Efficient Spike Timing Dependent Plasticity rule for Complex-Valued Neurons

Alex Baranski[1] and Tom Froese[1]

[1]Embodied Cognitive Science Unit
Okinawa Institute of Science and Technology Graduate University, 1919-1 Tancha, Onna-son, Okinawa, 904-0495 Japan
alexander.baranski@oist.edu

Abstract

Donald Hebb proposed in his 1949 book *The Organization of Behavior* that cell assemblies organized by temporally-asymmetric excitation form the basis of cognition. This basic idea has inspired a large body of research in neuroscience, and to a lesser extent in artificial intelligence. The modern manifestation of Hebb's principle is Spike-Timing Dependent Plasticity (STDP), and though we have a large body of experimental work investigating STDP, there is still little understanding of how networks of spiking neurons organize themselves into complex functional circuits, even though some progress has been made with models such as Liquid State Machines. Networks popular in artificial intelligence (e.g. MLPs) and in artificial life (e.g. CTRNNs) tend to eschew Hebb's insight and use error-backpropagation by gradient descent, in the case of AI, or an a-temporal Hebbian learning rule based on the outer product of neural activities, in the case of AL. Both of these approaches have greater interpretability than Spiking Neural Networks (SNNs), but both lack the mechanism that Hebb claimed was fundamental to cognition. This paper proposes to use complex-valued neurons (CVNs) to address this limitation, simultaneously promoting biological interpretation and computational tractability. The CVNs encode the firing rate and spike-time of a spiking neuron in the magnitude and angle, respectively, of a complex number. We also introduce an unsupervised piecewise-linear STDP learning rule compatible with CVNs, which for brevity we call complex-valued STDP (CVSTDP). We demonstrate both learning through error-backpropagation, and the spontaneous formation and dissolution of cell assemblies via the CVSTDP rule.

Introduction

A popular modeling choice for simplifying neurons in neural networks is to assume that only the firing rate of the neuron carries information. The firing rate is generally modeled as a nonlinear function of the neuron's membrane potential, which itself is a weighted sum of the inputs to the neuron. This assumption has become embedded in both discrete-time systems such as used in Machine Learning (most notably Deep Learning) and in continuous-time systems such as CTRNNs. Universal approximation results have been obtained for both kinds of systems, function approximation for feedforward Machine Learning architectures (Kratsios, 2021), and dynamical system approximation for CTRNNs (Funahashi and Nakamura, 1993), so in principle both approaches are capable of modeling any system. However, a more modern understanding of biological neurons has given merit to the idea that the relative timing, or phase, of spikes can also encode information, not just the firing rate. Even though feedforward discrete time networks and CTRNNs can approximate any function or dynamical system, that does not mean that any function or dynamical system is equally easy to approximate for these systems. In particular it's possible that some of the properties of biological neurons ignored by these models provide a strong structural prior that simplify particular aspects of cognition that are difficult to emulate in these systems. For instance, it has been found that under some (thought not all) physiological conditions (Inglebert et al., 2020) real neurons adjust their weights in response to the relative timing of pre- and post-synaptic spikes (Song et al., 2000), a learning paradigm that is called Spike Timing Dependent Plasticity (STDP).

For STDP to be implemented in artificial neural networks (ANNs), there must be spikes with times on which plasticity can depend, motivating the development of spiking neural networks (SNNs). While SNNs are attractive because they are naturally suited to low-energy event-based computing, there has been relatively little adoption in the Machine Learning community of spiking networks (notable exceptions include Liquid State Machines (Maass and Markram, 2004)) or of STDP. We think that this is in part due to the disparity in the mathematics and software infrastructure needed for both kinds of models: linear algebra for Machine Learning models, and numerical integration and event-based computing for SNNs.

Many attempts to link STDP to backpropagation find connections between the STDP update rule and the change in the average firing rate of some kind of spiking neuron model (Leaky Integrate and Fire, for example). However, these formulations generally involve some kind of aggregation over the effect of spikes, treating spikes as coming from a Poisson distribution and neglecting the effect of relative timing (Tavanaei and Maida, 2019) which are so important for

STDP. In other words, STDP becomes nothing more than an event-based approximation of backpropagation for rate-coding neurons.

Complex-Valued Neural Networks (CVNNs) are a kind of ANN in which the weights and biases are complex-valued, and the nonlinear activation function maps complex values to complex values, see (Bassey et al., 2021a) for a recent review. In general, these networks have found limited use in deep learning despite evidence of their utility (Rahman and Geiger, 2016; Danihelka et al., 2016; Arjovsky et al., 2016). In general, the motivation for using CVNNs in deep learning is that complex numbers have some beneficial properties with respect to gradient descent, namely that critical points of CVNNs are saddle points instead of local minima (Nitta, 2002). From a biologically perspective it is believed that neural oscillations play an important role in cognitive processing (Ward, 2003), and because complex numbers provide a natural way to represent oscillations, some work has been done using CVNNs to model perceptual binding through neural synchrony (Rao and Cecchi, 2010). Some work (Woźniak et al., 2020) has modeled spiking neurons as threshold neurons with an accumulator, but spike times can only take discrete values, whereas our approach allows for continuous spike times. To our knowledge there has been no investigation of the possibly fruitful connection between STDP-based learning and the temporal information that the phase of a CVN can carry. In particular, (Reichert and Serre, 2013) used CVNs with a similar interpretation to the one used in this paper (that is, firing rate and spike-time are related to complex number magnitude and angle), but we introduce some additional constraints to our nonlinear activation function motivated by biological plausibility. In addition, their work focused exclusively on training CVNNs using gradient descent, whereas here we introduce a computationally efficient variant of STDP that is applicable to CVNNs, and demonstrate that this rule is sufficient to enable the spontaneous and transient self-organization of phase-assemblies, which we hope will lead to the development of computationally tractable, biologically plausible, and mathematically interpretable neural networks that would enable translation of results among the neurophysiology, deep learning, and artificial life communities. Our hope is that the model presented in this paper can serve as a computationally simple but phenomenologically rich substrate for further study. To be clear, the model proposed here is not, per se, more powerful than other computational models of learning. Rather, it has the potential to make more transparent (from a machine learning perspective) some aspects of the behavior or real neurons.

Neuron model

Probably the simplest choice that can return a notion of relative spike timing to what is fundamentally a rate-coding neuron is to add another number, a phase offset. This encourages us to think of the neuron as an oscillator, with a firing rate f and a phase offset ω. In order to have these terms be well defined, as well as to stay within the mathematical framework common to machine learning, we pick an arbitrary binning of time, and have the neuron's firing rate and phase be constant within a single time bin. Our convention in this paper will be to treat time within a bin as a real number t between 0 and 1. We assume that all connections between neurons have a transmission delay of one time bin. This helps us conceptually, because we can treat individual neurons as isolated from feedback loops within a time bin. Essentially, feedback can only happen over multiple time bins (where feedback from A to itself means A stimulates neuron B, which reciprocally stimulates neuron A).

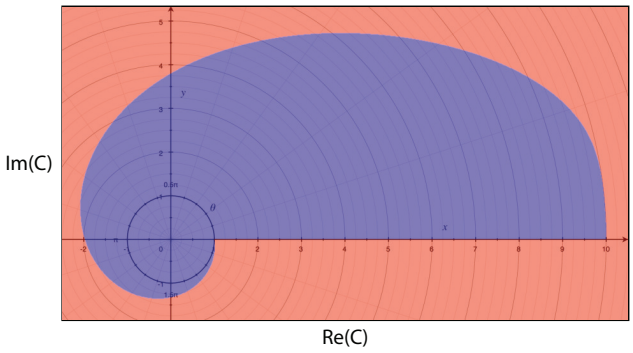

Figure 1: The blue region represents the range of permissible states for a CVN of the type described in this paper with a maximum firing rate f_{max} of 10. It is visually clear here that higher-frequency neuron states have a smaller range of possible phases.

At this point, we seem to have two options: (A) treat the phase offset ω as the time of the first spike (without loss of generality a value between 0 and 1), or (B) treat the phase offset as the phase of a periodic function (period $T = \frac{2\pi}{f}$). The difference between these two is that in (A) we find that the phase component of the neuron's state always conveys information, whereas in (B) the phase only conveys information when the firing rate is low, because the effective phase $\omega_{eff} = \omega \bmod T$, meaning that the range of effective phases is $[0, \frac{2\pi}{f}]$. We will explore the consequences of only the second model, because we feel that it better captures the spirit of a trade-off between rate-encoding and temporal-encoding. The reasons this trade-off should exist are relatively simple. Consider a spike train with temporal jitter (that is, individual spikes do not have completely precise timing). If the firing rate is low, it will still be possible to encode information in the relative timing of spikes, because there are few spikes with large temporal gaps between them, making spike disambiguation possible. However, for higher firing rates, we will find that individual spikes are

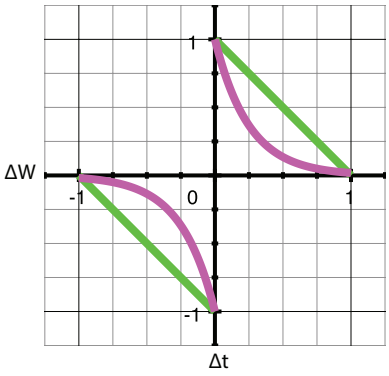

Figure 2: Magenta curve is symmetric STDP with A=1 and $\tau = \frac{1}{4}$. Green curve is the linear approximation given in equation (10).

more likely to "swap places" temporally with each other, meaning that spike disambiguation becomes difficult or impossible. Thus, from a theoretical perspective, we feel that explicitly modeling this trade-off is warranted, especially in light of the evidence that relative timing of spikes becomes less important for plasticity at high firing rates (Froemke et al., 2010).

A natural way to encode the state of a neuron with both firing rate and firing time is in a complex number c, with the firing rate of the neuron encoded in the magnitude or radius r of c and the phase offset ω encoded in the angle θ of c. Using this convention, we have $c = r\cos(\theta) + i \cdot r\sin(\theta)$, where i is the complex unit. We note here briefly that "phase offset" and "firing time" are used interchangeably through a scalar factor of 2π, so the "firing time" of a CVN is a number in $[0, 1]$ for the purposes of STDP calculation, but is a number in $[0, 2\pi]$ for the purposes of representation as a complex number.

As per the usual formulation in machine learning models, we construct our network as an alternating sequence of affine and nonlinear transformations. The affine transformation, which models the integration of inputs, is formulated as

$$a_k = \sum_j (w_{j,k} \cdot z_j) + b_k \tag{1}$$

which is a weighted (by $w_{j,k}$) sum of inputs z_j to neuron k. In matrix and vector notation, this is

$$\mathbf{a} = \mathbf{W}\mathbf{z} + \mathbf{b} \tag{2}$$

In this paper, we consider only $\mathbb{R}$-valued weights, and for now we let $\mathbf{b}$ be the zero-vector $\mathbf{0}$, so we can re-write the above as

$$\mathbf{a} = \mathbf{W}(\mathbf{z}_r \odot \cos(\mathbf{z}_\theta)) + i \cdot \mathbf{W}(\mathbf{z}_r \odot \sin(\mathbf{z}_\theta)) \tag{3}$$

Where $\odot$ is an element-wise product, and $\mathbf{z}_r$ and $\mathbf{z}_\theta$ are the vectors of element-wise magnitudes and angles of the complex vector $\mathbf{z}$, respectively. We define $\mathbf{a}_{real} = \mathbf{W}(\mathbf{z}_r \odot \cos(\mathbf{z}_\theta))$ and $\mathbf{a}_{imag} = \mathbf{W}(\mathbf{z}_r \odot \sin(\mathbf{z}_\theta))$, which lets us re-write $\mathbf{a}$ as $\mathbf{a} = \mathbf{a}_{real} + i \cdot \mathbf{a}_{imag}$

Because $\mathbf{a}$ is a complex vector, each of its elements has a radius and an angle. Let $\mathbf{a}_r$ and $\mathbf{a}_\theta$ be the element-wise radius and angle vectors of $\mathbf{a}$, respectively. More concretely, the k-th elements of $\mathbf{a}_r$ and $\mathbf{a}_\theta$ are

$$a_{r_k} = \sqrt{{a_{real_k}}^2 + {a_{imag_k}}^2} \tag{4}$$

$$a_{\theta_k} = \text{atan2}(a_{real_k}, a_{imag_k}) \tag{5}$$

Where atan2 is the 2-argument arctangent function. Now, we can introduce the CV-nonlinearity used for our model $g : \mathbb{C} \to \mathbb{C}$, defined as

$$g(\mathbf{a}) = \mathbf{y}_f \odot \cos(\mathbf{y}_\omega) + \mathbf{y}_f \odot \sin(\mathbf{y}_\omega) \tag{6}$$

with

$$\mathbf{y}_f = f_{max} \cdot \tanh(\frac{\mathbf{a}_r}{f_{max}}) \tag{7}$$

$$\mathbf{y}_\omega = 2\pi \cdot (\frac{\mathbf{a}_\theta}{2\pi} \bmod \frac{1}{\mathbf{a}_r}) \tag{8}$$

In this formulation, f_{max} is the maximum firing rate of a neuron, and the tanh function serves the same role as a saturating non-linearity that controls overall activity in the network. Figure 1 provides a visualization of the domain (all complex numbers, i.e. the red and blue regions) and the range (only the blue region) of the our CV-nonlinear function. We note that the nonlinearity we use is actually differentiable, meaning that a network of CVNs can be trained using gradient descent, without any modification. There are a number of benefits to the neuron model outline above from the perspective of biological plausibility and interpretability, in particular in comparison to CTRNNs (Beer, 1995) . First, $\mathbf{a}_r$ is always positive, and since $\tanh(x) > 0 \; \forall x > 0$, we are never faced with the situation of trying to find an interpretation of negative outputs for neurons; the output is always positive, so an interpretation as a firing rate is always unproblematic, at least from a mathematical point of view. Second, the CVN's *state* is equal to a bounded function rather than having the state *derivative* equal to a bounded function, as is the case in CTRNNs. This is beneficial, because the state space of a CTRNN can technically be unbounded over time, meaning that dynamics can, in theory, diverge to either infinite negative or positive values. The dynamics of a neuron governed by g however are bounded to the blue region (in the case of $f_{max} = 10$). Biological neurons themselves are only capable of finite firing rates, so it makes sense to directly enforce this constraint with a saturating nonlinearity.

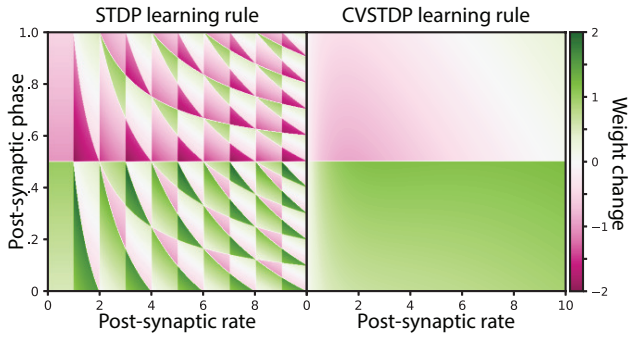

Figure 3: The pre-synaptic neuron has a fixed firing rate equal to 1 and a fixed firing phase = $\frac{1}{2}$. The post-synaptic rate varies from 0 to 10, while the post-synaptic phase varies from 0 to 1. There is a large amount of detailed structure in real STDP learning that is lost in CVSTDP learning, though the overall shape of the potentiation distribution is preserved.

Learning rule

Our choice of neuron model has radically reduced the complexity of the state-space of a spiking neuron down to just two dimensions, but even so, the dynamics of STDP are non-trivial. It is possible to explicity generate the spike-train from a given firing rate and phase offset, and then directly calculate the weight change due to a given STDP learning rule. Doing so is computationally quite expensive from the point of view of a Machine Learning update rule, and furthermore yields a very intricate dependency between the phase and rate of the pre- and post-synaptic neurons (see the left side of Figure 3 for an example, where the pre-synaptic neuron has a firing rate of 1 and a phase offset of $\frac{1}{2}$). Furthermore, there is evidence that the dependence of synaptic potentiation and depotentiation on relative timing of individual spikes breaks down for higher firing rates, and that there is a transition to a more rate-based outer-product Hebbian learning rule (Froemke et al., 2010). In order to capture at least some of the dependence of synaptic dynamics on temporal order, as well as model this transition to a simpler outer-product Hebbian rule at higher firing rates, we construct our STDP learning rule for CVNs as explicitly interpolating between two edge cases: STDP per-se, which heavily relies on spike timing, and simple rate-based Hebbian learning based on the product of firing rates. The classic form of STDP is

$$\Delta W = \begin{cases} A_+ e^{-\frac{\Delta t}{\tau_+}}, & \text{if } \Delta t \geq 0 \\ 0, & \text{if } \Delta t = 0 \\ -A_- e^{\frac{\Delta t}{\tau_-}}, & \text{if } \Delta t \leq 0 \end{cases} \tag{9}$$

Where Δt is the difference between the firing time of a post-synaptic and pre-synaptic spike, A_+ controls the strength of synpatic potentiation, A_- controls the strength

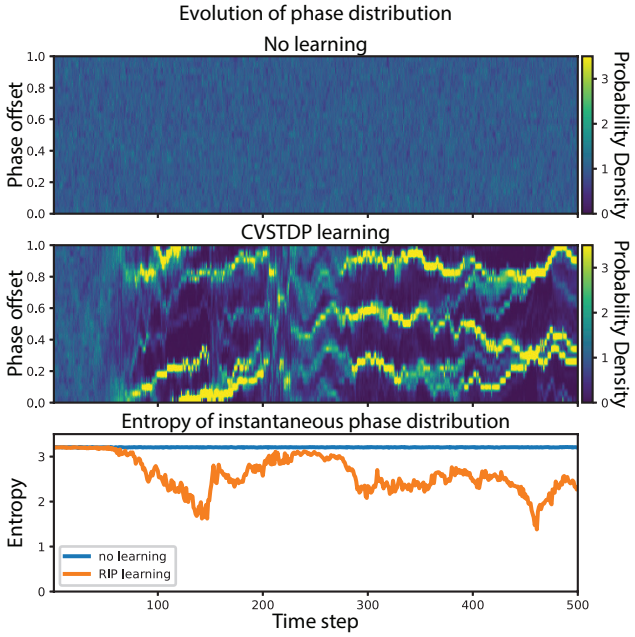

Figure 4: The top and middle images show the evolution of the distribution of phase offsets over time for a network of 1000 CVNs. The phase offset distribution is the empirical probability density of phase offsets. Brighter yellow colors indicate higher density, while darker blue indicates lower density. Without learning, the distribution of neuron firing phases has no obvious structure (top image). When CVSTDP-learning is introduced, complex phase synchronization spontaneously forms and un-forms (middle image). The phase distribution tends to rotate over time under the action of the affine transformation, making visual comprehension difficult. In order to clarify the structure in the phase distribution over time, we rotated the phase distribution at each time-step to maximize it's coherence with the previous time-step. The bottom image shows a plot of the entropy of the phase distribution over time both in the no-learning case (blue line) and the CVSTDP learning case (orange line).

of depotentiation, τ_+ is the time-constant of potentiation, and τ_- is the time-constant of depotentiation. For the purpose of fitting the dynamic range of this function within the temporal window of our model, we set $\tau_+ = \tau_- = \frac{1}{4}$ and $A_+ = A_- = 1$, which confines effective learning to cases were $|\Delta t| \leq 1$. In order to create an operation that will be easier to vectorize, we introduce a linear approximation of symmetric STDP that has the form

$$\Delta W' = 2\left\lfloor \frac{\Delta t}{2} \right\rfloor + 1 - \Delta t \tag{10}$$

Equations 9 and 10 are shown for comparison in Figure 2 to demonstrate that the simplification is in fact similar in overall shape to the original exponential form with the cho-

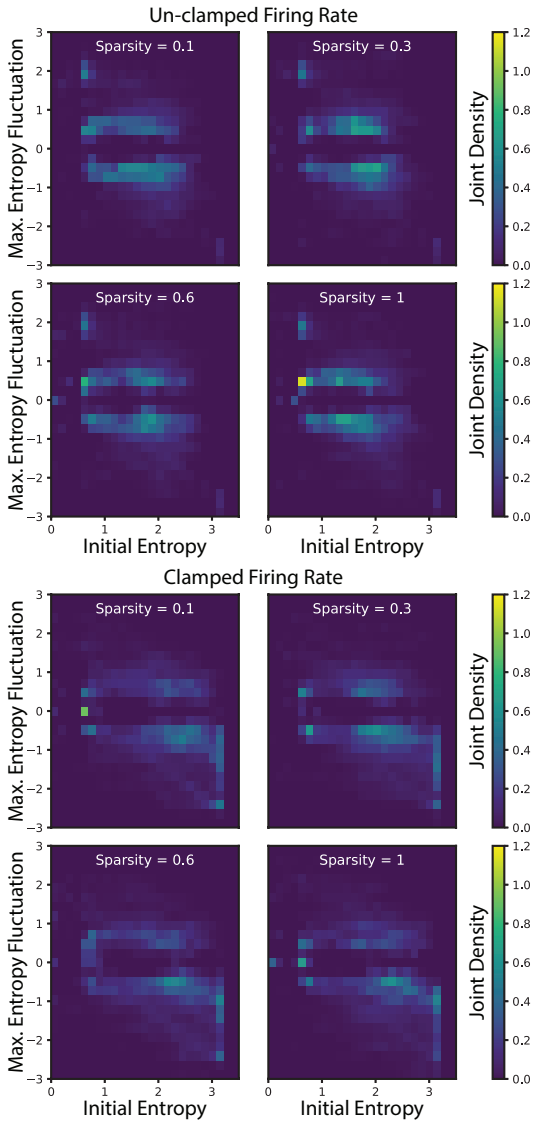

Figure 5: Each plot shows the joint probability distribution of entropy (x-axis) vs. largest future entropy fluctuation (in a time window K) (y-axis). High entropy implies low synchrony and visa-versa, so negative fluctuations are synchronizations, positive fluctuations are desynchronizations. There is relatively little change in the distribution over different levels of network sparsity in both clamped and unclamped cases. In the unclamped case we see a higher tendency for large spontaneous desynchronizations (higher-density area in top-left corner) vs. the clamped case, where spontaneous synchronizations are more common. Simulations run with CVSTDP.

sen parameters. The CVSTDP learning rule explicitly interpolates between this STDP term and a Hebbian term. The CVSTDP learning rule, given in terms of the pre-synaptic firing rate ${y_f}^{pre}$, the post-synaptic firing rate ${y_f}^{post}$, and the difference $\Delta\omega = {y_\omega}^{post} - {y_\omega}^{pre}$ between the post-synaptic phase ${y_\omega}^{post}$ and the pre-synaptic phase ${y_\omega}^{pre}$, is

$$\Delta W_{CVSTDP} = \gamma \cdot \alpha \cdot \delta_{HEBB} + \beta \cdot \delta_{STDP} \tag{11}$$

$$\delta_{HEBB} = {y_f}^{pre} \cdot {y_f}^{post} \tag{12}$$

$$\delta_{STDP} = \left\lfloor \frac{\Delta\omega}{2} \right\rfloor + 1 - \Delta\omega \tag{13}$$

$$\alpha = \frac{8 \cdot \phi \cdot \delta_{HEBB}}{(\phi + 2 \cdot \delta_{HEBB})^2} \tag{14}$$

Where γ and β are hyperparameters that determine the relative strength of the STDP and Hebbian terms, respectively (we set $\gamma = 10$ and $\beta = 100$ for all experiments). The α term controls the frequency-dependent transition from STDP-learning to Hebbian-learning, and is a unimodal function in the positive domain that has a maximum of 1 at 2ϕ, which makes ϕ another hyperparameter (we set $\phi = 4$ in all experiments). We note that all of these formulas can be calculated in a vectorized fashion, making computer implementation particularly easy.

Figure 3 demonstrates an example of the differences between STDP and the greatly simplified CVSTDP learning rule for a fixed pre-synaptic firing rate and phase. Note that while much of the detail of STDP is lost, the broad structure of depotentiation and potentiation is preserved.

Our aim in this paper is to demonstrate that CVNNs can be a useful tool for connecting the formalisms and insights associated with those formalisms in deep learning and computational neuroscience. To that end, we check that (1) a feedforward network of CVNs can successfully be trained using error-backpropagation by gradient descent to memorize a small toy dataset without any higher-order structure (noise vectors), and that (2) a network of recurrently connected CVNs can autonomously synchronize its activity.

Methods

All of our computational experiments were performed with a PyTorch (Paszke et al., 2019) implementation of the described CVN and CVSTDP rule. Weight matrices were always initialized with values drawn from a uniform distribution over the range $[\frac{-1}{\sqrt{N_{in}}}, \frac{1}{\sqrt{N_{in}}}]$, where N_{in} is the number of input neurons for a weight matrix.

Rate-Phase translation

Since the network model is end-to-end differentiable, we can train networks of CVNs using error-backpropagation, a result that has already been well explored elsewhere, see (Bassey et al., 2021b) for a review. Here, we demonstrate that networks of this type can learn to extract information encoded in firing rates, translate that information into an encoding using spike timings, and then translate this information back into firing rates. To do this, we built a CVN autoencoder (top image of Figure 7), and clamped the firing

rates of the neurons in the encoder network to 1, so that information could only be transferred through the network via phases. The input dimension of the autoencoder was 32 (32 input CVNs), with subsequent layers of 24 and 16 CVNs. The encoding layer of 8 CVNs also had firing rates clamped to 1. The dataset was a small set of random vectors, where the phases of the vectors were all zero, but the firing rates were random, taking values between 0 and 1.

We treated the autoencoder as a denoising autoencoder (DAE). The network was fed noisy versions of the dataset vectors, and trained with error backpropagation through stochastic gradient descent (SGD) to recover the noiseless vectors. This only makes sense because of the size of the dataset, for a larger dataset the network would have needed to generalize the dataset, but it's impossible to generalize noise, which is what our dataset is. We did not use the CVSTDP learning rule for this experiment.

Synchronization

We ran simulations of networks of CVNs with CVSTDP learning (not backpropagation) for 1000 timesteps, using 32, 100, 316, and 1000 neurons and differing levels of sparsity, i.e. the fraction of connections in a network that are clamped to 0 throughout the entire life of the network, with sparsity values of 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1. We then calculated for each time step in a given simulation, the discretized distributions of phase across the entire network (discretized with 25 bins). This allowed us to directly calculate the instantaneous entropy of the phase distribution (an example of which is shown in the top and middle images of Figure 4). Low entropy indicates a higher degree of synchronization while high entropy indicates a lower degree of synchronization (notice in Figure 4 that when there are phase clumps in the middle image, the entropy goes down in the bottom image). We observe in a representative time series of this distribution that the entropy has the potential for large fluctuations, indicating spontaneous increases and decreases in the phase coherence of the network, which we think can be interpreted as the formation and dissolution of neuronal phase assemblies.

In order to quantify these fluctuations, we did the following: For each time point in our series, we looked at the next K=200 time points, calculated the minimum and maximum values of the entropy over those time points, and found the range (minimum - maximum) of entropy over this time span. Then, we checked if the maximum occurred after the minimum (indicating a positive fluctuation) or before the minimum (indicating a negative fluctuation), which allowed us to assign to each point in a time series a signed maximum fluctuation magnitude. In this formulation, K acts as a scale term, since it controls the time horizon over which maxima and minima can be calculated. If we plot a two-dimensional distribution of the entropy value at a time-point vs. maximum entropy fluctuation over the next K time points, it can give us a sense of what kinds of fluctuations occur in the future for a given entropy value (examples are shown in Figure 5 and Figure 6).

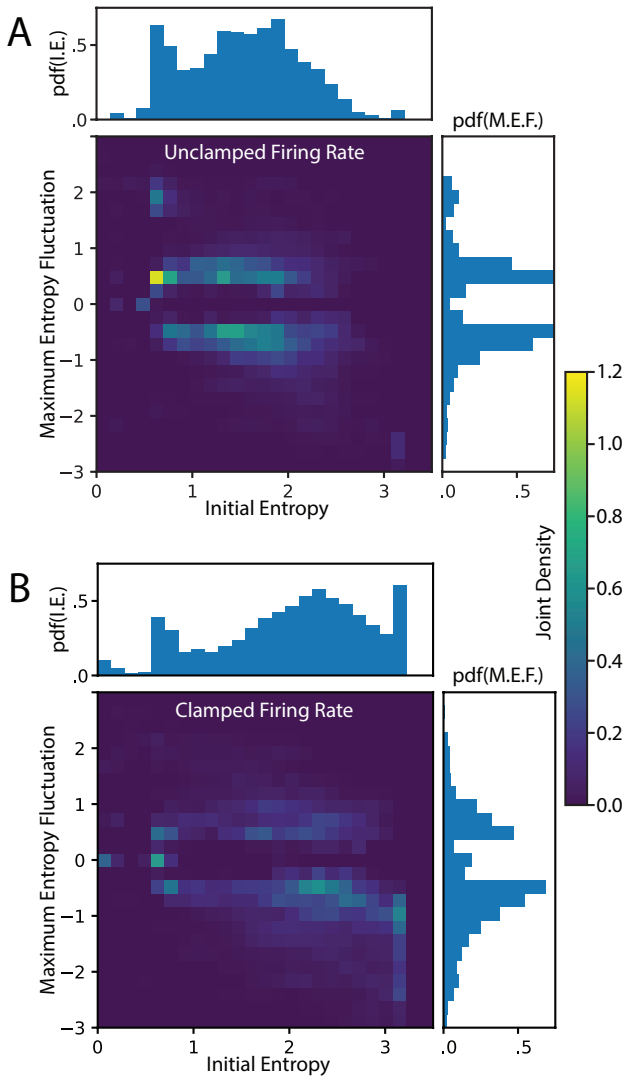

Figure 6: Detailed look at entropy vs. future entropy fluctuation distribution for dense 1000 neuron network. For both the unclamped and clamped firing-rate cases, we show the joint distribution (mostly blue image) and the marginal distributions of entropy (top histogram in each case) and future entropy fluctuation (side histogram).

Results

Here we present the result of some preliminary investigations of the properties of our CVN and the CVSTDP learning rule.

Rate-Phase translation

We were able to successfully store a small dataset of 32-dimensional random firing rate, zero-phase vectors in the CVN DAE. The network successfully memorized the dataset (sample error plot shown in bottom image of Figure 7), which is interesting because this means that the CVNs converted a spike-rate signal (firing rates taking values between 0 and 1) into a phase-coded signal (because no information could possibly be passed through the firing rate of the encoder neurons). The decoder network had unclamped firing rates, and was able to translate the 8-dimensional phase-encoded information back into 32-dimensional rate-encoded information.

Synchronization

Figure 4 shows an example of the evolution of the phase distribution for a network of CVNs without learning (top image) and with learning (middle image). Without learning, no clear structure in the phase distribution ever emerges, but with learning we see after about 100 time steps the formation of high-density areas of the phase distribution ("phase clusters"), indicating that many neurons have similar phase offsets. In many of our simulations, we see that the initial synchronization happens via a phase/anti-phase pairing. However, we often see much more complex (but non-uniform) phase distributions. For instance, the middle image of Figure 4 shows that after about 280 time-steps, 3 distinct "phase clusters" emerge. There appears to be complex reorganization of these phase clusters over time, with phase clusters splitting, drifting, and merging over time. It is possible that the larger reorganizations that are observed, such as the transition from two phase assemblies to three phase assemblies in Figure 4, are analogous to the meta-state transitions described in (Tseng and Poppenk, 2020).

For our investigation of the dependence of synchronization and desynchronization on hyperparameters we studied three main axes: was firing rate clamped or not, what was the enforced sparsity of the network connectivity matrix, and what was the maximum firing rate of the neurons (this last condition only makes sense when firing rates are not clamped). In general, we found relatively little change in the joint distribution of entropy vs. fluctuation with respect to sparsity (notice in Figure 5 that over different sparsity conditions the shape of the distribution is unchanged), but fairly large differences in this distribution with respect to presence/absence of firing rate clamping and maximum firing rate (compare the top four images with the bottom four images in Figure 5). In particular, it appears that between the clamped and unclamped cases, desynchronizations are favored in the unclamped case vs. synchronizations being favored in the clamped case (compare in Figure 6 the marginal distributions in of maximum entropy fluctuation $pdf(M.E.F.)$).

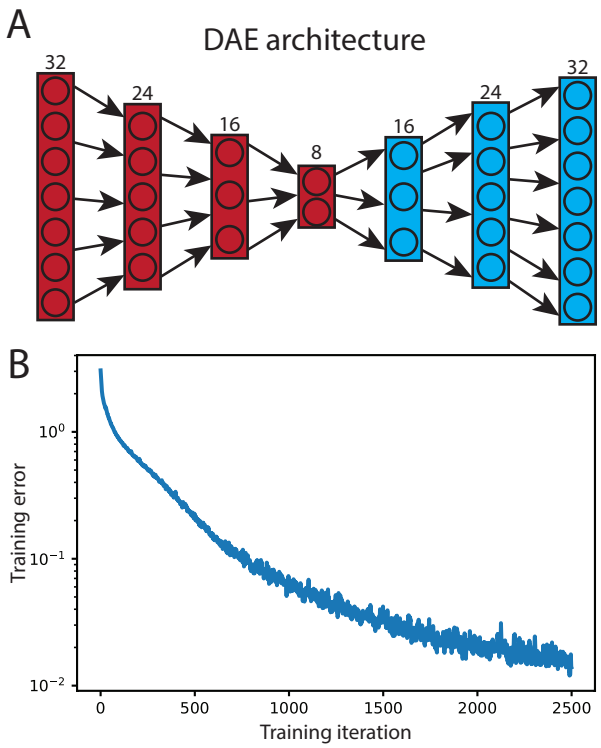

Figure 7: (A) The architecture of the CVN denoising autoencoder, with an input dimension of 32 and an code dimension of 8. The red layers had a firing rate clamped to 1, while the decoding (blue) layers were unclamped. (B) The model's error (MSE) over learning iterations.

it becomes more likely to enter low-entropy (high-synchrony) regimes, but it is also more likely that these high-synchrony regimes will spontaneously decay. When maximum firing rate is a little higher than 1, we find that no synchrony ever occurs in the time series.

Discussion

The spontaneous formation and dissolution of phase assemblies was suggested by (Woodward et al., 2015) as a possibly important component of a biologically plausible extension of Hopfield networks. We show here that simply allowing neurons to take values in $\mathbb{C}$ instead of just $\mathbb{R}$ enables this. While our network was given no extrinsic goals, these transient assemblies could be treated as the cell assemblies that Donald Hebb proposed as the basic building blocks of neural processing (Hebb, 1949). Following (Buzsáki, 2019), giving these transient assemblies meaning could be as simple as coupling a network of these neurons to an external environment, allowing the environment to "pick" different assemblies via temporal correlations. If environmental information was also encoded via a CVNN, we would expect similar dynamics as observed in our study, since the coupled

environment-network system would itself act like a single larger CVNN. If a particular behavior was desired, the observed variability of phase assemblies might support exploration of behaviors, while selection of particular behaviors could occur through reinforcement learning. This model is very amenable to interpretation via deep learning models, so it is fairly easy to reason about how information flows through networks of these neurons.

Discrete-time CVNs using CVSTDP leverage the mathematical simplicity of the networks used in machine learning, while also replicating some of the key properties of biological spiking neurons such as temporal coding of information and STDP. If it's the case that some aspects of cognition, learning, and behavior take advantage of these unique properties of real neurons, then explicitly modeling them as is done here might make construction of more sophisticated cognitive architectures simpler and more straightforward. For example, it seems that the order and relative timing of discrete events is important for causal reasoning and learning. A real-valued discrete feedforward network or a CTRNN could surely be trained to replicate this sensitivity, but perhaps it would be easier if the network had access to explicit relative timing information, as is the case for spiking networks with a temporally asymmetric learning rule.

There are several features of this model which render it unrealistic, and which might be overcome with simple, though ultimately computationally more costly, modifications. The most non-physical feature is that effectively, spikes in CVNs can suppress spikes in the past. Consider a CVN with strongly weighted (on the order of 1) connections from two input CVNs (call them n_a and n_b), such that a single spike (firing rate $f = 1$) from either neuron will illicit an in-phase spike. Suppose the output phase of n_a and n_b is $\omega_a = \frac{1}{4}$ and $\omega_b = \frac{3}{4}$, respectively. If our CVN received an input from n_a, it would fire a single spike with phase $\frac{1}{4}$. If it received an input from n_b, it would fire a single spike with phase $\frac{3}{4}$. n_a and n_b have opposite phases when projected onto the complex plane, and if they have the same magnitude, then their sum will be 0. This means that if our neuron receives input form both neurons, it will not fire. A physical interpretation of this is difficult to justify, though eliminating this effect would require something like performing a cumulative sum of inputs ordered by phase, which would be computationally expensive. Notably, (Reichert and Serre, 2013) proposed a solution to this problem, though we did not follow their formulation.

We find it interesting that the CVN autoencoder with clamped firing rates in the encoder was able to translate rate-encoded information into phase-encoded information and back again. The nonlinearity we use mixes information about phase and rate in the phase component via equation 8, which means that information in the frequency channel can theoretically be transferred to the phase channel. The nonlinearity itself, however, has no mechanism for transferring information from the phase to the frequency. Presumably, this must happen via interactions through the affine transformations of the network. The weight matrix operates on the real and imaginary components separately; however, the real and imaginary components of a complex number themselves represent a mixture of the phase and magnitude of the complex number. The presence of this phase-to-rate/rate-to-phase transition in CVN networks suggest the possibility of the same phenomena in biological brain networks. This could be used by neural networks to segment different information streams, allowing for something like the dynamical emulation of virtual parallel networks in the same physical network. Information encoded in phase is possibly sensitive to co-incidence in a way that information encoded in rate is not, and on the other hand information encoded in firing rate is amenable to relative comparison in a way that might be difficult for phase-encoded information. Enabling translation between these two domains might allow for complex logical comparisons embedded in dynamical systems. Future investigations should explore the robustness of this phenomenon and possible computational and representational consequences.

While training the DAE, we found that enabling biases eased training. A possible biological interpretation of this is that biases that adjust the phase of firing act like transmission delays. A further extension would be to allow $\mathbf{W}$ and $\mathbf{b}$ to take values in $\mathbb{C}$. Doing this would effectively allow for learnable transmission delays between neurons, although this extra complication might motivate a more sophisticated mechanism for handling phase-overshooting (going over 1 in our model). Since currently phase wraps around back to 0, delaying a spike enough can actually result in it happening earlier in the time-bin. An alternative that might be better motivated in the context of complex-valued parameters might be to allow for a spike to spill over into the next time-bin, though again this would increase the computational complexity of the network.

For the synchronization experiments, we observed that clamped-firing rate networks more commonly densynchronized as compared to unclamped networks. This is likely because in the clamped case, low-magnitude ($f \ll 1$) vectors that are out-of-phase cannot be as easily recruited, because their magnitude is reset to 1, increasing their influence.

We found that under the CVSTDP learning rule, the spectral radius of the weight matrix (absolute value of the largest eigenvalue) of the network tended to increase over time, which has actually been observed in symmetric STDP learning rules. A future extension of this work might try and counteract this tendency by adjusting the learning rule itself, or simply implementing some form of weight decay or maybe even transient re-sets of the spectral radius.

A PyTorch implementation of the CVN neuron and CVSTDP learning rule is available on the ECSU gitlab (https://gitlab.com/oist-ecsu/) in the DT_STDP repository.

References

Arjovsky, M., Shah, A., and Bengio, Y. (2016). Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128. PMLR.

Bassey, J., Qian, L., and Li, X. (2021a). A survey of complex-valued neural networks. *ArXiv*, abs/2101.12249.

Bassey, J., Qian, L., and Li, X. (2021b). A survey of complex-valued neural networks. *arXiv preprint arXiv:2101.12249*.

Beer, R. D. (1995). On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509.

Buzsáki, G. (2019). *The Brain from Inside Out*. Oxford University Press.

Danihelka, I., Wayne, G., Uria, B., Kalchbrenner, N., and Graves, A. (2016). Associative long short-term memory. In *International Conference on Machine Learning*, pages 1986–1994. PMLR.

Froemke, R. C., Debanne, D., and Bi, G.-Q. (2010). Temporal modulation of spike-timing-dependent plasticity. *Frontiers in synaptic neuroscience*, 2:19.

Funahashi, K.-i. and Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806.

Hebb, D. O. (1949). The organization of behavior; a neuropsycholocigal theory. *A Wiley Book in Clinical Psychology*, 62:78.

Inglebert, Y., Aljadeff, J., Brunel, N., and Debanne, D. (2020). Altered spike timing-dependent plasticity rules in physiological calcium. *bioRxiv*.

Kratsios, A. (2021). The universal approximation property. *Annals of Mathematics and Artificial Intelligence*, pages 1–35.

Maass, W. and Markram, H. (2004). On the computational power of circuits of spiking neurons. *Journal of Computer and System Sciences*, 69(4):593–616.

Nitta, T. (2002). On the critical points of the complex-valued neural network. In *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02.*, volume 3, pages 1099–1103. IEEE.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Rahman, M. and Geiger, D. (2016). Quantum clustering and gaussian mixtures. *arXiv preprint arXiv:1612.09199*.

Rao, A. R. and Cecchi, G. A. (2010). An objective function utilizing complex sparsity for efficient segmentation in multilayer oscillatory networks. *International Journal of Intelligent Computing and Cybernetics*.

Reichert, D. P. and Serre, T. (2013). Neuronal synchrony in complex-valued deep networks. *arXiv preprint arXiv:1312.6115*.

Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926.

Tavanaei, A. and Maida, A. (2019). Bp-stdp: Approximating backpropagation using spike timing dependent plasticity. *Neurocomputing*, 330:39–47.

Tseng, J. and Poppenk, J. (2020). Brain meta-state transitions demarcate thoughts across task contexts exposing the mental noise of trait neuroticism. *Nature communications*, 11(1):1–12.

Ward, L. M. (2003). Synchronous neural oscillations and cognitive processes. *Trends in cognitive sciences*, 7(12):553–559.

Woodward, A., Froese, T., and Ikegami, T. (2015). Neural coordination can be enhanced by occasional interruption of normal firing patterns: A self-optimizing spiking neural network model. *Neural Networks*, 62:39–46.

Woźniak, S., Pantazi, A., Bohnstingl, T., and Eleftheriou, E. (2020). Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nature Machine Intelligence*, 2(6):325–336.