Learning of Spatiotemporal Behavior in Cellular Neural Networks *

Samuel Xavier-de-Souza, Johan A.K. Suykens, Joos Vandewalle

K.U. Leuven, ESAT-SCD-SISTA

Kasteelpark Arenberg 10

B-3001 Leuven (Heverlee) Belgium

e-mail: samuel.xavierdesouza@esat.kuleuven.be

September 22, 2005

*Submitted in 2005 to the Special Issue on CNN Technology of the International Journal of Circuit Theory and Applications.

Abstract

In this paper the problem of learning spatiotemporal behavior with cellular neural networks is analyzed and a novel method is proposed to approach the problem. The basis for this method is found in trajectory learning with recurrent neural networks. Despite of similarities, the two learning problems have underling differences which make non-trivial a direct mapping into the problem at hand. In order to solve the problem, a new cost function is proposed, which also assimilates time instants as parameters to be optimized. As a consequence, it does not force the desired spatiotemporal behavior to be learned in its original speed and thus different speed versions of the desired behavior are allowed to be learned. Hence also providing a promising direction for increasing the speed of existing applications. Learning examples are presented for different classes of spatiotemporal dynamics including spiral autowaves. Results of simulation and on-chip learning shows that the proposed approach is able to learn these dynamics with cellular neural networks.

Keywords: Trajectory Learning; Spatiotemporal Behavior; Autowaves; Cellular Neural Networks.

1 Introduction

In recent years, the existence of a variety of complex phenomena have been discovered in Cellular Neural Networks (CNN) systems. Many of these phenomena can also be encountered in different disciplines. Hence such systems have proven to be very good tools for modeling and simulation. Additionally, with the growing interest and success in implementing CNNs on silicon, excellent results have been achieved for different applications within the new paradigm of *active wave computing* [1][2]. This new form of information processing allows for silicon based implementations of algorithms for robot navigation, artificial retina, finger print enhancement, etc. Among these complex phenomena, two- and higher dimensional spatiotemporal behavior are of special importance. They appear in diverse fields of physics, chemistry, and biology. This type of behavior is observed in an active medium which can be modeled by arrays of coupled non-linear circuits like CNN cells. Autowaves, spiral waves, traveling waves and many other phenomena have been reported to emerge from an active medium consisting of an array of CNN cells. Nevertheless, a systematic method for learning such phenomena has not yet been developed. Although learning and tuning of CNN operations with static output has already been approached with different methodologies [3, 4], the training of spatiotemporal behavior demands accounting for time-varying outputs in a similar way as trajectory learning in recurrent neural networks (RNNs).

In this paper a novel methodology to train spatiotemporal behavior with CNNs

is proposed within the framework of trajectory learning, taking into account issues like initial conditions and training sets. The novelty lies on the derivation of a cost function that also assimilates the time instants into the set of parameters to be optimized allows for the learning of the desired behavior with different speeds rather than restricting it to the original speed of the dynamics. Besides the positive effect that this feature may have for increasing the speed of existing applications, it also reduces the necessity of generating a perfect training set, which is certainly the most important issue for the learning of 2-dimensional spatiotemporal behavior on CNNs.

This paper is organized as follows. An overview of the state of the art in trajectory learning and RNNs is given (Section 2) followed by the presentation of a few examples of 2-D spatiotemporal behavior (Section 3). Then the description of the proposed methodology for training such behaviors in CNNs is presented (Section 4). Finally, conclusions are made and important points are summarized.

2 Trajectory Learning and Recurrent Neural Networks

Trajectory learning, the problem of modifying parameters of dynamical systems to achieve that their outputs follow a given function of time has been analyzed by many authors [5, 6, 7, 8, 9, 10, 11, 12, 13] most frequently using recurrent neural networks (RNN) as a model for such systems. Consider the following RNN model

that describes the dynamics of these systems

$$\frac{d\mathbf{x}}{dt} = -\mathbf{x}(t) + W\mathbf{y} + W'\mathbf{u} + \mathbf{z} \tag{1}$$

where the output $\mathbf{y} \in \mathbb{R}^N$ is a function of the state vector $\mathbf{x} \in \mathbb{R}^N$, and the matrices W, $W' \in \mathbb{R}^{N\times N}$, are the weight matrices for the output $\mathbf{y}$ and the constant input $\mathbf{u} \in \mathbb{R}^N$ respectively. The term $\mathbf{z} \in \mathbb{R}^N$ denotes the bias of the network. The problem consists in minimizing a cost function which is not defined at a fixedpoint but rather is a function of the model's temporal behavior. The problem is often addressed in the literature by gradient descent methods [7, 14, 8, 15] (see [16] for earlier survey). These methods are known to coexist with problems like local minima [9] and vanishing gradients with the evolution of the dynamics [10]. On the other hand, methods that do not use gradient information of the learning process [11, 12, 10, 13] (e.g. simulated annealing, genetic algorithms, multi-grid random search, etc.), are persistently slower, but are more frequently able to converge to a globally optimal solution. The learning problem can be described by

$$\min_{W,W'} \left[E = \sum_{i=1}^{M} \int_{t_0}^{t_f} (y_i^d(t) - y_i(t, W, W'))^2 dt \right], \tag{2}$$

where the square difference between the desired trajectory function $y_i^d(t)$ and system's output function in time $y_i(t)$ is integrated from time t_0 to t_f and then summed over all M output neurons. This cost function $E(W, W')$ may also assume different forms [11, 15, 13], but this one will be used here to devise a new cost function for

the training of spatiotemporal behavior on CNNs.

The mapping of trajectory learning with RNNs into learning of spatiotemporal behavior with CNNs is straightforward. One only needs to consider the equivalence between output neurons and output cells. Although trajectory learning becomes increasingly complicated with the number of neurons N in the RNN as the number of weights increases quadratically, by assuming zero elements in the weight matrices, the computational burden can be reduced. This is essentially what happens with CNNs. More precisely, only weights of neighboring cells are taken into account with the remaining values of the weight matrices equating zero. Moreover, CNNs weights are often also space invariant. This means that the description of a neuron, or cell in CNNs, is sufficient to describe the whole system. These local and invariant weights are called CNN templates. Templates are local and invariant equivalents of the matrices W and W' together with the bias term $\mathbf{z}$ in Eq. (1). Therefore, the description of a first order CNN system with local and invariant weights can be reduced to the description of the behavior of a single cell:

$$\frac{dx_{i,j}}{dt} = -x_{i,j}(t) + Ay_{i,j} + Bu_{i,j} + z, \tag{3}$$

with A and B being the local and invariant equivalent of W and W', and i,j denoting the indices of the given cell in a regular grid.

Other models that are especially important for modeling of complex behavior are second order CNN systems. Their dynamics can be represented by the following

equations:

$$\frac{dx_{i,j;1}}{dt} = -x_{i,j;1}(t) + A_{1,1}y_{i,j;1} + A_{1,2}y_{i,j;2} + B_1u_{i,j;1} + z_1$$
$$\frac{dx_{i,j;2}}{dt} = -x_{i,j;2}(t) + A_{2,2}y_{i,j;2} + A_{2,1}y_{i,j;1} + B_2u_{i,j;2} + z_2$$

where the index after the semi-column identifies the layer and the indices i,j locate the given cell within the layer. This equation can also be written in its condensed form:

$$\frac{d\mathbf{x}_{i,j}}{dt} = -\mathbf{x}_{i,j}(t) + A\mathbf{y}_{i,j} + B\mathbf{u}_{i,j} + z \tag{4}$$

with

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2.1} & A_{2,2} \end{bmatrix}; \quad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}; \quad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix};$$

and

$$\mathbf{x}_{i,j} = \begin{bmatrix} x_{i,j;1} \\ x_{i,j;2} \end{bmatrix}; \quad \mathbf{y}_{i,j} = \begin{bmatrix} y_{i,j;1} \\ y_{i,j;2} \end{bmatrix}; \quad \mathbf{u}_{i,j} = \begin{bmatrix} u_{i,j;1} \\ u_{i,j;2} \end{bmatrix}.$$

These equations describe the behavior of a second order cell, which when isolated from the neighboring cells may behave as an oscillator [17] and when coupled to other cells is able to generate interesting spatiotemporal patterns. Moreover, these equations closely describe the model of the CACE1k chip [18] which is nowadays a major advance in VLSI implementation for generation of complex behavior.

Locality and invariance are the main advantages of CNNs. Despite the condensed set of values that describe these systems, a large variety of dynamical phenomena

can be observed. Moreover, locality and invariance alone make these systems very suitable for VLSI implementation which is a trend that has been emerging in the past years. Nowadays, high end silicon versions of CNN Universal Machines (CNN-UM) are already available for the development of image processing application at extremely high speed [19].

3 Examples of Spatiotemporal Behavior

A simple example of spatiotemporal behavior can be seen of Fig. 1(a) where a traveling dot can be observed traveling across the image at an arbitrary angle. In the same way as the dot, one might think of a traveling wave of a specific width also moving across the image, like in Fig. 1(b).

[Figure 1 about here.]

Aperiodic traveling waves like in Fig. 1(b) can be of two different kinds: "convergent" and "divergent"[1]. A typical case of a "convergent" wave is a wave that starts from a line and propagates with decreasing length in order to form a pyramid. This operation converges when the top of the pyramid is built like Fig. 2(a). An example of an application for this operation can be found in [20]. When aperiodic traveling waves do not converge to a fixed image[2], it is called here "divergent". A simple example of a "divergent" wave is the well known *shadow* template. Another

[1]here, these concepts do not have a strict mathematical meaning but are rather limited to geometric boundaries

[2]a fictitious image of infinite size is considered here to avoid misunderstanding

interesting aperiodic wave was used in [21] to compute the shortest path of flat and wrinkled labyrinths. This wave simulates e.g. the waves of combustion where an active medium cannot return to the same state after the propagation of the wave. Fig. 2(b) shows a typical combustion wave where the burning effect starts at the center of the image.

[Figure 2 about here.]

Another interesting class of spatiotemporal behavior is autowaves. The term autowave is an abbreviation of "autonomous wave" commonly used to characterize self-sustained signals that induce a local release of stored energy in an active medium, and use this energy to trigger the same process in neighbor regions. The term have been used in the CNN community to describe periodic traveling waves (often 2-dimensional) that have the following properties:

- the waveform and the amplitude of wave remain constant during propagation;
- the waves do not reflect at obstacles;
- colliding waves are annihilated and thus no interference emerges;
- diffraction can be observed in the same way as for classical waves.

Many waves occurring in nature share the same properties. Typical examples include: waves in the cerebral cortex, epidemic waves, combustion waves, reaction-diffusion processes, etc. In cellular neural networks, autowaves represent the means for the new paradigm of Active Wave Computing and have been used for example to

guide robots along obstacles towards a target [22]. An example of autowaves known as spiral wave is shown in Fig. 3.

[Figure 3 about here.]

In CNN systems, autowaves have been observed to emerge in 2-D arrays of second or higher order cells [21, 23] and delayed type first order cells [24]. Due to the fact that the arrays of regular first order cells can not generate the necessary active local dynamics, autowaves can not be observed in these systems. However, in [25] this type of waves were observed in a VLSI implementation that was designed to be a first order 2-D array of CNN cells [26]. Although internal sources of autowaves (chip-inherent) could not be avoided, external sources could also be placed generating a competition between the sources. Explanations for this observation are not fully understood yet although no important malfunctioning has been observed in this chip. One may assume that the physical system behaves more like a second order or delayed type first order CNN system rather than as a first order system as intended for the original design. Therefore, for all means the system that is used throughout this paper to produce autowaves is chosen to be a 2-layer array of second order CNN cells described by Eq. (4).

4 Spatiotemporal Learning with Cellular Neural Networks

The fact that the training of spatiotemporal behavior in CNN systems is reduced to the learning of the matrices A, B, and the bias z makes the process much simpler. Nevertheless, there are two important issues that limit this simplicity:

- First, in spite of being very efficient and widely used, gradient descent techniques are hard to be applied in this case. In CNNs, the non-linear output function $y = f(x)$ often assumes the piecewise linear non-differentiable form $f(x) = \frac{1}{2}(|x+1| - |x-1|)$, which complicates the derivation of an analytical form for the gradient of the cost function and would require techniques from non-differentiable optimization.

- Second, the generation of training sets is not a straightforward task. While a trajectory can be described by a sequence of values for every output, e.g. $\sin(t)$, spatiotemporal behavior in a grid of cells needs to be described by a continuous 2-D image sequence where the values of the pixels correspond to the output of a single cell. Due to the couplings, the desired values of every individual pixel trajectory can not be derived independently and need to be considered as a whole. The manual generation of the desired image sequence can thus be tricky and may result in a behavior that is physically impossible to be learned by a CNN system.

4.1 Learning sequences of images

The lack of the derivative for the cost function is not a serious problem. Although gradient methods are more efficient to solve convex and local optimization problems, there are no guarantees about the global optimality of the task to be learned for the problem considered here. Global optimization methods can be used here with reasonable confidence since the number of parameters to be optimized is not very high. Additionally, these methods are not limited to convex problems and can do surprisingly well on problems with many local minima. The second issue though, needs more care and will be addressed here in a case-by-case fashion. As a consequence of this last issue, the images in a training set may represent snapshots of the desired system's output which are irregular in time. In order to avoid the necessity of a strict match between a irregular time evolution of the desired and resulting behavior, a new cost function is proposed here. This cost function assimilates the time intervals between snapshots of the system's output into the set of optimization parameters. The problem of learning spatiotemporal behavior with CNNs can be presented in this way as

$$\min_{A,B,z,\Delta t_k,\Delta t_{k+1},\cdots,\Delta t_T} \left[E = \sum_{i,j} \sum_{k=1}^{T} (y^d_{i,j;k} - y_{i,j}(A,B,z,t_k))^2 \right], \tag{5}$$

where $\Delta t_k = t_k - t_{k-1} \ \forall \ k = 0, \cdots, T$, representing the time interval between two output samples with T being the finite number of samples and $t_0 = 0$. $y^d_{i,j;k}$ denotes the desired output value of a pixel in the kth image of a given sequence of T images.

The value $y_{i,j}(A, B, z, t_k)$ denotes the output value of a pixel as the system has evolved to the time instant t_k with weight matrices A and B, and bias z.

The initial conditions $x_{ij}(0)$ are assumed to be given. In actual CNN chips or simulators, initial conditions can be easily set by loading an Initial State image prior to template execution. This image will determine the initial state of all x_{ij}. A straightforward way to establish the initial conditions for learning of spatiotemporal behavior is to assume the Initial State image as the first image of the sequence to be learned.

Observe that the desired outputs $y^d_{i,j;k}$ do not depend on time but rather on the index k of time instants whereas the obtained outputs $y_{i,j}(A, B, z, t_k)$ do depend on the time instants t_k but these are also parameters to be optimized. A way to exemplify the results of such measure in view of trajectory learning is to think that if the trajectory to be trained is e.g. $\sin(t)$, the resulting trajectory is allowed to be any approximation of $\sin(\omega t)$, i.e. the desired trajectory becomes not only $\sin(t)$ but also any frequency modulation of it. Additionally, also observe that Δt_k is the parameter to be optimized rather than t_k itself. In this case all the desired time instants t^d_k are irrelevant, only the order in which the images are learned matters, not at which specific time instants the images are generated by the CNN. This is also the reason why the error term $(y^d_{i,j;k} - y_{i,j}(A, B, z, t_k))^2$ is summed over all k samples rather than integrated over time the interval $[0, t_T]$ like in Eq. (2). The right choice for the value of T will depend on how difficult the learning problem is and on how much time and processing resources are available.

The relaxation of the schedule for the resulting spatiotemporal behavior is an important issue for CNNs. Fixing a rigid schedule for the spatiotemporal trajectory to be learned would magnify the importance of a physically feasible training set and this needs to be avoided for a simple reason: in many cases nothing can guarantee that suggested time stamps for the desired spatiotemporal trajectory have a fixed relation between themselves in a real system.

It is also important to notice that hints about the task to be learned can be used to constrain the template form, e.g. symmetry, zero elements, etc., in order to reduce the number of weight elements to be optimized. For example, in the case of wave propagation in a given direction, one might want to impose the assumption of symmetry along the direction of the propagation. In this way the number of parameters to be optimized is reduced and the process can converge faster. However, if the direction of the propagation is not cardinal (i.e. north, northeast, etc.) these assumptions do not hold and the full template matrices need to be trained. This type of assumptions do not affect the unknown template values. Although the number of weights to be learned can be reduced this way, the remaining weights are still assumed to be totally unknown. This constrains can be later in learning process dropped to give a more defined form to the template.

The training of complex behavior like autowaves can be done in the same way as for other spatiotemporal behavior. Nevertheless, it can be given an extra degree of freedom which concerns the number of CNN layers involved. In a 2-layer CNN for example, autowaves occur simultaneously in both layers but often with different

waveforms. Frequently the outcome of what happens in one layer is sufficient for some applications and in this case only the output of this layer needs to be taken into account for the calculation of the cost function. However, besides the cases where the outcome of both layers is important, the inclusion of the output of the second layer in the cost function calculation can sometimes bring more insight about the location of a globally optimal solution. For this case it is only necessary to include a sum over the number of layers in Eq. (5), and an index for the layers in the output and desired output values.

4.2 Modifying speed

The speed or the amount of time necessary for a given dynamical system to reach a specified state will depend on its time constant. In trajectory learning, time constants are included in what are called *scale parameters* because of the nature of what happens in the system when these parameters are changed. In CNN systems, two systems with identical templates will reach a given state in a different amount of time depending on their time constants. The time constant of a system is thus one factor which can change the speed of evolution of such systems. For CNN systems, the speed can also be modified in another way. Two CNN systems with the same time constant τ can still present similar dynamical behavior with different speeds if their template values are modified accordingly, without changing τ.

After a given task is learned, one could apply constraints to a new optimization process and force the learned behavior to evolve faster (or slower), with as starting

point the first learned template. This can be done in an incremental way. In trajectory learning, this sort of learning is called *incremental learning* and is often used to gradually transform an existing trajectory into a different one by using intermediate target trajectories [13]. This feature might be very promising especially for VLSI CNN-UMs to increase the speed of existing template operations, bringing execution time to the speed limit and making a variety of applications working faster without performing any structural changes.

5 Simulation and On-Chip experiments

A variety of experiments were performed to evaluate the method proposed here. The optimization method used to minimize the cost function in Eq. (5) was Adaptive Simulated Annealing (ASA) [27]. This method has proved to be efficient for tuning fixed output templates for VLSI implementations [4]. It can be observed that if one makes $T = 1$ in Eq. (5) and t_1 is removed from the optimization and made sufficiently long, this cost function is reduced to the fixed output case. The same approach of relaxation of constraints and search boundaries used in [4] can also be made useful for learning in the following way: (a) in the beginning of the learning process no limits are imposed to the weight values and thus the maximum range of values are available; (b) after this process converges, better solutions are obtained by limiting the weight values to values that are close to the first solution and/or incrementally relaxing existing constraints, e.g. symmetry, zero values, etc; (c) the

last step is then repeated until any stopping criteria is reached.

Only immediate neighbor cells are assumed to have a non-zero weight, which makes the matrices A and $B \in \mathbb{R}^{3\times3}$. However, if prior knowledge about the template matrices is known, the number of parameters that actually needs to be optimized can be reduced, e.g. with symmetry this number can go from 9 to 5 parameters per matrix. Without any prior knowledge, the number of parameters to be optimized is the sum of the number of weights of a full template (in the single layer case: 19) plus the number of time intervals T, which will depend on the necessary accuracy. For simplification of the illustrative examples presented here, the input images $\mathbf{u}$ in Eqs. (3) and (4) will be set to zero, and therefore the input weight matrix B will be assumed zero. The inclusion of input images and input weights in the optimization is straightforward but brings little or no extra clarity to these examples.

The experiments were performed on the same set of examples presented in the last subsection. Learning was performed in simulation but also on-chip, using the ACE4k (64×64 cells) CNN-UM [28]. On-chip learning has a great advantage with respect to execution time. The speed with which tasks can be learned on CNN chips may open new possibilities with simulations that are difficult to achieve, even in the most modern digital computers.

The results of the operation of two templates trained using as training sets the images of Fig. 1(a) and Fig. 1(b) are presented in Fig. 4 for on-chip learning.

[Figure 4 about here.]

Aperiodic propagating waves were trained using the images of Fig. 2(a) and

Fig. 2(b) as training sets for the corresponding behavior. Results for on-chip learning can be seen in Fig. 5. Although this type of spatiotemporal behavior can also be learned with a single fixed output (just make $T = 1$ in Eq. (5)), with many intermediate samples one can also specify how the wave propagates, e.g. one can specify if the pyramid has be grow from left to right rather than bottom up.

[Figure 5 about here.]

Concerning autowaves, initial conditions are an important aspect to consider when trying to generate this kind of behavior. Here, when training autowaves in 2-layer systems, the initial condition of the first layer will be given as an image and the image for the second layer it will be assumed to be the same image of the first layer inverted and shifted one or two pixels in the direction of the desired propagation. Observe that, this is not the only way for generating initial conditions for autowaves [17] but it is certainly a simple one.

The images in Fig. 3, which were produced by simulation, were used as training set for experiments with autowaves, since manual derivation of this kind of images could lead to impractical and physically infeasible behavior. Although the results shown in Fig. 6 were also obtained by performing the training in simulation, the final template was obtained without any prior knowledge of the original. As a result, the original and the resulting templates are not the same but have a similar structure.

[Figure 6 about here.]

Observe that there was a good generalization of outputs further in time that were not

used for the learning process. Another important remark to be made w.r.t. Fig. 6 is that the speed of the template execution changed. The speed of the evolution of the learned behavior is approximately 20% faster than the original one although no measures were taken with respect to this. One may assume that the learned template could also have randomly generated a behavior which was slower than the original one. Hence changes in speed of a given CNN operation are possible, without modifying time constants.

The following table presents the values of the final templates obtained by the learning procedure.

[Table 1 about here.]

6 Conclusion

In this paper the problem of learning 2-dimensional spatiotemporal behavior with cellular neural networks was approached. Trajectory learning with recurrent neural networks was used as a starting point for the formulation of the problem at hand. It was shown that although trajectory learning with RNNs and learning of spatiotemporal behavior with CNNs have many elements in common, two key points distinct the two problems: (a) due to the locality and space invariance of CNN weights, the number of parameters to be optimized in these networks is much smaller and thus learning is considerably easier favoring the use of global optimization methods to avoid the need of an expression for the gradient of the cost function; and (b) the

generation of an efficient training set for the CNN problem is not straightforward as in classical trajectory learning and thus customized solutions need to be devised. Taking into account these two points, a new cost function and methodology was proposed for learning of spatiotemporal behadifvior. This cost function assimilates time intervals also as parameters to be optimized and with that also reduce the importance of generating perfect training sets. This cost function also allows for the learning of the desired behavior at arbitrary speed of the dynamics. Results for the learning of different examples of spatiotemporal behavior were obtained with experiments made in simulations and CNN chip implementations. These results show that qualitatively good operations can be learned in the simulator and on-chip. Another important feature of the cost function proposed here is the possibility to modify the speed of an operation. A deeper exploration of this feature is ongoing research but it is believed that existing CNN applications can benefit from faster execution by pushing the template operations to its speed limit.

Acknowledgments Research supported by: • Research Council KUL: GOA-Mefisto 666, GOA-AMBioRICS, BOF OT/03/12, several PhD/postdoc & fellow grants; • Flemish Government: ◦ FWO: PhD/postdoc grants, G.0407.02, G.0080.01, G.0211.05, G.0499.04, G.0226.06, research communities (ICCoS, ANMMM); ◦ IWT: PhD Grants, ◦ Tournesol 2005 • Belgian Federal Science Policy Office IUAP P5/22 ('Dynamical Systems and Control: Computation, Identification and Modelling'). J. Suykens and J. Vandewalle are an associate and full professor with K.U. Leuven Belgium, respectively.

References

[1] B. Shi, P. Arena, and A. Zarándy, editors. *IEEE Trans. on Circuits and Systems-I - Special issue on CNN technology and active wave computing*, volume 51 (5). May 2004.

[2] T. Roska. Computational and computer complexity of analogic cellular wave computers. *J. Circuits, Systems, and Computers*, 12(4):539–56, 2003.

[3] P. Földesy, L. Kék, Á. Zarándy, and G. Bártfai T. Roska. Fault-Tolerant Design of Analogic CNN Templates and Algorithms—Part I: The Binary Output Case. *IEEE Trans. on Circuits and Systems-I*, 46(2):312–322, February 1999.

[4] S. Xavier-de-Souza, M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle. Toward CNN Chip-specific robustness. *IEEE Trans. on Circuits and Systems-I*, 51(5):892–902, May 2004.

[5] Werbos P. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), 1990.

[6] Narendra K. S. and Parthasarathy K. Gradient methods for the optimization of dynamical systems containing neural networks. *IEEE Trans. Neural Networks*, 2(2):252–262, 1991.

[7] M. Galicki, L. Leistritz, E. B. Zwick, and H. Witte. Improving generalization capabilities of dynamic neural networks. *Neural Computation*, 16(6):1253–1282, Jun 2004.

[8] P. Zegers and M. K. Sundareshan. Trajectory generation and modulation using dynamic neural networks. *IEEE Trans. Neural Networks*, 14(3):520–533, may 2003.

[9] M. Bianchini, M. Gori, and M. Maggini. On the problem of local minima in recurrent neural networks. *IEEE Trans. Neural Networks*, 5(2):167–172, Mar 1994.

[10] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, Mar 1994.

[11] B. Cohen, D. Saad, and E. Marom. Efficient training of recurrent neural network with time delays. *Neural Networks*, 10(1):51–59, Jan 1997.

[12] P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans. Neural Networks*, 5(1):54–65, Jan 1994.

[13] M. K. Sundareshan and T. A. Condarcure. Recurrent neural-network training by a learning automaton approach for trajectory learning and control system design. *IEEE Transactions on Neural Networks*, 9(3):354–368, Jan 1998.

[14] M. Galicki, L. Leistritz, and H. Witte. Learning continuous trajectories in recurrent neural networks with time-dependent weights. *IEEE Trans. Neural Networks*, 10(4):741–756, Jul 1999.

[15] L. Leistritz, M. Galicki, H. Witte, and E. Kochs. Training trajectories by continuous recurrent multilayer networks. *IEEE Trans. Neural Networks*, 13(2):283–291, Mar 2002.

[16] B. A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks - a survey. *IEEE Trans. Neural Networks*, 6(5):1212–1228, Sep 1995.

[17] P. Arena, S. Baglio, L. Fortuna, and G. Manganaro. Self-organization in a two-layer CNN. *IEEE Trans. on Circuits and Systems-I*, 45(2):157–162, Feb 1998.

[18] R. Carmona-Galan, F. Jimenez-Garrido, C.M. Dominguez-Mata, R. Dominguez-Castro, S.E. Meana, I. Petras, and A. Rodriguez-Vazquez. Second-order neural core for bioinspired focal-plane dynamic image processing in cmos. *IEEE Trans. on Circuits and Systems-I*, 51(5):913–925, May 2004.

[19] L. O. Chua, T. Roska, T. Kozek, and Á. Zardy. CNN Universal chips crank up computing power. *IEEE Circuits and Devices*, 12(4):18–28, 1996.

[20] S. Xavier-de-Souza, J. A. K. Suykens, and J. Vandewalle. Real-time tracking algorithm with locking on a given object for VLSI CNN-UM implementations. In *Proceedings of IEEE Int. Workshop on Cellular Neural Networks and their applications*, pages 291–296, Budapest, Hungary, Sep 2004.

[21] V. Pérez-Muñuzuri, V. Pérez-Villar, and L. O. Chua. Autowaves for image processing on a two-dimensional cnn array of excitable nonlinear circuits: flat

and wrinkled labyrinths. *IEEE Trans. on Circuits and Systems-I*, 40(3):174–181, Mar 1993.

[22] A. Adamatzky, P. Arena, A. Basile, R. Carmona-Galan, B.D.L. Costello, L. Fortuna, M. Frasca, and A. Rodriguez-Vazquez. Reaction-diffusion navigation robot control: from chemical to VLSI analogic processors. *IEEE Trans. on Circuits and Systems-I*, 51(5):926–938, May 2004.

[23] A.P. Munuzuri, V. Perez-Munuzuri, M. Gomez-Gesteira, L.O. Chua, and V. Perez-Villar. Spatiotemporal structures in discretely-coupled arrays of nonlinear circuits: a review. *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, 5(1):17–50, Feb 1995.

[24] T. Roska, L. O. Chua, D. Wolf, T. Kozek, R. Tetzlaff, and F. Puffer. Simulating nonlinear waves and partial differential equations via CNN—part I: Basic techniques. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 42(10):807–815, Oct 1995.

[25] M. E. Yalcin, J. A. K. Suykens, and Vandewalle J. Spatiotemporal pattern formation in the ACE16k CNN chip. In *IEEE International Symposium on Circuits and Systsems (ISCAS 2005)*, pages 5814–5817, Kope, Japan, May 2005.

[26] A. Rodríguez-Vázquez, G. Liñán Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Gálan, F. Jiménez-Garrido, R. Domínguez-Castro, and S.E.

Meana. ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs. *IEEE Trans. on Circuits and Systems-I*, 51(5):851–863, May 2004.

[27] L. Ingber. Adaptive simulated annealing (ASA): lessons learned. *J. Control and Cybernetics*, 25(1):33–54, 1996.

[28] G. Liñán, S. Espejo, R. Domínguez-Castro, and A. Rodríguez-Vázquez. ACE4k: An analog I/O 64x64 visual microprocessor chip with 7-bit analog accuracy. *Int. J. of Circuit Theory and Applications*, 30(2-3):89–116, 2002.

List of Figures

1 Initial conditions and desired snapshots in time of: (a) a dot traveling across the image at a given angle; and (b) a wave of given width propagating across the image in a specific direction. 27

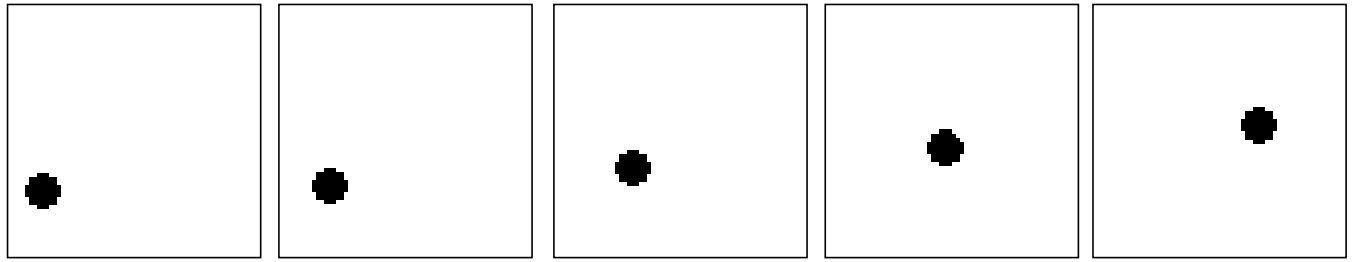
2 Initial conditions and desired snapshots in time of: (a) a wave that propagates to form a horizontal pyramid and a pyramid in a given angle and shape; and (b) a combustion wave which burns from the center to the borders of the image. 28

3 Spiral autowave phenomena in a 2-layer CNN simulator: initial conditions and time snapshots of the output in the two layers. Each row represents one layer. 29

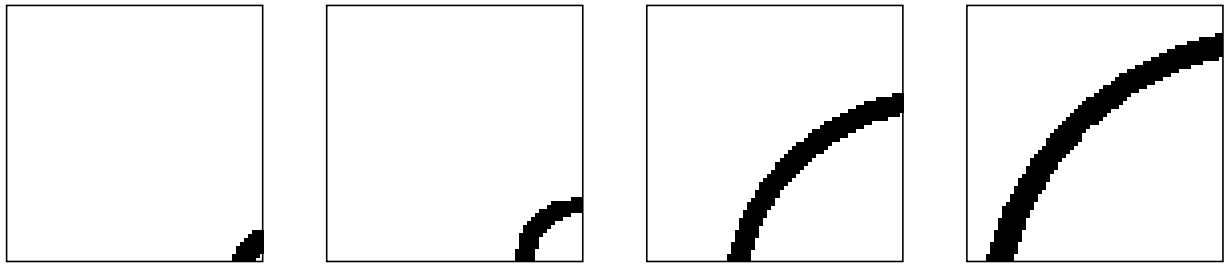
4 Results for on-chip learning in the ACE4k (64×64) of two CNN templates: (a) a traveling dot at a given angle; (b) a wave traveling in a specific direction. 30

5 Results of on-chip learning in the ACE4k (64×64) of: (a) "convergent" waves that form a pyramid; and (b) a "divergent" wave that models a combustion wave (b). 31

6 Spiral autowave that was learned from the three first columns of Fig. 3. (a) Trained behavior; and (b) generalized behavior further in time. 32

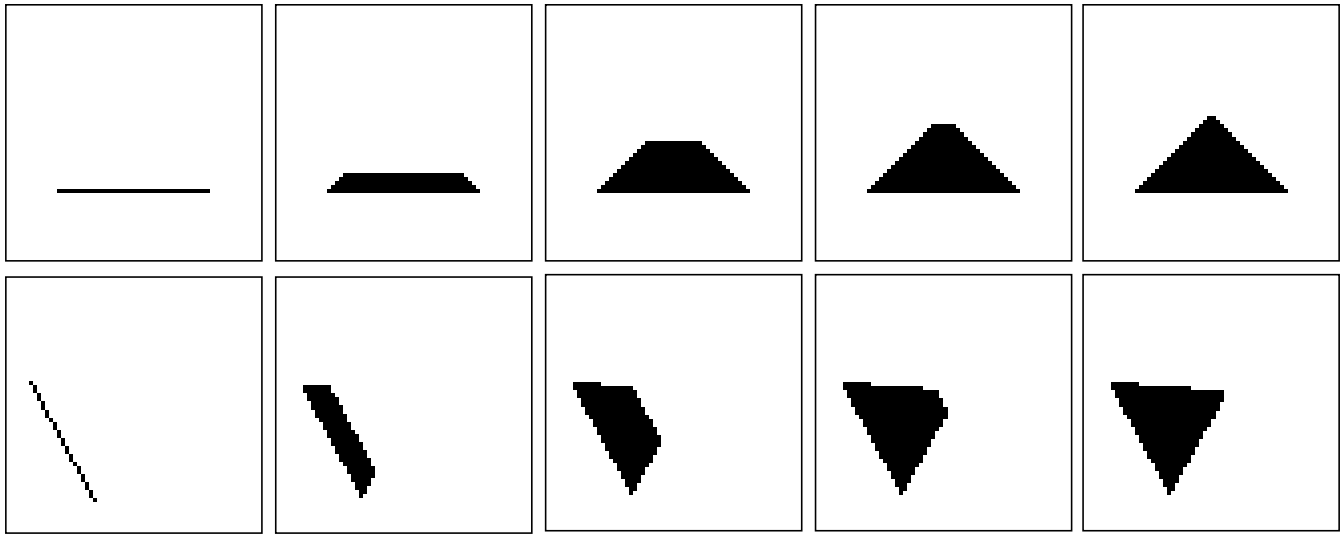


(a)

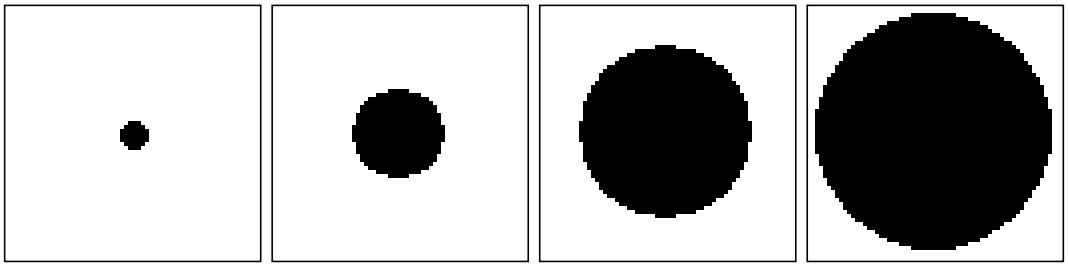


(b)

Figure 1: Initial conditions and desired snapshots in time of: (a) a dot traveling across the image at a given angle; and (b) a wave of given width propagating across the image in a specific direction.



(a)



(b)

Figure 2: Initial conditions and desired snapshots in time of: (a) a wave that propagates to form a horizontal pyramid and a pyramid in a given angle and shape; and (b) a combustion wave which burns from the center to the borders of the image.

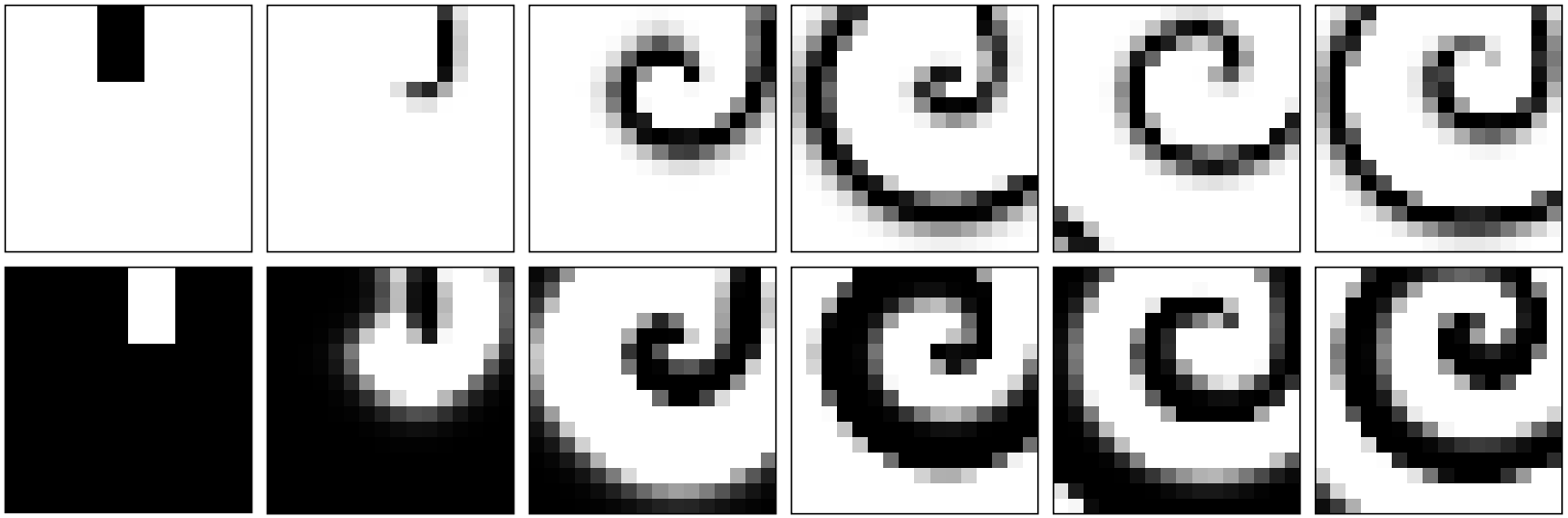
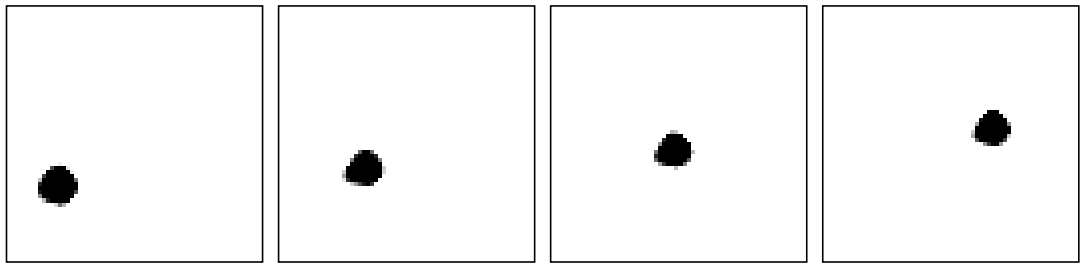
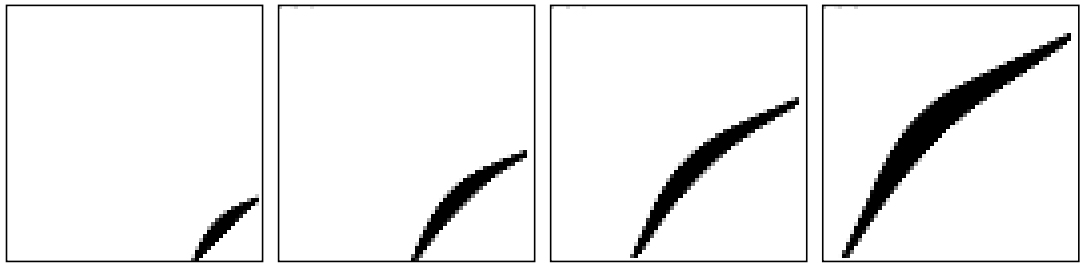


Figure 3: Spiral autowave phenomena in a 2-layer CNN simulator: initial conditions and time snapshots of the output in the two layers. Each row represents one layer.

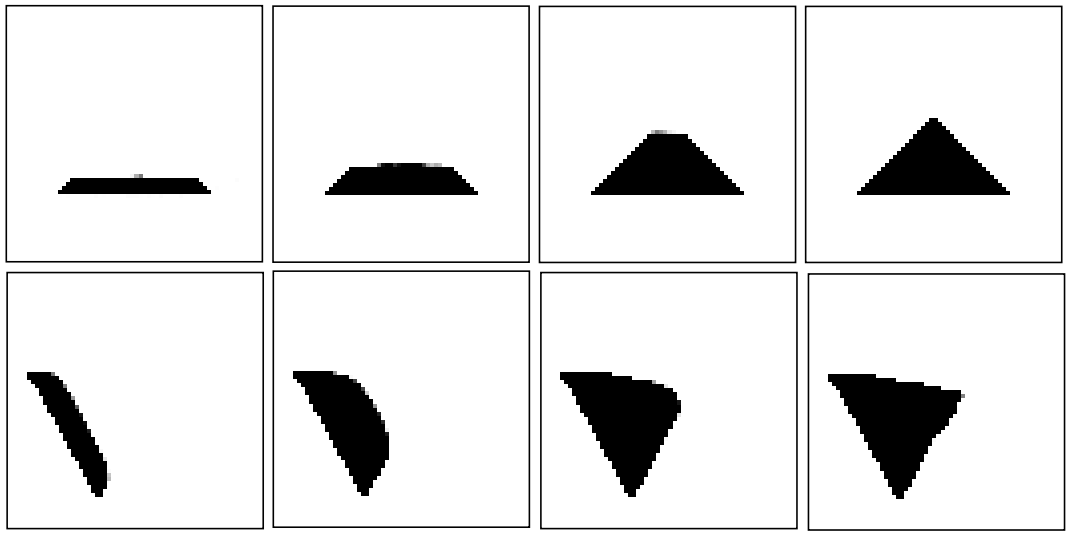


(a)

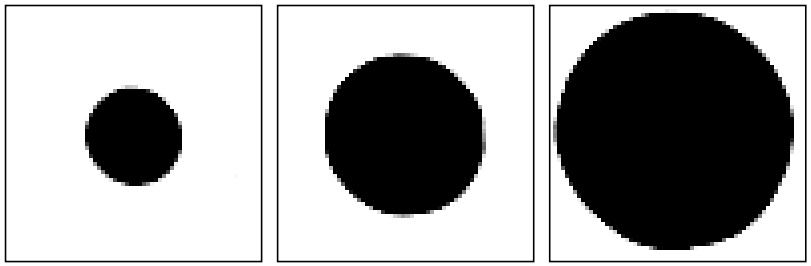


(b)

Figure 4: Results for on-chip learning in the ACE4k (64×64) of two CNN templates: (a) a traveling dot at a given angle; (b) a wave traveling in a specific direction.

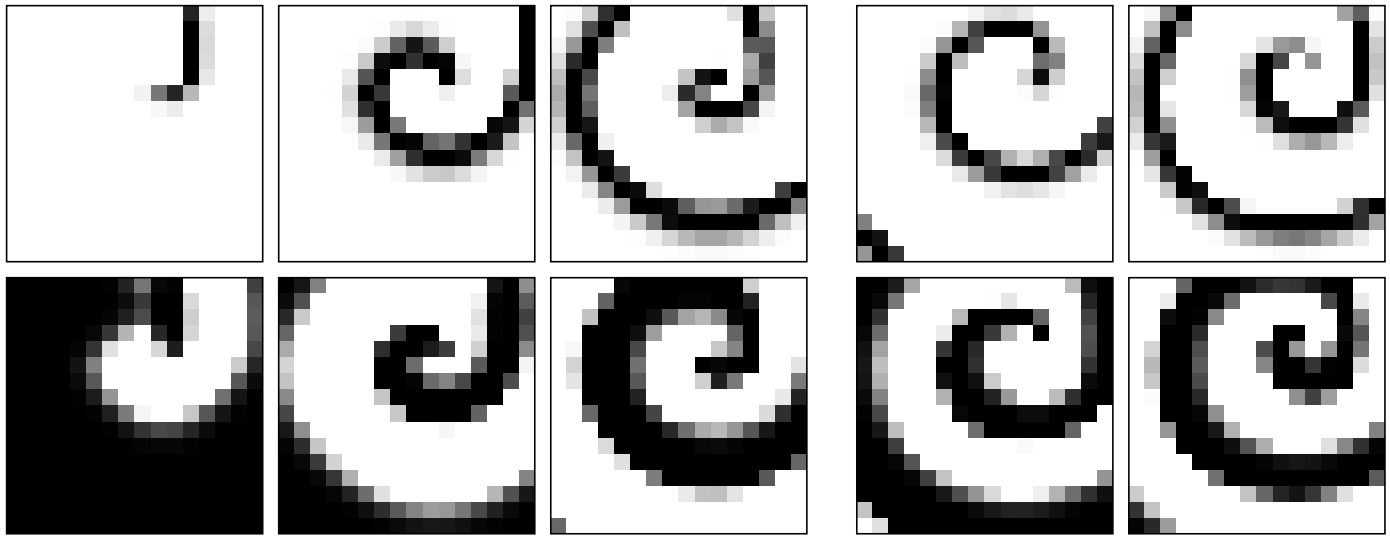


(a)



(b)

Figure 5: Results of on-chip learning in the ACE4k (64×64) of: (a) "convergent" waves that form a pyramid; and (b) a "divergent" wave that models a combustion wave (b).



(a) outputs required to be learned

(b) generalization of the evolution

Figure 6: Spiral autowave that was learned from the three first columns of Fig. 3. (a) Trained behavior; and (b) generalized behavior further in time.

List of Tables

1 Resulting template values after the learning process for different spatiotemporal behaviors. 34

Operation	Fig.	Feedback Matrices	Bias Term
Traveling Dot	4(a)	$A = \begin{bmatrix} 1.42 & 1.45 & 0.15 \\ 2.11 & -0.29 & -1.27 \\ 2.18 & 1.71 & 1.60 \end{bmatrix}$	$z = 0.21$
Traveling Wave	4(b)	$A = \begin{bmatrix} -2.96 & 1.19 & 0.76 \\ 0.94 & -1.02 & 1.96 \\ 0.78 & 1.75 & -1.57 \end{bmatrix}$	$z = 0.12$
Vertical Piramid	5(a)	$A = \begin{bmatrix} 2.72 & -1.51 & -0.15 \\ 1.29 & 2.71 & 2.37 \\ 2.96 & -1.08 & 2.50 \end{bmatrix}$	$z = 1.69$
Arbitrary Piramid	5(a)	$A = \begin{bmatrix} 1.11 & -0.44 & -1.44 \\ 0.59 & 2.89 & 2.54 \\ 0.59 & 1.40 & -0.70 \end{bmatrix}$	$z = 1.43$
Combustion Wave	5(b)	$A = \begin{bmatrix} 1.31 & 2.53 & 1.16 \\ 2.30 & -2.90 & 2.34 \\ 0.90 & 2.05 & 1.23 \end{bmatrix}$	$z = 3.99$
Spiral Wave	6	$A_{11} = \begin{bmatrix} -0.0019 & 0.1182 & -0.0019 \\ 0.1182 & 1.4921 & 0.1182 \\ -0.0019 & 0.1182 & -0.0019 \end{bmatrix}$ $A_{12} = -1.3845$ $A_{22} = \begin{bmatrix} -0.0019 & 0.1182 & -0.0019 \\ 0.1182 & 1.4921 & 0.1182 \\ -0.0019 & 0.1182 & -0.0019 \end{bmatrix}$ $A_{21} = 1.3845$	$z_1 = -0.4298$ $z_2 = 0.4298$

Table 1: Resulting template values after the learning process for different spatiotemporal behaviors.