In the quest of efficient hardware implementations of dynamic neural fields: an experimental study on the influence of the kernel shape

Benoît Chappet de Vangel, Jérémy Fix

In the quest of efficient hardware implementations of dynamic neural fields: an experimental study on the influence of the kernel shape

Benoît Chappet de Vangel
Université de Lorraine, 54000 Nancy, France
Email: bdevangel@gmail.com

Jérémy Fix
UMI 2958, Georgia Tech - CNRS,
CentraleSupélec, Université Paris-Saclay, 57070 METZ
Email: Jeremy.Fix@centralesupelec.fr

***Abstract*—Dynamic neural field (DNF) is a popular mesoscopic model for cortical column interactions. It is widely studied analytically and successfully applied to physiological modelling, bio-inspired computation and robotics. DNF behavior emerges from distributed and decentralized interactions between computing units which makes it an interesting candidate as a cellular building-block for unconventional computations. That is why we are studying the hardware implementation of DNF on digital substratum (eg. FPGA). As shown in previous papers, this implementation requires several modifications to the equations in order to obtain decent hardware surface utilisation and clock speed. Here we show that the modification of the lateral weights kernel function is possible as long as certain conditions, enumerated in Amari's seminal work are respected. Thank to metaheuristic optimisation it is possible to find the right parameters for two behavioral scenarii of bio-inspired computation interest. We show that the two most hardware-friendly kernels (difference of linear functions and piece-wise function) are as easy to tune as the traditional Mexican hat kernel. However the difference of exponential kernel is more difficult to tune.**

I. Introduction

Dynamic neural fields (DNF) is a mean field model for cortical neural population behavior. In DNF, both excitatory and inhibitory neural population share the same equation and the propagation delays are neglected for the afferent and lateral contributions. It has interesting applications for physiological modeling [1], autonomous robotics [2]–[4] and bio-inspired computing like multi-level clustering [5], classification [6] or tracking [7]. This wide spectrum of application makes it an interesting tool for unconventional computing substratum. Unconventional computing is the field which study new computing paradigms to escape from von-Neumann bottleneck. New hardware have to be developed together with new algorithms both often based on a biological inspiration. Unconventional computing include quantum computing, chemical computer for reaction diffusion computations, cellular computing, membrane computing etc. Neuro inspiration can also be included in unconventional computing with neuro-morphic hardware [8] together with neuron-based computations. For most of these new computing paradigms, the common idea is to have massively parallel computation in a decentralized and unsupervised fashion. The resulting behavior is an emergent property of the hardware substratum. It is attractive because it is generally very robust, scalable and fast on a certain category of problems [9]. We are studying the dynamic neural fields as a candidate for emergent computation with efficient hardware implementation [10] and by studying the spectrum of DNF computation abilities.

After the seminal work of Amari, a lot of analytical results were shown on the dynamical properties of the DNF. Bump stability, oscillations, travelling waves etc [11], [12]. On the contrary, in [3], [7], [13], the concern was more on the utilisation of the DNF as a building block for bio-inspired cognition behaviors with visual attention, working memory and tracking working together to obtain the right behavior.

Hardware implementation of DNF is not straightforward. The heaviside activation function has to be use to diminish the bandwidth (some time even with a change in the potential equation to reduce the activation time with a spiking behavior like in [14]). But the main issue persists: the global connectivity between the computing units. As it is not imaginable to connect so many units together we are looking for a von-Neumann neighbourhood communications between our units thus facilitating the hardware implementation. In previous work we proposed an approach to bypass the global connectivity in a cellular way by using a stochastic bit stream approach [15]. A direct consequence was a modification of the lateral-weights function of the DNF from a difference of Gaussian to a difference of exponential.

More generally, for any hardware implementation we often want to modify the equations and mainly the lateral weights kernel equation. Are the main properties of the DNF still present with unconventional lateral weights kernel shapes?

To answer this problematic we will use meta-optimisation to show that it is possible to find parameters which preserve the DNF behaviors for the most interesting abilities of the DNF: selection and memory.

II. Methods

A. Dynamic neural fields

Dynamic neural fields were introduced in [16]–[18] as a mean-firing rate model of coupled populations of inhibitory and excitatory neurons. The evolution of the potential $u(x,t)$ of a neural population reads:

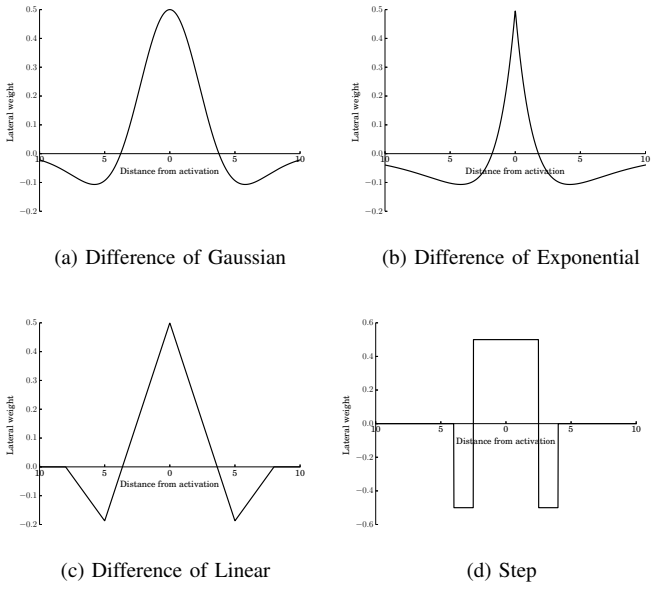
$$\tau \frac{\partial u}{\partial t}(x,t) = -u(x,t) + \sum_y w(x,y) f(u(y,t)) + I(x,t) + h \tag{1}$$

where f is an activation function (generally a sigmoid or a step function), I is the afferent input and h is the resting potential. In this paper we define the transfer function f, which links the firing rate to the membrane potential, as a Heaviside function ($f(x) = \mathcal{H}(x)$) as its implementation is much easier in hardware. The lateral weights kernel $w(x, y)$ is responsible for the lateral interactions within the field.

In [16], [19] the lateral weights kernel general shape is described to ensure convergence and stability:

1) w(x,y) depends only on x-y (homogeneous);
2) w(x,y) is positive around its origin (excitatory part);
3) w(x,y) is negative and tends towards 0 outside this neighbourhood (inhibitory part);
4) w(x) = w(-x) (symmetry).

A traditional choice can be a Mexican hat function (see fig. 1a) but we will study in this paper different other shapes which are convenient for hardware and software implementations. These weight functions keep the above properties and roughly the excitatory and inhibitory influences of the Mexican hat.



(a) Difference of Gaussian (b) Difference of Exponential

(c) Difference of Linear (d) Step

Fig. 1. The different lateral weights function that we are studying in this paper. The difference of Gaussian is our control and the three other are modified to improve hardware implementation.

a) Mexican hat or difference of Gaussian (DoG): this is our control for the behavior of the DNF.

$$w(x) = A_e \exp^{\frac{-x^2}{2\sigma_e^2}} - A_i \exp^{\frac{-x^2}{2\sigma_i^2}} \tag{2}$$

b) Wizard hat or difference of exponential (DoE): this is a kernel that we already studied experimentally and implemented in hardware in [10]. It has also been studied analytically by several authors because it is easily tractable [11], [19].

$$w(x) = A_e \exp^{\frac{-4|x|}{\sigma_e^2}} - A_i \exp^{\frac{-4|x|}{\sigma_i^2}} \tag{3}$$

The terms to the power of x have been defined in order to keep a shape of the kernel similar to the difference of Gaussian when using the same parameters.

c) Difference of linear function (DoL): this is a new kernel that we will use in a new architecture. The linearity of the function greatly increases the computation speed in a hardware substratum.

$$w(x) = A_e \left[1 - \frac{|x|}{2\sigma_e}\right]^+ - A_i \left[1 - \frac{|x|}{2\sigma_i}\right]^+ \tag{4}$$

with $A_e > A_i$, $\sigma_i > \sigma_e$ and $[.]^+$ denotes the positive part of the inner term.

d) Rectangular Mexican Hat or step function (Step): already used in [20] to generate a working memory behavior, the step function is a piecewise constant kernel whose equation reads:

$$w(x) = A_e \mathcal{H}(|x| - \sigma_e) - A_i \mathcal{H}(|x| - \sigma_i) \tag{5}$$

with $A_e > A_i$, $\sigma_i > \sigma_e$ and $\mathcal{H}$ denotes the Heaviside function. This is the cheapest implementation possible as its computation is based on simple comparisons.

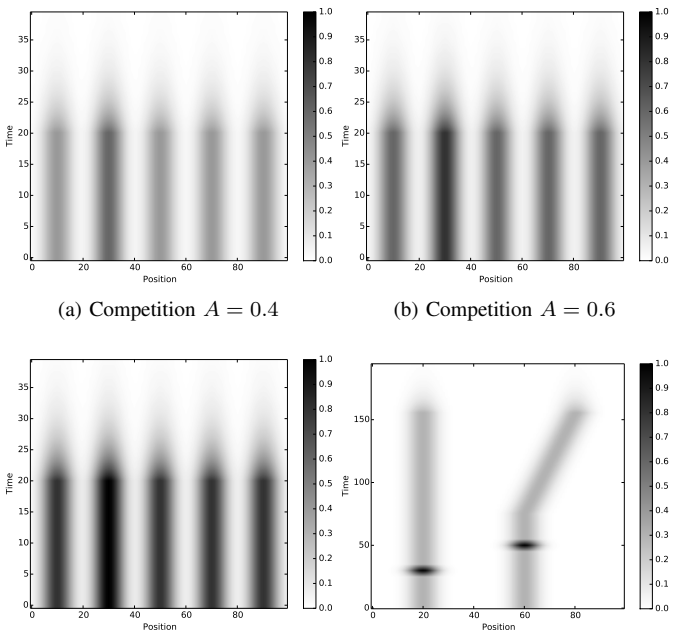
B. Optimization problem

Dynamic neural fields have been shown to exhibit various properties such as the ability to create a localized decision (a so-called *bump*) where the input is locally maximal as well as to retain an information and therefore to act as a memory of some gated inputs. In order to determine if a dynamic neural field with the previously introduced kernel functions are still able to exhibit these properties, we follow the optimisation procedure of [21]. It consists in introducing some scenarios with a set of inputs and particular cost functions to be minimized. The cost function or fitness captures the expected behavior when the neural field is fed with the proposed input. The optimization of the scenarios is performed with the particle swarm optimizer (PSO) [1], a metaheuristic which only requires to evaluate the cost function to be minimized and not any of its derivatives which is handful given the non linearities in the cost function and neural field equations. In this paper we consider two scenarios: a competition scenario and a working memory scenario. In the competition scenario, the neural field has five excited regions among which one region is slightly more excited than the others, as depicted on fig. 2. The input consists in four Gaussian stimuli of variance $\sigma = 4$ and amplitude $A \in \{0.4, 0.6, 0.8\}$ and one Gaussian stimulus with the same variance but amplitude $A + 0.2$.

The cost function to be minimized is designed to ensure that a bump appears in the neural field at the location where the input is maximal while vanishing when the input is switched off and, formally, reads :

$$J = \sum_{x,|x-x_0|\leq\sigma} (1 - f(u(x, \frac{T}{2})) + \sum_{x,|x-x_0|>\sigma} f(u(x, \frac{T}{2})) + \sum_x f(u(x, T))$$

[1] but any other black box optimizer such as CMA-ES would have been well suited



(a) Competition $A = 0.4$ (b) Competition $A = 0.6$

(c) Competition $A = 0.8$ (d) Working memory

Fig. 2. In the competition scenario, three conditions (a),(b),(c) are simultaneously optimized with various input amplitudes in order to optimize the ability of the neural field to select the most salient input. In the working memory scenario (d), the neural field is optimized to keep track, in memory, input stimuli that get transiently over-excited.

where $T = 40$ denotes the maximal time of the scenario and x_0 the mean of the input stimulus with the highest amplitude. The cost function simply captures the requirement to get a bump centered on x_0 at half of the scenario, i.e. $f(u(x, \frac{T}{2})) = 1$ for $|x - x_0| \leq \sigma$ and $f(u(x, \frac{T}{2}))$ when $|x - x_0| > \sigma$, and no firing rates at the end of the scenario and hence the term $\sum_x f(u(x, T))$ to be minimized. In order to get a neural field that is more robust than one we would obtain by optimizing on a single input condition, we simultaneously optimize three conditions with various input amplitudes. Indeed, if the neural field is optimized on a scenario with strong inputs it might not be able to perform a competition when the inputs are weak. Therefore, three conditions are optimized simultaneously with the amplitude of the weak stimuli set to $A \in \{0.4, 0.6, 0.8\}$ and the amplitude of the strong stimulus set to $A + 0.2$. Intuitively, such a behavior is obtained with lateral weights that are essentially globally inhibitory and locally weak but not necessarily excitatory as a positive firing rate is sufficient to get a bump with a Heaviside transfer function. It might be noted that if a single scenario was considered, it would have been straightforward to find optimal parameters. Indeed, setting the baseline to $h = -(A + 0.1)$ and discarding the lateral weights should produce parameters pretty close to an optimal solution. Optimizing simultaneously three scenarios discards such a trivial solution.

For the working memory scenario, two regions are excited in the input with an initial weak amplitude. At $t = 30$, the left-most stimulus get transiently excited to trigger the entrance in memory. At $t = 50$, the second stimulus is triggered. At $t = 75$, the right most stimulus starts moving to the right by $\delta x = 20$, ending at $t = 155$. The neural field is expected to track the motion of the input and to update its content respectively. The scenario ends at $t = 195$ after a period during which the inputs are switched off and the stimuli are expected to disappear from the memory. The scenario is challenging for a neural field because the memory requires stronger lateral excitation than the competition and this strong lateral excitation might actually induce such a strong local positive feedback that the field might get blind to motions of input stimuli. The inhibitory component of the lateral weights must be local and rather strong to keep the positive firing rates local to the memorized stimulus and to avoid interference with the other locations in the field. In order to capture this behavior, an appropriate cost function is designed. Its definition is similar to the one for the competition for capturing the presence or absence of a bump. However, it is here built from the contribution of the neural field activities at 5 stages :

- before the first stimulus gets focused, we minimize $\sum_x f(u(x))$
- after the first stimulus was focused, we minimize $\sum_{x,|x-x_1|\leq\sigma}(1 - f(u(x)) + \sum_{x,|x-x_1|>\sigma} f(u(x))$
- after the second stimulus was focused, we minimize $\sum_{x\in|x-x_1|\leq\sigma\cup|x-x_2|\leq\sigma}(1 - f(u(x)))$ $+\sum_{x\notin|x-x_1|\leq\sigma\cup|x-x_2|\leq\sigma} f(u(x))$
- after the second stimulus moved to $x'_2 = x_2 + dx$, while the first stimulus is at $x'_1 = x_1$, we minimize $\sum_{x\in|x-x'_1|\leq\sigma\cup|x-x'_2|\leq\sigma}(1 - f(u(x)))$ $+\sum_{x\notin|x-x'_1|\leq\sigma\cup|x-x'_2|\leq\sigma} f(u(x))$
- at the end of the scenario, we minimize $\sum_x f(u(x))$

For all the simulations, a time discrete neural field equation is used :

$$\begin{aligned} u(x, t+1) &= u(x, t) + \Delta u(x, t) \\ \frac{\tau}{\Delta t}\Delta u(x, t) &= -u(x, t) + \sum_y w(x, y) f(u(y, t)) + I(x, t) + h \end{aligned}$$

The cost functions introduced previously are optimized with respect to 6 parameters: the time constant $\frac{\Delta t}{\tau}$, baseline h as well as a specific parametrization of the weight functions to ensure the locally-excitatory/globally-inhibitory shape of the kernel. The kernels have been introduced with A_e, σ_e for the excitatory component and A_i, σ_i for the inhibitory component. Actually, to ensure the appropriate shape of the kernels, we set $A_i = k_a A_e, k_a \in [0, 1]$ and $\sigma_e = k_\sigma \sigma_i, k_\sigma \in]0, 1]$. Therefore, the cost functions are optimized with respect to $\frac{\Delta t}{\tau}, h, A_e, k_\sigma, k_a, \sigma_i$ with the bounds given in the table below.

Parameter	Bounds
$\frac{\Delta t}{\tau}$	$[0, 0.3]$
h	$[-1, 1]$
A_e	$[0, 5]$
k_σ	$[1e^{-3}, 1]$
k_a	$[0, 1]$
σ_i	$[1, 100]$

C. Optimization algorithm : Particle Swarm Optimization

As in [21], the cost functions previously introduced are minimized using the Particle Swarm Optimization (PSO) algorithm. The PSO algorithm is an optimization heuristic inspired by the flocking behaviors of birds. Formally, it consists in considering a collection of N so called particles which have a position $p_i(t)$ in the parameter space which evolves according to the following equations :

$$\begin{aligned} p_i(t+1) &= p_i(t) + v_i(t+1) \\ v_i(t+1) &= wv_i(t) + c_1 r_1 (p_i^l(t) - p_i(t)) + c_2 r_2 (p_i^g(t) - p_i(t)) \end{aligned}$$

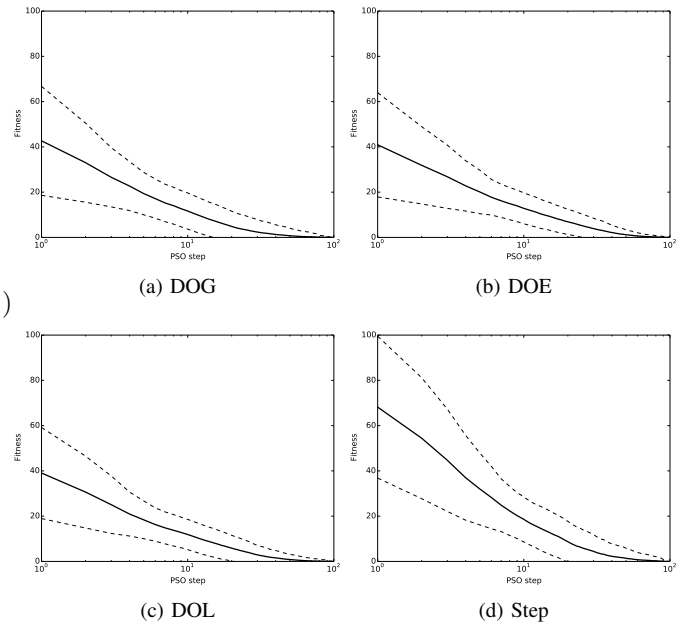
where w, c_1, c_2 are parameters of the algorithm, r_1, r_2 are vectors in $[0,1]^n$ randomly generated for every particle at every iteration, $p_i^l(t)$ is the best position the particle i ever had and $p_i^g(t)$ is the best position ever occupied by a particle in the neighborhood of the i-ith particle. The particles are not necessarily interacting directly with all the other particles and a neighborhood is introduced. Several neighborhoods have been considered in the literature, for example all-to-all, ring, von Neumann (grid with cyclic boundary conditions), random, etc. Here we stick to the standard PSO algorithm 2006 as defined in [22] and therefore use a random informants topology where each particle informs $K = 3$ other particles. In addition, at every iteration, if no improvement in the best fitness is observed, the topology is regenerated. The parameters are set to $w = \frac{1}{2\log 2}$ and $c_1 = c_2 = \frac{1}{2} + \log 2$ as suggested in [22].

III. Results

A. Competition

The competition scenario was optimized with $n = 20$ particles during 100 epochs (i.e. 2000 evaluations of the cost functions) and repeated 1000 times. In order to get a robust analysis, we keep only the trials which ended with a cost function at 0. This is clearly a conservative choice but actually, most of the runs successfully reached a null cost. Out of the 1000 trials, the success rate was respectively 98.9 % for the DOG, 84.1 % for the DOE, 95.2 % for the DOL and 95.2% for the step weight function. On the figure 3 we plot the mean fitness and their standard deviation as a function of the epoch of PSO for all the successful trials for each weight kernel.

To better compare the evolution of the cost functions for each kernel, the mean cost function of the successful trials for all the kernels is shown on the figure 4. The optimization is slower for the step function and harder for the wizard hat (difference of exponential). More interestingly, there are common patterns that can be observed from the distribution of the optimal parameters. We shall comment first the distribution of the parameters that are not shown in the paper but the optimal parameters are available online [23]. There is not much to say about the time constant $\frac{\Delta t}{\tau}$ which is confined in $[0.15, 0.3]$. The optimal time constant is clearly influenced by the dynamics that the scenario imposes as the field is requested to build up its decision within the $T/2$ first time steps. For all the kernels, the baseline h is confined within $[-0.5, 0]$, monomodal with a mean around $h_0 \approx -0.15$. For the weights, there are also common patterns. The amplitude of the inhibitory component is always close to the amplitude of the excitatory component with $k_a \approx 1$. For the DOG, DOE and DOL, the excitatory amplitude tends to be uniformly distributed within



(a) DOG (b) DOE

(c) DOL (d) Step

Fig. 3. Mean and standard deviation of the cost function for every weight kernel. The optimization reached the global minimum of the cost function for 98.9 % of the trials for the DOG (a), 84.1 % for the DOE (b), 95.2 % for the DOL (c) and 95.2% for the step (d) weight function.

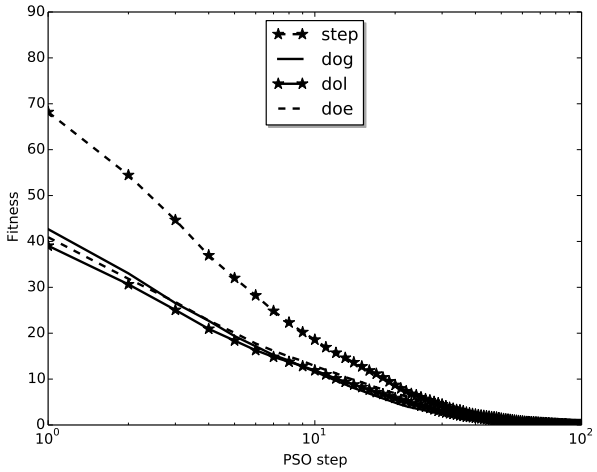
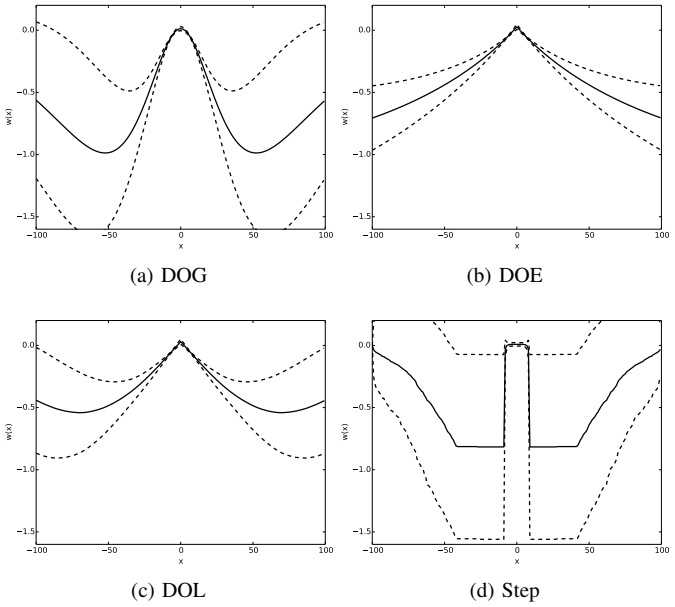


Fig. 4. Comparison of the evolution of the mean cost function for every the successful trials of every kernels for the competition scenario.

the bounds while the variance of the excitatory component tends to be half the variance of the inhibitory component and the variance of the inhibitory component tends to be uniformly distributed within the bounds but lower bounded by 20 which is clearly linked to the inter-stimulus distance. The distribution of the parameters is different for the step function for which the excitatory amplitude A_e is much more concentrated to small values, the variance of the excitatory component is strongly concentrated around one tenth of the inhibitory variance and the inhibitory variance is still uniformly distributed within the bounds but lower bounded by 40. As a summary, the mean weights and their standard deviation, still for the successful trials, are depicted on the figure 5. The

plotted mean to which the standard deviation is added can be sometimes misleading. For example, for the step function, none of the weights have a positive component for large distances but the distribution of the weight values are actually strongly asymmetric, densely packed around small values but with strong negative components. The plots of all the considered weights are given at [23].



(a) DOG (b) DOE

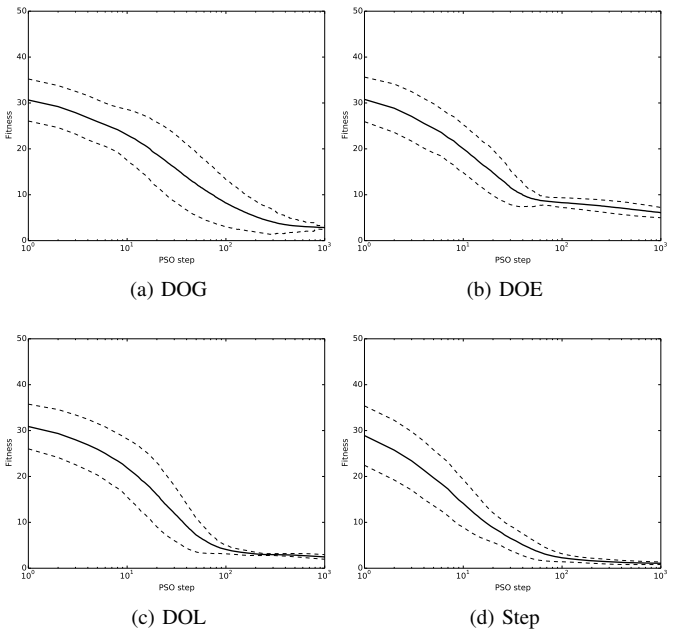
(c) DOL (d) Step

Fig. 5. Mean and standard deviation of the weights for each of the kernels for the successful trials of the competition scenario. More details about the shapes of the kernels are given in the text.

B. Working memory

The working memory scenario was optimized with $n = 200$ particles during 1000 epochs (i.e. 200000 evaluations of the cost function) and repeated 1000 times[2]. Contrary to the selection scenario, the choice of the threshold cannot be made too conservative otherwise too few runs would be considered. The 1000 runs of the 4 kernel setups (i.e. a total of 4000 neural fields) were ordered by increasing fitness and the behavior of all the runs were visually inspected in order to set a threshold on the fitness of the trials to consider. It turns out that all the runs with a fitness lower than 8 successfully performed the scenario while all the scenarios with a fitness larger failed with a systematic behavior. The neural fields of the failing runs were all able to memorize the input stimuli after the transient increase of amplitude but failed to track the change of position and lost the moving stimulus. Therefore, a fitness threshold of 8 was considered. Out of the 1000 trials, the success rate were respectively 100 % for the DOG, 46 % for the DOE, 100 % for the DOL and 100% for the step weight function. It is clear that the neural fields are much harder to optimize with the DOE. On the figure 6 we plot the mean fitness and their standard deviation as a function of the epoch of PSO for all the successful trials for each weight kernel.

[2]one trial took approximately one hour on an Intel(R) Xeon(R) 3075 at 2.66GHz. As a comparison, a trial with the competition scenario took one minute.



(a) DOG (b) DOE

(c) DOL (d) Step

Fig. 6. Mean and standard deviation over 1000 trials of the cost function for every weight kernel for the working memory scenario. The optimization reached the global minimum of the cost function for 100 % of the trials for the DOG (a), 46 % for the DOE (b), 100 % for the DOL (c) and 100% for the step (d) weight function. The y-scale is identical for the DOG, DOE and DOL but different for the Step function.

To better compare the evolution of the cost functions for each kernel, the mean cost function of the successful trials for all the kernels is shown on the figure 7. The speed of convergence of the optimization is similar whatever the kernel function. However, as mentioned previously, in the case of the DOE (wizard hard) only half of the runs successfully converged to the desired behavior. The optimization problem is harder for the difference of exponential probably because the kernel is very pointy. In order to correctly track a moving stimulus, the kernels has to spread excitation around excited regions in order to stimulate the propagation of excitation within the field, an unfavorable requirement for the difference of exponential.

As in the competition scenario, there are common patterns in the optimal parameters. From the histograms of the optimal parameters (not shown but available at [23]) the time constant $\frac{\Delta t}{\tau}$ is usually distributed within $[0.15, 0.3]$ but tends to be more packed toward large values (compared to the competition scenario) especially for the step kernel. The baseline h is still monomodal but much densely centered around -0.5 except for the step kernel for which the mean value is -0.33. For all the kernels, there is a clear tendency to get small amplitudes for the excitatory components. The excitatory component tends to be local and usually 40% to 90% of the variance of the inhibitory component. The inhibitory component amplitude tends to be 70% of the excitatory component for the DOG and DOL and more around 90% for the step. For the DOE, the inhibitory amplitude has much more freedom and tends to be uniformly distributed between 10% and 90% of the excitatory amplitude. The variance of the inhibitory component is rather local with mean values around 7 to 14 for the DOG, DOL and DOE and

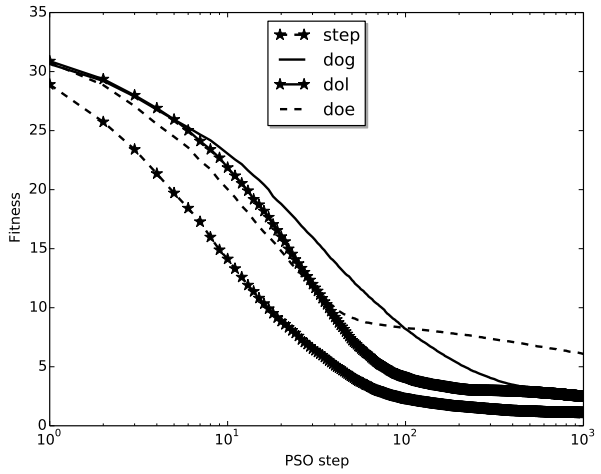
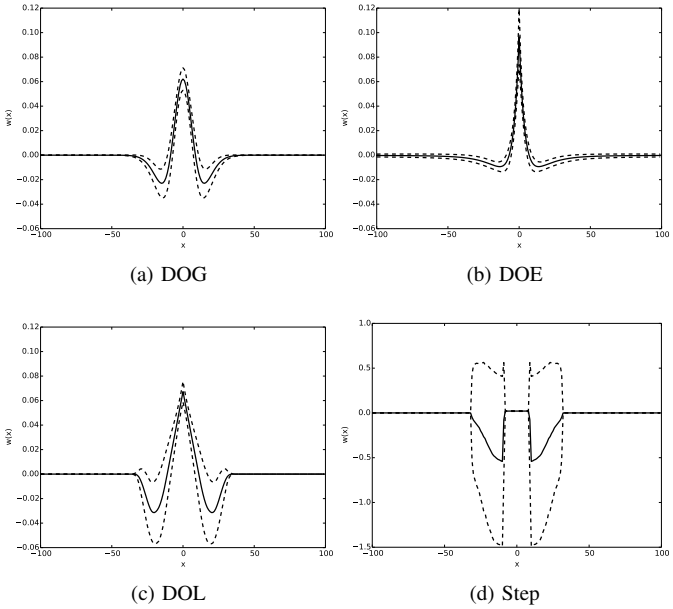


Fig. 7. Comparison of the evolution of the mean cost function for every successful trials of every kernels for the working memory scenario.

more around 20 for the step. The variance of the inhibitory component has clearly much less freedom for the working memory than for the competition scenario. Indeed, the role of the inhibitory component is to keep the memorized stimulus local while not interacting with the neighboring stimulus that is memorized which imply that the inhibitory component must exist and be local. As a summary, the mean weights and their standard deviation, still for the successful trials, are depicted on the figure 8.



(a) DOG (b) DOE

(c) DOL (d) Step

Fig. 8. Mean and standard deviation of the weights for every kernel for the successful trials. More details about the shapes of the kernels are given in the text. The y-scale is the same for all but the step kernel.

IV. Discussion

Following optimization guidelines introduced in [21] we were able to find parameters for 4 different kernel on two relevant scenarios. These results are very encouraging for efficient hardware implementation as they indicate that designers have a large choice of lateral weights function shape and precision and can therefore adapt the kernel to the targeted hardware substratum.

For instance, in our previous work we compared centralized hardware dynamic neural field implementation with a decentralized and a cellular approach [24]. The centralized implantation is based on address event representation (AER) communication between units [25]. We adapted it so as every time a neuron receives an activation information, the distance of the event (and not its address) is provided to the target unit. The modification of the neuron's potential then depends on a local look-up-table which associates a weight to the distance. Present results show that we can dramatically reduce the precision of the weights and even reduce them to a step function with two values using a simple test on the distance thus reducing the memory needs of the implementation.

We also investigated a decentralized, cellular architecture. We are interested by cellular computing as it is a massively parallel architecture with emergent behavior and a good candidate to escape the von Neumann bottleneck as memory and computations are distributed on the same substratum [9]. In this approach the challenge is to design all-to-all lateral interactions with local update rule. It is thus necessary to propagate the activation information sufficiently fast so as the propagation delay can be neglected. In [10] we introduced the randomly spiking dynamic neural fields (RSDNF) were random propagation of the information results in a lateral weights function in a shape of a difference of exponential (DoE) $w(d) = k_e p_e^d - k_i p_i^d$ were p_e and p_i are the probability of propagation for the excitation and inhibitions information respectively. We shown the behavior of this lateral weights function was correct for robust tracking. However the implantation performance in term of speed and area is limited by the generation of a tremendous amount of pseudo random numbers using a cellular automaton which takes half of the area on a FPGA chip. Moreover we show in the present paper that the optimization of the DoE is more difficult.

That is why we are interested by the DoL or step lateral weights kernel which are easier to optimize than the DoE and would not require random numbers generation. The gains in area and speed are not yet evaluated and will be shown in a future work. The first approach is to use a counter to count time since the activation of the units. Assuming that the data is propagated without delay, the cells know the distance of activated unit as it is the value of the counter. With this approach it is easy to derive the lateral weight using a look-up-table to associate one weight with one distance. Using a step function would again replace the look-up-table by a simple test and diminish the memory needs. This approach is cellular but not decentralized as a global synchronization signal has to trigger the start of the propagation. We are currently looking for a way to decentralize this computation as in [15].

It might however be noticed that using a step function for the lateral weights significantly improves the performance for software implementation. Indeed, when computing the lateral contributions for all the cells with a step function, starting from the lateral contributions for the first cell (involving 3 products and at most N sums), the lateral contributions for the other cells can be deduced, step by step, by only considering

the change in contribution of the four units on the splits between the positive/negative/null parts of the weights, which makes the complexity of evaluating such a neural field linear in the number of units rather than $N \log(N)$ with a general convolutive kernel.

Future work will also explore analytical tools to derive the lateral weights parameters in order to obtain an expected behavior. This problem known as the inverse problem is well defined and was studied in the context of DNF in [26].

To our knowledge similar study does not exist as we are among the first one to tackle the problematic of neural field implementation in digital hardware. However, a lot of work was done to implement spiking recurrent neural networks on digital or analogous substratum using locally cooperative globally competitive connectivity. For example the work of Neftci et al. [27], [28] which uses soft winner-take-all (sWTA) network as a building block for a VLSI neuromorphic chip. The sWTA dynamic is the result of global inhibitory connections and local cooperation with excitatory connections to close neighbours. This is similar to our step lateral weights function with infinite w_i for global inhibition.

REFERENCES

[1] W. Erlhagen and G. Schöner, "Dynamic field theory of movement preparation," *Psychological Review*, vol. 109, no. 3, pp. 545–572, 2002.

[2] C. Engels and G. Schöner, "Dynamic fields endow behavior-based robots with representations," *Robotics and Autonomous Systems*, vol. 14, no. 1, pp. 55 – 77, 1995.

[3] G. Schöner, M. Dose, and C. Engels, "Dynamics of behavior: Theory and applications for autonomous robot architectures," *Robotics and Autonomous Systems*, vol. 16, no. 24, pp. 213 – 245, 1995, moving the Frontiers between Robotics and Biology.

[4] J. Vitay, N. P. Rougier, and F. Alexandre, "A distributed model of spatial visual attention," in *Biomimetic Neural Learning for Intelligent Robotics*, ser. Lecture Notes in Computer Science, G. P. S. Wermter and M. Elshaw, Eds. Springer-Verlag, 2005, vol. 3575, pp. 54–72.

[5] D. Jin, J. Peng, and B. Li, "A new clustering approach on the basis of dynamical neural field," *Neural computation*, vol. 23, no. 8, pp. 2032–2057, 2011.

[6] M. Cerda and B. Girau, "Bio-inspired visual sequences classification," in *From Brains to Systems - Brain-Inspired Cognitive Systems*, 2010.

[7] N. P. Rougier and J. Vitay, "Emergence of attention within a neural population," *Neural Networks*, vol. 19, no. 5, pp. 573 – 581, 2006.

[8] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[9] M. Sipper, "The emergence of cellular computing," *Computer*, vol. 32, no. 7, pp. 18–26, 1999.

[10] B. Chappet de Vangel, C. Torres-Huitzil, and B. Girau, "Randomly spiking dynamic neural fields," *ACM Journal of Emerging Technologies in Computing Systems*, vol. 11, no. 4, pp. 1–26, Apr 2015.

[11] S. Coombes, "Waves, bumps, and patterns in neural field theories," *Biological Cybernetics*, vol. 93, pp. 91–108, 2005.

[12] P. C. Bressloff, "Spatiotemporal dynamics of continuum neural fields," *Journal of Physics A: Mathematical and Theoretical*, vol. 45, no. 3, p. 033001, 2012.

[13] J. Fix, N. Rougier, and F. Alexandre, "A dynamic neural field approach to the covert and overt deployment of spatial attention," *Cognitive Computation*, vol. 3, no. 1, pp. 279–293, 2011. [Online]. Available: http://dx.doi.org/10.1007/s12559-010-9083-y

[14] R. Vazquez, B. Girau, and J. Quinton, "Visual attention using spiking neural maps," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 31 2011-aug. 5 2011, pp. 2164 –2171.

[15] B. Chappet de Vangel, C. Torres-Huitzil, and B. Girau, "Stochastic and asynchronous spiking dynamic neural fields," in *Neural Networks (IJCNN), 2015 International Joint Conference on*, July 2015, pp. 1–8.

[16] S.-i. Amari, "Dynamics of pattern formation in lateral-inhibition type neural fields," *Biological Cybernetics*, vol. 27, pp. 77–87, 1977, 10.1007/BF00337259.

[17] H. R. Wilson and J. D. Cowan, "Excitatory and inhibitory interactions in localized populations of model neurons," *Biophysical Journal*, vol. 12, no. 1, pp. 1 – 24, 1972.

[18] R. L. Beurle, "Properties of a mass of cells capable of regenerating pulses," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 240, no. 669, pp. 55–94, 1956.

[19] Y. Guo and C. C. Chow, "Existence and stability of standing pulses in neural networks: I. existence," *SIAM Journal on Applied Dynamical Systems*, vol. 4, no. 2, pp. 217–248, 2005.

[20] M. A. Giese, "Technical applications of neural fields," in *Dynamic Neural Field Theory for Motion Perception*. Springer, 1999, pp. 173–199.

[21] J. Fix, "Template based black-box optimization of dynamic neural fields," *Neural Networks*, vol. 46, no. 0, pp. 40 – 49, 2013.

[22] M. Clerc, "Standard particle swarm optimization," OpenArchive HAL, Tech. Rep., 2012.

[23] J. Fix, "The source codes and results for the simulations of the paper," http://jeremy.fix.free.fr/Simulations/ijcnn_2016.html, 2016.

[24] B. Chappet de Vangel, C. Torres-Huitzil, and B. Girau, "Spiking dynamic neural fields architectures on fpga," in *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, Dec 2014, pp. 1–6.

[25] M. Mahowald, "Vlsi analogs of neuronal visual processing: A synthesis of form and function," Ph.D. dissertation, California Institute of Technology, 1992.

[26] R. Potthast and P. B. Graben, "Inverse problems in neural field theory," *SIAM Journal on Applied Dynamical Systems*, vol. 8, no. 4, pp. 1405–1433, 2009.

[27] E. Neftci, J. Binas, U. Rutishauser, E. Chicca, G. Indiveri, and R. J. Douglas, "Synthesizing cognition in neuromorphic electronic systems," *Proceedings of the National Academy of Sciences*, vol. 110, no. 37, pp. E3468–E3476, 2013.

[28] G. Indiveri and E. Chicca, "A vlsi neuromorphic device for implementing spike-based neural networks," ser. Proceedings of the 21st Italian Workshop on Neural Nets, B. Apolloni, S. Bassis, A. Esposito, and C. F. Morabito, Eds., 2011, pp. 305–316.