Self-Organizing Maps Based on Limit Cycle Attractors

Di-Wei Huang[*,a], Rodolphe J. Gentili[b,c,d], and James A. Reggia[a,c,d,e]

[a] *Department of Computer Science,*
[b] *Department of Kinesiology,*
[c] *Neuroscience and Cognitive Science Program,*
[d] *Maryland Robotics Center,*
[e] *University of Maryland Institute for Advanced Computer Studies,*
University of Maryland, College Park, MD 20742, United States

Abstract

Recent efforts to develop large-scale brain and neurocognitive architectures have paid relatively little attention to the use of self-organizing maps (SOMs). Part of the reason for this is that most conventional SOMs use a static encoding representation: each input pattern or sequence is effectively represented as a fixed point activation pattern in the map layer, something that is inconsistent with the rhythmic oscillatory activity observed in the brain. Here we develop and study an alternative encoding scheme that instead uses sparsely-coded limit cycles to represent external input patterns/sequences. We establish conditions under which learned limit cycle representations arise reliably and dominate the dynamics in a SOM. These limit cycles tend to be relatively unique for different inputs, robust to perturbations, and fairly insensitive to timing. In spite of the continually changing activity in the map layer when a limit cycle representation is used, map formation continues to occur reliably. In a two-SOM architecture where each SOM represents a different sensory modality, we also show that after learning, limit cycles in one SOM can correctly evoke corresponding limit cycles in the other, and thus there is the potential for multi-SOM systems using limit cycles to work effectively as hetero-associative memories. While the results presented here are only first steps, they establish the viability of SOM models based on limit cycle activity patterns, and suggest that such models merit further study.

Keywords: self-organizing maps, limit cycle attractors, oscillatory dynamics, multi-self-organizing map architectures

[*]Corresponding author.
Email addresses: dwh@cs.umd.edu (Di-Wei Huang), rodolphe@umd.edu (Rodolphe Gentili), reggia@cs.umd.edu (James Reggia)

1 Introduction

Self-organizing maps (SOMs) are artificial neural networks initially inspired by maps in biological neural systems [20, 23]. They learn, using unsupervised methods, to project vectors in a high-dimensional input space onto a low-dimensional (usually 2D) lattice of artificial neurons. This projection preserves topological relationships of the input space — nearby neurons in a trained map are typically sensitive to similar input patterns, although sudden jumps may also occur. SOMs are important neuocomputational tools in part due to their frequent use for clustering and visualizing complex high-dimensional data that may otherwise be difficult to understand [15, 20, 24]. They have been applied in a wide range of fields, such as robotic control, weather monitoring, genome analysis, and economic modeling, to name just a few examples. At the same time, SOMs are also important as brain models. Sensorimotor maps are a common phenomenon in the brain, most famously in the neocortex, and SOMs have been successful in modeling many biologically observed cortical properties, such as self-organization of somatosensory and visual cortical regions [27, 35], the alignment of multiple feature maps [7], the formation of mirror-symmetric topographic maps [36], and related cognitive phenomena [2].

The importance of SOM models for understanding brain function is likely to increase substantially over the next decade as work on developing large-scale brain and neurocognitive models accelerates. By *large-scale models* we mean recent and ongoing research efforts to create neuroanatomically grounded simulations of all or major portions of human/mammalian brain structure and function, or at least major subsystems of the brain that span multiple cortical regions. These models vary from extremely large networks of biologically-realistic spiking neurons to those that are more abstract, based on a higher level of components such as cortical columns, or are focused on simultaneously supporting cognitive functions [9, 10, 39, 41]. Work in this area is increasing in part due to major recent research funding initiatives (the Human Brain Project in Europe, the BRAIN Initiative in the US, etc. [1]).

Self-organizing maps have received relatively little attention so far in large-scale brain/neurocognitive models, something that is surprising given the tremendous interest in neuroscience in "mapping the brain." Part of the reason for this is that, in spite of their computational successes, most conventional SOM models are substantially limited from the viewpoint of neuroscience — they face several

significant barriers to more widespread use in multi-region brain models. Fortunately, work during recent years has been addressing some of these issues. For example, many SOMs follow Kohonen [20] in having a computationally-efficient single "winner" node for each input pattern, something that is inconsistent with widespread distributed activity over biological cortical regions, limits an N-node network to representing only N distinct activation patterns, and requires an implausible (from a biological perspective) global selection of a "winner" cortical node whose incoming weights best match the current input pattern. A variety of approaches have been taken in addressing this limitation, such as the use of softmax functions [17], allowing non-winner nodes to retain non-zero activation levels [19, 22, 38], allowing a SOMs' output to be the weights or location of a winner node [8, 14, 26, 34], running SOMs as dynamical systems to produce multiple winners [23, 27], or supporting one-step multi-winner activation patterns [33]. The spatial activation patterns in multi-winner architectures are potentially much richer in expressiveness than with single-winner SOMs, so we adopt multi-winner maps in our work described below.

A second barrier to adopting SOMs in large-scale brain/neurocognitive models is that many SOMs do not address temporal sequence processing. SOM models often take a single fixed pattern as input, and when the next input is received, effectively reset the map region before processing the following input, with the order of inputs in each sequence thus being largely ignored. In contrast, biological systems receive a continuing stream of input patterns, and even if these patterns are discretized, often their order is critically significant (e.g., phonemes forming a word). To date, only a limited number of SOMs have been developed for processing temporal sequences. These work in a variety of ways, for example by superimposing a map's activation pattern from one input pattern with decayed activation patterns from previous inputs [5, 6, 16, 21], by allowing activity to propagate laterally as each input pattern is processed [11, 40], or by transforming previous activation patterns based on a separate set of weights [8, 14, 26, 34, 38].

A third barrier to adopting SOMs in large-scale brain models, and the one most central to the work reported below, is that most past SOMs have used a *static representation* of information. By this we mean that each input pattern or sequence of patterns is typically represented by a single fixed activation pattern over the map layer, without considering how that activation pattern evolves with time. In conventional SOMs, activation patterns are driven directly by fixed input stimuli, and thus remain static until external control mechanisms alter or reset the map layer. This remains true even

in SOMs for sequence processing: even though the map regions have changing activation patterns over time, the representation that encodes a whole input sequence is still a single spatial activity pattern that is chosen from the series of changing activation patterns (e.g., the activation pattern that occurs in response to the last element of an input sequence [33]). A sequence's representation is therefore still static. Importantly, such a static representation is typically transient — it occurs in a SOM for a very short time in-between updates of the SOM's activation. For a downstream neural component to access a transient representation like this, it has to either implement a precise timing control or artificially "freeze" the SOM's activation state. In both cases, a priori knowledge about the timing of the transient representation is required.

In the context of building large-scale brain architectures, using static representations for SOMs can introduce both substantial overhead and biologically-implausible processing. For example, biological neural systems generally do not exist in static activation states, and they are unlikely to operate solely based on fixed spatial patterns. On the contrary, biological systems are clearly characterized by prominent ongoing rhythmic oscillatory activity [4, 29], and there is substantial evidence that cognitive functions such as memory are strongly related to this rhythmic oscillation of neural activities [12]. The oscillatory nature of this ongoing activity has rarely been accounted for in past work on SOMs, although there are exceptions. In Kaipainen & Ilmonen [18], a SOM with explicit phase variables for each node is used to detect the periods of oscillatory input stimuli, while in Rumbell et al. [32], a conventional SOM is based on spiking neurons that exhibit periodic firing behaviors. However, the oscillatory activation in both of these studies is a direct result of the input stimuli being periodic, not of the map layer's activation dynamics. It is unclear if they would exhibit any oscillatory activation at all should the input become aperiodic or unavailable, and the issue of whether oscillations would continue or what they would represent if they did occur is not considered.

Motivated by the above issues, here we investigate a new *dynamic representation* for use in SOMs that involves not only spatial patterns but also temporal extension in the form of map limit cycle attractors. To our knowledge, this is the first study of SOMs that encode external stimuli using activity limit cycles. Each input pattern/sequence becomes encoded by a short limit cycle attractor in the state space of a multi-winner SOM, regardless of whether input stimuli form temporal sequences or are static spatial patterns. The limit cycles obtained are self-sustained and

persist after the input that triggered them terminates, and they can therefore be viewed as a candidate representation for working memory. The length of a learned limit cycle representation is found to generally not be the same as that of the corresponding input sequence that it represents. Further, these representations can be accessed at relatively arbitrary times compared with using static representations. As shown later, they are also more resistant to noise and more unique when compared with static representations.

Computational experiments are used to address several questions concerning the proposed use of limit cycle representations in SOMs. First, since it is unclear a priori, we establish that map formation still occurs over a lattice, and does so robustly, in the context of persistent limit cycle activity. Second, for SOMs to be useful in large-scale brain models involving multiple cortical regions, these SOMs need to be able to interact with each other and exchange useful information. It is not obvious in advance that this can be achieved effectively in systems of SOMs using limit cycle representations. We therefore study two possible ways of applying limit cycle representations in architectures containing two interacting SOMs where each SOM encodes external stimuli for a distinct modality (auditory and visual modalities). The task in these latter experiments is to establish appropriate associations between the corresponding representations of the two SOMs, such that when the input in one modality is absent, the representation for that absent input can be restored automatically by using input via the other modality alone. In one situation, associations between limit cycle representations and static ones are established. This shows that SOMs using our dynamic representation method are compatible with those using existing static representations. In the other situation, pairs of distinct limit cycles are associated, showing that it is possible for a multi-map architecture to operate solely based on limit cycle representations.

The rest of this paper is organized as follows. Section 2 describes our limit cycle SOM model as well as the training and experimental methods. Section 3 characterizes different types of dynamic representations (including limit cycles), and compares them with static representations. In Section 4, a sample architecture associating limit cycle and static representations is studied. The results are compared with a control experiment that associates pairs of static representations. In Section 5, a modified architecture is introduced to associate pairs of limit cycle representations. Finally, Section 6 concludes by examining the implications of this work.

2 Methods

One-shot multi-winner SOMs with locally recurrent feedback connections, modified from Schulz & Reggia [33] so that they now support continuous post-stimulus activation dynamics, are used in this study. A key difference that separates this study from other work is that SOMs with feedback connections are allowed to continue running and adapting after input sequences are over. The modified SOMs generate sparsely-coded activation patterns with multiple simultaneous node activations, and retain time-varying activity indefinitely after external input ends.

The artificial neurons (nodes) in a SOM are organized in a 2D rectangular grid. The neighborhood of a node i is the set of nodes that are located within a radius r of the node i. Box distances are measured between nodes on the grid. Formally, let $N_{i,r}$ denote the neighborhood of a node i within radius r:

$$N_{i,r} = \{k \mid k \neq i \text{ and } d(i,k) \leq r\}\,, \tag{1}$$

$$d(i,k) = \max\left(|row_i - row_k|, |column_i - column_k|\right). \tag{2}$$

Here we use a planar grid to model a SOM, as opposed to using a toroidal configuration in which boundaries of a grid "wrap around" to the opposite side of the grid. Although the latter has been adopted by many SOMs to avoid discontinuity around grid boundaries, it lacks biological grounding. Further, our preliminary simulations indicated that a planar grid tends to result in more unique activation sequences.

In the computational experiments that follow, we study models having one or more SOMs, the latter generally involving inter-connected map regions. If more than one SOM is present in a multi-SOM architecture, each SOM receives streams of input via synaptic connections from one or more upstream components, such as external inputs, other SOMs, and/or itself (i.e., via recurrent feedback connections). The set of connections coming from the same upstream component is called a "channel". The set of connections for a channel can be full or topographic. Full connections are often used for connecting a non-SOM component to a SOM, where a link exists between every pair of upstream and downstream neurons. Topographic connections are used for connecting two SOMs (including direct feedback connections in a SOM), where a node i in the downstream SOM

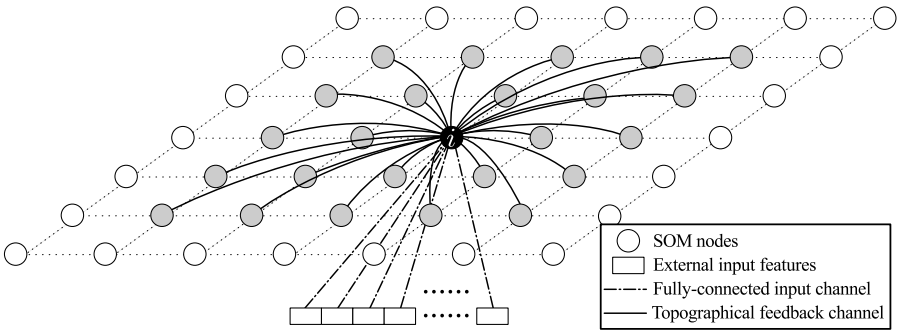

Figure 1: A small example 7×7 SOM for illustrative purposes. The SOM has two channels: a fully-connected afferent channel and a topographic recurrent channel (radius 2). For readability, only a small number of nodes are illustrated and incoming links for only node i are drawn. SOMs similar to this one serve as a basic component for the experiments in the following sections.

receives connections only from the neighborhood of its corresponding node i' in the upstream SOM ($row_i = row_{i'}$, $column_i = column_{i'}$). Figure 1 shows a simple example of a SOM whose structure resembles those in the experiments in this study. It has two channels. The fully-connected channel provides afferent input, and the topographic recurrent channel provides time-delayed feedback. The recurrently connected SOM shown here can be interconnected with other SOMs via additional channels, as occurs in some of the experiments described below.

Let $\boldsymbol{x}_j(t)$ denote the input vector that a SOM receives from channel j at time t. The net input for each node i receiving one or more channels j at time t is calculated as

$$in_i(t) = \sum_j \alpha_j \left(\boldsymbol{w}_{ij}^{\mathsf{T}} \boldsymbol{x}_j(t) \right), \tag{3}$$

where $\boldsymbol{w}_{ij}$ denotes the weight vector at node i for channel j, and the α_j's are constant parameters specifying the relative weighting among channels. For recurrent connections, the input is delayed by one time step, i.e., $\boldsymbol{x}_j(t) = \boldsymbol{a}(t-1)$, where $\boldsymbol{a}(t)$ is the activation vector (output vector) of the SOM. It is assumed the activation is initially zero, i.e., $\boldsymbol{a}(t=-1) = \boldsymbol{0}$. A one-shot multi-winner activation rule is used to determine $\boldsymbol{a}(t)$ for $t \geq 0$, where each $a_i(t) \in [0,1]$ is calculated as:

$$winners(t) = \{k \mid in_k > in_l, \forall l \in N_{k,r^{\text{comp}}}\}, \tag{4}$$

$$a_i(t) = \min\left(1, \textstyle\sum_{k \in N_i \cap winners(t)} \gamma^{d(i,k)}\right). \tag{5}$$

Winners are nodes that have the highest net input in their local competition neighborhood (radius r^{comp}). The activation level for each node is determined by how close it is to the winner nodes. The winners themselves are maximally activated, while their surrounding nodes have lower activation levels based on their distance from the winners. Together each winner node and its neighbors form a peak of activation centered at the winner. The shape of the peaks depends on the decay parameter γ ($0 \leq \gamma < 1$). A smaller γ makes steeper peaks.

The weight adaptation for each afferent channel j follows competitive Hebbian learning:

$$\hat{\boldsymbol{w}_{ij}}(t+1) = \boldsymbol{w}_{ij}(t) + \mu_j a_i(t) \boldsymbol{x}_j(t), \tag{6}$$

$$\boldsymbol{w}_{ij}(t+1) = \hat{\boldsymbol{w}_{ij}}(t+1) / ||\hat{\boldsymbol{w}_{ij}}(t+1)||_2, \tag{7}$$

where μ_j is the learning rate.

Each direct feedback channel, such as the recurrent connections in Figure 1, is adapted using temporally asymmetric Hebbian learning [33], which is based on biological evidence that the efficacy of a synapse is strengthened if presynaptic firing precedes the postsynaptic firing in a 20 to 50 ms time window [3, 25, 42]. The weight value $w_{ijk} \in \boldsymbol{w}_{ij}$ for each connection from node k to node i via channel j is updated as:

$$\hat{w_{ijk}}(t+1) = w_{ijk}(t) + \mu_j a_k(t-1) \max\left(0, a_i(t) - a_i(t-1)\right), \tag{8}$$

$$w_{ijk}(t+1) = \hat{w_{ijk}}(t+1) / \textstyle\sum_l \hat{w_{ijl}}(t+1). \tag{9}$$

The term $\max\left(0, a_i(t) - a_i(t-1)\right)$ specifies that the increase, rather than the value, of the activation level is taken into consideration. Note that if each w_{iji} (a self-link that is originated and terminated at the same node i) was not treated differently form non-self-links, it would be strengthened during learning whenever a_i increases and soon dominate $\boldsymbol{w}_{ij}$. For this reason, each self-link w_{iji} is fixed to a constant parameter β.

The data set that we use to train and test SOMs consists of sets of input stimuli, each representing a different modality. The input data entries are varying-length temporal sequences of spatial patterns, including the special case of length 1. The entries that describe the same event in different modalities are associated, i.e., presented simultaneously. In this study, we use the same

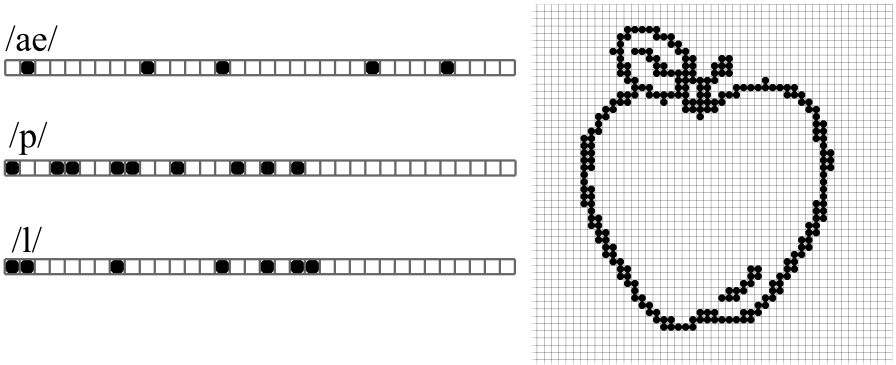

Figure 2: Sample data. Left: a phoneme sequence for the word "apple". The 34-bit input patterns corresponding to each phoneme are shown (black = 1, white = 0). Right: the corresponding bitmap image of an apple.

data set as in Weems & Reggia [39], which describes 50 objects using two modalities: English phoneme sequences (auditory words) and bitmap images (visual) of corresponding objects. Let $P = \{P_1, P_2, ...\}$ be the set of phonetic stimuli, where each P_k is itself a temporal sequence of 2 to 9 phonemes that is the name of an object, e.g., P_{apple} ="/ae/, /p/, /l/" for apple (see Figure 2). Each phoneme is encoded by a binary vector of 34 distinctive features. Similarly, let $V = \{V_1, V_2, ...\}$ be the set of corresponding pictorial stimuli, where each V_k is a single bitmap image (50×50) of the object that P_k names, e.g., the image of an apple (Figure 2). The associated entries are organized in tuples, such as $\langle P_k, V_k \rangle$, and the stimuli in the same tuple are always presented concurrently during learning (the sequence of phonemes in P_k serve as inputs, one at a time, as the corresponding image V_k is held fixed as the visual inputs).

A SOM-based architecture is trained for a number of epochs. In each epoch, each tuple in the data set is presented to the architecture once in a random order. Suppose a SOM takes afferent input from the phoneme data P using channel j, and that the current tuple being presented is $\langle P_k, V_k \rangle$. Then the input vectors are received as

$$\boldsymbol{x}_j(t) = \begin{cases} P_k(t) & \text{if } 0 \leq t < |P_k|; \\ \mathbf{0} & \text{if } t \geq |P_k|. \end{cases} \tag{10}$$

where $|P_k|$ denotes the length of the temporal sequence P_k. The activation and adaptation are then performed accordingly. Learning occurs both while the temporal sequence P_k is being presented

Table 1: Parameters for nonlinearly decreasing functions used during training

z	z_{init}	z_{fin}	z_{infl}	z_σ
μ_j for afferent channels	0.44	0	0.4	0.0001
μ_j for direct feedback channels	0.62	0	0.8	0.04
γ	0.37	0	0.2	0.16

and afterwards, during which the activity of SOM evolves without afferent input. The learning rates μ_j and the peak parameter γ decrease nonlinearly as the epoch number progresses, according to the function $z = z_{\text{fin}} + (z_{\text{init}} - z_{\text{fin}})/(1 + \exp((\phi - z_{\text{infl}})/z_\sigma))$, where z is a parameter (μ_j or γ) and ϕ is the fraction of epochs completed. Relevant parameter values are listed in Table 1. This function simulates the widely-adopted two phase training often used with SOMs, namely a rough organization phase with large learning rates and neighborhood sizes, followed by a fine-tuning phase with small learning rates and neighborhood sizes.

A critical element in our approach is that SOMs with direct feedback connections are allowed to continue running after input sequences are over for another T time steps until $t = |P_k| + T$ (exclusive), where continuation time T is a variable. After $t = |P_k| + T$ steps, a SOM switches to the next input sequence and reset time $t = 0$. The same T value is always used for all input entries in the same pass of the data set. During the last T time steps for each data entry, activation and adaptation are performed as usual, although no external input is being provided. This results in the afferent weights being unchanged in this time period because $\boldsymbol{x}_j(t) = \mathbf{0}$ (see Eq. 6), while recurrent weights adaptation is continued. Notice that except at the beginning of each input sequence, the activation levels of a SOM are never reset to $\mathbf{0}$, so that the recurrent connections can learn the temporal relations between activation patterns in consecutive time steps. This method allows us to obtain and observe the on-going activation dynamics of SOMs.

After training, the activation dynamics naturally occurring in a SOM during the last T time steps are designated to be the dynamic representation that encodes the corresponding input stimuli. The peak parameter is set to $\gamma = 0$, i.e., all winners have activation level 1 and non-winners 0, and all learning rates $\mu_j = 0$. For large enough T (e.g., $T = 200$), the dynamics can be qualitatively classified as three types of attractors: fixed point, limit cycle, and "complex". Denote the dynamic representation encoding the stimuli $\langle P_k, V_k \rangle$ as R_k, which is a time-ordered list of map activity

spatial patterns. With a fixed point attractor, the dynamics eventually reach a state where further updating of the activation levels results in the same state. In this case, R_k contains only this particular fixed-point state. For limit cycles, R_k consists of an ordered list of periodically repeating distinct states. The number of states in R_k is the length of the limit cycle. Fixed points and limit cycles are both referred to as *simple attractors* in the following. Dynamics that do not show apparent regularity during the last T time steps are classified as being a *complex attractor* (this includes chaotic attractors and potentially very long limit cycles where no state has yet repeated). In this case, R_k contains the whole list of states occurring in the last T time steps.

In the following experiments, a representation R_k is considered successfully *recalled* in a SOM if the ordered list of states in R_k is accurately recreated in the SOM, possibly without it being provided with the original input training sequence.

For the sake of comparison, we also conduct control experiments with $T = 0$. In the control experiments, we simulate static representations by taking the single activation pattern occurring at the end of each input sequence to be that sequence's representation, as has been done in some other past studies [33]. Setting $T = 0$ means that during training, there is no additional adaptation time after each input sequence, and that during testing, no attractor dynamics are observed beyond the end of each input sequence. Instead, the static representation during testing is a single pattern, denoted as $R_k^{\text{static}} = \boldsymbol{a}(|P_k| - 1)$, if the SOM takes phonemes as input. A successful recall of R_k^{static} in a SOM requires recreating the only activation pattern in R_k^{static}.

3 Results: Characterizing Dynamic Representations

This first experiment aims to study basic properties of dynamic representations in SOMs. A priori, it is unknown what types of attractors will naturally emerge as a result of training, or if smooth maps like those in conventional SOMs can still be formed when using non-fixed point (limit cycle) dynamics. It is also unclear what the relative advantages are for different types of attractors to serve as representations, and how dynamic and static representations compare. As will be shown later, limit cycle attractors tend to be quite suitable to serve as representations.

Our results are obtained using a 30×20 SOM with two channels. The rectangular layout of the SOM, as opposed to a square one, is because of evidence that this helps to stabilize the

Table 2: Summary of SOM Parameters

Parameter	Meaning
r^{comp}	radius of local competitive neighborhoods (Eq. 4)
α_j	relative weighting for channel j when computing net input (Eq. 3)
β	fixed weight for same-node feedback connections
μ_j	learning rate for channel j (Eq. 6, 8)
γ	peak parameter determining the steepness of activation peaks (Eq. 5)
τ	post-stimulus continuation time T during training

learning process [20, p. 159]. The first channel receives input phoneme sequences via fully-connected connections, and the second forms direct feedback via topographic connections, providing much the same structure as in Figure 1 but with a large map region. For clarity, let the continuation time $T = \tau$ during training, where τ is an adjustable parameter. $T = 200$ is fixed after training. Table 2 summarizes the parameters and their meanings. The SOM is trained for 1000 epochs with different τ and β (the fixed weight of same-node feedback links) values, while the other parameters are fixed as: $r^{\text{comp}} = 2, \alpha_1 = 0.64, \alpha_2 = 0.36$. After training, the SOM is run one last time without adaptation, using the same data set and $T = 200$. Then both the static and the dynamic representations are recorded for analysis. The results reported below are based on 20 independent simulations per result with different initial weights (random). The error bars indicate one standard deviation.

3.1 Attractor Formation

The map region activity state attractors formed are largely dependent on two parameters: β, the constant same-node recurrent weight for each node, and τ, the value of post-input continuation time T during training.

Large $|\beta|$ values would dominate all other recurrent connections in the same channel, and marginalize their contributions in the attractor dynamics. As a result, the SOM would operate based on blind activation (for $\beta \gg 0$) or deactivation (for $\beta \ll 0$) of the current winners, rather than on self-organization of the adaptable recurrent weights. Small $|\beta|$ values, e.g., $\beta \in [-1, 1]$, on the other hand, make the contributions of the same-node recurrent weights comparable to the other adaptable recurrent weights. Figures 3(a)-(b) show the types of attractors, (c) the number

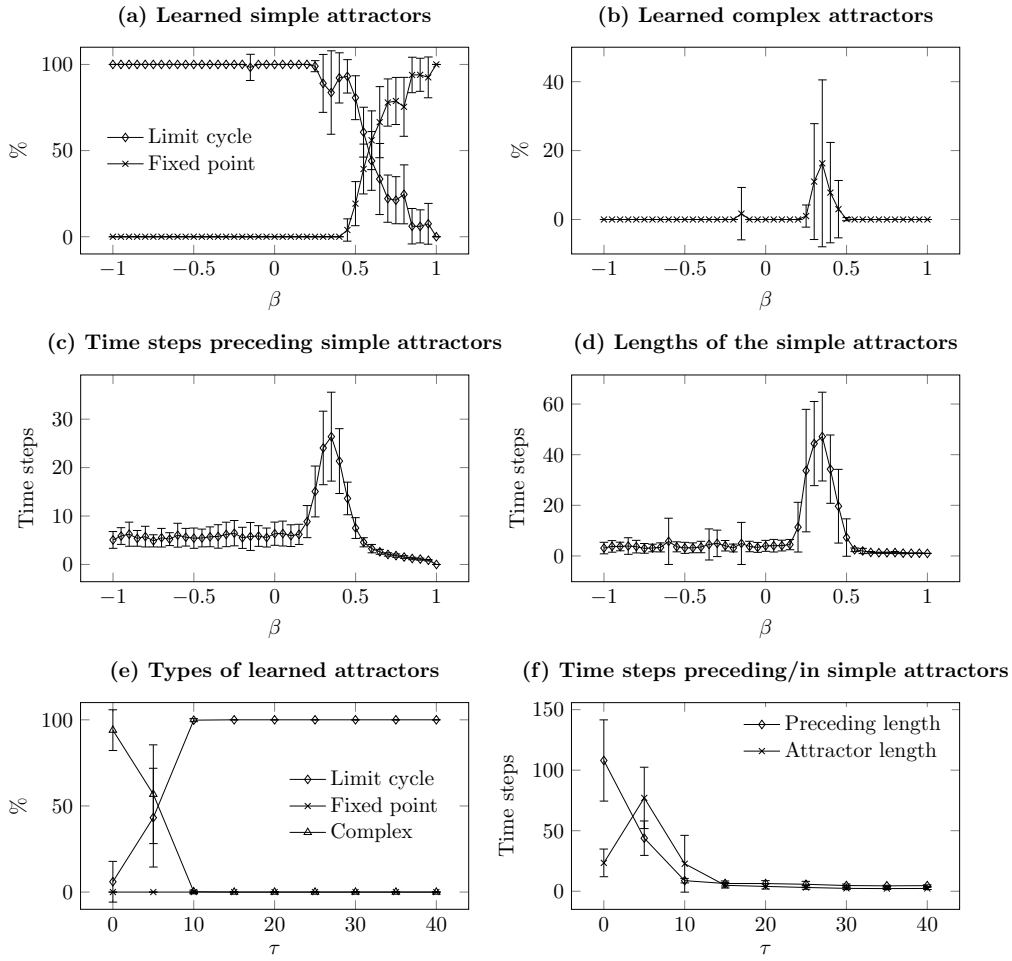

Figure 3: (a)-(d) show the effects of β (while τ is fixed at 20), and (e)-(f) the effects of τ (while β is fixed at 0), on the formation of attractors. Among all attractors formed after training, (a), (b), and (e) show the percentages that are fixed points, limit cycles, and complex attractors. For simple attractors (i.e., fixed points and limit cycles), (c) and (f) show the numbers of time steps it takes for the dynamics to settle in the attractors. (d) and (f) show the lengths of the simple attractors.

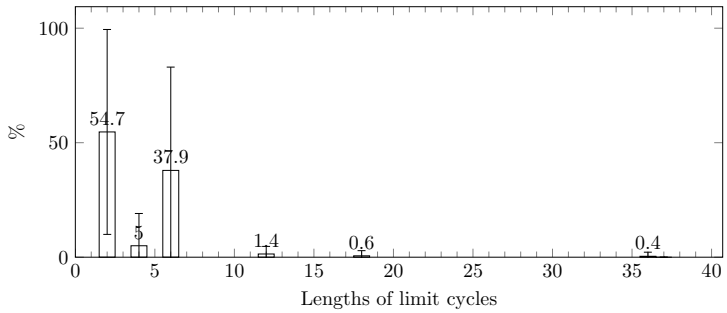

Figure 4: The length distribution of the limit cycles that are formed using $\beta = 0$ and $\tau = 20$.

of time steps preceding simple attractors, and (d) the lengths of simple attractors, that are formed using different β values. For most of the negative and small positive β, i.e., $\beta < 0.15$, nearly all are small limit cycles. In the range $0.15 < \beta < 0.5$, sizes of the limit cycles increase and a peak occurs at $\beta = 0.35$, at which point the dynamics also take longer to settle in the limit cycles. In the range $0.2 < \beta < 0.5$, complex attractors are also formed in addition to large cycles. As β increases beyond 0.5, fixed points soon take over.

Figures 3(e)-(f) show how τ affects attractor formation. At $\tau = 0$, which is what conventional training methods imply, most of the resulting attractors are complex. This is because the recurrent connections are under-trained for the situation where the SOM is run without external input. At $\tau = 5$, some complex attractors are reduced to large limit cycles. For $\tau > 15$, the SOM stably produces small limit cycles. A histogram of the cycle lengths are depicted in Figure 4. The cycle lengths tend to be multiples of 2 or 3.

To be useful for subsequent processing, learned representations need to be quickly reproducible, or retrievable, in SOMs. Retrieving a dynamic representations means restoring all constituent activation states. Therefore, complex attractors need to be avoided because their irregular activation makes them nearly irreproducible. Large cycles are undesirable as well because restoring them takes a longer time than small cycles or fixed points. Limit cycles that occur late are not preferred for the same reason. Therefore, according to the results shown in Figure 3, a feasible range of parameters lies in $\tau \geq 15$ and $\beta \in [-1, 0.15] \cup [0.5, 1]$. In the following subsections, this range will be narrowed down when stability and uniqueness are taken into consideration.

3.2 Attractor Stability

The robustness of dynamic representations directly depends on the stability of their attractors. To assess this stability, an activation state $\boldsymbol{b}$ in each attractor representation R_k is perturbed for all i as

$$b'_i = \begin{cases} b_i - \nu Z & \text{if } b_i = 1; \\ b_i + \nu Z & \text{if } b_i = 0, \end{cases} \tag{11}$$

where ν is the amplitude of perturbation and Z a uniform random number in $[0, 1)$. Then the SOM's initial activation is set to this perturbed pattern, $\boldsymbol{a}(t = -1) = \boldsymbol{b}'$, and run for $T = 200$ time

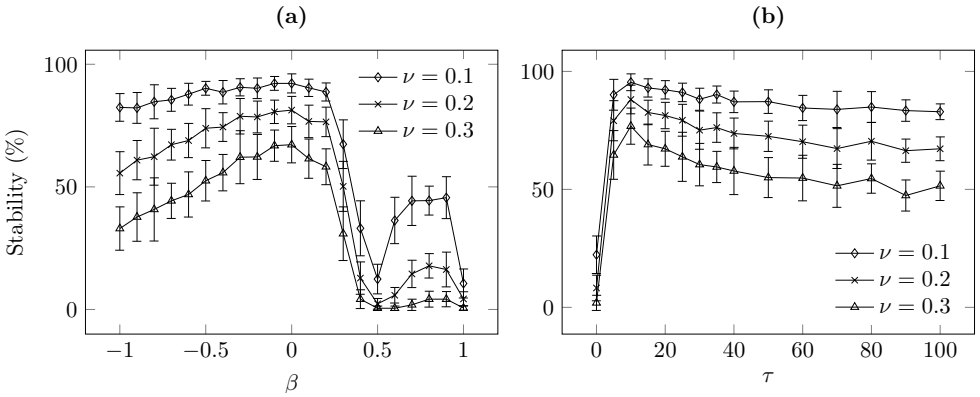

Figure 5: Stability of the attractors as measured by the percentages of attractors that are restored from random perturbation. ν denotes the amplitude of noise added to a state in each attractor. In (a), τ is fixed at 20; in (b), β is fixed at 0.

steps without external input. If the resulting dynamics successfully restore the attractor R_k, R_k is marked stable. The stability metric in Figure 5 is measured as the percentage in all attractors that are marked stable.

In Figure 5(a), stability reaches the highest value around $\beta = 0$. Larger β causes stability to drop dramatically, which coincides with the β range where large cycles, complex, and fixed point attractors are formed (see Figure 3). As β decreases from 0, although the attractors remain small cycles, stability drops gradually. In Figure 5(b), stability increases quickly as τ increases from 0 to 10, and gradually decreases as τ increases further. This indicates that prolonged post-input sequence training does not improve stability. Therefore, a preferred range of parameters is $\beta \in [-0.1, 0.1]$ and $\tau \in [15, 25]$.

3.3 Representation Uniqueness

Generally, uniqueness of representations is regarded as a desirable property for encoding a set of items (words, images, etc.). The more unique a representation is, the less likely its corresponding item is to be confused with other items in subsequent processing, such as in a classification task. The distinction between two conventional representations can be calculated as the distance between the two static states in a straightforward way. However, since our encoding method allows each representation to contain multiple states, the distance function must be generalized to accommodate

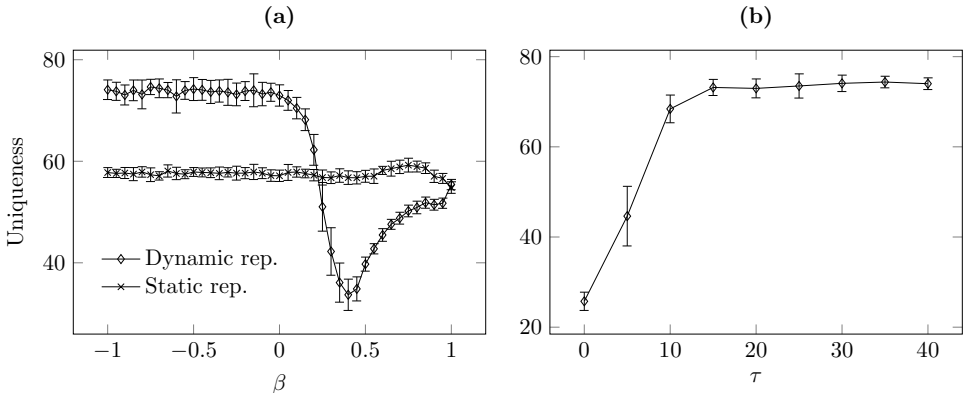

Figure 6: Uniqueness of the dynamic representations (attractors) as measured by the averages of the nearest distance among all combinations of the patterns in each pair of attractors. For comparison, the average distances of static representations are also shown in (a). Higher degrees of uniqueness reduce the chances that two representations are confused in subsequent processing. In (a), τ is fixed at 20; in (b), β is fixed at 0.

two sequences of states:

$$\mathrm{dist}(R_k, R_l) = \min_{\boldsymbol{p}\in R_k, \boldsymbol{q}\in R_l} \|\boldsymbol{p} - \boldsymbol{q}\|_1, \tag{12}$$

where $\boldsymbol{p}, \boldsymbol{q}$ are any constituent activation states in R_k, R_l, respectively. This distance reflects the minimum number of nodes whose activation must be inverted (i.e., changing 0 to 1 or vice versa) to cause one representation's attractor (e.g., a limit cycle attractor) to be converted into another's. It is a "conservative" lower bound on distances between two sequences; if it is larger than the distance between two static representations, one can be confident that the dynamic representations are more distinct in general. Note that this distance function also works for representations that contain one state, such as static representations or fixed point attractors. Using this distance function, the uniqueness metric can be defined as the average distance over all pairs of representations.

Figure 6 shows the uniqueness metric for different β and τ values. In Figure 6(a), the range of β that produces small limit cycles ($\beta < 0.15$) yields about 30% better uniqueness than static representations. For $\beta > 0.15$, uniqueness drops significantly for large cycle, complex, and fixed point attractors. In Figure 6(b), uniqueness reaches its highest when $\tau \geq 15$.

Table 3: Comparison between pre- and post-training attractors.

Properties	Pre-training	Post-training ($\tau = 20$)
Types of attractors (%)		
Fixed points	0 (SD=0)	0 (SD=0)
Limit cycles	61.9 (SD=20.1)	100 (SD=0)
Complex	38.1 (SD=20.0)	0 (SD=0)
Time preceding simple attractors	99.9 (SD=11.2)	6.4 (SD=2.4)
Simple attractor length	15.6 (SD=5.3)	4.0 (SD=2.2)
Stability metric with $\nu = 0.1$ (%)	9.0 (SD=7.4)	92.2 (SD=3.9)
Uniqueness metric	35.4 (SD=3.3)	73.0 (SD=2.1)

3.4 Effects of Training

To show how training affects the resulting limit cycles, Table 3 summarizes the properties of the limit cycles obtained in a SOM before and after training. The values are obtained based on 20 independent simulations initialized using different random weights and $\beta = 0$. Before training, the attractors are either complex or limit cycles that occur long after inputs end. After training, all attractors become limit cycles that are both shorter and start much earlier. In both cases, there are no fixed point attractors. Further, the post-training limit cycles are much more stable and unique compared with the pre-training attractors.

3.5 Oscillatory Activation: An Example

As an example, two limit cycle representations in a typical simulation, using $\beta = 0, \tau = 20$, are visualized in Figure 7. The learned representation for the phoneme sequence of "butterfly" is a limit cycle of length 2 (Figure 7(a)). The two alternating activation patterns of the SOM are shown in the top of Figure 7(a). Each cell corresponds to a node in the SOM, and dark cells indicate nodes that are activated. Designate all activated nodes in one pattern (left) to be group 1, and all activated nodes in the other (right) to be group 2. The bottom of Figure 7(a) shows the number of nodes in group 1 and 2 that are turned on at each time step. The dashed line at $t = 6$ indicates when the phoneme input sequence ends. Neither group is significantly activated until the input sequence is finished, shortly after which both groups become fully activated and demonstrate oscillatory patterns.

Another limit cycle of length 6 that encodes the phoneme sequence for "apple" is illustrated in

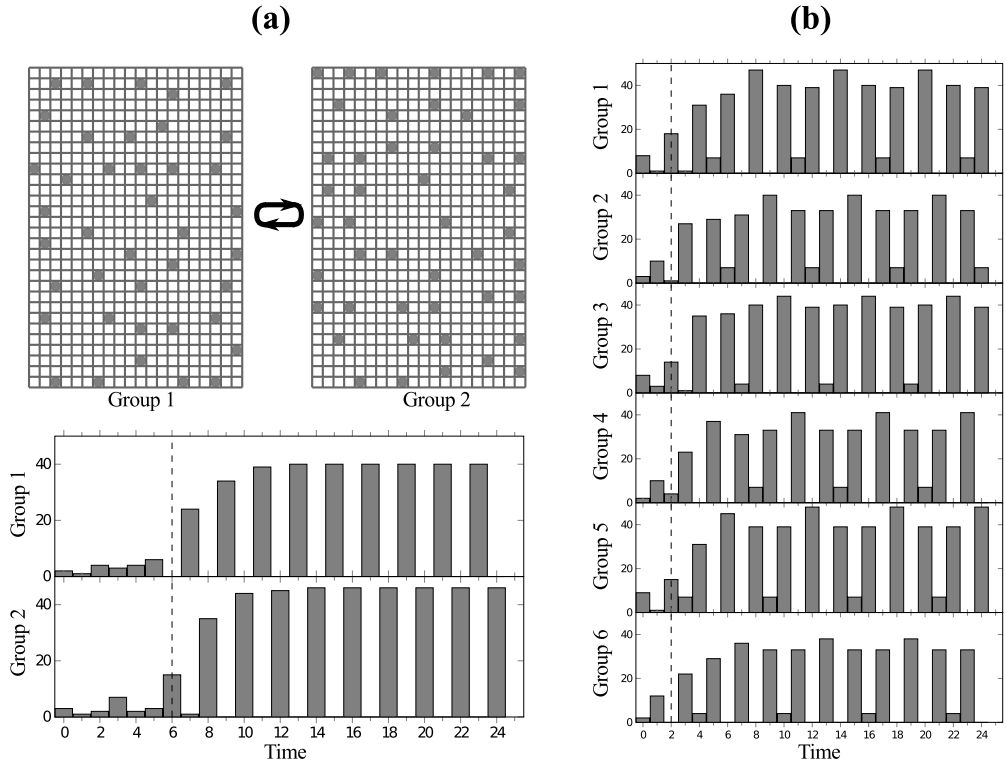

Figure 7: Illustration of two examples of limit cycles representing the phoneme sequences (a) "butterfly" and (b) "apple". The limit cycles are obtained using $\beta = 0$ and $\tau = 20$. In (a), a limit cycle of length 2 is shown. The top part shows the alternating activation patterns in the limit cycle, where activated SOM nodes are marked using dark cells. All activated nodes in one pattern are arbitrarily designated to be group 1, and those in the other pattern group 2. The bottom part of (a) shows the number of nodes in each group that are turned on at each time step. The dashed line indicates the end of the input phoneme sequence. In (b), another limit cycle of length 6 is shown. Nodes may be assigned to multiple groups if they are activated more than once in one pass through the limit cycle.

Figure 7(b). The nodes that are activated in each of the six distinct activation patterns in the limit cycle are designated to be groups 1–6. Nodes may be assigned to more than one group if they are activated more than once in a cycle. The times where a group is fully activated is indicated by the tallest bars in each chart. Although a majority of the nodes in each group is activated every other time step, at least some of them are activated every three time steps as indicated by the shorter bars. Closer inspection reveals that other than 2 and 3, some nodes have a period of 6 time steps.

Finally, note that these examples also illustrate a general observation of these experiments that the length of a representing limit cycle is not generally proportional to the length of the corresponding input sequence. The phoneme sequence for "butterfly" is significantly longer than that of "apple", yet their limit cycle representations are 2 and 6, respectively.

3.6 Map Formation

With multi-winner activation, limit cycle SOM dynamics, and post-input continuing adaptation, it is unclear in advance whether map formation similar to that in conventional SOMs will still occur. We used our model for phoneme sequence processing to examine this issue, based on the parameters $\beta = 0, \tau = 20$ that are known to consistently generate small limit cycles. In each of 20 independent simulations, we found that self-organizing maps of input patterns formed reliably in the presence of limit cycle representations.

Figure 8 shows a representative example of the afferent weights of a SOM from a typical simulation. Map formation is clearly observed. By comparing pre-(a) and post-training(b) weight values, the overall lighter shades after training indicate that the weight vectors are more tuned to the phoneme data. Similar phonemes are grouped in close proximity. For instance, the vowels are grouped in "islands" of nodes (highlighted in the figure by thick borders) in a "sea" of consonants. That there exist multiple vowel islands instead of a single island is caused by the multi-winner activation, where local winners separate from each other coexist. Also notice that most of the vowels are often surrounded by /l/ and /r/, the consonants having the most similar features to vowels.

Limit cycle representations formed in a SOM have no obvious correlations with the SOM's afferent weights. For example, limit cycles that activate a node labeled /f/ in Figure 8(b) do not necessarily encode phoneme sequences containing /f/. Limit cycle representations are a result of post-stimuli self-organization of feedback weights, which is only indirectly related to afferent inputs.

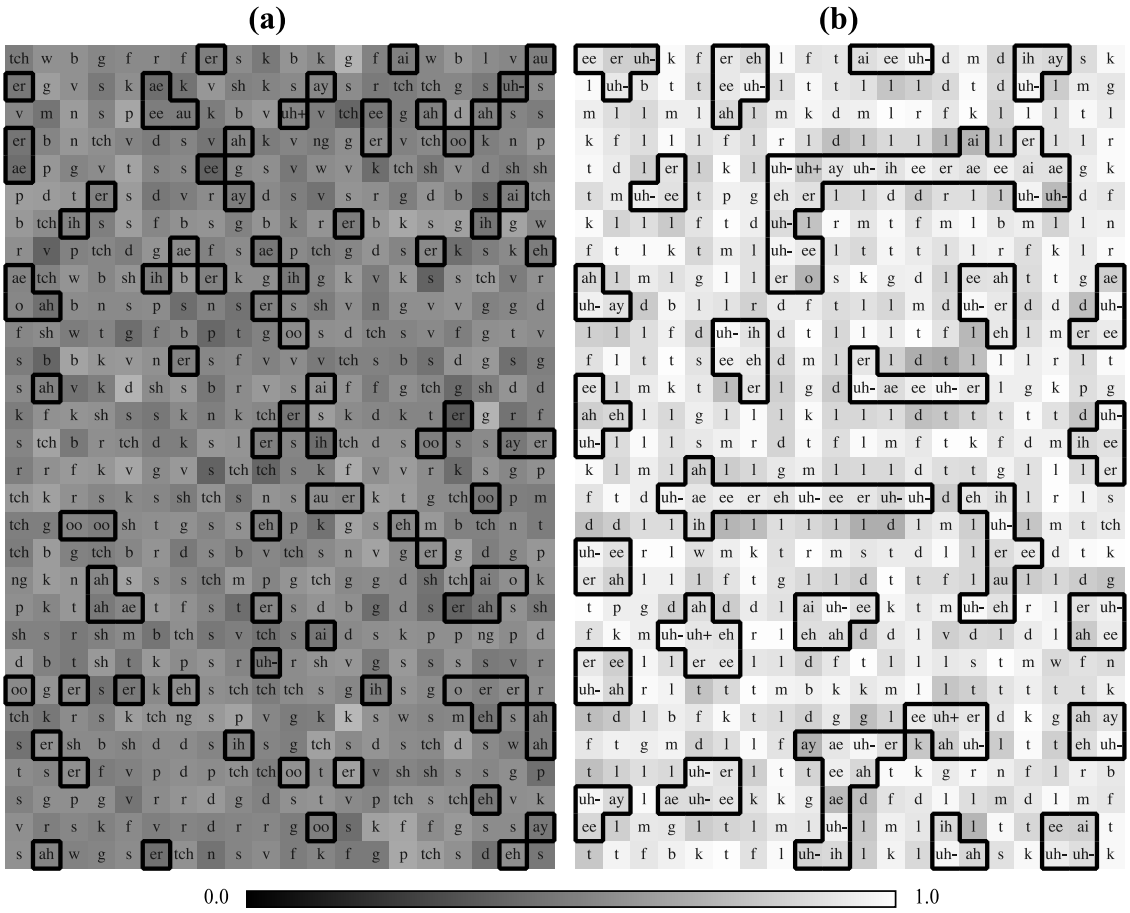

Figure 8: A labeled SOM in a typical simulation (a) before and (b) after training using phoneme sequence data. The phoneme label in each node is the one that best matches (using an inner product metric) the node's weight vector in the afferent channel. Degrees of matching are shown by shading in the cells. Lighter color indicates a better match. Cells labeled with vowels are highlighted by thicker borders and are seen to be clustered in (b). ($\beta = 0$, $\tau = 20$)

They can be viewed as abstractions of the input sequences.

A U-matrix [37] analysis of the phoneme map is shown in Figure 9. The U-matrix values, calculated as the similarity (inner product) of the adjacent weight vectors, are shown as greyscale in each cell. The lighter the shade, the smoother the weight changes are at the location. Before training (Figure 9(a)), the U-matrix values are mostly distributed in the mid-range (0.5–0.7) without prominent pattern, whereas the post-training map (Figure 9(b)) shows improved contrast: clusters of lighter-shaded regions (valleys) emerged and are separated by narrow and darker-shaded regions (walls). The shades of most cells (74.5% of the cells) are lighter after training, indicating that the map becomes smoother in most regions after training. The darker cells indicate that greater sudden

jumps of weights occur after training, although cells of this type are relatively few. Although the overall smoothness may not be as good as in a conventional SOM, due to the use of a multi-winner SOM, the U-matrix visualization clearly suggests the formation of a piecewise smooth map. To further improves map smoothness, we performed a separate experiment where the hard binary inputs 0 and 1 were replaced by 0.1 and 0.9, respectively. The resulting U-matrix is shown in Figure 9(c). While the shading patterns are not qualitatively different, the overall shades become lighter than in Figure 9(b), including those "walls", which indicates an overall smoother map (95.6% of the U-matrix cells became lighter after training). Unfortunately, this change causes the number of unique limit cycles to decrease too, and therefore we continue using hard binary inputs for the rest of the study. Overall, the above visualizations indicate that map formation indeed emerges in the SOM variant and the modified training process that this study uses.

A separate SOM was trained using fixed image inputs. The SOM model and the parameters are identical to that used in the above experiments, except that the number of afferent weights is increased to account for 50×50 pixels of the images. Each image is presented for a fixed number of time steps, while the SOM is trained using $\beta = 0$ and $\tau = 20$. In all of the 20 independent simulations, limit cycles form, even though there is a fixed input pattern rather than a temporal sequence of inputs. As a typical example, the afferent weights of the SOM after learning are visualized in Figure 10 (top). An overview of the entire map (left in Figure 10 (top)) shows map formation where images occupying similar spatial locations tend to occur in clusters having similar overall shapes. For instance, horizontally flat images are grouped together in the top two rows of the magnified view of the encircled region that is pictured on the right in Figure 10 (top). U-matrix analysis in Figure 10 (bottom) shows that some regions become smoother after training, although most of the other regions do not change significantly, possibly due to the large number of weights (2500).

In summary, this first experiment studied the formation of attractor representations in a SOM for temporal sequence inputs. Although three qualitatively different types of attractors can be formed using different parameter sets, the stability and uniqueness analysis indicates that the small limit cycles emerging around $\beta = 0$ and $\tau = 20$ are the most robust and unique. These are important properties for limit cycles that are intended to serve as internal representations because they make the limit cycles more resistant to noise and easier to tell apart. The small number of

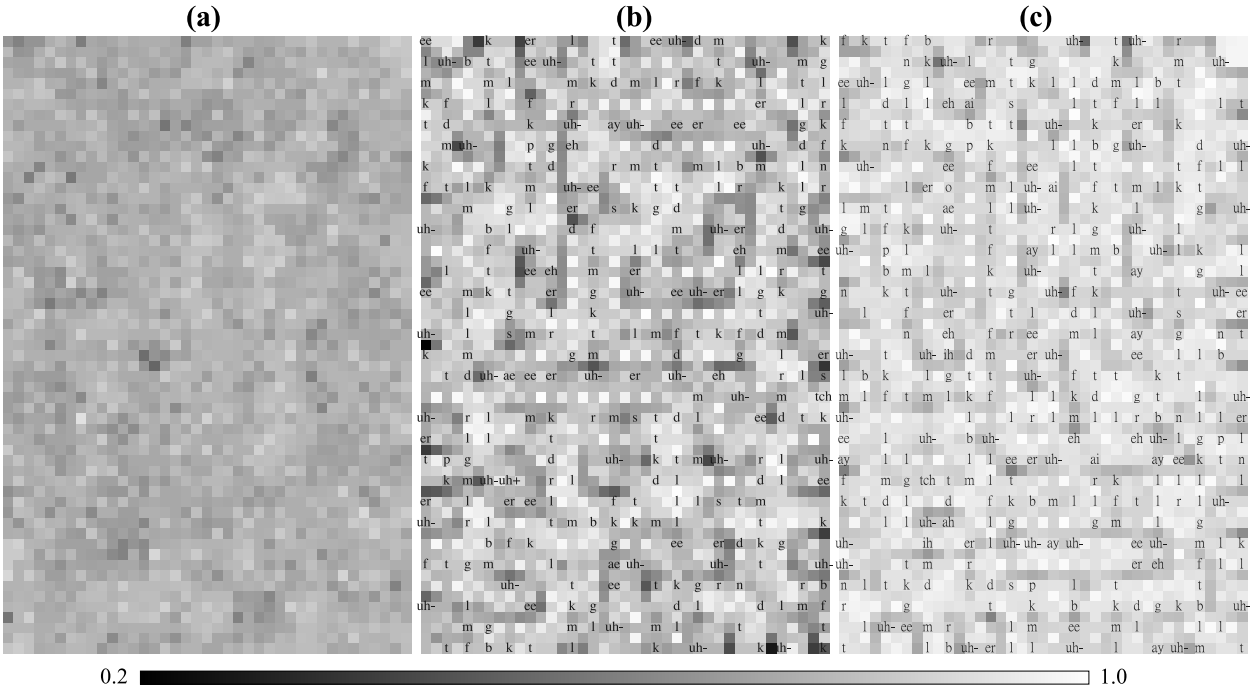

Figure 9: The U-matrices corresponding to the SOM in Figure 8. (a) and (b) are the results before and after training, respectively. (c) shows the result of training the SOM using blurred binary inputs (i.e., using values $\{0.1, 0.9\}$ to replace hard binary values $\{0, 1\}$). The shade of each cell represents its U-matrix value, which indicates the similarity (inner product) of the weight vectors between that node and its neighbors. The greater the value (the lighter the shade), the smoother the weight changes are in the local region. After training (i.e., (b), (c)), the U-matrix shows the emergence of clusters of smooth regions (light shade) separated by less smooth walls (dark shade). The overlaid phoneme labels are the same as in Figure 8, but omit the ones that are not significantly similar to a phoneme vector (i.e., inner product ≤ 0.9). ($\beta = 0$, $\theta = 20$)

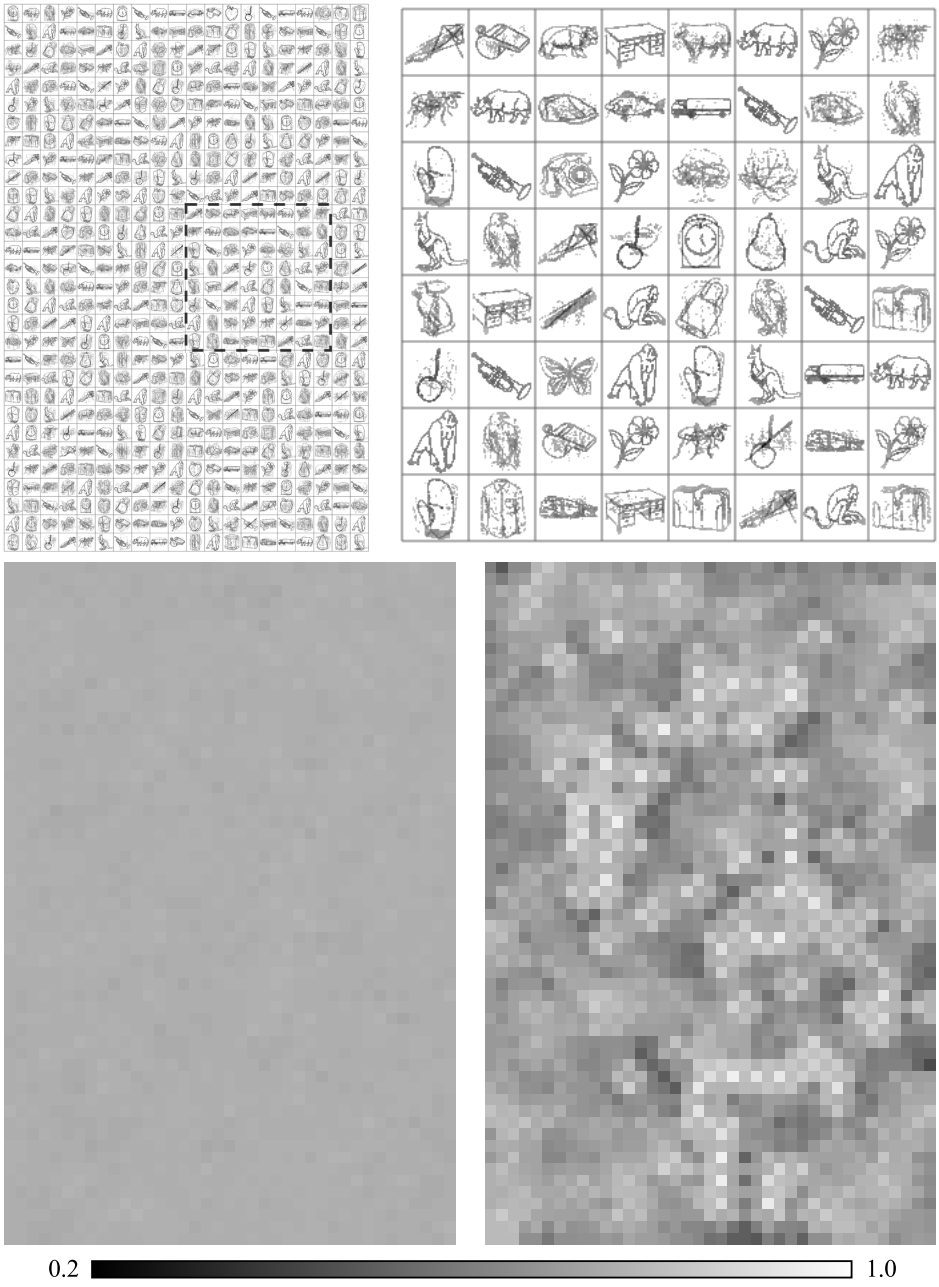

Figure 10: Top: afferent weights of a limit cycle representation 30×20 SOM, trained using fixed images as input ($\beta = 0, \tau = 20$). Darker pixels indicate higher weight values. A threshold is used to filter out low weight values for clearer visualization. The left part shows an overview of the entire map. The region enclosed by the dashed lines is magnified and shown in the right. Bottom: the U-matrices for the SOM before (left) and after (right) training. The U-matrix visualization method is identical to Figure 9.

time steps it takes to settle in these limit cycle attractors and the short cycle lengths also mean faster recall. This is in contrast to static representations, which have no mechanisms to fix noisy states and have lower uniqueness that makes them harder to be distinguished. Examination of the activity patterns forming limit cycles showed that they reflect an oscillatory nature of individual nodes. Finally, self-organized map formation, a defining property of all SOMs, also occurs in our modified SOM model in spite of the presence of limit cycles.

4 Associating Limit Cycles with Static Representations

If one is to ultimately construct large-scale neurocognitive models that incorporate a network of interacting SOMs, these model SOM regions must have a means to associate corresponding representations occurring in different SOMs. For example, while they do not involve map formation, neuroanatomically-based models of language and symbol grounding need to associate spoken names with seen objects [28, 39]. Our goal here is to study the practicality of limit cycles to form associations. At a minimum, a represented entity in one SOM (e.g., temporal sequence of activation patterns representing the spoken name of a specific object) should be recallable not only by repeating the original external stimuli it encodes, but also by an associated representation appearing alone in another SOM (e.g., activation pattern representing the corresponding image of the named entity). Despite the good properties demonstrated by isolated limit cycle representations in Section 3, it is unclear in advance whether associations between representations in different SOMs can be learned when using limit cycles. To address this issue, here we investigate the ability of a SOM to recall a target limit cycle by referencing only an associated static representation learned independently in another SOM. If associations are possible with limit cycles, it is also important to compare the effectiveness of using limit cycle representations versus static ones.

Establishing associations between representations in two multi-winner SOMs is a much harder problem than between two single-winner SOMs, especially if one or both SOMs exhibit limit cycle dynamics. With two single-winner SOMs, the association between each pair of winner nodes, one in each SOM, is a simple one-to-one mapping problem. Forming associations between two multi-winner SOMs, on the other hand, requires correlating many activated nodes for each pair of states, while coping with the multi-winners-take-all processes that make the input-output relationship nonlinear

for each SOM. Further, when using limit cycle representations, the associations are required to support correlations between varying numbers of patterns in each representation, while avoiding explicit temporal alignments of the limit cycles.

In this second experiment, we apply the limit cycle representations developed in Section 3 to a simple association task involving two SOMs. The goal is for a SOM to learn to recall the limit cycles that represent learned phoneme sequences, without being provided with the input sequences directly, but instead based on activation of another SOM which has previously learned independently to process stimuli in a different modality (visual images). This is a kind of "name that object" task, and it relates to the difficult issue of symbol grounding [28].

Figure 11 shows the architecture used in this experiment. It consists of two 30×20 SOMs, each processing stimuli in a different modality, and each trained separately before being connected together. Associative connections exist between the two SOMs, via a hidden layer of 20 nodes. For simplicity, the image SOM in this experiment has no recurrent connections, and thus its activation forms a static representation, driven directly by the image input. The question being asked is whether a static representation in one map can be associated with limit cycle representations in another map (or more generally, can limit cycle representation SOMs be combined effectively with more conventional static representation SOMs in a single system). The limit cycles that are to be recalled in this case are in the phoneme SOM, which has three channels. Channels 1 and 2 are phoneme input and topographic feedback, respectively, similar to the SOM in the previous experiment ($\alpha_1 = 0.64$, $\alpha_2 = 0.36$). Channel 3 is composed of associative connections from the image SOM via the hidden layer (20 nodes). The goal of the task is, after learning, to recall the limit cycle representing each phoneme sequence P_k in the phoneme SOM, when the architecture is provided with only an image stimulus V_k. The procedure in this experiment is outlined in Figure 12 and elaborated below.

Training is performed in two stages. In the first stage, the two SOMs are trained independently using the data in their respective modality. Channel 3 of the phoneme SOM is disabled ($\alpha_3 = 0$, see Eq. 3) during this time. The parameters for training the phoneme SOM are $\beta = 0, \tau = 20$. The image SOM does not have recurrent feedback connections, and thus it is trained in the same way as conventional SOMs (although multi-winners-take-all is used). At the end of the first stage, the static representations in the image SOM are recorded as a set of activation states $\left\{R_1^V, R_2^V, ...\right\}$, and

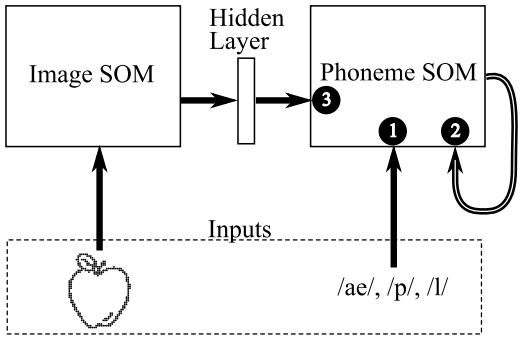

Figure 11: The architecture for the task of establishing associations between static and limit cycle representations. The two SOMs, one for images and the other for phoneme sequences, are connected via a hidden layer. The three channels of the phoneme SOM are numbered by circles. Full connections are drawn as solid arrows, and topographic connections are drawn as hollow arrows.

the limit cycle representations in the phoneme SOM are recorded as $\left\{R_1^P, R_2^P, ...\right\}$. Additionally, for each phoneme input sequence P_k, the single state in the phoneme SOM at $t = |P_k| + \xi$, where ξ is a discrete uniform random variable in $[0, 9]$, is recorded as S_k^P. In the second stage, each pair $\langle R_k^V, S_k^P \rangle$ is used as the input and the target output to train the associative connections between the two SOMs. Training is done by a standard error-backpropagation method, RPROP [31]. Error backpropagation is not biologically plausible, but here we use it simply as a first step to see whether inter-map associations can be formed at all in this situation.

In a separate control experiment, a model identical to that shown in Figure 11 is used, except that the phoneme SOM is based on static representations. The two training stages described above are performed on this model as well, except for the following differences. In the first stage, the phoneme SOM is trained using $\beta = 0, \tau = 0$. Each static representation $R_k^{P,\text{static}}$ in the phoneme SOM is taken to be the final activation pattern occurring at $t = |P_k| - 1$. In the second stage, each $R_k^{P,\text{static}}$, as opposed to a randomly selected state, is used as the target output for learning the associative connections.

Notice that with each S_k^P for training, the association is recorded at a random time step after the input phoneme sequence ends. The timing of S_k^P does not depend on the timing of the cycle R_k^P or its length. In fact, S_k^P may not even be a state in the cycle R_k^P. The intent here is to determine whether associative learning can be effective with this relaxation of timing requirements, and whether it can restore an entire limit cycle based on an association with part of it. In contrast,

Training Phase 1:

Disable the associative connections between the two SOMs
Train both SOMs independently using unsupervised learning
Temporarily store activation states for later use:
    foreach k in the dataset:
        $R_k^V \leftarrow$ the static state in the image SOM for input V_k
        $R_k^P \leftarrow$ the limit cycle of states in the phoneme SOM for input P_k
        $S_k^P \leftarrow$ a single state that occurs at $t = |P_k| + \texttt{random\_integer}\{0, ..., 9\}$ in the phoneme SOM for input P_k

Training Phase 2:

Train the associative connections between the two SOMs using error backpropagation with training data $\langle R_k^V, S_k^P \rangle$, $\forall k$

Testing Phase:

Enable the associative connections between the two SOMs
Disable the afferent input of the phoneme SOM
Test if the limit cycles in the phoneme SOM are restored by image inputs alone:
    foreach image V_k in the dataset:
        Provide V_k as input to the image SOM for the first λ time steps
        Update activation states throughout the architecture for each time step until a limit cycle occurs in the phoneme SOM
        if the above limit cycle == R_k^P:
            Mark data entry k as "successful"

Figure 12: Outline of the second experiment: associating static representations with limit cycles.

recording each static representation $R_k^{P,\text{static}}$ as has been done in some previous SOM models [33] for training the associations requires the knowledge of the exact time when the target activation state appears, i.e., at $t = |P_k| - 1$, something that in general would differ with different length input sequences.

After training, the architecture is tested by enabling channel 3 ($\alpha_3 = 0.64$) and disabling channel 1 ($\alpha_1 = 0$) for the phoneme SOM. Consequently, the architecture receives only image stimuli via the image SOM. Each image V_k is presented for λ (a constant parameter) time steps, during which the learned static pattern R_k^V occurs in the image SOM. The activation of the phoneme SOM is then triggered by R_k^V via the hidden layer. To assess robustness of association, noise of amplitude ν (a constant parameter) is added in channel 3, in the same way as described in Eq. 11. The added noise persists throughout the λ time steps during which the image input is presented. If the eventual state trajectory of the phoneme SOM recreates the corresponding cycle R_k^P, the data entry k is considered successfully recalled. For the control experiment, a recall is considered successful if $\boldsymbol{a}(\lambda - 1) = R_k^{P,\text{static}}$ in the phoneme SOM.

Figure 13 shows the percentages of all representations in the phoneme SOM that are recalled this way, with respect to different parameter values for β (same-node feedback weight), λ (number of time steps the image input is presented), and ν (amplitude of noise added to channel 3 of the phoneme SOM). The results reported here are averaged over 20 independent simulations with different random initial weights. The highest recall rate is achieved with β values around 0, which is more evident when noise is added ($\nu > 0$). This echoes the observations in Section 3 that small cycles around $\beta = 0$ are robust.

When compared with using static representations in the phoneme SOM, using limit cycle representations consistently results in higher recall rates (see Figure 13(b)(c)). The limit cycle representations also require the image input to last for a shorter time period ($\lambda = 2$) in order to reach the maximum recall rate. Notice that the static representations have very low recall rate for $\lambda = 1$. This is the case even when error-backpropagation training produces very low mean-square errors. The reason is that the multi-winners-take-all process can result in incorrect winners even when these nodes receive a very small amount of net input from channel 3, if their net inputs are the highest in their local neighborhoods. Allowing the phoneme SOM activation to update for multiple time steps ($\lambda > 1$) via both the associative and the recurrent feedback connections helps eliminate some of these incorrect winners for both static and limit cycle representations, but only to a certain extent (the recall rates are somewhat the same for $\lambda \geq 5$ in the case of static representations). For limit cycle representations, attractor dynamics provide an additional means to restore incorrect activation, which results in higher recall rate than static representations.

As noise amplitude ν increases, the recall rates for both the static and the limit cycle representations decline, although the limit cycle representations consistently maintain higher recall rate than the static ones (Figure 13). The *difference* in performance actually monotonically increases as ν increases until noise becomes quite substantial ($\nu > 0.3$). The limit cycles perform slightly better with a small amount of noise, which on closer examination, helps prevent incorrect winners in the attractor dynamics, and thus they are robust in the presence of noise.

In summary, this experiment shows that it is possible to recall limit cycle representations (in the phoneme SOM) using static activation patterns (in the image SOM), via associative connections. This suggests that SOMs based on limit cycle representations are compatible with and might be combined with the more prevalent SOMs using static representations. The training data for

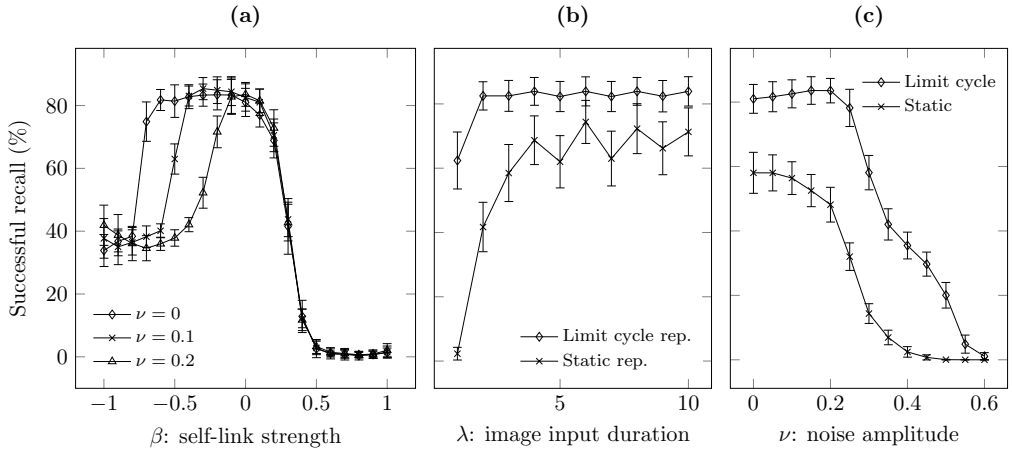

Figure 13: The percentages of the dynamic and the static representations recalled in the phoneme SOM of Figure 11 when image stimuli alone are provided to the architecture. The results in (a) and (c) are averaged over $\lambda \in [1, 10]$. $\beta = 0$ in (b) and (c); $\nu = 0$ in (a) and (b); $\tau = 20$ in all cases. The error bars indicate one standard deviation over 20 independent simulations.

limit cycle associations are sampled at relatively arbitrary times. This shows one possible way in which timing requirement for limit cycle representations are fairly relaxed. Recalling limit cycle representations is consistently more successful than static representations. They require referencing the source of association for fewer time steps and are more resistant to noise.

5 Associating Pairs of Limit Cycle Representations

A neural architecture operating solely based on limit cycle representations must be able to establish associations between these attractors in different SOMs. This is a more difficult task than that considered in Section 4, because the limit cycles in the different SOMs each contains multiple spatial patterns, and the limit cycles can be of different lengths and aligned temporally in different ways. This third experiment aims to associate pairs of limit cycle representations, such that a limit cycle in one SOM can be recalled by referencing only the associated limit cycle in another SOM. Unlike the last experiment, where a limit cycle is recalled by attempting to recreate one of its preceding or constituent states, here multiple states in the cycle are learned by the association. To demonstrate the relaxation of operation timing requirements, limit cycles in different SOMs are not explicitly aligned during and after training, as doing so requires timing information about the limit cycles.

Figure 14 shows that the architecture used in this experiment is similar to that in the previous

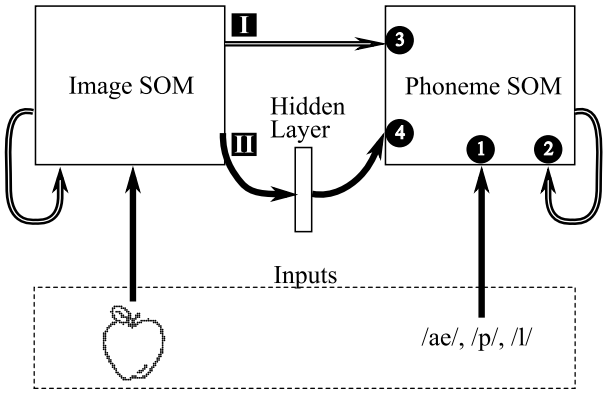

Figure 14: The architecture for the task of establishing associations between limit cycle representations occurring in two different SOMs. Both SOMs have direct feedback. The four channels of the phoneme SOM are numbered in circles. Full connections are drawn as solid arrows, and topographic connections are drawn as hollow arrows. The two SOMs are connected using associative connections containing two routes. Route I is topographic and involves direct connections. Route II is full and involves indirect connections through a hidden layer.

experiment. Again, the goal after learning is to recall each limit cycle representing a phoneme sequence in the phoneme SOM by giving the architecture only the corresponding image. However, the image SOM now also has topographic feedback connections in addition to afferent connections, and thus is able to generate limit cycle representations for image stimuli. Further, the associative connections between the two SOMs now contain two routes. Route I consists of direct topographic connections between the two SOMs. Route II consists of indirect full connections via a hidden layer (60 nodes), which were the only associative connections in the previous experiment. These two routes are trained separately and tested jointly. The procedure of this experiment is outlined in Figure 15 and elaborated below.

Training of this architecture is again divided into two stages. In the first stage, training of the two SOMs is performed independently (channels 3 and 4 of the phoneme SOM are disabled) to acquire limit cycle representations in their respective modalities. Small limit cycle attractors are learned independently in both SOMs. Each image stimulus V_k is represented by a limit cycle R_k^V, and each phoneme sequence P_k is represented by a limit cycle R_k^P, where the lengths of R_k^V and R_k^P are in general different.

In the second stage of training, each tuple of data $\langle V_k, P_k \rangle$ is presented to the architecture. The two SOMs are normally activated through time while $\alpha_3 = \alpha 4 = 0$ (i.e., activation in the

Training Phase 1:

Disable the associative connections between the two SOMs
Train both SOMs independently using unsupervised learning
$AdaptationPeriod \leftarrow t = 30, 31, ..., 34$ (exact timing not important; see text)
Temporarily store activation states for later use:
    foreach k in the dataset:
        $A_k^V \leftarrow$ states that occurs in *AdaptationPeriod* in the image SOM for input V_k
        $A_k^P \leftarrow$ states that occurs in *AdaptationPeriod* in the phoneme SOM for input P_k
        $R_k^P \leftarrow$ the limit cycle of states in the phoneme SOM for input P_k

Training Phase 2:

Train route (I) of the associative connections using Eq. (13) and $\langle A_k^V, A_k^P \rangle$, $\forall k$
Train route (II) of the associative connections using error backprop and $\langle A_k^V, A_k^P \rangle$, $\forall k$

Testing Phase:

Disable the afferent input of the phoneme SOM
Test if the limit cycles in the phoneme SOM are restored by image inputs alone:
    foreach image V_k in the dataset:
        Provide V_k as input to the image SOM
        Disable both routes of the associative connections at $t = 0$
        Enable both routes of the associative connections at $t = \theta$ (parameter)
        Update activation states throughout the architecture for each time step until a limit cycle occurs in the phoneme SOM
        if the above limit cycle == R_k^P:
            Mark data entry k as "successful"

Figure 15: Outline of the third experiment: associating two sets of limit cycles.

image SOM does not affect that of the phoneme SOM). Both routes of the associative connections are then adapted within an *adaptation period*. The adaptation period is chosen rather arbitrarily, beginning at the 30th time step after each input and lasting for 5 time steps. The number 30 can instead be any number such that adaptation occurs sufficiently late, i.e., after the dynamics of both SOMs have settled into the limit cycle attractors R_k^V and R_k^P. The duration of 5 time steps can instead be any small number. Large numbers tend to prematurely over-fit the associative weights for a particular pair of limit cycles. The same duration of the adaptation period is used regardless of the different lengths of the limit cycles. The number 5 is chosen here to demonstrate that the length of the adaptation period does not have to be the same or a multiple of the lengths of any limit cycles (no limit cycles of length 5 are known to form across all simulations; see Figure 4). This signifies that the timing and the lengths of the limit cycles are not important in training the associative connections.

For the direct route (I), weight adaptation for channel 3 is performed during the adaptation period using modified Eq. 6 (for each node i, channel $j = 3$):

$$\hat{\boldsymbol{w}}_{ij}(t+1) = \boldsymbol{w}_{ij}(t) + \mu_j \left[a_i(t) - \boldsymbol{w}_{ij}(t)^{\mathsf{T}} \boldsymbol{x}_j(t) \right] \boldsymbol{x}_j(t), \tag{13}$$

and Eq. 7 (unchanged). The quantity in the square brackets here represents the difference between activation level a_i of node i in the phoneme SOM that is triggered by an input phoneme sequence, and $\boldsymbol{w}_{ij}^{\mathsf{T}} \boldsymbol{x}_j$ that represents the net input from channel 3. Since $\alpha_3 = 0$, $\boldsymbol{w}_{ij}^{\mathsf{T}} \boldsymbol{x}_j$ ($j = 3$) does not contribute to a_i but adapts passively. For the indirect route (II), weight adaptation is again performed using RPROP, using the input-driven limit cycles in the phoneme SOM as target outputs. The training data are recorded within the adaptation period for each data entry $\langle V_k, P_k \rangle$. As a result, they contain multiple pairs (5 in this case, which is the length of the adaptation period) of input and target output states for each data entry k, as opposed to only one pair each in the previous experiment.

After training, only image input is provided to the architecture ($\alpha_1 = 0$ for the phoneme SOM). Each R_k^V is recalled in the image SOM as expected. The phoneme SOM needs to recreate the corresponding cycle representation R_k^P by referencing the cycle representation R_k^V in the image SOM, via both routes of the associative connections. Channels 3 and 4 are initially disabled

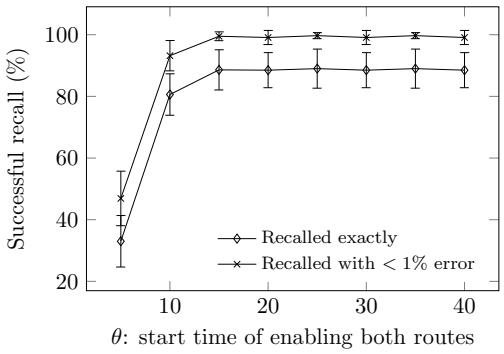

Figure 16: Percentages of limit cycle representations that are successfully recalled in the phoneme SOM and percentages that are nearly recalled (at most 5 nodes out of 600 having incorrect activation) when input is given only to the image SOM.

($\alpha_3 = \alpha_4 = 0$) to prevent pre-limit cycle states from passing through. They are then enabled at $t = \theta$ ($\alpha_3 = 0.1$, $\alpha_4 = 0.15$) and remain so, where θ is a parameter. At this point the phoneme SOM starts to access the image SOM via both routes. Note that the information about the start time and the length of each cycle in the image SOM is not conveyed in any explicit way to the phoneme SOM. Only node activities are accessed.

Percentages of limit cycle representations correctly recalled in the phoneme SOM are shown in Figure 16. The figure also shows percentages that include "closely recalled" cycles, which are within one-norm distance of 5 away from the correct cycles. This means that for each limit cycle to be considered closely recalled, there must be 5 or fewer nodes (out of 600) having different activation from the correct cycle R_k^P, which translates to less than a 1% difference. This threshold is also much smaller than the average distance ~ 70 between any limit cycles (see Figure 5).

As the results show, nearly all limit cycles are closely recalled, $\sim 89\%$ of which are exactly recalled, for $\theta \geq 20$. The exact time of θ is not critical. It does not have to be set according to the timing information about the input sequence, the representations, or the training process, as long as it is large enough to allow the image SOM to settle into a limit cycle R_k^V. This robustness with respect to timing indicates that the model is able to access limit cycle representations at relatively arbitrary times.

6 Concluding Remarks

SOMs conventionally work in isolation for simple tasks such as clustering and visualization. Since SOMs are widely accepted as models of individual cortical maps, organizing SOMs into larger neural architectures in a way resembling cortical regions that are connected as part of a large brain network, is a reasonable goal for pursuing machine intelligence and for advancing to large-scale brain models. Recent work on multiple interacting SOMs organized into tree-like structures provides pioneering examples of the potential value of multi-SOM systems [13, 30].

In an attempt to facilitate building such architectures, this study has developed and studied a novel spatiotemporal encoding scheme for SOMs, using limit cycle attractors that are part of on-going map activation dynamics. Each external event is represented by cycles of recurring states that are encoded in a distributed fashion. Such multi-winner SOMs can express many more states with a sparse coding scheme than conventional one-hot coding, and ultimately the information they pass on to other neural components in the same architecture can be much richer. Additionally, the problem of controlling information flow in neural architectures is made easier by the relaxed timing constraints that occur when using limit cycle representations. This is because a limit cycle representation can be accessed at a relatively arbitrary time, as opposed to having to retrieve a transient state at a highly specific time. The latter requires fine-grained executive control, which both increases complexity and impedes scalability of a neural architecture, and in the past has largely been implemented by external control mechanisms. The relaxed timing requirement is demonstrated in the experiments reported here by accessing SOMs' activities at random or arbitrary times.

Limit cycle representations are also robust. Stable limit cycle attractors are able to recover themselves from nearby states. In the first experiment, we identified the parameters necessary to reliably obtain stable limit cycle attractors. When used in a SOM model, the limit cycle representations were found to be more resistant to sources of noise that the architecture was exposed to than occurred with static representations. Noise may include direct perturbations to the node activations (first experiment) and/or noisy associative connections between SOMs (second experiment).

We also showed that it is possible to establish associations between limit cycle representations in one SOM and information represented in another SOM. This is important because forming

associations is a basic operation shared by many neural models and the brain. In the second experiment, limit cycle representations were associated with single (static) states. The success of this experiment demonstrates that SOMs using limit cycle representations and those using the more prevalent static representations are compatible and can coexist in the same architecture. In the third experiment, static representations were completely eliminated. Slightly more complicated inter-map connectivity was used to associate pairs of limit cycle representations in the two SOMs. This shows the potential of using multiple maps based on limit cycle representations in an architecture.

Although the SOM model in this study makes a one-shot decision at each time step to determine what the winning nodes are, this winner selection process may potentially be implemented in a more biologically realistic way by using local recurrent connections that are short-range excitatory and long-range inhibitory [23, 27]. In particular, the LISSOM model in Miikkulainen et al. [27] has a set of local recurrent connections that learn to activate multiple winners through an iterative process that stabilizes after a few iterations. Note that currently the local recurrent connections in LISSOM are for multi-winner activation only, not to be confused with those in this study, which are for maintaining temporal relations among winner nodes at consecutive time steps that eventually lead to limit cycles. To obtain limit cycles in biologically-realistic SOMs such as LISSOM, a second set of local recurrent connections may need to be added. A tradeoff for using biologically-realistic SOMs is the increase of computational expense, since multiple iterations of calculating activation levels are required for each time step, as opposed to a quick one-shot decision. Such a tradeoff would clearly be justified, for example, if one was trying to model biological neural systems.

As an enormous amount of neuro-physiological data indicates, biological neural systems are unlikely to operate based solely on static or fixed states. They are highly rhythmic and oscillatory, and the limit cycle representations studied here bring SOM models one step closer to biological realism. This study tries to address this fundamental issue that not only applies to conventional SOMs but also to many other neurocomputational models that use fixed states. Moreover, models for temporal sequence processing often base computations on what happens when the input is presented, but ignore what happens *beyond* that point. On the other hand, information processing in biological neural systems is commonly attributed to the neural activities that are on-going, spontaneous, and oscillatory. Limit cycle representations can persist for extended periods of time, without requiring input sequences to be constantly available, and thus they can be viewed as

implementing a type of working memory [41]. Individual nodes in a limit cycle representation SOM also exhibit oscillatory activity patterns, even though they are not provided with explicit oscillatory abilities.

Acknowledgements

This work was supported by ONR award N000141310597.

References

[1] Abbott, A. (2013). Solving the brain. *Nature*, *499*(7458), 272–274.

[2] Bednar, J. A. & Miikkulainen, R. (2000). Tilt aftereffects in a self-organizing model of the primary visual cortex. *Neural Computation*, *12*(7), 1721–1740.

[3] Bi, G.-q. & Poo, M.-m. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience*, *18*(24), 10464–10472.

[4] Buzsaki, G. (2006). *Rhythms of the Brain*. Oxford University Press.

[5] Carpinteiro, O. (1999). A hierarchical self-organizing map model for sequence recognition. *Neural Processing Letters*, *9*(3), 209–220.

[6] Chappell, G. J. & Taylor, J. G. (1993). The temporal Kohønen map. *Neural Networks*, *6*(3), 441 – 445.

[7] Chen, Y. & Reggia, J. A. (1996). Alignment of coexisting cortical maps in a motor control model. *Neural Computation*, *8*(4), 731–755.

[8] Chow, T. W. S. & Rahman, M. K. M. (2009). Multilayer SOM with tree-structured data for efficient document retrieval and plagiarism detection. *IEEE Transactions on Neural Networks*, *20*(9), 1385 –1402.

[9] de Garis, H., Shuo, C., Goertzel, B., & Ruiting, L. (2010). A world survey of artificial brain projects, part I: Large-scale brain simulations. *Neurocomputing*, *74*(1–3), 3 – 29.

[10] Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science*, *338*(6111), 1202–1205.

[11] Euliano, N. R. & Principe, J. C. (1999). A spatio-temporal memory based on SOMs with activity diffusion. In E. Oja & S. Kaski (Eds.), *Kohonen Maps* (pp. 253–266). Elsevier.

[12] Fell, J. & Axmacher, N. (2011). The role of phase synchronization in memory processes. *Nat Rev Neurosci*, *12*(2), 105–118.

[13] Furukawa, T. (2009). SOM of SOMs. *Neural Networks*, *22*(4), 463–478.

[14] Hagenbuchner, M., Sperduti, A., & Tsoi, A. C. (2003). A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, *14*(3), 491–505.

[15] Hsu, C.-C. & Lin, S.-H. (2012). Visualized analysis of mixed numeric and categorical data via extended self-organizing map. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(1), 72–86.

[16] James, D. L. & Miikkulainen, R. (1995). SARDNET: A self-organizing feature map for sequences. In *Conference on Neural Information Processing Systems*, volume 7, (pp. 577–584).

[17] Johnsson, M., Balkenius, C., & Hesslow, G. (2011). Multimodal system based on self-organizing maps. In K. Madani, A. Correia, A. Rosa, & J. Filipe (Eds.), *Computational Intelligence*, volume 343 of *Studies in Computational Intelligence* (pp. 251–263). Springer.

[18] Kaipainen, M. & Ilmonen, T. (2003). Period detection and representation by recurrent oscillatory self-organizing map. *Neurocomputing*, *55*(3–4), 699 – 710.

[19] Kayacik, H. G., Zincir-Heywood, A. N., & Heywood, M. I. (2003). On the capability of an SOM based intrusion detection system. In *International Joint Conference on Neural Networks*, volume 3, (pp. 1808–1813).

[20] Kohonen, T. (2001). *Self-organizing maps* (3 ed.)., volume 30 of *Springer Series in Information Sciences*. Springer.

[21] Koskela, T., Varsta, M., Heikkonen, J., & Kaski, K. (1998). Temporal sequence processing using recurrent SOM. In *International Conference on Knowledge-Based Intelligent Electronic Systems*, volume 1, (pp. 290–297).

[22] Lichodzijewski, P., Nur Zincir-Heywood, A., & Heywood, M. (2002). Host-based intrusion detection using self-organizing maps. In *International Joint Conference on Neural Networks*, volume 2, (pp. 1714–1719).

[23] Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, *14*(2), 85–100.

[24] Manukyan, N., Eppstein, M., & Rizzo, D. M. (2012). Data-driven cluster reinforcement and visualization in sparsely-matched self-organizing maps. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(5), 846–852.

[25] Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, *275*(5297), 213–215.

[26] McQueen, T. A., Hopgood, A. A., Tepper, J. A., & Allen, T. J. (2004). A recurrent self-organizing map for temporal sequence processing. In A. Lotfi & J. M. Garibaldi (Eds.), *Applications and Science in Soft Computing*, volume 24 of *Advances in Intelligent and Soft Computing* (pp. 3–8). Springer.

[27] Miikkulainen, R., Bednar, J. A., Choe, Y., & Sirosh, J. (2005). *Computational maps in the visual cortex*. Springer.

[28] Monner, D. & Reggia, J. A. (2012). Emergent latent symbol systems in recurrent neural networks. *Connection Science*, *24*(4), 193–225.

[29] Niedermeyer, E. & da Silva, F. (2005). *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Lippincott Williams & Wilkins.

[30] Rauber, A., Merkl, D., & Dittenbach, M. (2002). The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, *13*(6), 1331–1341.

[31] Riedmiller, M. & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *International Conference on Neural Networks*, (pp. 586–591).

[32] Rumbell, T., Denham, S., & Wennekers, T. (2014). A spiking self-organizing map combining stdp, oscillations, and continuous learning. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(5), 894–907.

[33] Schulz, R. & Reggia, J. A. (2004). Temporally asymmetric learning supports sequence processing in multi-winner self-organizing maps. *Neural Computation*, *16*(3), 535–561.

[34] Strickert, M. & Hammer, B. (2005). Merge SOM for temporal data. *Neurocomputing*, *64*, 39 – 71.

[35] Sutton, G. G., Reggia, J. A., Armentrout, S. L., & D'Autrechy, C. L. (1994). Cortical map reorganization as a competitive process. *Neural Computation*, *6*(1), 1–13.

[36] Sylvester, J. & Reggia, J. (2009). Plasticity-induced symmetry relationships between adjacent self-organizing topographic maps. *Neural Computation*, *21*(12), 3429–3443.

[37] Ultsch, A. (1993). Self-organizing neural networks for visualisation and classification. In O. Opitz, B. Lausen, & R. Klar (Eds.), *Information and Classification*, Studies in Classification, Data Analysis and Knowledge Organization (pp. 307–313). Springer.

[38] Voegtlin, T. (2002). Recursive self-organizing maps. *Neural Networks*, *15*(8–9), 979 – 991.

[39] Weems, S. A. & Reggia, J. A. (2006). Simulating single word processing in the classic aphasia syndromes based on the Wernicke–Lichtheim–Geschwind theory. *Brain and Language*, *98*(3), 291 – 309.

[40] Wiemer, J. C. (2003). The time-organized map algorithm: Extending the self-organizing map to spatiotemporal signals. *Neural Computation*, *15*(5), 1143–1171.

[41] Winder, R., Cortes, C. R., Reggia, J. A., & Tagamets, M.-A. (2007). Functional connectivity in fMRI: A modeling approach for estimation and for relating to local circuits. *NeuroImage*, *34*(3), 1093 – 1107.

[42] Zhang, L. I., Tao, H. W., Holt, C. E., Harris, W. A., & Poo, M.-m. (1998). A critical window for cooperation and competition among developing retinotectal synapses. *Nature*, *395*(6697), 37–44.