undergoing stuck-at faults. In this study, it was shown that defective SOMs can eventually re-organize themselves, by relearning, if the defective neuron stuck-output is larger than a critical value. Only a linear array was analyzed and discussed, where defective neurons were concentrated in one place in the array, forming what they called a defective-neuron cluster. In the experiments, 100 neurons, including six defective ones, were tested to show that the SOM can learn in spite of some faulty neurons. In [15], authors addressed fault tolerance improvement in SOMs by using a technique called fault immunization of the synaptic connections. Stuck-at-a faults, a is a real value, were considered. Only one neuron was faulty at any time, but no restriction on the number of faulty links of the neuron was assumed. Weights are immunized by adding a constant value that is increased or decreased as much as possible without creating any misclassification. Fault immunization was formulated as an optimization problem on finding the corresponding constant value for each neuron. The application of this study was oriented to enhancing the reliability of the lossless image compression by a SOM. Other related works are related to modifications of the Kohonen SOM on-line learning algorithm so as to improve the fault tolerance of the learned SOM. Section V describes more precisely these works.

III. Self Organizing Maps

A. Fundamentals

Self-organizing maps (SOMs), as proposed by Kohonen [13], consist of neighboring neurons commonly organized on one- or two- dimensional arrays that project patterns of arbitrary dimensionality onto a lower dimensional array of neurons. All neurons receive the same input pattern and an iterative mechanism updates the neuron's weights so as to learn to quantize the input space in an unsupervised way. This mechanism first selects the neuron whose weights best fit the given input pattern, and then brings the weights of this neuron and of its neighbors closer to the current input pattern.

More precisely, each neuron in a SOM is represented by a d-dimensional weight vector, $\mathbf{m} \in \mathbb{R}^d$, also known as prototype vector, $\mathbf{m} = [m_1, ..., m_d]$, where d is the dimension of the input vectors, $\mathbf{x}$. Neurons are connected to adjacent ones by a neighborhood relationship, which define the structure of the map. The mechanism for selecting the winning neuron requires a centralized entity, so that the Kohonen SOM is not a fully distributed model as the cortex organization [16]. After learning, or self-organization, two vectors that are close in the input space will be represented by prototypes of the same or of neighboring neurons on the neural map. Thus, the learned prototypes become ordered by the structure of the map, since neighboring neurons have similar weight vectors.

In the literature, two main types of SOM can be distinguished, considering the number of neurons in the map with respect to the number of expected clusters: SOMs in which the number of neurons is roughly equal to the number of expected clusters to be found in the input, and SOMs with a large number of neurons (thousands or tens of thousands), which are used to allow the emergence of intrinsic structural features of the data space; these SOMs are referred to as Emergent Self-Organizing Maps (ESOMs) [17].

B. *On-line mode training*

The basic version of SOM learning is an on-line stochastic process which has been inspired by neurobiological learning paradigms. Several other extensions of the algorithm have been proposed [18]. In the SOM on-line mode algorithm, the learning phase starts with an appropriate (usually random) initialization of the weight vectors, $\mathbf{m}_i$. The input vectors are presented to the neural map in multiple iterations. For each iteration, i.e. for each input vector $\mathbf{x}$, the distance from $\mathbf{x}$ to all the weight vectors is calculated using some distance measure. The neuron whose weight vector gives the smallest distance to the input vector $\mathbf{x}$ is usually called the best matching unit (BMU), denoted by c, and determined according to:

$$\|\mathbf{x} - \mathbf{m}_c\| = \min_i \|\mathbf{x} - \mathbf{m}_i\| \tag{1}$$

where $\|\cdot\|$ is the distance measure, typically the Euclidean distance, $\mathbf{x}$ is the input vector and $\mathbf{m}_i$ is the weight vector of neuron i. The winner c and its neighboring neurons $i \in N_w$ update their weights according to the SOM rule:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)h_{ci}(t)[\mathbf{x}(t) - \mathbf{m}_i(t)] \tag{2}$$

where t denotes the time, $\mathbf{x}(t)$ is an input vector randomly drawn from the input data set at time t, $\alpha(t)$ the learning rate at time t, and $h_{ci}(t)$ is the neighborhood kernel around c. The learning rate $\alpha(t)$ defines the strength of the adaptation, which is application-dependent. Commonly $\alpha(t) < 1$ is constant or a decreasing scalar function of t.

The neighboring kernel $h_{ci}(t)$, which is a function of the distance between the winner neuron c and neuron i, can be computed using a Gaussian function:

$$h_{ci}(t) = \exp(-\frac{\|\mathbf{r}_c - \mathbf{r}_i\|^2}{2\sigma^2(t)}) \tag{3}$$

The term $\|\mathbf{r}_c - \mathbf{r}_i\|$ is the distance between neuron i and the winner neuron c. The precise value of $\sigma(t)$ does not really matter, as long as it is fairly large in the beginning of the process, for instance in the order of 20% of the longer side of the SOM array, after which it is gradually reduced to a small fraction of it, e.g. 5% of the shorter side of the array [13].

A batch version of SOM learning exists. It is also iterative, but instead of using a single input vector at a time, the whole dataset (batch) is presented to the map before updating any weight [19]. In each training step, the dataset is partitioned according to the Voronoi regions of the map weight vectors, i.e., each data vector belongs to the dataset of the closest map unit. This algorithm is deterministic, and one of its advantages is that the limit states of the prototypes depend only on the initial choices. However, in this paper, we are interested in the way fault tolerance evolves with SOM learning. The on-line mode training provides us with a more detailed evolution to observe. Moreover, we not only consider the standard Kohonen SOM learning algorithm, but also some variants among which several ones cannot be applied in batch mode. Therefore, we will only focus on on-line mode training.

IV. Fault tolerance in neural models

At a high level of abstraction, neural networks fault tolerance can be analyzed by the effects of errors in the main operators, rather independent from their intended physical implementation. In a more comprehensive approach [20], after this initial step, physical faults affecting a specific implementation can be mapped onto such errors to estimate fault tolerance of a given neural model, and by identifying critical components, complementary and ad-hoc fault tolerance policies can be further applied to enhance the properties of the neural model implementation. In order to study fault tolerance, researchers develop models of them to examine the variety of faults that need to be tolerated during the operation of a given system.

A. *Faults and fault models*

The following fault models have been widely used as abstractions of physical defect mechanisms in digital electronics devices/systems [21], and also applied to neural networks:

- **Stuck-at**, a data or control line appears to be held exclusively high (stuck-at-1) or low (stuck-at-0).
- **Random bit flips**, a data or memory element has some incorrect, but random value.

The stuck-at fault model is very popular since many defects at the transistor and interconnection structures can be modeled, as permanent faults, at the logic level with reasonable accuracy. This model is a binary model that does not capture the indeterminate states that faults may induce while occurring. It is applied to model faults on communication channels and arithmetic-logic operators. The random bit-flip model models transient faults that usually happen at registers or memory elements due to external perturbations, for instance, a single event upset. Under this model, damage/corruption is done only to the data and not to the physical circuit itself. Conceptually, it consists of a register bit that is switched randomly, resulting in a memory element that holds a wrong logic value.

B. *Fault injection*

A fault injection method is required for gaining insights of the behavior of a system [22]. Faults are probabilistically introduced into a neural model and the degree of failure, impact on the performed task, is evaluated according to some measures, which basically measure the performance distance (closeness) in performing a task by a fault-free neural network and the derived faulty networks. The measure of fault tolerance from many experiments is evaluated against the number of faults injected into the neural model. The limit of the network fault tolerance is problem-dependent and determined by operating scenarios of multiple faults leading to a violation of the performance constraints. Exhaustive testing of all possible single faults is prohibitive, not to mention that even multiple faults might occur concurrently. Hence, the strategy of randomly testing a small fraction of the total number of possible faults in a network has been adopted for tractability. It yields partial fault tolerance estimates that are statistically very close to those obtained by exhaustive testing.

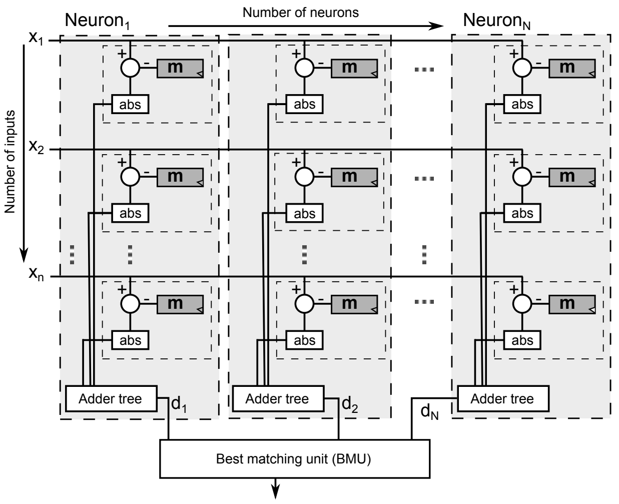


Fig. 1: Block diagram of an abstract digital hardware implementation for a two-dimensional SOM.

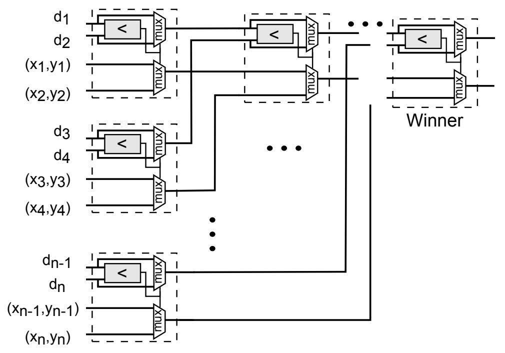


Fig. 2: BMU block diagram for a two-dimensional SOM.

C. SOM implementation choice

In this paper, we analyze SOM fault tolerance when a map is targeted to a fully parallel digital hardware implementation. We consider a parallel implementation for the SOM recall phase following a general architecture as shown in figure 1. A b-bit fixed-point representation for the internal computations and storage is used, as most SOMs hardware implementations use this number system so as to optimize hardware resources. A neuron essentially consists of registers (to store weights), adders and absolute computation modules that as a whole computes the Manhattan distance. The original SOM uses the Euclidean distance to measure the distance between input patterns and weight vectors, but it requires hardware-greedy multiplications and square roots, thus the Manhattan distance is a preferable choice so that even the distance between the inputs $\mathbf{x}$ and the vector prototypes $\mathbf{m}$ can be computed concurrently at every clock cycle. The BMU can be implemented as a comparator binary tree and a set of multiplexers, which receive the Manhattan distances and spatial coordinates from all neurons, as shown in figure 2. The BMU's outputs are the spatial coordinates (x, y) of the winner neuron. The neighborhood function plays a key role in SOM learning, but the direct implementation of the exponential function is not practical due to its complexity, instead look-up tables to store precomputed values can be used for resource saving. However, herein we only consider the on-chip implementation of the recall phase and an off-chip implementation for the learning phase.

D. SOM fault tolerance assessment

For SOM fault tolerance assessment, we consider that faults occur only in weight registers in the hardware architecture shown in figure 1. This choice considers plausible hardware faults and not only the unprecise notion of faulty neuron, yet all possible hardware faults are not considered this way (see IV-A). More precisely, we assume that each bit in defective neuron weights, which are stored in b-bit registers, is flipped with some probability and remains the same during the SOM processing. The fault rate u is defined as the percentage of flipped bits (b_{flips} such bits) in neuron weights with respect to the total number of bits in the SOM map. It is used so as to evaluate fault tolerance in SOMs. Considering a $w \times w$ SOM map, with N inputs and b-bit wordlength for weights, yields:

$$u = \frac{b_{flips}}{w^2 \times N \times b} \tag{4}$$

It must be pointed out that the number of faulty bits and the rate of faulty neurons increase very rapidly with the above-defined fault rate: to give an example, if $b = 16$, $N = 4$, and $u = 0.05$, 5120 faulty bits appear (out of 102 400) in a 40×40 SOM (more precisely, for this case, an ESOM), and 96 % of all neurons contain at least one faulty weight. For space consideration, we have focused on such fault rates as they represent a more devastating effect on the SOM behavior than single, double or triple faults, which can be well masked in SOM as it has been reported in related works [14][15], and confirmed by our experiments. Moreover, this fault model takes into account the fact that all the introduced faults are not equivalent: faults on least significant bits do not have the same effect as faults on most significant bits.

The severity of faults in the SOM when performing a task can be characterized with respect to the fault rate u: it consists in considering the artificially introduced faults at a given fault rate as the independent variable, and a selected performance measure of the SOM as the dependent variable. As SOM performance criteria for fault tolerance, the average quantization error (AQE) and distortion measure are used, which have been proposed as a way to trust the result of the SOM convergence on a given dataset [23][24][25]. The AQE measures the representation fidelity of the training data, and it is computed as the average distance between each data vector and the prototype of its BMU that should be very close:

$$AQE = \frac{1}{n} \sum_{1}^{n} \left\| \mathbf{x}_i - \mathbf{m}^{c(\mathbf{x}_i)} \right\|^2 \tag{5}$$

where n is the number of input vectors, $\mathbf{x}_i$ is an input, $\mathbf{m}^{c(\mathbf{x}_i)}$ is the weight of the BMU in the map for $\mathbf{x}_i$. The less the AQE, the better the fidelity of the representation of input data.

The distortion measure considers both vector quantization and structure preservation of the SOM [26] and it is suitable for comparing maps of the same size [24]. Let's consider a SOM represented by prototypes $\mathbf{m}^j$, the distortion measure is defined as the average over all input patterns (a set of n input data vectors $\mathbf{x}_i$) of the weighted sums of the squared distances between the pattern and all neuron prototypes, with weights that depend on the neighborhood function between each neuron and the winner neuron for the given pattern. According to [27], the original distortion measure would be biased towards the 2D SOM. If the input's distortion is redefined to be the average distortion over its BMU's neighborhood, the results will not be affected by the different topologies of the grid and the distortion measure becomes:

$$\xi = \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_{j=1}^{m} h_{c(\mathbf{x}_i)j} \left\| \mathbf{x}_i - \mathbf{m}^j \right\|^2}{\sum_{j=1}^{m} h_{c(\mathbf{x}_i)j}} \tag{6}$$

where $c(\mathbf{x}_i)$ is the BMU of $\mathbf{x}_i$, and $h_{c(\mathbf{x}_i)j}$ is the neighborhood function of neurons $c(\mathbf{x}_i)$ and j.

V. Enhancing fault tolerance in SOM

This section briefly introduces three techniques for SOM that have been proposed to improve its fault tolerance: weights restriction, fault insertion, and noise injection during training.

A. Restricting weights during training

The basic idea of this technique is that critical synapses have high magnitude weights [28][29]. Restricting the weight magnitudes to be low during training potentially makes the SOM more fault-tolerant. For instance, by i) penalizing high magnitude weights in the learning rule; $m_i^j(t+1) \in [m_{min}, m_{max}]$, the lower and upper bound of the weight vectors components, respectively, or ii) weight balancing to distribute the weight values more uniformly, e.g., to distribute the absolute weight values around the average absolute value. Here, a weight is updated only if its absolute value, computed by the learning rule, does not exceed the following threshold:

$$\theta(t) = \frac{1}{nw^2} \sum_{\mathbf{m}^j} \sum_{i=1}^{n} \left| m_i^j(t) \right|$$

B. Inserting faults during training

This technique randomly inserts faults during training [30] so as to improve fault tolerance. In each training iteration, a fixed number of weights (neurons) are randomly chosen, then faults (bit-flip) are injected according to these steps:

1) Randomly choose a predefined number of weights and insert faults in them
2) Apply the learning rule of on-line (batch) training for a pattern (all patterns) in the training set
3) Restore faulty weights to their unfaulty state and repeat the process until a stop criterion is reached

In our tests, only one faulty bit is introduced (and then restored) for each learning iteration.

C. Noise and weight perturbations

During training, in each iteration, a small number of inputs/weights are selected randomly to be perturbed by adding some type of noise [15][31]. The basic idea is to add noise to the inputs or weights of each neuron. Two main options exist, multiplicative and additive noise; additive noise being the most used. We consider such an additive noise in our tests.

VI. Experimental results

This section presents the experiments that have been carried out to assess SOM fault tolerance. We have focused on studying the performance of 2D maps on artificial datasets, for different bit-flips fault rates in weight registers. We have developed a software environment for fault tolerance evaluation, which handles computations using fixed-point representations (even during learning) as in the proposed hardware implementation. Experiments are limited here to 40×40 2D SOMs, i.e, ESOMs, whose weights are initialized to random values, evenly distributed between $[0, 1]$ and then scaled to a fixed-point representation Q6.10. As SOM performance depends on many other factors, to reduce the scope of the analysis, all SOMs use the same learning rate, update radius and Gaussian neighborhood function, during 150 epochs with 100 iterations each. Different map initializations are tested, but all learning techniques apply to the same initialized maps and use the same randomly chosen input pattern for each iteration.

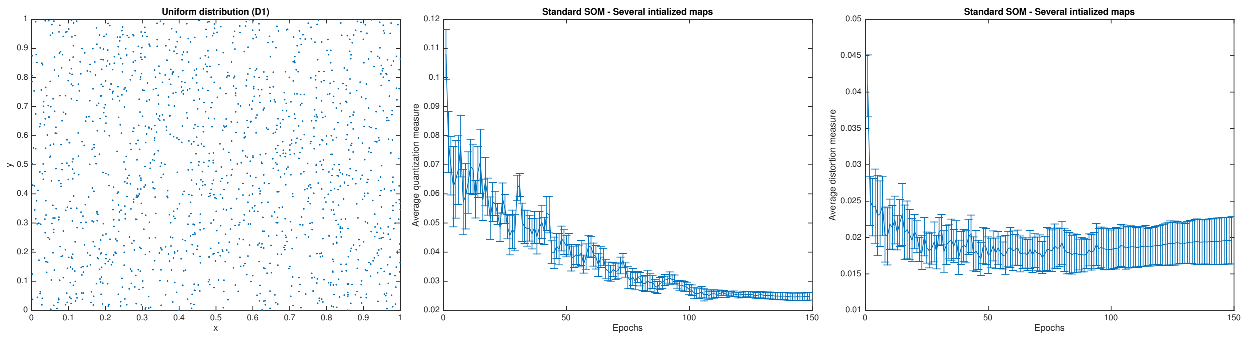
A. Datasets

Two simple datasets, called D1 and D2, with 2D points, were used in the experiments. D1 contains uniform observations in $[0, 1] \times [0, 1]$ and D2 observations drawn from a mixture of gaussians: three clusters with a Gaussian distribution, which have equal variance and some overlap. The standard deviation is 1 in each dimension. and the clusters centers are: (0.35, 0.35), (0.65, 0.35), and (0.5, 0.65). An example of the distributions in D1 and D2 are shown in figure 3 a), and b), respectively. It is important to point out that new points for D1 and D2 are generated at each learning iteration according to the chosen distribution, instead of pre-generating a fixed-size dataset. Thus, no cross validation technique is required.

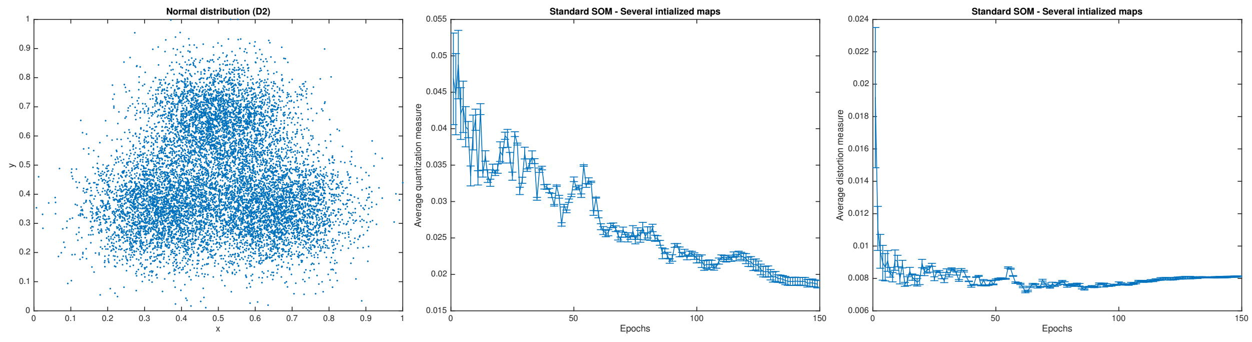
B. Results

The AQE and distortion measure profiles during learning are shown in figure 3. The presented results are averaged over several randomly initialized maps, but all learning techniques and all maps use the same learning patterns, randomly generated at each epoch. It explains why the different curves have similar irregular profiles. Nevertheless, experiments performed with other random patterns provide the same kind of results. Both measures decrease during training for all SOMs, but for D2 the convergence is faster and the measures show less variance for different random initializations. This can be explained by the usual problems known for Kohonen SOMs that unfold themselves in a D1-like input space: some maps spread well while some others remain partially twisted.

Figures 4(a) and 4(d) shows the performance metrics profiles for the SOM learning algorithms with noise injection and thresholding. SOM trained with thresholding yields larger AQE and distortion than the others, and a slower convergence rate. These results tend to show that this technique does not well extend to the kind of fault model we study here. This is even more noticeable for the fault injection based technique: its results are not even included, for readability, in the figure since this method performs worst among the implemented algorithms. This is directly linked to the number representation system that we use herein, where faulty bits can be injected in the 6-bit integer part and induce excessively large weights



(a) Uniform distribution of 2D samples (D1)



(b) Normal distribution of 2D samples (D2)

Fig. 3: Quantization error and distortion for SOM stochastic on-line learning for different initialized maps on D1 and D2.

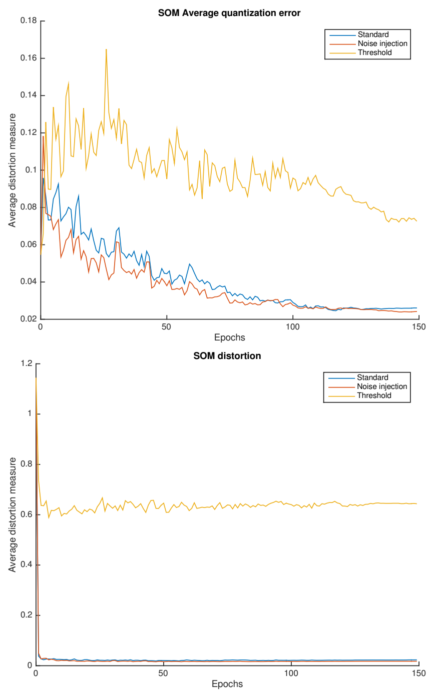
even after bit restoration (e.g. 000000 becomes 100000 with a faulty bit, then 011111 with learning, and recovering the initial value of the faulty bit is useless here). Using different integer part sizes induces more or less bad results for this method.

Figures 4(b) and 4(e) shows the AQE and distortion measures for D1 and D2 versus the bit-flip percentage introduced into SOMs. Recall that a $1\,\%$ fault rate already corresponds to 512 faulty bits in the kind of SOM we consider herein. As the same results are not expected for different faulty SOMs, even on the same dataset, for each fault rate many faulty versions of the trained SOMs were generated and averaged in the presented results. Distortion rapidly increases even with a small percentage of faults, and a criterion (threshold value) should be defined below which the SOM might be considered performing properly. The AQE also increases with the fault percentage, but in a more attenuated way for small percentages. The SOM trained with noise injection slightly outperforms the threshold method, the map is less affected when the number of inserted faults in weights increases. It can also be observed for dataset D1 that the AQE increases slowly at fault rates below 10%, suggesting that faults in weights can be masked in the SOM when it performs quantization, the global internal representation being distributed enough to cover the input space even with faults. However, when an error measure that considers the local behavior of the map is used, results clearly show that the structure preservation is affected even at small fault rate percentages, as the distortion measure indicates.

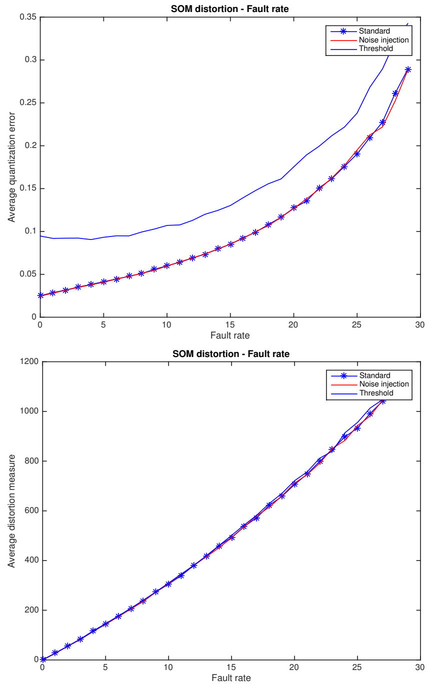
Nevertheless, all these results confirm that SOMs may stand as quite fault-tolerant models, since any abrupt degradation of their behavior with faults is observed. Figures 4(c) and 4(f) show in more detail how fault tolerance evolves through learning for different fault rates for stochastic online learning. The level of fault tolerance is estimated as the ratio between the average (over 100 faulty versions of each map) performance obtained for the given fault rate and the performance without faults. Thus the “optimal” or “reference” value is 1, and a ratio increase means a degradation of fault tolerance. After an immediate decrease of this ratio from the initialization to epoch 1, we see that it increases quite slowly for the AQE measure, and even stabilizes from epoch 100 at a value that depends on the fault rate for D1. For the distortion measure, there is an abrupt increase of the fault tolerance ratio up to very large values, after it stabilizes or even slightly decreases after 50 epochs. Such increase is a direct consequence of the introduction of faulty bits in the integer parts of the weights, as it was the case to explain the bad performance of the fault injection learning technique. The order of magnitude of the ratio goes up to $\frac{10^2}{10^{-2}}$ with such faults, since the distortion measure takes into account the weights of all neurons, whereas the AQE measure only considers the winner neuron. Again, it is tightly linked to the size of the integer part, e.g. a Q4.12 instead of Q6.10 fixed point representation would induce approximately 16 times lower values for the maximum distortions. This negative effect on distortion is observed even when introducing only a few faulty bits in our other experiments, whereas the quantization-based fault tolerance remains optimal for such scarce faults.

To summarize, these results tend to show that:

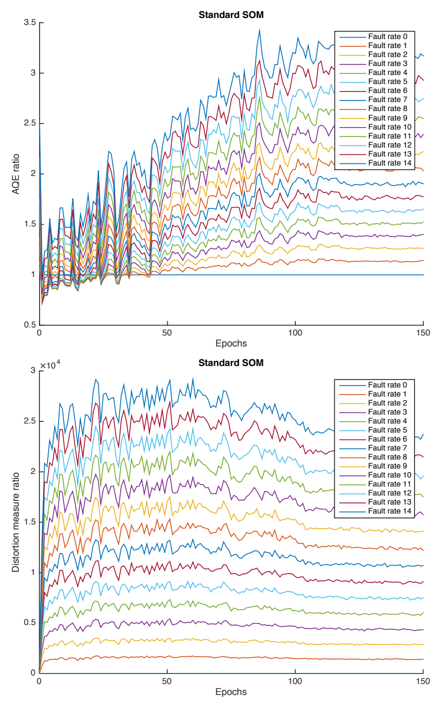
- Pre-trained SOMs are quite tolerant to faults for vector quantization since their performance gradually degrades with the fault rate. But the specific properties of structure preservation degrade more drastically with faulty bits.



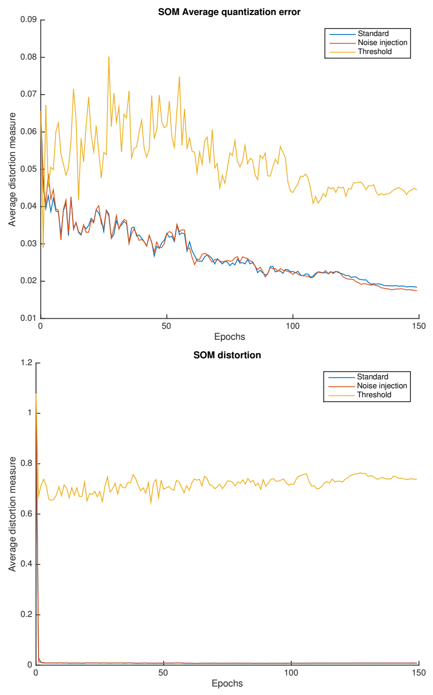
(a) AQE and DM profile during learning (D1)



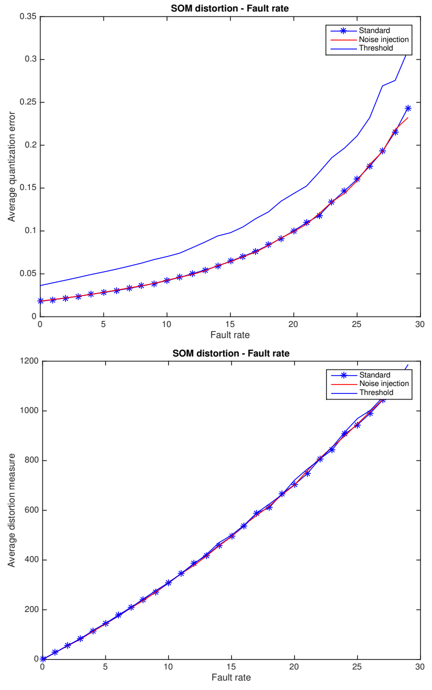
(b) AQE and DM vs fault percentage (D1)



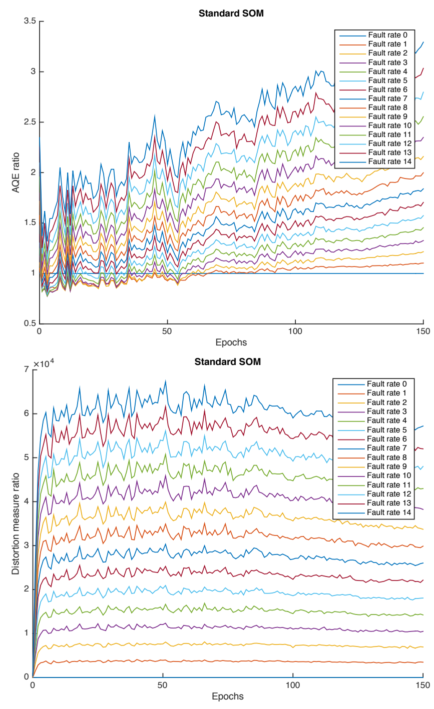
(c) Fault tolerance along learning (D1)



(d) AQE and DM profile during learning (D2)



(e) AQE and DM vs fault percentage (D2)



(f) Fault tolerance along learning (D2)

Fig. 4: Experimental results for SOM fault tolerance assessment. Figures 4(a) and 4(d) shows the AQE and distortion profiles for SOMs with noise injection and thresholding during learning. Figures 4(b) and 4(e) shows the AQE and distortion measures for D1 and D2 versus the percentage of bit-flips introduced into SOMs. Figures 4(c) and 4(f) show in more detail how fault tolerance evolves through learning for different fault rates when stochastic online learning is used.

- Among the different learning modifications that have been proposed to increase SOM fault tolerance, only the noise injection technique appears to be robust to the kind of fault model considered here (figures 4(b) and 4(e)).

- When observing the evolution of fault tolerance during learning, self-organization induces an improvement with respect to the AQE criterion only during the first learning iterations (initial unfolding of the map). If the distortion criterion is used, even the initial self-organization phenomenon is not able to induce more fault tolerance (figures 4(c) and 4(f)).

VII. Conclusion

This paper presents a study of fault tolerance of SOMs. We compare the performances of SOMs obtained with three variants of the on-line training algorithm: weight restriction, fault injection and noise injection. We also study how fault tolerance evolves during learning. The obtained results show that training by inserting noise injection provides slightly better results than the other techniques but it is far from producing highly fault tolerant SOMs. Nevertheless, SOM fault tolerance is globally confirmed since the quantization quality degrades gradually with increasing fault rates. SOM fault tolerance with respect to structure preservation has been studied using the distortion measure, and results from this initial study show that this particular property of SOMs is not necessarily capable of supporting a large number of faults. To address this limit, we consider that a more collective behavior of neurons is required: instead of considering the weights of each neuron separately, a more fault tolerant behavior is expected from a population-coded approach of the prototypes, where single highly faulty weights would not directly affect the global performance. To this end, we currently study how a soft-winner-takes-all approach by means of gaussian filtering of prototypes may increase the fault tolerance of SOMs. Then in order to decentralize the process and reduce its dependence to new possibly faulty registers, we intend to replace gaussian filtering by the emergence of bubbles of activity in dynamic neural fields coupled to the SOMs.

Acknowledgment

The authors would like to thank the partial support received through the Conacyt grant program to conduct research abroad. This work is also partially supported by Erasmus+ 2016-2017 and by the French embassy of Ukraine.

References

[1] C. Alippi, "Selecting accurate, robust, and minimal feedforward neural networks," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, no. 12, pp. 1799–1810, Dec 2002.

[2] A. Avižienis, "Framework for a taxonomy of fault-tolerance attributes in computer systems," *SIGARCH Comput. Archit. News*, vol. 11, no. 3, pp. 16–21, Jun. 1983.

[3] P. W. Protzel, D. L. Palumbo, and M. K. Arras, "Performance and fault-tolerance of neural networks for optimization," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 600–614, Jul 1993.

[4] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 860–880, May 2014.

[5] T. Sejnowksi and T. Delbruck, "The language of the brain," *Scientific American*, vol. 307, pp. 54–59, Oct 2012.

[6] S. S. Venkatesh, "Robustness in neural computation: random graphs and sparsity," *IEEE Transactions on Information Theory*, vol. 38, no. 3, pp. 1114–1119, May 1992.

[7] Z. Wang, K. H. Lee, and N. Verma, "Overcoming computational errors in sensing platforms through embedded machine-learning kernels," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 8, pp. 1459–1470, Aug 2015.

[8] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata Studies*, pp. 43–98, 1956.

[9] A. Rahimi, L. Benini, and R. K. Gupta, "Variability mitigation in nanometer cmos integrated systems: A survey of techniques from circuits to software," *Proceedings of the IEEE*, vol. 104, no. 7, pp. 1410–1448, July 2016.

[10] N. C. Hammadi and H. Ito, "Improving the performance of feedforward neural networks by noise injection into hidden neurons," *Journal of Intelligent and Robotic Systems*, vol. 21, no. 2, pp. 103–115, 1998.

[11] X. Zeng and D. S. Yeung, "Sensitivity analysis of multilayer perceptron to input and weight perturbations," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1358–1366, Nov 2001.

[12] E. B. Tchernev, R. G. Mulvaney, and D. S. Phatak, "Investigating the fault tolerance of neural networks," *Neural Computation*, vol. 17, no. 7, pp. 1646–1664, 2016/06/01 2005.

[13] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep 1990.

[14] M. Yasunaga, I. Hachiya, K. Moki, and J. H. Kim, "Fault-tolerant self-organizing map implemented by wafer-scale integration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 2, pp. 257–265, June 1998.

[15] R. Talumassawatdi and C. Lursinsap, "Fault immunization concept for self-organizing mapping neural networks," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 09, no. 06, pp. 781–790, 2001.

[16] N. Rougier and Y. Boniface, "Dynamic self-organising map," *Neurocomputing*, vol. 74, no. 11, pp. 1840 – 1847, 2011.

[17] A. Ultsch, "Data mining and knowledge discovery with emergent self-organizing feature maps for multivariate time series," in *Kohonen Maps*, S. K. E. Oja, Ed., June 24–27 1999, pp. 33–46.

[18] M. Cottrell, M. Olteanu, F. Rossi, and N. Villa-Vialaneix, *Theoretical and Applied Aspects of the Self-Organizing Maps*, 2016, pp. 3–26.

[19] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 13, pp. 1 – 6, 1998.

[20] V. Piuri, "Analysis of fault tolerance in artificial neural networks," *Journal of Parallel and Distributed Computing*, vol. 61, no. 1, pp. 18 – 48, 2001.

[21] J. A. Abraham and W. K. Fuchs, "Fault and error models for vlsi," *Proceedings of the IEEE*, vol. 74, no. 5, pp. 639–654, May 1986.

[22] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, "Fault injection techniques and tools," *Computer*, vol. 30, no. 4, pp. 75–82, Apr 1997.

[23] E. de Bodt, M. Cottrell, and M. Verleysen, "Statistical tools to assess the reliability of self-organizing maps," *Neural Networks*, vol. 15, no. 89, pp. 967 – 978, 2002.

[24] G. Pölzlbauer, "Survey and comparison of quality measures for self-organizing maps," in *Proceedings of the Fifth Workshop on Data Analysis (WDA'04)*, June 24–27 2004, pp. 67–82.

[25] D. Beaton, I. Valova, and D. MacLean, "Cqoco: A measure for comparative quality of coverage and organization for self-organizing maps," *Neurocomputing*, vol. 73, no. 1012, pp. 2147 – 2159, 2010.

[26] J. Rynkiewicz, "Self-organizing map algorithm and distortion measure," *Neural Networks*, vol. 19, no. 67, pp. 830 – 837, 2006.

[27] Y. Wu and M. Takatsuka, "Spherical self-organizing map using efficient indexed geodesic data structure," *Neural Networks*, vol. 19, no. 67, pp. 900 – 910, 2006, advances in Self Organising Maps - WSOM'05.

[28] C.-T. Chin, K. Mehrotra, C. K. Mohan, and S. Rankat, "Training techniques to obtain fault-tolerant neural networks," in *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*, June 1994, pp. 360–369.

[29] S. Cavalieri and O. Mirabella, "A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks," *Neural Networks*, vol. 12, no. 1, pp. 91 – 106, 1999.

[30] C. H. Sequin and R. D. Clay, "Fault tolerance in artificial neural networks," in *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, June 1990, pp. 703–708 vol.1.

[31] H. Allende, S. Moreno, C. Rogel, and R. Salas, *Robust Self-organizing Maps*. Springer Berlin Heidelberg, 2004, pp. 179–186.