Deep Phasor Networks: Connecting Conventional and Spiking Neural Networks

Wilkie Olin-Ammentorp  
Department of Medicine  
University of California, San Diego  
La Jolla, CA  
wolin.ammentorp@health.ucsd.edu

Maxim Bazhenov  
Department of Medicine  
University of California, San Diego  
La Jolla, CA  
mbazhenov@health.ucsd.edu

Abstract—Artificial neural networks (ANNs) are powerful but require many orders of magnitude more energy than biological systems capable of solving similar tasks. One critical difference is that ANN units communicate using continuous signals, as opposed to the binary spike events employed by biological networks. In this work, we extend standard ANN design by building upon an assumption that neuronal activations correspond to the angle of a complex number lying on the unit circle, or ‘phasor’ – the same domain employed by the Fourier Holographic Reduced Representation (FHRR) Vector-Symbolic Architecture (VSA). Each layer in such a network produces new activations by taking a weighted superposition of the previous layer’s phases and calculating this sum’s phase value. This generalized architecture allows models to reach high accuracy and carries the singular advantage that mathematically equivalent versions of the network can be executed with or without regard to a temporal variable. Importantly, the value of a phase angle in the temporal domain can be sparsely represented by a periodically repeating series of delta functions or ‘spikes.’ We demonstrate the atemporal training of a phasor network on standard deep learning tasks and show that these networks can then be executed in either the traditional atemporal domain or spiking temporal domain with no conversion step needed. This provides a novel basis for constructing deep networks which operate via temporal, spike-based calculations suitable for low-energy neuromorphic computing hardware.

Keywords—neural network, temporal executing, spiking, neuromorphic

I. INTRODUCTION

Efficient learning and inference processes remain a challenge for deep-learning based artificial intelligence methods [1]. Problems include poor generalization beyond properties of the training data [2], catastrophic forgetting during new task learning [3], lack of transfer of knowledge [4], very high-energy consumption [5]. Many efforts focus on addressing this by carrying out deep learning via the use of networks which communicate via binary events - ‘spikes,’ bringing artificial neurons closer to their highly efficient biological counterparts [6]–[8]. While recent efforts to link traditional deep neural networks (DNNs) to spiking neural networks (SNNs) via the usage of rate-coding have yielded SNNs which can attain high accuracy, this achievement comes with several caveats. Firstly, converting a suitable DNN to a rate-based SNN can require complex conversion methods to normalize weights and activation values [9]–[11]. Secondly, the resulting spike-based networks lack many established characteristics of biological computation, such as fast inference and sensitivity to the timing of individual spikes [12]–[16]. Lastly, even when executing on specialized neuromorphic hardware, these spiking networks require processing long sequences of spikes to evaluate rate and as a result provide at best a marginal gain in energy efficiency when compared to traditional networks running on conventional hardware [17]. For these reasons, to create a spiking network which can achieve the goals of neuromorphic computing (such as high performance, energy efficiency, and biological relevance), alternate approaches from rate-coding will likely be required [18]–[21]. Latency-based codes can be applied to resolve some of these issues, but sacrifice robustness for efficiency; the loss of a single spike in a latency code greatly changes the meaning of a transmitted message [22], [23]. To provide maximum utility, spike-based computing would ideally use a code which is both efficient and robust.

We build on previous works which assume that the state of a neuron may be represented by the angle of a complex number, commonly referred to as a “phasor” [24], [25]. Phasors are often used in electrical engineering and physics to provide convenient representations and manipulations of sinusoidal signals with a common frequency [26]. A phasor describes a sinusoidal signal’s phase relative to a reference signal, and can be scaled by a real magnitude to describe any complex number in polar form.

\[
A \angle \phi = Ae^{i(\omega t + \phi)} = \cos(\omega t + \phi) + i \sin(\omega t + \phi) \tag{1}
\]

Any set of sinusoidal signals with a common frequency can be represented by a vector of phasors, and as their superposition produces another sinusoid, it can also be represented by a single new phasor. This value is calculated by summing the original vector of phasors in their complex form. The phase of the resulting complex number is calculated via a non-linear trigonometric operation, allowing the superposition of phasors to form the basis of a neuronal activation function suitable for a deep neural network. The representation of information through phasors and non-linear properties of their superpositions provide the basis of information processing within phasor networks. While the apparent phase of a signal varies with respect to time, either an absolute starting time or reference signal can be used to decode phases from a set of time-varying signals.

In this work we propose an extension of the “classic” deep neural network architecture by replacing real-valued activations by phasors. We describe the operations needed to calculate propagation of the activation through the phasor network and show that the phasor network can be executed either in an atemporal domain or temporal domain. In the former, values are passed between layers in standard tensors of phasor values. In the latter, values are passed between layers by series of precisely-timed binary spikes, referred to as ‘spike trains’. Atemporal execution is well-suited to existing computer architectures (e.g. CPU and GPU), and temporal execution is suited to current or future neuromorphic platforms. We demonstrate that either...
execution mode leads to similar performance on standard machine learning datasets. We compare the temporal phasor networks to alternatives employing different representations and show that phasor networks offer a unique trade-off between efficiency and robustness. Furthermore, this architecture shares a number of benefits with the Fourier Holographic Reduced Representation, a Vector-Symbolic Architecture, potentially allowing it to be integrated with novel VSA-based computing architectures [27], [28].

RESULTS

A. Atemporal Evaluation

1) Activation Function

Let us assume that the input to a single neuron consists of a vector of phases, \( \mathbf{x} \). For convenience and ease of integration within existing deep learning frameworks, in this work all phase angles are reported after being normalized by \( \pi \), so that \( \mathbf{x} \in [-1,1] \). Multiplying these values by \( \pi \) converts to a standard angle in radians.

To compute a single neuron’s output \( y \) given an input vector \( \mathbf{x} \) with \( n \) elements, the real valued phases \( \mathbf{x} \) are converted into an explicit complex representation. These complex elements are then scaled by a vector of corresponding weights \( \mathbf{w} \), which we currently restrict to be entirely real-valued. The sum of the scaled complex elements produces a new complex value \( \mathbf{x}' \) (2). To extract only its phase, the two-argument arctangent (atan2) is used (3). Through these operations, a nonlinear neuronal operation is obtained. Additionally, the local continuity of these operations ensures deep networks employing them can learn each neuron’s weights \( \mathbf{w} \) using standard backpropagation techniques.

\[
\mathbf{x}' = \sum_{i=1}^{n} \mathbf{w}_i e^{i\pi x_i} \quad (2)
\]

\[
y = \text{atan2}(\text{Im}(\mathbf{x}'), \text{Re}(\mathbf{x}')) \quad (3)
\]

2) Output and Loss Functions

In an image classification task, the output of a standard network is often a vector of real values normalized between zero and one using the softmax function [29]. The outputs of this network are then taken to be the probability that the input image belongs to the corresponding output class, and a loss function using cosine similarity between the network’s predictions and the input image is often used. A neuronal dropout rate of 25% is used for all neurons. A ReLU activation function is used, and in phasor networks, the previously described activation method is disabled for all neurons. A neuronal dropout rate of 25% is used for regularization (Figure 1a). In the standard networks, a ReLU activation function is used, and in phasor networks, the previously described activation method is used.

In these results, we train groups of 12 models and report the group’s mean accuracy plus or minus its standard deviation. Phase projection layers are included even in standard networks as they can affect test accuracy, particularly for the MNIST dataset.

3) Image-to-Phase Conversion

One issue in implementing phasor-based networks arises at the input layer of such a network. Inputs such as images are almost always encoded on a domain with pixel intensities normalized between 0 and 1. However, as previously stated, a phasor network utilizes inputs on the domain \([-1,1]\). We show below that an initial conversion step between domains can assist phasor networks in reaching performance levels that match conventional networks. A simple linear scaling \((2x - 1)\) between domains is insufficient, as this will lead to ones and zeros being encoded into phases which have an identical cosine similarity (-1 and 1). Instead, we utilize two intensity-to-phase conversion methods: the first is a ‘normalized random projection’ (NRP), and the second a ‘random pixel phase’ (RPP).

The NRP method constructs a random projection by sampling values from a uniform distribution on the domain \([-1,1]\). The input image is then multiplied by this square matrix to produce a new input vector of the same dimension with pixel data distributed across multiple values. A simplified batch-normalization layer with two moving moments learned during training (mean and standard deviation) is applied to the random projection to keep approximately 99% of projected phases within the range \([-1,1]\). Lastly, outlier values are clipped to this domain.

The RPP method randomly projects input pixels for all images by 1 or -1. Collectively, this spreads values across the domain \([-1,1]\) although the absolute range of an individual pixel does not increase.

4) Model Architecture and Accuracy

To show that deep phasor networks can be effectively trained using the approach described above, we train a series of standard and phasor-based image classifiers on the standard MNIST, FashionMNIST, and CIFAR-10 datasets [30]–[32].

First, we demonstrate a simple multilayer-perceptron (MLP) model with one hidden layer. The architectures of the MLP networks trained on MNIST-format images \((28x28x1\text{ pixels})\) are identical, consisting of an input layer, intensity-to-phase conversion method, hidden layer of 100 neurons, and an output layer of 10 neurons with biases disabled for all neurons. A neuronal dropout rate of 25% is used for regularization (Figure 1a). In the standard networks, a ReLU activation function is used, and in phasor networks, the previously described activation method is used.

In these results, we train groups of 12 models and report the group’s mean accuracy plus or minus its standard deviation. Phase projection layers are included even in standard networks as they can affect test accuracy, particularly for the MNIST dataset.

\[
\hat{c} = \text{argmin} |\text{abs}(\hat{y} - \frac{n}{2})| 
\]

(6)
Each layer is labeled with its output shape. (b) Classification accuracy of this model on the standard MNIST dataset and (c) FashionMNIST dataset over 12 trials. ‘Standard’ models use a ReLU activation function and phasor models use the phasor activation method described in the main text. NRP models apply a normalized random projection to convert intensities to phases and RPP models randomly select pixels to produce positive or negative phases. These results suggest that the phasor activation function can provide an effective basis for constructing deep networks.

With an NRP conversion, the standard models reached a test accuracy of 96.4±0.1% on standard MNIST after 2 epochs of training. Similarly, the phasor models reached a median accuracy of 95.0±0.3%. Training instead on FashionMNIST dataset, the standard models with NRP conversion reached 85.7±0.4% accuracy on the test set. The highest-performing phasor networks for FashionMNIST used the RPP conversion, with an accuracy of 83.7±0.7% (Figure 1).

To classify the CIFAR-10 dataset, a convolutional architecture was used. It consists of an input block, two convolutional blocks, and a dense output block (Figure 2a). The input block consists of an optional phase projection method followed by a batch-normalization step. The first convolutional block consists of two convolution layers with 32 channels and 3x3 kernels, followed by a 2x2 max-pool and 25% dropout. The second convolutional block is identical but uses convolutional layers with 64 channels. An additional L2 regularization is applied to kernels in all convolutional layers. The dense block flattens the convolutional output, and applies a dense layer with 1000 neurons, a dropout of 25%, and a final dense layer of 10 neurons. Biases are again disabled on all neurons. All layers in this standard model use a ReLU activation except for the final dense layer which uses a softmax. No phase projection is used for the standard network, and RPP is used for the phasor network.

For the convolutional architecture, one additional change was made between standard and phasor-based networks. In the latter case, a minimum-pool (min-pool) was substituted for the more common max-pool on the basis that in the temporal representation of a phase value, lower values correspond to earlier spikes. In biological networks, earlier spikes would result from stronger synaptic inputs and would more likely contribute to the later processing while later spikes may be ignored or canceled [33]. Min-pooling also may be possible to approximate based on a combination of winner-take-all circuits and network dynamics [34]. We found that the substitution of a min-pool operation for max-pool operation in the phasor CIFAR-10 networks leads to no significant change in performance. However, average pooling leads to significant performance loss and was not used.

Both network types (standard and phasor) were trained using an augmented dataset to prevent overfitting. A rotation range of 15°, a width and height shift range of 10%, and 50% chance of a horizontal flip were used. All networks were trained using 70312 batches of 128 augmented images, corresponding approximately to 180 epochs on the original training dataset.

We found that in case of this more complex network architecture, no significant difference is observed between the standard and phasor networks’ performance on the test set (Figure 2b). This contrasts with small performance differences observed in the MNIST and F-MNIST tasks, and may be due to the higher regularization penalties imposed.
on both networks and the more complex image recognition task.

To summarize, these results demonstrate that applying phasor-based representation and using the phasor neuron described in Equations 2-3 can create networks which achieve results on-par with standard neural networks. Some differences in performance between phasor and standard networks can be attributed to many factors, including the image-phase conversion method, regularization techniques, and the dataset. Next, we investigate an alternate inference mode for phasor networks which is inherently temporal in nature.

B. Temporal Evaluation

As shown in (1), the invariants of a temporal sinusoidal signal can be used to represent it in a compact, atemporal form which can be easily manipulated. This equivalence allows for the convenient analysis of systems such as alternating-current (AC) electrical circuits. And similarly to the case of AC circuits, a neural network utilizing phasor-based activations also has an equivalent, temporal form which may be implemented by the dynamics of a virtual or potentially physical system. This equivalence was previously reported by Frady et al., who used it to execute an associative memory [35]; here it is employed to execute deep neural networks.

In this section, we describe how one version of a temporal phasor model can be constructed. In contrast to the previous atemporal execution mode, phases are communicated between neurons not by tensors of floating-point values, but by periodic and precisely-timed binary ‘spikes.’ The integration of currents induced by these spikes excite the neuron to fire in response, allowing for an equivalent temporal calculation of what was previously computed solely via standard linear algebra. This equivalence allows for different hardware systems to carry out the computations required for a deep neural network.

1) Spike-Based Representation of Phase

In our model, the tensors of phase values communicated between neurons represent the relative phases between a sinusoidal signal and a reference. Previously, in atemporal model, each value was represented by a single, real number restricted to the domain $[-1, 1]$. However, the relative phase can be represented in a different form: an instantaneous pulse or ‘spike’ can be used to mark whenever a sinusoid reaches its maximum. This creates a sparse, periodic representation of the underlying signal which communicates the same phase information. For instance, a signal in-phase to a reference has an atemporal representation of $0$. In the spiking, temporal domain, this is represented instead by the spikes which occur in the middle of the period defined by the reference signal. Signals with earlier phase values will be represented by spikes that precede the cycle’s midpoint, and signals with higher values will be represented by spikes that follow it.

In this manner, the representation of a phase is changed from a continuous value to a series of instantaneous spikes. To compute activation function as was previously demonstrated, the phases represented by these signals must be weighted, superimposed, and the phase of the output signal calculated. In temporal model, this is accomplished by each spike exciting a sinusoidal current within a resonate-and-fire (R&F) neuron which then accumulates through time into a voltage. The peak of this voltage is detected and used in turn to create an output spike representing the same activation value as in the atemporal calculation. Below we discuss this operation in detail.

2) Equivalent Resonate and Fire Model

Taking a derivative of a complex number $z$ representing a harmonic signal with respect to time (1) gives (7), which is identical to the case of a resonate and fire (R&F) neuron with no ‘leakage’ (attraction to the rest state) [36]. Leakage can be re-introduced and it shows that the dynamics of the R&F neuron are inherently linked to the activation function forming the basis of inference in a phasor network. In the model of an R&F neuron with no leakage, the voltage-current oscillation produced after a current pulse is identical to the rotation of a phasor with respect to time represented in (8). Following the original R&F convention, we refer to the ‘current’ $U$ of an R&F neuron as the real part of its complex potential $z$, and its ‘voltage’ $V$ as the imaginary part of $z$ [36].

$$z = Ae^{i\omega t + \varphi}$$

$$\frac{dz}{dt} = (0 + ia)z \rightarrow (-b + ia)z$$

With this approach, the superposition and phase detection which was previously calculated atemporally using (2) and (3) can be carried out exactly in the temporal domain using an R&F neuron with no leakage. First, the duration of one ‘cycle’ of time in the system is defined as $T$, giving the neuron a natural angular frequency $\omega$ of $2\pi/T$. Phases of an input vector $x$ are represented by Dirac delta functions. The amount by which this spike is offset from the temporal midpoint of the period is set proportionally to its value (e.g. an $x$ of -1 is represented by a spike at the beginning of the period, and an $x$ of 1 a spike at the end). Essentially, a spike represents the ‘peak’ of its original sinusoidal signal. Real-valued weights are represented again with $w$ (9).

$$\frac{dz}{dt} = i\omega z + \sum_{i=1}^{n} w_i \delta(t - x_i)$$

$$z(T) = \sum_{i=1}^{n} w_i \cdot e^{-\pi x_i} \quad \text{if} \quad z(0) = 0$$

Integrating (9) through a single cycle ($t=0$ to $2\pi$) produces another form of the superposition of complex values (10, proof in methods). The timing of the inputs $x$ defines the phase of the output $z$, which can be detected after the integration period by determining the time when two conditions (such as $V > 0$ and $dV/dt = 0$) are met. This allows the full calculation required for a phasor neuron (complex superposition & phase measurement) to be carried out in the temporal domain using input spikes.

However, leakage must be retained within R&F neurons if they are to carry out different computations through time (allowing their potentials to gradually return to the initial condition required in (10)). By increasing the level of leakage $b$ in the R&F neuron from 0, the same approximate calculation can be carried out, ideally without having a major effect on the phase of the superposition. Too high a
leakage value will cause the ‘memory’ of the neuron to be too short, leaving it unable to calculate an approximation of (10) as the induced oscillations will decay too quickly to superimpose. However, too low a leakage value will prevent the neuron from returning to a rest state which is required to allow it to adapt flexibly to new computations through time. To strike a balance between these extremes, we use a leakage set to 1/5th the value of the integration period $T$ (Table 1).

The representation of an input spike by an instantaneous Dirac delta function at a time $x$ can be relaxed by convolving it with a kernel such as a box function ($\Pi(t)$ with width scale factor $s$). These alterations (leakage and box kernel) are included in (11), which is solved numerically through time to calculate the complex potential $z$ of a phasor neuron. To reiterate the other parameters in this equation, $T$ is the R&F neuron’s fundamental period, $b$ is a positive value which sets the neuron’s leakage, $w$ is the vector of the neuron’s $n$ real-valued input weights, and $t$ is time. In Table 1 we present the values used for these parameters in our experiments.

$$\frac{dz}{dt} = (-b \cdot T + i\frac{2\pi}{T}) z + \sum_{i=1}^{n} w_i \Pi(s \cdot t - x_i) \quad (11)$$

The spiking threshold of an R&F neuron is an important parameter. To meet the requirements of a phasor network, a spike is produced from the R&F neuron when it reaches a certain phase of its current/voltage oscillation. This phase can be found by determining when the neuron’s complex potential $z$ of a phasor neuron reaches its maximum, i.e., $\partial Im(z)/\partial t = 0$ and (b) has a positive imaginary value $Im(z) > 0$; (c) its voltage is above a set threshold ($Im(Z) > V_{th}$), and (d) the time after last preceding spike exceeds refractory period ($T/4$). The last condition reduces the occurrence of multiple spikes in the same period. This gradient-based method allows the output spikes to be sparsely produced without referencing an external clock. For conciseness and given their approximate equivalence, we term an R&F neuron using this spike detection method as a temporal phasor neuron.

| Parameter       | Value   |
|-----------------|---------|
| Period ($T$)    | 1.0 s   |
| Leakage ($b$)   | 0.2     |
| Box width scale ($s$) | 0.05 |
| Threshold ($V_{th}$) | 0.03 |
| Refractory Period | 0.25 s |

3) Demonstration of Equivalent Neuron

Next, we first demonstrate that a temporal phasor neuron can carry out the identity function: given an input consisting of binary spikes repeating once a period, a phasor neuron resonates with this stimulus to produce another series of output spikes (Figure 3a-b). As each input spikes once per cycle, it is the relative timing of each spike which communicates a corresponding phase. In this case, the excitement of three input spikes gives the neuron sufficient ‘momentum’ that it produces 7 output spikes before its potential decays below the spiking threshold (Figure 3b). Earlier spikes communicate low phase values, and later spikes higher ones (Figure 3a).

Each stage of integration produces a quarter-period of delay ($T/4$) relative to its input. Adding this delay relative to the absolute time reference of the network’s starting point, the phases of the output spikes can be decoded and compared to the ideal values encoded within the input spike trains. These decoded output phases successfully reproduce the values encoded in the input (Figure 3c). This result shows that the complex summation and phase-detection can succeed at integrating and replicating a single input. However, the superposition of multiple inputs being calculated correctly through time is not addressed here, and is evaluated in the next section.

4) Demonstration of Equivalent Layer

To demonstrate that beyond the identity function, a temporal phasor neuron can calculate an approximately correct weighted superposition of its inputs (as described in (11)), we create a series of neurons corresponding to a ‘layer’ in a conventional network. The input weights to this temporal layer are taken from the hidden layer of an atemporal phasor network after it was trained on the standard MNIST dataset. A series of stimuli representing random input phases are applied to each neuron, which produce a series of output responses (Figure 4a). The conversions between spikes and phases at each layer inputs and outputs are identical to what we previously described. However, each neuron is now a subject to spikes from multiple (784) sources and it has to accurately integrate the
weighted sum of these inputs to produce a correctly-timed output spike.

![Figure 4](image1.png)

**Figure 4**: (a) A spike raster shows one example of a spiking input which stimulates a layer of phasor neurons that produces a series of spikes in response. Horizontal red lines demarcate the boundaries between integration periods. (b) After several periods of integration, the spike phases decoded from the temporal network using an absolute time reference and the ideal values produced by the atemporal network are highly correlated. (c) This correlation reaches its peak value during the last executed integration cycle. Importantly, the approximations used in (11) (i.e., introduction of leakage and box kernel) do not prevent a temporal phasor neuron from producing a value which is highly correlated to its atemporal value ($R=0.89$ in the final integration period) (Figure 4c-e). This demonstrates that a temporally-executed layer of phasor neurons can produce a good approximation of its atemporal counterpart.

5) *Demonstration of Equivalent Network*

Given the approximate equivalence of a single phasor layer executed via spikes in the temporal domain to its atemporal execution mode (Fig. 4), we next tested performance of the full networks used for image classification tasks. The networks were created identically to the networks demonstrated in the experiments above: both the MLP and convolutional networks were trained in the atemporal domain using standard backpropagation to reduce the loss function described in 0. However, to test performance of the temporal model execution, instead of executing a trained network by passing tensors of values from layer to layer (atemporal execution), here the networks execute by sending precisely-timed binary spikes between layers (temporal execution). Network parameters remain identical between execution modes.

![Figure 5](image2.png)

**Figure 5**: (a) An input image from the MNIST dataset of the digit ‘1’. This image is (1) converted from intensities to phases and (2) into a spike train (b) which drives the phasor network in the temporal domain (c). Horizontal red lines demarcate the boundaries between integration periods. The network has been trained to produce an out-of-phase spike on the image label ‘1’, (d). (e) 2 sets of 8 networks trained on the MNIST and FashionMNIST datasets are evaluated in both their atemporal and temporal execution methods, and the resulting accuracies on the test set are reported. (f) Running in temporal evaluation mode, most models lost an average of only 0.57% accuracy compared to atemporal evaluation.

Inputs to the network executed in temporal domain are provided by stimulating the input layer with a series of spikes encoding the phase of input into their relative timing as previously described. These input impulses repeat a set number of times during the execution of the network – in this case, spikes representing the image are presented 5 times (Figure 5b). The predicted output class of an image is decoded from the output spike train by detecting the phase of spikes produced during the output layer’s last full execution cycle (i.e. the final full cycle before the simulation’s stopping time is reached). The phase is decoded by referencing to the initial start time, and the predicted label is defined by the neuron with the phase closest to the target value of $\frac{1}{2}$ (Figure 5d).

We find that despite the significant change in underlying execution strategy – from standard matrix multiplication and activation function to the integration of spike-driven currents and phase detection through time – the final accuracy of MLP networks differs little between execution modes (Figure 5f). After an input image is flattened and converted to spike trains, each spiking layer of the network can perform an integration through time with sufficiently high accuracy to produce the desired output (Figure 5a-d). Results from 2 sets of 8 networks trained on the standard MNIST or F-MNIST datasets show that the two execution modes are very similar and, on average, only 0.57% accuracy is lost by switching from atemporal to temporal execution (Figure 5e-f). Next, a phasor-based convolutional network was executed temporally to test consequences of having a wider and deeper network architecture when using the temporal execution method. To temporally execute the min-pool operation, spikes within the pooling groups were examined cycle-to-cycle, and the earliest spike time in the pool was selected to be output.

In this work, the same conventional CPU/GPU hardware used to simulate both temporal and atemporal networks. On
this hardware, the integration of a current through time requires more calls to a multiply-and-accumulate operation than the atemporal mode, but each operation is applied to inputs which are sparse: on average, few or no neurons fire during each integration step. In our current implementation we were not able to leverage this sparsity, leading to a simulation of temporal networks which was slower than the atemporal mode. For this reason, the temporal execution of the convolutional network with 70,514 neurons (versus 110 for the previous networks) was limited to a smaller test set of 1024 images.

Running a single network on this reduced CIFAR test set, the network reached accuracies of 71.4% and 73.5% for temporal and atemporal execution modes, respectively. This corresponds to 2.1% accuracy loss by switching to the temporal execution mode. Importantly, we expect execution time of the temporal mode and its efficiency to improve greatly when using neuromorphic hardware which specializes in executing with sparse signals and avoiding the high-cost movement of data which dominates energy consumption in network execution [37]–[39].

The greater depth and width of the network used for CIFAR data set allows for a more in-depth investigation of the propagation of information through layers with respect to time. This was measured by calculating the mean squared error (MSE) between the phases encoded by spikes in the network using temporal mode and the ideal phases calculated in the network under atemporal execution. We found that the initial execution cycles show the largest decreases in MSE, which continue to decay in time as more spikes propagate forward (Figure 6a). Different layers reach different lower bounds in error, and these differences stem from the approximate integration carried out by the resonate-and-fire neuron. A low error (approximately 1% MSE) is required at the dense output layer in order to reach temporal mode network classification performance comparable to atemporal execution mode (Figure 6b). As the temporal execution network approximates the same calculations carried out under atemporal execution, misclassified images are usually shared between execution modes (Figure 6c).

6) Efficiency & Sensitivity Analysis

There exist a variety of methods which allow for the execution of deep neural networks via sparse, spike-based dynamics. Each method is often related to the encoding of information in spikes via a certain coding scheme [23]. Methods utilizing rate-based and time-to-first-spike (TTFS) coding have previously been successful in allowing spiking neural networks to achieve high accuracies on image classification tasks [10], [22]. However, each method carries a set of trade-offs between factors such as accuracy, efficiency, robustness, and more.

Rate-coding requires moderate integration periods in order for each neuron in the network to accumulate sufficient spikes from the previous layer to begin producing an accurate output in turn [10]. This can limit the propagation of information from layer to layer, and different inputs can produce different spiking statistics (e.g. brighter images with higher initial activations may require more spikes). High spiking rates may lead to issues such as routing congestion on a neuromorphic chip [21]. However, the redundancy of information within a rate-coded network can provide robustness of execution. Perturbation of an individual spike’s timing is not highly detrimental, as only the long-term accumulation of spikes is used as the basis of computation.

In contrast, temporal codes use the timing of spikes relative to a local or global reference to communicate values. Time-to-first-spike (TTFS) is a common temporal code, in which the duration between a global starting point and the arrival of a single spike is used to encode values. This can greatly reduce the number of spikes required for a spiking network inference compared to a rate-coded equivalent [22]. However, the sparsity enforced by this
scheme gives it a higher vulnerability to perturbations such as the loss of spikes or disruptions in their timing; in contrast to rate-coding, the delay or loss of a single spike will communicate a different value.

We posit that phasor networks can strike a balance between the robustness provided by rate-coding and the efficiency provided by TTFS. The coding scheme used in phasor networks enforces one spike per integration period, but multiple integration periods are used per inference (Figure 7b). The ‘momentum’ of a neuron’s potential from one cycle to the next can, over time, allow it to compute accurate outputs despite perturbations such as the loss of spikes between hidden layers (Figure 7a) or slight jitter added to the timing of spikes (Figure 7b). Additionally, one potential challenge of temporal codes is that they may require a higher level of time discretization used in their execution to be effective. We find that this is not the case for phasor networks; in the networks tested, approximately 40 points per integration cycle were found to be sufficient to execute both the MLP and convolutional networks (Figure 7c).

![Figure 7](image)

**Figure 7:** The effect of various perturbations on a network’s performance is measured by comparing the performance of the perturbed network in comparison to its original value. (a) By randomly removing spikes transmitted between layers, the sensitivity of phasor networks to loss of information between neurons is estimated. Networks remain robust to a moderate amount of dropout, and higher levels of dropout begin to affect performance more severely. These higher levels of dropout may not provide many neurons with sufficient drive to fire, leading to a larger loss in accuracy. (b) Randomly perturbing the timing of spikes produced by each layer estimates the sensitivity of the network to disruption of a temporal nature, such as random delays. Networks can tolerate a small amount of jitter, with the deeper convolutional network exhibiting a higher sensitivity to disruptions. (c) Related to timing jitter is the precision of the underlying integration being carried out. By varying the number of points per integration cycle used to calculate a solution, we find that 40 or more steps per cycle appears to be sufficient. (d) Synaptic operations provide a basis to estimate the computational resources required for a task (in this case, a classification inference). In comparison to a rate-coded network, a convolutional phasor network requires 29% of the synaptic operations to reach high accuracy.

While these points demonstrate the robustness of phasor networks against perturbations, it does not yet demonstrate their efficiencies relative to rate-coding or other temporal codes. By utilizing open-source software provided by Rueckauer et al., we created an alternate version of our convolutional network (Figure 2a) which executes via rate-coding [10]. The spikes produced by each neuron in conjunction its fanout was used to compute the number of synaptic operations required for each network in comparison to the accuracy of its current predictions. Our results suggest that phasor networks do indeed provide a ‘middle point’ between rate-coding and TTFS; to reach a high level of accuracy compared to a non-spiking version, a phasor networks requires approximately 3.4x fewer synaptic operations than a rate-code based equivalent (Figure 7d). This is lesser than the 7-10x reduction reported for TTFS encoding [22], but phasor networks are much more robust than TTFS and we believe that the efficiency of phasor networks may be further improved by incorporating sparsity schemes. This point is elaborated in greater detail in the discussion below.

II. DISCUSSION

In this study, we showed that the real valued activation function of a standard artificial neural network can be replaced by one that represents the angle of a complex value, or ‘phasor’ – the same domain employed by the Fourier Holographic Reduced Representation (FHRR) [40]. With this approach, each neuron in the network integrates the phases of a set of inputs and generates a new output phase, which then propagates to the next layers. This allows the network to operate in a traditional manner, in which inputs are presented and processed via tensors of floating-point values as is done in standard machine learning. However, the mathematical properties of the phasor-based activation function allow it to be approximately calculated through time by using resonate-and-fire neurons which integrate temporally sparse impulses. This makes phasor-based networks uniquely suited to processing temporally sparse, event-based data and execution on neuromorphic processors. Furthermore, we believe that this model brings deep learning closer to biological relevance while maintaining key advantages over other spike-based deep learning models.

A. Execution

One of the important advantages of phasor representation is its ability to present the network with a complete input within a set time interval defined by the neuron’s resonant frequency; each phase is encoded by the spike timing (its offset within the defined interval) and not by the accumulated rate of spikes (as is the case in rate-coding). This allows the network to produce outputs much more efficiently (as shown in Figure 5b) and on an established time basis, rather than only after waiting an arbitrarily defined amount of time. Additionally, the dynamic coupling between layers transmits information through the network with small delays, although time is required for the temporal outputs to reach a high degree of precision. Further latency characteristics still need to be tested, as well as the performance of phasor networks in modern, very deep networks with real-world applications such as image detectors [41].

An argument which can be presented against temporal phasor networks is that they trade off the long integration times of rate-based networks by instead using a continuous time domain which may consist of an equivalent or even greater number of discrete steps when calculated on digital, clocked hardware. This is a salient point which must be addressed by future work which examines in more depth the effect of time and weight quantization on the execution of phasor networks. However, even if using an equivalent number of time steps, phase-based encoding maintains the
of information in the FHRR vector-symbolic architecture [27], [28]. This enables vectors produced by each layer of a phasor network to be manipulated through vector-symbolic manipulations such as binding and bundling, allowing rich data structures to be built within the framework of the network’s evaluation [49]. These operations can be used to build new generation computation architectures with capabilities such as the factorization of components into symbols, linking them to the emerging field of neurosymbolic computation while maintaining spike-compatible computation [50], [51].

III. CONCLUSION

We demonstrated that by replacing the activation function of a standard feed-forward network by one created via complex operations, a novel ‘phasor’ network may be designed that can be executed either temporally or atemporally with no conversion process and only slight differences in output. The temporal execution mode’s spiking and oscillatory computations have strong parallels to biological computation and can be adapted to current or future neuromorphic hardware. Additionally, phasor networks’ domain is linked to a vector-symbolic architecture, providing a rich basis to apply them towards the construction of novel training methods and architectures.

IV. ACKNOWLEDGMENTS

This work was supported by NIH T-32 Training Grant (5T32MH020002), the Lifelong Learning Machines program from DARPA/MTO (HR0011-18-2-0021), ONR (MURI: N00014-16-1-2829), and Intel (00018020-001).

The authors declare no competing interests.

We would like to thank Friedrich Sommer, E. Paxon Frady, Stefan Preble, and Matthew van Niekerk for their comments and discussions related to this work.

V. REFERENCES

[1] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green AI,” Commun. ACM, vol. 63, no. 12, pp. 54–63, 2020, doi: 10.1145/3381831.
[2] I. Sucholutsky and M. Schonlau, “‘Less than one’-shot learning: Learning N classes from M-N samples,” arXiv, 2020, [Online]. Available: http://arxiv.org/abs/2009.08449.
[3] G. M. van de Ven and A. S. Tolias, “Three scenarios for continual learning,” pp. 1–18, 2019, [Online]. Available: http://arxiv.org/abs/1904.07734.
[4] G. Marcus, “The next decade in AI: Four steps towards robust artificial intelligence,” arXiv, no. February, 2020.
[5] E. Strubell, A. Ganesh, and A. McCallum, “Energy and policy considerations for modern deep learning research,” AAAI 2020 - 34th AAAI Conf. Artif. Intell., no. 1, pp. 1393–13966, 2020, doi: 10.1609/aaai.v34i01.1393.
[6] C. D. Schuman et al., “A Survey of Neuromorphic Computing and Neural Networks in Hardware,” pp. 1–88, 2017, [Online]. Available: http://arxiv.org/abs/1705.06963.
[7] C. S. Thakur et al., “Large-Scale Neuromorphic Spiking Array Processors: A quest to mimic the brain,” arXiv, vol. 12, no. December, pp. 1–37, 2018, doi: 10.3389/fnins.2018.00891.
[8] D. V. Christensen et al., “2021 Roadmap on Neuromorphic Computing and Engineering,” 2021, [Online]. Available: http://arxiv.org/abs/2105.05956.
[9] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer, “Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing,” Proc. Int. Jt. Conf. Neural Networks, vol. 2015-Septe, 2015, doi:
