NP-SOM: network programmable self-organizing maps

Yann Bernard, Emeline Buoy, Adrien Fois, Bernard Girau

NP-SOM: network programmable self-organizing maps

Yann Bernard, Emeline Buoy, Adrien Fois, Bernard Girau
Université de Lorraine, CNRS, LORIA, F-54000 Nancy, France
Contact emails: Yann.Bernard@inria.fr - Bernard.Girau@loria.fr

Abstract Self-organizing maps (SOM) are a well-known and biologically plausible model of input-driven self-organization that has shown to be effective in a wide range of applications. We want to use SOMs to control the processing cores of a massively parallel digital reconfigurable hardware, taking into account the communication constraints of its underlying network-on-chip (NoC) thanks to bio-inspired principles of structural plasticity. Although the SOM accounts for synaptic plasticity, it doesn't address structural plasticity. Therefore we have developed a model, namely the NP-SOM (network programmable self-organizing map), able to define SOMs with different underlying topologies as the result of a specific configuration of the associated NoC. To gain insights on a future introduction of advanced structural plasticity rules that will induce dynamic topological modifications, we investigate and quantify the effects of different hardware-compatible topologies on the SOM performance. To perform our tests we consider a lossy image compression as an illustrative application.

Introduction

The structural organization of the brain is already a source of inspiration of several neuromorphic integrated circuits, but several dynamic properties of the brain computation have not really been explored as design principles for computational architectures. We are interested in the introduction of brain-inspired self-organization principles into the design of adaptive and dynamic digital computing architectures. Self-organizing neural models have already been studied, especially for vector quantization tasks. In project Saturn [1] it has been studied how neural self-organizing maps (SOMs) may control the development of computing areas in a many-core substrate, thus applying principles of synaptic plasticity to a hardware configuration. In [1], different input streams are analysed so as to quantify the complexity of the information they carry, then these measurements are fed into a SOM in which each neuron is associated to a core of the computing substrate: the codeword that is learned for a neuron determines which input stream must be processed by the associated core (e.g. visual input). But computing resources need to exchange data while they are dynamically allocated to different tasks, therefore communications between resources also need to re-organize themselves with the same constraints of decentralization. Existing self-organizing models are limited in this context, because they rely on a predefined topology and they do not take into account the communication costs nor the afferent connectivity. Therefore, we want to extend the usual self-organization mechanisms to account for a combination of bio-inspired principles of synaptic and structural plasticities. A few self-organizing models already consider structural plasticity, such as growing neural gas (GNG), but their induced connections are only data-dependent, so that they are usually hardware incompatible. We are currently developing variants of SOMs that include local pruning and sprouting rules while preserving underlying topologies that fit hardware communication constraints [2]. But before introducing advanced rules for structural plasticity that induce changes to the network topology, an important first step is to study how different hardware-compatible topologies affect the behaviour and the performance of SOMs. This is precisely the aim of this paper, in which we analyse how different pre-designed topologies impact the SOM operation, in order to gain insights into the potential effects of an upcoming introduction of structural plasticity rules. For this analysis, we consider an application of vector quantization in which the SOM properties are known to improve the results [3, p.743] : lossy image compression. On the hardware side, we use basic principles on the operation of locally configurable network-on-chips (NoCs) to define the hardware constraints in our model. We consider three different configurations of a NoC that induce three SOM topologies. We compare the performance of the different SOMs in terms of pixel error, peak signal-to-noise ratio, and we compare how the final differential and entropy coding steps perform with the different SOMs. These comparisons help us to analyse the precise effects of the topological modifications, and to interpret them in terms of future applications for reconfigurable reconfiguration. In Section 1, we propose a short survey of vector quantization and its application to the concept of a self-organizing machine architecture. In Section 2, the basic Kohonen model for self-organizing maps is sketched out, and its application to lossy image compression is precisely described. Section 3 introduces our network-programmable version of SOMs (NP-SOM), first defining the kind of NoC we take into account. Section 4 gathers our experimental results, from which we derive several conclusions and discussions in Section 5.

1 Background

1.1 Vector quantization

We are interested in this work by brain-inspired input-driven self-organization. More specifically, we want a network to be able to represent an input distribution (e.g. visual input) by learning it in an unsupervised way. The learning of the input distribution is usually conducted in the network by the modification of the synaptic weights and/or the creation or suppression of connections and/or neurons. The aim of this

self-organized learning procedure is to start from an disorganized state (e.g. with randomly initialized synaptic weights) and converge toward a more ordered state. This increase in order in the network is characterized by the gradual formation of spatially segregated clusters of neurons that are coding - and hence are selectively tuned to - certain features of the input distribution. In this regard vector quantization (VQ) techniques seem to be tightly related to the kind of input-driven self-organization we expect. VQ is a lossy source coding technique in which block of samples are quantized together [4]. It consists in approximating the probability density of the input space (which is split in blocks of samples) with a finite set of prototype vectors. A prototype vector is often referred to as a codeword and the set of codewords as the codebook.

Many VQ techniques exist such as k-means [5], self-organizing maps (SOM) [6], neural gas (NG) [7], growing neural gas (GNG) [8] or growing when required (GWR) [9]. Algorithms such as NG, GNG or GWR present good performance in terms of quantization error minimization which is measured by the mean squared error. However this performance may come during the learning phase with the creation in the network of a very large number of new prototypes and/or connections between prototypes. Additionally, the induced modifications are only data-dependent, without any topological restrictions. This is problematic as the connectivity and the computational resources are limited in reconfigurable hardware, therefore making those algorithms hardly able to reflect hardware communication constraints.

On the other hand the SOM is based on a static underlying topology and a fixed number of prototypes, making it more suited to the concept of a self-organizing hardware substrate. Moreover in a SOM and in contrast to standard VQ techniques, each prototype has an associated position in a map that has a predefined topology (the map is usually a 2D lattice). This spatial arrangement of the prototypes in the map makes it able to capture the topographic relationships (i.e. the similarities) between the inputs, in such a way that similar blocks of samples tend to be represented by spatially close prototypes in the map. This clustering results from the projection of the input space onto a lower-dimensional space (the map), which embodies pre-designed neighborhood relationships and competitive learning through a winner-takes-all mechanism. This property can be measured by the distortion that is computed by the Gaussian averages of distances between prototypes and input samples around the sample-related winners. As a matter of fact, the SOM is a well-known and biologically plausible model of the topographical mapping of the visual sensors to the cortex [3]. Besides its biological plausibility, the SOM has shown to be effective in a wide range of applications such as image compression and segmentation, data visualization or text mining [3, 10]. However, despite the interesting features that the SOM offers, the static underlying topology remains a limiting factor in the context of a self-organizing hardware project. Indeed, it corresponds to a manycore substrate where only neighbouring nodes can communicate. To extend it to long-range and adaptive communications, we intend to use more flexible network-on-chip solutions (see 3.1) and we therefore need to develop a SOM variant that takes into account this additional topological flexibility.

1.2 The SOMA project

The work presented in this paper is part of a broader project that follows a previous one [1], namely the SOMA project (*Self-Organizing Machine Architecture* [11]). The SOMA project aims at developing an architecture based on brain-inspired self-organization principles in a digital reconfigurable hardware, focusing on how structural plasticity principles can inspire new hardware mechanisms. As we wish to deploy the architecture in a manycore substrate, the scalability of the architecture needs to be ensured. For this reason, the communication system between the cores shall be carefully chosen to avoid the Von Neumann bottleneck. Consequently, rather than using a shared bus, a network-on-chip (NoC) ensures a decentralized communication between the cores in the SOMA architecture. Following [1], a SOM controls the configuration of computing nodes by analysing the complexity of the different input streams and deriving clusters of neurons that are directly related to computing areas in the manycore substrate. However, any configuration of the cores induces communication costs in terms of interconnects utilization in the underlying NoC. The idea of SOMA is to simultaneously learn the task allocation of nodes and the structural connectivity that the NoC must implement, so that the NoC constraints are taken into account while defining the computing areas. Thus, the principle of a free topology as in the Growing Neural Gases (GNG) is interesting to dynamically create topologies that better fit the input distribution, but it results in connections between neurons that induce too costly NoC-based communications between the associated nodes. Therefore, even if both SOM and GNG have interesting properties, they do not take into account the hardware communication restrictions (GNG) nor NoC configurability (SOM). This is why we propose here the model of a reconfigurable SOM. Such a model is able to dynamically change its underlying topology and adapt its codebook learning to these changes. In the context of this paper, we do not yet take into account such dynamic changes. We first want to estimate the impact of different hardware compatible topologies onto the behaviour of the SOM. We thus consider NoC-based topologies that differ from the basic 2D lattice of Kohonen SOMs, and we test them for a typical application for which various and complementary performance criteria are available.

2 Kohonen SOM

2.1 The model

The map

The neurons of the Kohonen SOM [10] are spatially arranged in a discrete map which usually consists in a two-dimensional grid. It shall be reminded that we will consider this topology as well as other hardware-compatible topologies in Section 3. Each neuron n is fully connected to all the source nodes and has a weight vector w_n, or codeword, whose dimension is equal to a source node.

The algorithm

First, all codewords are initialized with random weights. The training of the SOM lasts a certain number of epochs. One epoch includes enough training iterations such that the whole training dataset has been used once for learning. For

each training iteration, a training vector v is picked among the inputs. The best matching unit (BMU) g is then found, it corresponds to the neuron with the minimal L^2 distance between w_g and v. Then the weights of all neurons are updated according to the following equation:

$$w_i(t+1) = w_i(t) + \epsilon(t) \cdot \Theta(\sigma(t), d_{i,g}) \cdot (v - w_i(t)) \quad (1)$$

with ϵ and σ time functions (more details on them later), $d_{i,g}$ is the normalized L^1 distance between neuron i and the BMU. Θ is a normalized centred Gaussian function with standard deviation σ.

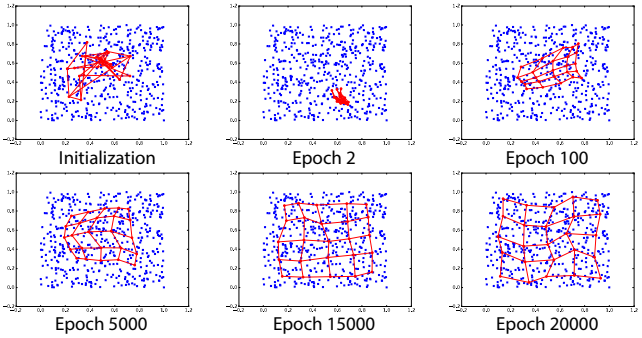

Figure 1: Example of a Kohonen SOM learning a sample of random uniformly distributed data in $[0, 1]^2$

Figure 1 illustrates how a SOM unfolds in the input space (here simply 2D): the codewords that are learned are shown in red in the input space (from which random inputs are drawn, see blue points), red links being virtually added between the codewords of neighbouring neurons in the map.

Parameters

$\sigma(t)$ is a parameter that influences the neighbourhood function. The higher it is, the more the BMU influences other neurons. We have set it to start at 0.5 and linearly decrease to a final value of 0.001, so that at the beginning of the training all neurons are significantly influenced by the BMU, and at the end, nearly none except the BMU are. $\epsilon(t)$ is the learning parameter, it starts at 0.6 and linearly decreases to a final value of 0.05. In our tests, we ran the SOM for 50 epochs.

2.2 Application: lossy image compression

We consider a well-known application of vector quantization: lossy image compression [12]. A picture or series of pictures to be compressed is split into smaller $k \times k$ pixels wide thumbnails. When the image height or width is not divisible by k, we crop it on the right and bottom. We then use these thumbnails as training samples of a VQ model. Once the training is finished, the compressed image is composed of the whole codebook, and the index of the BMU for each thumbnail extracted from the image or images. A final lossless entropy coding such as Huffman coding is used to further compress the file. Decompression is performed by reverting the final lossless compression, and recomposing the original image from the stored codebook and the indexes of the BMUs: each index is replaced by the corresponding codeword thumbnail. The result is similar to the original image, but with every sub-image replaced by the codeword learned by its BMU. Figure 2 illustrates this compression/decompression process, without the entropy coding step.

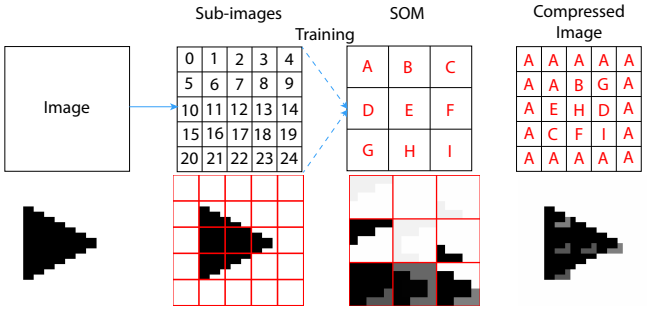

Figure 2: Simplified scheme of the image compression/decompression process (with only 25 sub-images and 9 neurons) with a simple test example underneath.

In the case of a SOM used as VQ model, an additional step can be introduced just before entropy coding to further improve its efficiency. A differential coding can be applied to the stored BMU indexes, as in [13]. Each index is replaced by the difference between this index and the BMU index of the immediately neighbouring thumbnail in the direction which maximizes the image smoothness (see [13] for details). This process results in rather small differences thanks to SOM properties, and the final entropy coding performs better on such small values.

Compression rate

We consider here the compression of a single greyscale image. A raw compression rate can be calculated before the final differential and entropy coding steps. It depends on the number of neurons $N \times N$ and the size of the side of each square sub-image. This raw compression rate (R_c) is the ratio between the initial image size and the size of the file that contains the codebook and the binary coded BMU indexes. It is expressed as below:

$$R_c = \frac{S_i}{S_f} = \frac{8hw}{8N^2s^2 + 2\frac{hw}{s^2}\log_2(N)} \quad (2)$$

S_i and S_f represent respectively the size of the original image and of the compressed image, h and w are respectively the height and width of the image, $N \times N$ is the number of neurons and s is the side length of a sub-image.

As seen in Figure 3, for each number of neurons, R_c first increases with s, then it decreases, which reflects the influence of the two terms in the denominator of equation 2. Furthermore, the lesser neurons there are, the better the compression is. However, a compromise between compression rate and quality of the image has to be found.

The final compression rate depends on the efficiency of the entropy coding, which itself depends on the distribution of the values computed during the differential coding step.

2.3 Quality assessment of the compression

There are many possible metrics to quantify the differences between two images. But none of them can really give a definitive judgement on the visual quality of the compression. This is because evaluating quality is a multi-criteria problem

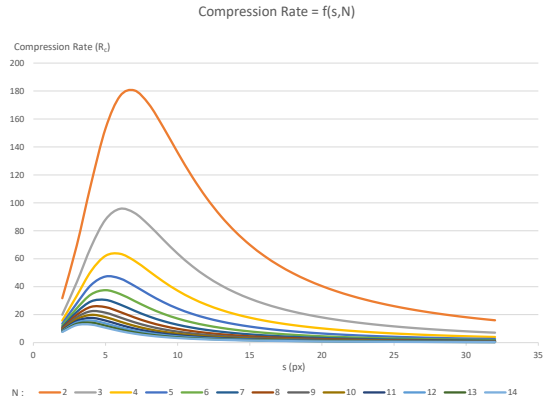

Figure 3: Compression rate function of the side s of sub-images and the number $N \times N$ of neurons.

dependent on the observer (one observer may prefer colour correctness over finely detailed edges and another one the inverse), and the fact that knowing the impact each pixel has in an observer's evaluation of quality is a non-trivial problem. We will present some of the metrics we used and the reasons behind each of these, but keep in mind that the eyeball is currently our best tool.

Mean pixel difference

The most obvious metric is to compute the mean error per pixel. It is obtained with the following equation:

$$mean = \frac{1}{w \times h} \sum_{x=1}^{w} \sum_{y=1}^{h} |P_{x,y} - P'_{x,y}| \quad (3)$$

with $P_{x,y}$ the pixel at the x^{th} column and y^{th} of the original image, and $P'_{x,y}$ its equivalent in the compressed image.

There are many problems with this metric for image comparison, because it puts the same weight on all pixels (some background pixels may have the same impact on the result as a high information foreground pixel). Moreover, it doesn't guarantee that the error is well distributed: 100 pixels that are off by 1 will have the same error as 5 pixels off by 20, although the visual result would be quite different. But it is also a generalizable measurement that would work equivalently with any other data, and it is the measure that our neurons try to minimise locally, so it is the best one to evaluate if the learning task of the SOM is correctly performed.

Peak Signal to Noise Ratio

A common metric used in image compression is the Peak Signal to Noise Ratio (PSNR). The goal is to measure the noise added to the signal (in our case the image) during compression. It is calculated from the Mean Square Error (MSE) with the following formulas, and expressed in decibels (dB):

$$MSE = \frac{1}{w \times h} \sum_{x=1}^{w} \sum_{y=1}^{h} (P_{x,y} - P'_{x,y})^2 \quad (4)$$

$$PSNR = 10 \times \log_{10}(\frac{255^2}{MSE}) \quad (5)$$

PSNR has been widely used in image compression because it is a loose approximation to the human perception of quality, and as such it is a rather satisfactory metric for what we want to measure. It still has some of the defaults of the mean pixels difference (all pixels have the same weight regardless of usefulness in the recognition), but now error spikes are more penalizing for the quality score than small diluted errors.

2.4 Image database used

Because of long training times we chose to limit ourselves to small greyscale images of 256*256 pixels. They can be found here [14].

3 The model of a topologically restricted SOM

3.1 Locally configurable Network-on-Chip

In the SOMA architecture, the proposed self-organizing mechanisms will be exploited by user-defined applications running on a multicore array in the form of a NoC-based manycore system. The NoC will provide different features such that computation nodes may:

- Communicate between neighbouring and remote computing nodes in the manycore array. Constraints apply to simultaneous communications.
- Communicate to the underlying self-organizing layer in order to influence the self-organization.
- Get inputs and dynamic configurations from the self-organization layer.

Different NoC technologies are available. A popular approach is packet switching, where channels of communications are created along the packet transfer path, using the information in the header to locally determine the next node of the path. Channels are occupied during the transmission of data, and then released. This approach is well adapted to applications where nodes communicate irregularly. The idea behind the SOMA architecture is to take into account hardware communications channels that evolve like some kind of dendritic tree. For example if the nodes perform neural computations, connected nodes will continuously exchange data. A possibility is to use a NoC architecture able to combine as many channels as possible at a time, e.g. using the HERMES routing scheme [15] for inter-node communication, for which an adapted version of the NoC architecture to support dynamic reconfiguration has been proposed [16]. Such a NoC architecture is able to build multiple and steady communication channels simultaneously. Another option is to use a locally configurable NoC dedicated to neural architectures, such as in the FPNA concept [17]. Despite their differences (such as Hermes' ability to handle type-based routing), these NoC architectures eventually reduce to a common principle: configuring local connections in an oriented graph where each node behaves as a local network switch between incoming arcs (input gates) and outgoing arcs (output gates). To ensure a simple hardware compatibility, we consider in this paper that this graph is based on a simple grid, see 3.

3.2 Link between SOM topology and Networks-on-Chip

In the self-organizing hardware architecture targeted by the SOMA project, each computing core is virtually associated to a neuron of the SOM that controls task allocation. Since the

learning process of this SOM must reflect the communication constraints among the cores according to their allocated task, we propose that this SOM learns its codebook and its underlying topology together, while constraining this topology to be NoC compatible. After learning, the computing areas will be associated to groups of neurons that have similar codewords, and the fundamental properties of SOMs will result in the fact that such neurons will be close with respect to the NoC compatible underlying topology. We consider that the best way to ensure this NoC compatibility is to directly define the SOM topology as the result of a specific configuration of the NoC. Our goal is then to define learning mechanisms at the level of the local configurations. Before defining such local learning rules on a bio-inspired basis, we want to study in this paper the effect of various NoC-based SOM topologies on VQ properties. To that purpose, we propose below the network programmable SOM model (NP-SOM), first without any mechanism to learn the topology, but evaluating various static NoC-based topologies.

A first obvious configuration corresponds to the fact that each routing node only sends the received data to the local core/neuron. Since the underlying graph of the NoC is a grid, this results in the usual grid topology of SOMs. Many other configurations are possible. Among them, we focus on configurations that somehow would minimize the resulting communication times for distant cores. Two such configurations are chosen, they will be depicted in the next section.

3.3 The NP-SOM model

Simple Model

Let us consider a couple $(\mathcal{N}, G)$, where $\mathcal{N}$ is a set of neurons and G is a set of gates. The neurons are positioned as in a grid. Each neuron is surrounded by 8 gates, 2 in the North, East, West and South directions of the neuron. The two gates that can be found in each direction around a neuron stand for the input and output gates of this neuron from and towards this direction. The set composed by a neuron and its gates is called a node.

Within each node, a local connection matrix defines which neuron or input gate is connected to which output gate or neuron. This matrix is filled with 0's and 1's: 0 if there is no connection and 1 if there is a connection between some input and output gates, or between an input gate and the local neuron n or between the local neuron n and an output gate.

The distance

As seen previously, neurons of a SOM need a distance metric in order to know the degree of closeness they have with the BMU during learning. In the perspective of associating each communication channel created in the NoC as a direct privileged link between computing nodes, we consider that any neuron that can be accessed by routing from a source neuron without having to go through another neuron (i.e. the only possible routing is through gate-to-gate connections except for the source and receiver nodes) is at a distance 1 from the source. In the general case, when there is k neurons between the source and the receiver, with k being minimal, the distance is equal to $k + 1$.

For example, as it can be seen in the bottom of Figure 4, the distance between neurons A and B is 6.

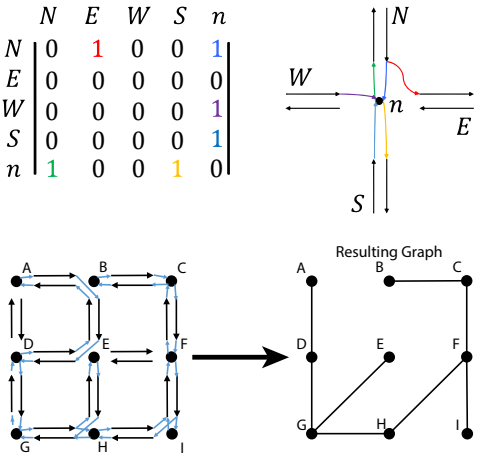

Figure 4: The top row illustrates an example of a neuron n local connection matrix. The bottom row represents a 3x3 NP-SOM and its resulting distance graph used for our SOM.

Three derived topologies

In the context of this paper, two topologies have been tested in addition to the grid topology often used for Kohonen SOMs. Figure 5 depicts these underlying topologies (or graph structure of the SOMs) on the right, while showing how to derive them from local interconnections on the left. From now on, we call `kohonen` a NP-SOM that uses a static grid as underlying topology, `star` a NP-SOM that uses the depicted scale-free star-like configuration as underlying topology, and `sw` a NP-SOM that uses the depicted small world-like configuration as underlying topology. Of course, `kohonen` behaves exactly as a standard Kohonen SOM using a grid structure.

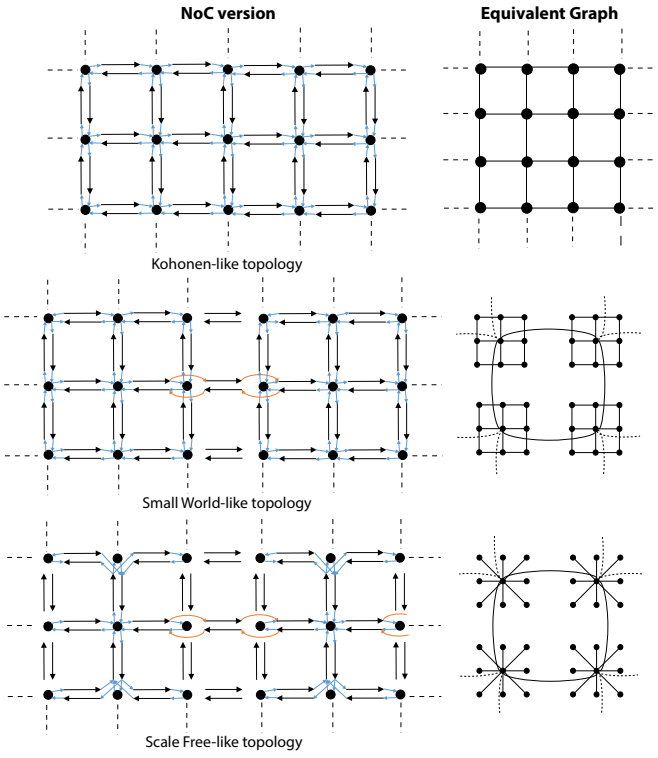

Figure 5: Examples of connection configurations on a NoC.

General Model

Let's consider an undirected graph $\Phi = (\mathcal{N}, A)$ where $\mathcal{N}$ is a finite set of neurons and A is the set of every possible link between neurons. For example for $n, n' \in \mathcal{N}$, $(n, n') \in A$ is a link between n and n' but it does not mean that n and n' are connected.

The set $L = \{(n, e, o)/n \in \mathcal{N}, e = (n', n) \in A \cup \{(n, n)\}, o = (n, n") \in A \cup \{(n, n)\}\}$ is defined as all possible local connections that can exist in Φ. At the level of a neuron n, a triplet (n, e, o) represents a direct connection from an incoming link e (from n' to n) or from neuron n itself (when $e = (n, n)$) to an outgoing link o (from n to n") or to neuron n itself (when $o = (n, n)$). The triplet (n,(n,n),(n,n)) is not allowed because it would represent a neuron connected to itself. A configuration of a NP-SOM based on Φ is a subset $C \subset L$ of instantiated connections.

Neuron distance

The definition of the distance first needs the definition of a simple path.

Let us consider two neurons $n_1, n_2 \in \mathcal{N}$, a simple path $\gamma(n_1, n_2)$ is a list of triplets of C, that represent successive connected links able to "go" from n_1 to n_2 within Φ, where there is no intermediate neuron in the path. This can be represented as follows:

$$\gamma(n_1, n_2) = ((n'_1, e_1, o_1), ..., (n'_m, e_m, o_m))$$

with

- $n'_1 = n_1$, $e_1 = (n_1, n_1)$, $n'_m = n_2$, $o_m = (n_2, n_2)$
- $\forall i \in \{1, \ldots, m\}$, $(n'_i, e_i, o_i) \in C$
- $\forall i > 1$ and $\forall n \in \mathcal{N}$, $e_i \neq (n, n)$
- $\forall j < m$ and $\forall n \in \mathcal{N}$, $o_j \neq (n, n)$
- $\forall i \in \{2, \ldots, m-1\}$, $e_i = (n'_{i-1}, n'_i)$ and $o_i = (n'_i, n'_{i+1})$

Thus, $\forall (n, n') \in \mathcal{N}^2$, if there is a simple path $\gamma(n, n')$ in Φ configured by C, the distance between n and n' is $d(n, n') = 1$. We then define a complex path between neurons n and n' as a list of simple paths $\Gamma(n, n') = (\gamma(n, n_1), \gamma(n_1, n_2), ..., \gamma(n_{m-2}, n_{m-1}), \gamma(n_{m-1}, n'))$. The length of a complex path is the size of the list (number of simple paths in the complex path). Eventually, the distance d between two neurons (n, n') is the minimum length of a complex path $\Gamma(n, n')$ in Φ configured by C. If there is no such complex path, the distance is $+\infty$.

Learning algorithm

The learning algorithm of a NP-SOM defined by a graph of neurons Φ and a configuration C of Φ is the same learning algorithm as for Kohonen SOMs, see Section 2, except that the distance used in equation 1 is defined as just above.

4 Results

4.1 Statistical analysis

In order to compare the different topologies on our set of metrics, we ran the three of them on three images from our dataset, and three configurations of picture sizes and neuron numbers so that all configurations approximately results in the same theoretical compression ratio (CR) of 9 (3x3 sub-pictures with 9x9 neurons has a CR of 9.23, 6x6 ones with 12x12 neurons is at 9.36 and 9x9 ones 9x9 neurons has a CR of 9.02). For each combination of all parameters, we ran 20 different seeds for the random weight initialisation and with each 10 other seeds that determine the order of presentation of the training vectors. So in Figure 7, each boxplot distribution corresponds to 200 random runs for each of the three topologies. Given that the variance of the distributions significantly differ, we used the Kruskal-Wallis rank based test. We ran this test for each of the 36 configurations represented in the Figure 7 and found p-values lower than $2e^{-12}$ for any configuration, except one, which is the total compression on the *Lake* image with 3*3 thumbnails.

On the resulting boxplots in Figure 7, we can observe that the distributions differ greatly between the topologies and images. For the mean error criterion, the `star` NP-SOM obtains the lowest value, and `kohonen` has the highest one, except when using small 3x3 thumbnails where it is equivalent to `star`. This could be due to the fact that a smaller dimensionality on data increases the density of samples in the training space, thus making stronger bonds between neurons an improving factor, unlike in larger dimensional spaces.

The PSNR column shows again `star` as a clear winner. The `sw` NP-SOM follows with a notably high variance. For `kohonen` we can observe that it is clearly less satisfactory than with `star`, even in the cases where it was better in mean error. The PSNR is based on the squared error, thus `star`'s better performance here implies that this NP-SOM is more capable of learning outliers training vectors.

The third criterion used in Figure 7 relates to differential coding. We computed a ratio by dividing the combination of differential coding followed by an entropy coding, by what the result would have been if only entropy coding was used without differential coding. This is meant to measure the gains of differential coding, by which we can estimate a SOM property that neighbouring neurons are also closely related in the input space, and as we expect similar sub-pictures to be close in the image, differential coding should give low numbers resulting in a more efficient entropy coding. The `kohonen` NP-SOM is as expected a clear winner here. The strong neighbourhood correlations aren't as present in the other layouts, consequently the gains here are less important because close sub-pictures may not be coded with close neurons.

The last column in Figure 7 is the total compression ratio. It is obtained by using the basic VQ compression followed by lossless differential and entropy coding. There is no clear winner here, but `kohonen` and `sw` seem a little bit ahead. The compression gains observed in differential coding for `kohonen` are somewhat mitigated by the density-respecting property of Kohonen SOM. As we can see in Figure 6 the codewords are more equally distributed for `kohonen` than for `star` or `sw` where a single neuron can represent a great number of pictures. This has an impact on the entropy coding compression, and explains why `kohonen` does not always reach the best compression ratio.

4.2 Examples

In this section we present some visual results and interpretations that complements the statistical data seen in the previous subsection, using a single random run of our SOM on a test image. On Figure 8 we can see the com-

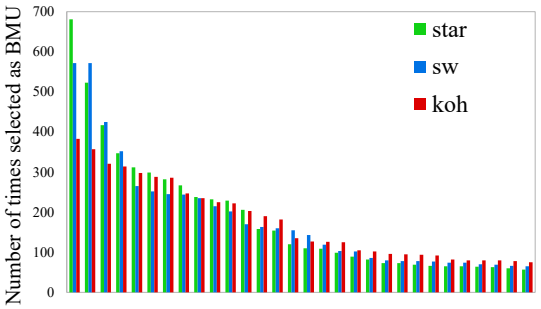

Figure 6: Histogram of the first 30 neurons sorted in decreasing order of times they were selected as BMU on one run.

pressed/decompressed version of the *Einstein* image (row 1: using 3×3 thumbnails, row 3: using 9×9 thumbnails) and the associated codebooks (rows 2 and 4) by showing the subpictures learned by the neurons arranged according to the neuron positions in the underlying topology. The left column corresponds to the results obtained with `kohonen` and the right one with `star`. For `kohonen` we can see a clear correlation between all neighbouring neurons in the codebook, but most neurons just learn some shade of grey with only a few features. On the other hand `star` better learns the most salient facial features (eyes, nose, moustache, ...). The `star` codebook shows a clear segmentation of blocks of 3 by 3 codewords. All codewords within each block are closely related but still have enough freedom to learn specific features, while block centers (connected in `sw`) are highly correlated.

Similar results are obtained for the other images, though they were chosen to stand for very different features (texture, saliency, ...). For example the compressed/decompressed *Pepper* image shows that edges are better defined and less blurry with `star`. The `star` codebook segmentation in blocks of 3 by 3 codewords is much slighter. Qualitative results obtained by `sw` are closer to those obtained by `star` than by `kohonen`.

5 Conclusion

In this paper, we defined the concept of network programmable self-organizing map (NP-SOM) from which we can define variants of Kohonen's SOMs that use different hardware-compatible topologies. Specific NP-SOMs using static underlying topologies have been derived to study the impact of topologies on the performance of the SOM. This is done in the context of the SOMA project to gain insights on the future introduction of advanced structural plasticity rules that induce changes to the network topology. By hardware-compatible topologies, we specifically mean topologies compatible with a simplified model of Network-on-Chip (NoC) which is a decentralized communication routing system.

The two proposed topologies, scale-free star-like and small-world-like, have numerous advantages for our proposed usage. They use less connections (approximately 45% less than in a grid for the scale-free star-like topology and 30% for the small-world-like one). The NP-SOMs that use one of these two topologies (called `star` and `sw`) have a lower error rate and higher PSNR than Kohonen SOM, at the cost of losing Kohonen's integration properties, which is reflected

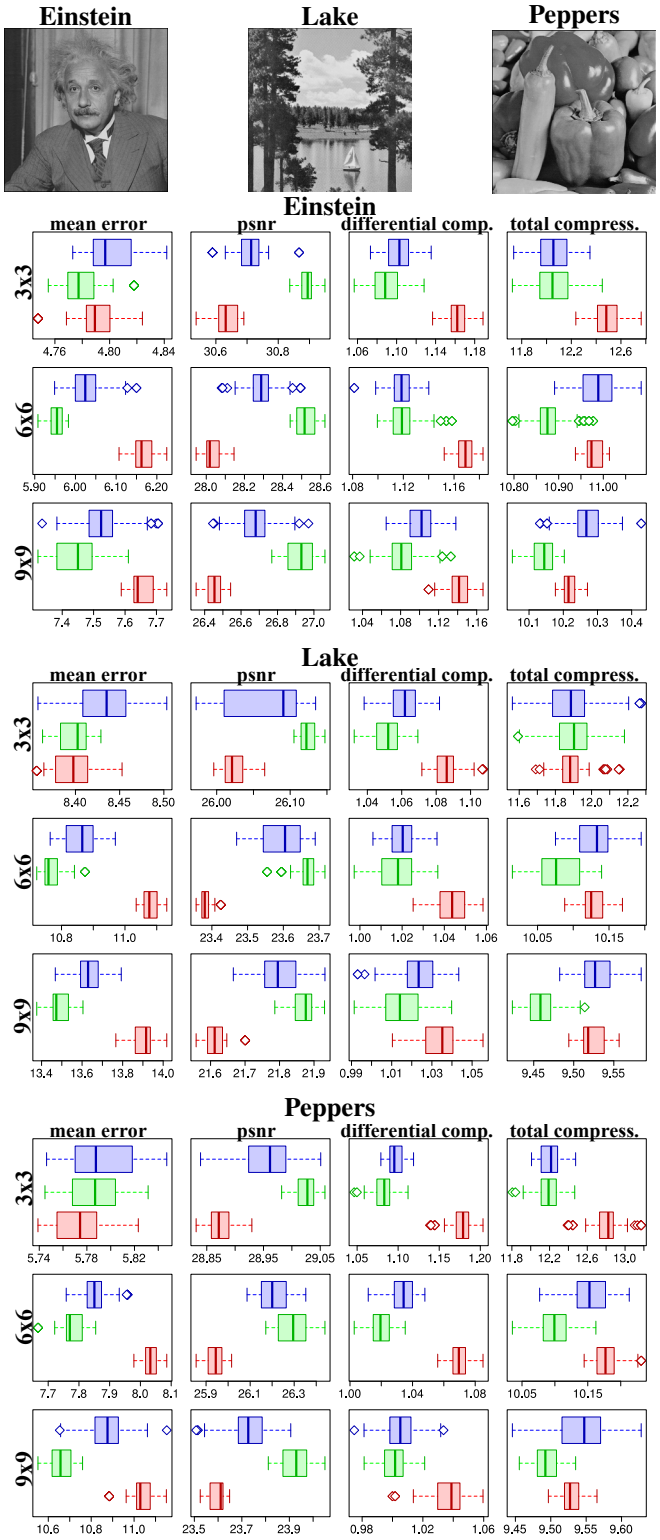

Figure 7: Resulting performance distributions on three images from our dataset. Each colour represents a topology, red corresponds to `Kohonen`, green to `Star` and blue to `Small-worlds`. For all criteria except mean error, the higher is the better.

with lower differential coding gains. On the other hand, lower neighbouring constraints gives `star`, and to a lesser extent `sw`, more freedom to explore the input space and adapt to the

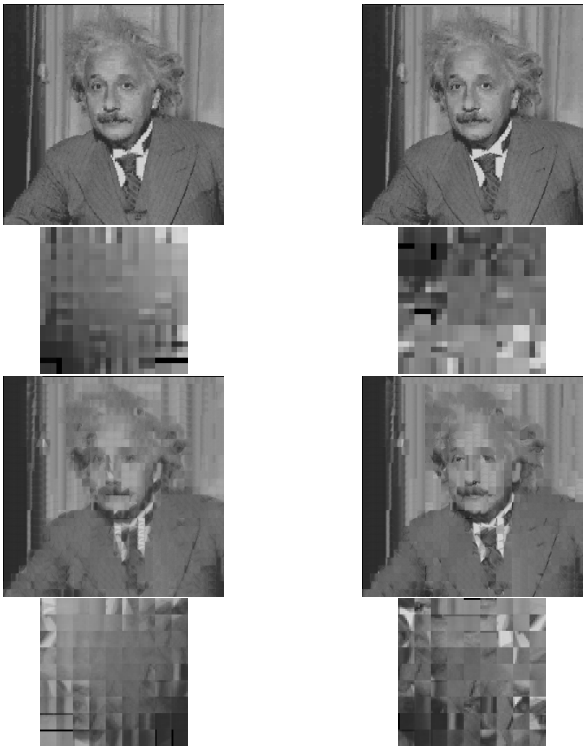

Figure 8: Compressed/decompressed image and learned sub-pictures of the *Einstein* image, using (top) 3×3 sub-pictures and 9×9 neurons, or (bottom) 9×9 sub-pictures and 12×12 neurons. `kohonen` is on the left and `star` on the right. (The horizontal and vertical black lines in the SOM are borders present in the original image.)

shape of the data. Consequently they follow a more segregatory approach, and are able to learn outliers that a strongly constrained Kohonen SOM would have ignored.

For the SOMA project this result suggests that the networking between neurons in SOM has a notable impact on it's performances and behaviour, and validates our approach to use structural plasticity as a mean to modify and improve SOM properties. For example `star` and `sw` are able to introduce mode abrupt transitions between neighbouring codewords when it better fits the input. This property may result in a more clearly defined task allocation for the cores at the border of processing areas.

Acknowledgement

The authors would like to thank the Swiss National Science Foundation (SNSF) and the Agence Nationale de la Recherche (ANR) for funding the SOMA project: <ANR-17-CE24-0036>.

References

[1] SATURN, "Self-adaptive technologies for upgraded reconfigurable neural computing, french research agency (anr), http://projet-saturn.ensea.fr," 2011-2014.

[2] A. Upegui, F. Vannel, B. Girau, N. Rougier, and B. Miramond, "Pruning self-organizing maps for cellular hardware architectures," in *NASA/ESA conf. on Adaptive Hardware and Systems*, 2018.

[3] H. Yin, "The Self-Organizing Maps: Background, Theories, Extensions and Applications," in *Computational Intelligence: A Compendium* (J. Kacprzyk, J. Fulcher, and L. Jain, eds.), vol. 115, Springer, 2008.

[4] A. Vasuki and P. Vanathi, "A review of vector quantization techniques," *IEEE Potentials*, vol. 25, pp. 39–47, July 2006.

[5] J. MacQueen, "Some methods for classification and analysis of multivariate observations," The Regents of the University of California, 1967.

[6] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, pp. 59–69, Jan. 1982.

[7] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "Neural-gas network for vector quantization and its application to time-series prediction," *IEEE Trans. on Neural Networks*, vol. 4, no. 4, 1993.

[8] B. Fritzke, "A Growing Neural Gas Network Learns Topologies," in *Advances in Neural Information Processing Systems 7*, pp. 625–632, MIT Press, 1995.

[9] S. Marsland, J. Shapiro, and U. Nehmzow, "A self-organising network that grows when required," *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 15, pp. 1041–1058, Nov. 2002.

[10] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, Jan. 2013.

[11] L. Khacef, B. Girau, N. Rougier, A. Upegui, and B. Miramond, "Neuromorphic hardware as a self-organizing computing system," in *Neuromorphic Hardware In Practice and Use - WCCI/IJCNN workshop*, 2018.

[12] Y. Xiao, C.-S. Leung, P.-M. Lam, and T.-Y. Ho, "Self-organizing map-based color palette for high-dynamic range texture compression," *Neural Computing and Applications*, vol. 21, pp. 639–647, June 2012.

[13] C. Amerijckx, J.-D. Legat, and M. Verleysen, "Image Compression using self-Organizing Maps," *Systems Analysis Modelling Simulation*, vol. 43, 2003.

[14] U. of Granada Computer Vision Group. Available at `http://decsai.ugr.es/cvg/dbimagenes/g256.php`.

[15] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "Hermes: an infrastructure for low area overhead packet-switching networks on chip," *INTEGRATION, the VLSI journal*, vol. 38, no. 1, pp. 69–93, 2004.

[16] L. Fiack, B. Miramond, A. Upegui, and F. Vannel, "Dynamic parallel reconfiguration for self-adaptive hardware architectures," in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2014)*, 2014.

[17] B. Girau, "FPNA: concepts and properties," in *FPGA Implementations of Neural Networks*, Springer, 2006.