Scaling Sparse Approximation with a Two-Layer Spiking Locally Competitive Algorithm

Albert Ting[1], Samuel Shapero[2]
[1]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA
[2]Georgia Tech Research Institute, Atlanta, USA

***Abstract*—Many applications, such as radio channel estimation, require solving for unknowns in overcomplete bases. Nonlinear solvers like Basis Pursuit Denoising (BPDN) can leverage sparse statistics to improve accuracy relative to linear solvers. The Locally Competitive Algorithm (LCA) is a nonlinear dynamical system that converges on the solution to BPDN, and can be implemented via a Spiking Neural Network that is orders of magnitude faster and more power efficient than CPU-based BPDN solutions. However, the Spiking LCA scales quadratically with the dimensionality of the state estimate, which can quickly make physical implementation impractical. In this work, we introduce a multi-layered complex-valued LCA architecture, which – by taking advantage of hierarchical sparsity in the channel estimation problem – allows sub-quadratic scaling of computational resource requirements, reducing resource requirements for a 357 complex channel estimation by 7.6x relative to a single layer solution. We implemented a 1470 complex channel, 2-layer Spiking LCA on Intel's Loihi chip, and demonstrated a 10x reduction in temporal smear relative to a linear solver.**

I. Introduction

Portable sensing systems are commonly constrained by their power consumption, as they do not have access to the power grid and batteries increase size and weight costs. Simultaneously, there is high demand for real-time processing of radio-frequency signals, especially with the advent of software-defined and 5G radio receivers.

Neuromorphic computing with spiking neural networks is an alternative to power-hungry artificial neural networks (ANNs) [1]–[3]. Spiking neural networks (SNNs) mimic the asynchronous, efficient, and sparse features of the brain by replacing the typical artificial neurons with time-dependent integrate-and-fire spiking neurons. Spiking neurons, when implemented on neuromorphic hardware, are more efficient than artificial neurons since interneuron messages are just binary spikes, and require no multiplication operations.

While SNNs can compute the same functions as ANNs and can solve similar problems [4], SNNs are particularly good at problems that are time-dependent and/or exploit a recurrent network structure [5]. One such problem in this class is Sparse Approximation, sometimes formalized as Basis Pursuit DeNoising (BPDN) [6], which can be used to decompose a signal into a sparse basis of a known dictionary. This aligns well with channel estimation, where the dictionary is a set of frequency and temporal offsets of a known, transmitted signal. More broadly, sparse approximation can be used to interpret cortical signals [7] and residual datastreams in large language models [8].

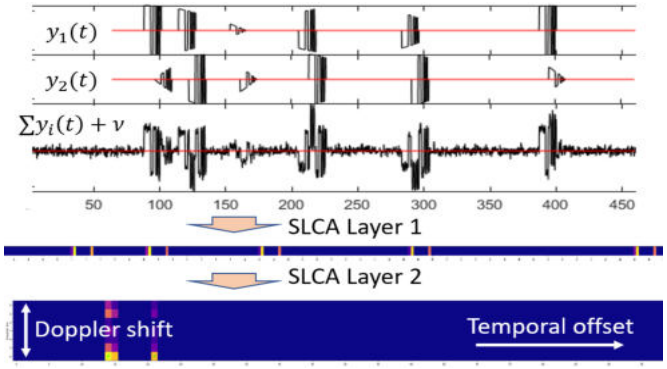

Fig. 1: The 2-layer Spiking Locally Competitive Algorithm (SLCA) is used to solve sparse overcomplete estimation problems, e.g. channel estimation. Signals consist of a sparse set of phase, time, and frequency offsets to a waveform ϕ. SLCA layer 1 computes sparse approximation on the short time features; layer 2 computes sparse approximation on the longer time features, resolving both time and Doppler offsets.

Rozell et al. [9] demonstrated BPDN can be solved with the Locally Competitive Algorithm (LCA), a fully recurrent ANN. The Spiking LCA [10], [11] benefits from both the temporal/power savings of being a recurrent SNN and from the fact that the solution is optimized to provide a sparse output – since most of the neurons don't spike, it uses dramatically less power.

Davies et al. [5] scaled the SLCA on Intel's neuromorphic Loihi chipset, showing 500x gains in speed and 100,000x gains in power efficiency. However, the SLCA also has a physical scaling challenge: because the network is fully recurrent, a naive approach requires $O(N^2)$ synapses, dramatically limiting the problem size achievable on a single chip.

In this paper we introduce a 2-layer SLCA, which decomposes a BPDN problem into smaller BPDN problems with fewer nonzero connections. We demonstrate estimation of 1470 complex channels on the Loihi chip (compare to 357 for a 1-layer SLCA). We further show that the 2-layer SLCA has superior performance to a linear solver in terms of temporal smear.

II. Sparse Approximation for Channel Estimation

In channel estimation, a radio transmitter produces a signal $\phi(t)$, and an emitter receives the modified signal $y(t) =$

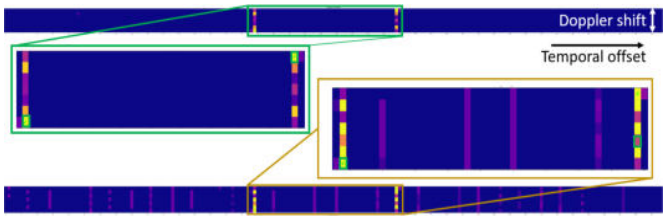

Fig. 2: The 2-layer SLCA (top, left insert) clearly identifies two true channels (green highlight), with only minor smearing in the adjacent Doppler bins, while a matched filter (bottom, right insert) leaves energy smeared across many channels.

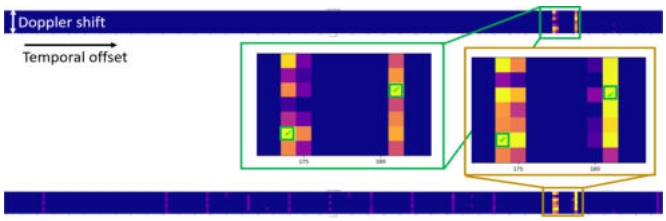

Fig. 3: The 2-layer SLCA (top, left insert) is able to clearly identify two channels in close temporal proximity, unlike a matched filter (bottom, right insert).

$H(\phi(t))$. We model the channel $H(\cdot)$ as the sum of a sparse number of linear channels with Additive White Gaussian Noise ν:

$$y(t) = \sum_i a_i e^{j2\pi(f_i t + \theta_i)} \phi(t - \tau_i) + \nu \quad , \tag{1}$$

where a_i is the amplitude gain, f_i is the Doppler shift, θ_i is the phase shift, and τ_i is the time delay of subchannel i.

This can be modeled as a linear, discrete valued equation:

$$\begin{aligned} y = \Omega x + \nu &= \sum_j D_j(\phi * x_j) + \nu \quad , \\ D_j &= diag(e^{j2\pi f_j t_k}) \, , \\ x_j &= \sum_{i, f_i = f_j} a_i e^{j2\pi\theta_i} \delta[t_k - \tau_i] \, , \\ x &= [x_1, \ldots, x_n]^{\mathrm{T}} \, . \end{aligned} \tag{2}$$

for discrete values of t_k and f_j, where x represents the sparse, complex gain for each discrete time and Doppler offset.

A traditional, linear approach to estimating x is to use a matched filter Ω^* [12], which is just the complex conjugate of $\Omega = [D_1\Phi, D_2\Phi, \ldots, D_n\Phi]$, where Φ is the Toeplitz matrix equivalent to convolution with ϕ. Because the dictionary Ω is overcomplete, (i.e. the number of observations is less than the number of unknowns), the resulting solution will have significant ambiguity. This ambiguity can be reduced by minimizing the cross-correlation between columns of Ω, guaranteeing that $||\Omega^*\Omega - I||_\infty < \epsilon$, known as the restricted isometry property [13].

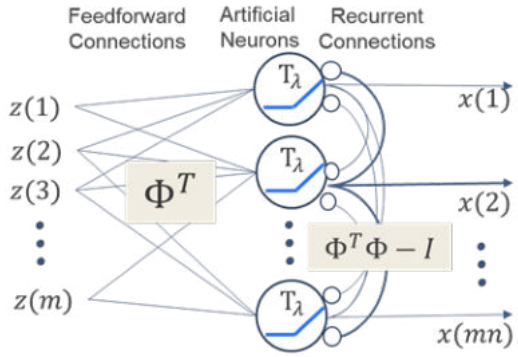

Fig. 4: The Locally Competitive Algorithm [11]. The input vector z is transformed by input weights to implement the $\Phi^T z$ operation. The result is then fed into the artificial neurons which implement the $T_\lambda(\cdot)$ operation. Then the neuron outputs are connected to each other as inhibitory connections in order to implement the $\Phi^T\Phi - I$ operation.

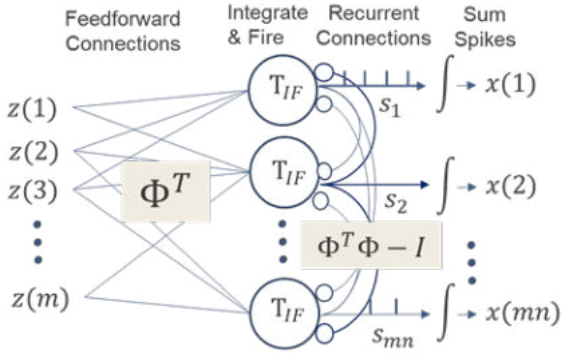

Fig. 5: The Spiking Locally Competitive Algorithm [11]. The implementation is analogous to the artificial neural network in Figure 4. The output is the spike rate of each neuron.

BPDN [6] is a technique for estimating unknowns with sparse statistics, where very few entries are nonzero. BPDN can be expressed as:

$$\begin{aligned} \hat{x} &= \arg\min_x \frac{1}{2}\|y - \Omega x\|_2^2 + \lambda\|x\|_1 \quad , \\ &= BPDN_{\Omega,\lambda}(y) \end{aligned} \tag{3}$$

where λ is a weighting parameter for the sparsity of the signal, modeled here by the $\ell 1$-Norm.

A. The Locally Competitive Algorithm

The Locally Competitive Algorithm (LCA) [9] is a system of nonlinear equations that converges on the solution to BPDN in finite time when Ω respects the Restricted Isometry Property. It can be described with the following equations:

$$\begin{aligned} T_u\dot{u}(t') + u(t') &= \Omega^* y - (\Omega^T\Omega - I)\hat{x}(t') \\ \hat{x}(t') &= T_\lambda(u(t')) \end{aligned} \tag{4}$$

where t' in the system's internal time, T_u is the convergence time constant, u is a hidden state variable, and T_λ is the soft-

threshold operator with threshold λ (similar to a rectified linear unit, but in both the positive and negative directions):

$$T_\lambda(x) = \begin{cases} \text{sign}(x)(|x| - \lambda), & \text{if } |x| > \lambda \\ 0\,, & \text{otherwise} \end{cases} \tag{5}$$

This network exponentially converges on a solution for $\hat{x}$.

The LCA has proven itself amenable to hardware implementations [10], including mathematically equivalent spiking neural networks (SNN) [11]. [5] and [14] demonstrated that a SNN-based LCA (SLCA) could outperform CPU-based BPDN solvers (such as FISTA [15]) in both speed and power costs, by multiple orders of magnitude, and that the relative advantage of the SLCA increases with scale.

B. Spiking Neural Network LCA Implementation

The LCA can also be implemented as a rate-encoded Spiking Neural Network as seen in Figure 5 [11].

The SLCA is a system of Integrate and Fire neurons and synapses described by:

$$\begin{aligned} T_u\dot{u}(t') + u(t') &= \Omega^* y - (\Omega^*\Omega - I)s(t') \\ v(t') &= \begin{cases} u(t') - \lambda - T_v\dot{v}(t')\,, & \text{if } v(t'^-) < 1 \\ 0\,, & \text{otherwise} \end{cases} \\ s(t') &= \begin{cases} 0\,, & \text{if } v(t'^-) < 1 \\ 1\,, & \text{otherwise} \end{cases} \\ \hat{x} &= \frac{1}{T_w}\int_0^{T_w} s\,dt' \end{aligned} \tag{6}$$

where $u(t')$, $v(t')$, and $s(t')$ represent the synaptic input current, internal voltage state, and binary output spike train of the assembly of neurons, as a function of system time t'. The output $\hat{x}$ is just the average spike rate of the neurons over the observation window $[0\ T_w]$.

C. Complex Valued Operation

One limitation in using the SLCA for channel estimation is that since spikes are binary, the weighted average output $\hat{x}$ can only be a non-negative real, which is obviously inadequate for estimating the complex values of x in (2).

This can be addressed by using four real,non-negative neurons to represent each complex-valued channel:

$$\begin{aligned} T_u\dot{u}_{R+} + u_{R+} &= \Re\{\Omega^* y\} - \Re\{(\Omega^*\Omega - I)\}s \\ T_u\dot{u}_{R-} + u_{R-} &= -\Re\{\Omega^* y\} + \Re\{(\Omega^*\Omega - I)\}s \\ T_u\dot{u}_{I+} + u_{I+} &= \Im\{\Omega^* y\} - \Im\{(\Omega^*\Omega - I)\}s \\ T_u\dot{u}_{I-} + u_{I-} &= -\Im\{\Omega^* y\} + \Im\{(\Omega^*\Omega - I)\}s \end{aligned}$$

The SLCA complex valued output is computed as:

$$\hat{x} = \frac{1}{T_w}\int_0^{T_w} s_{R+} - s_{R-} + j(s_{I+} - s_{I-})\,dt'\ . \tag{7}$$

Naturally, this increases the neuron count by a factor of 4, and the number of interneuron connections by a factor of 16.

III. Scaling Solutions

The primary scaling limitation for hardware instantiation of the SLCA is the lateral inhibitory term $(\Omega^T\Omega - I)$ in (6), which requires $O(N^2)$ interneuron synaptic connections. So, for example, the Loihi chip with 1,000,000 synapses can only have $N = 1000$ output neurons, or 250 complex channels.

In [16], this requirement was mitigated by use of a continuous Alltop waveform [17], where by clever factorization of Ω the number of synapses was reduced to $O(N\sqrt{N})$. They achieved a 1681 channel estimator on a single, custom chip. However, the factorization technique was only achievable due to a self-similarity property of that specific waveform, and furthermore required an equal number of time and frequency offset channels (41, as achieved). A more general solution is desired.

[5] introduced the convolutional SLCA, where Ω is constrained to be a block-Toeplitz matrix, where only a few blocks are nonzero. If ϕ has a only a few continuous, nonzero elements (so $l_\phi \ll l_y$), then Ω can be formulated as:

$$\Omega = \begin{bmatrix} 0 & \cdots & 0 \\ \omega & & 0 \\ & \omega & \vdots \\ 0 & & 0 \\ 0\ 0 & \ddots & \\ \vdots\ \vdots & & \omega \\ 0\ 0 & \cdots & \end{bmatrix}\ , \quad \omega = [D_1\phi,\ \cdots,\ D_n\phi] \tag{8}$$

This convolutional LCA decreases the number of nonzero lateral connections from $N^2 = (n(l_y - l_\phi))^2$ to $2n^2(l_y - l_\phi)l_\phi$, a reduction of $2l_\phi/(l_y - l_\phi)$.

Unfortunately, decreasing the number of nonzero elements in ϕ limits the time-bandwidth product, the combined effective resolution in time and Doppler. Adding too many Doppler bins means the restricted isometry property starts to be violated.

A. Multi-layer SLCA

The primary contribution of this work is the development of the multi-layered SLCA architecture, which minimizes the number of nonzero elements in each SLCA layer, while simultaneously allowing for longer waveforms.

By factoring $\Omega = BA$, we can convert (2) into:

$$y = Bz + \nu\ , \quad z = Ax\ , \tag{9}$$

which can be solved by applying BPDN twice:

$$\hat{x} = BPDN_{A,0} \cdot BPDN_{B,\lambda}(y)\ . \tag{10}$$

If we choose a $\phi = \phi_B * \phi_A$, then we can model Ω as:

$$\Omega = [D_1\Phi_B\Phi_A\,, \cdots,\ D_n\Phi_B\Phi_A]\ , \tag{11}$$

where Φ_A and Φ_B are Toeplitz matrices for ϕ_A and ϕ_B. We can approximate (11) as $\Omega \approx BA$ with:

$$\begin{aligned} B &= [D_{B1}\Phi_B,\ \cdots,\ D_{Bn_B}\Phi_B] \\ A &= I^{n_b} \otimes [D_{A1}\Phi_A\,, \cdots, D_{An_A}\Phi_A] \end{aligned} \tag{12}$$

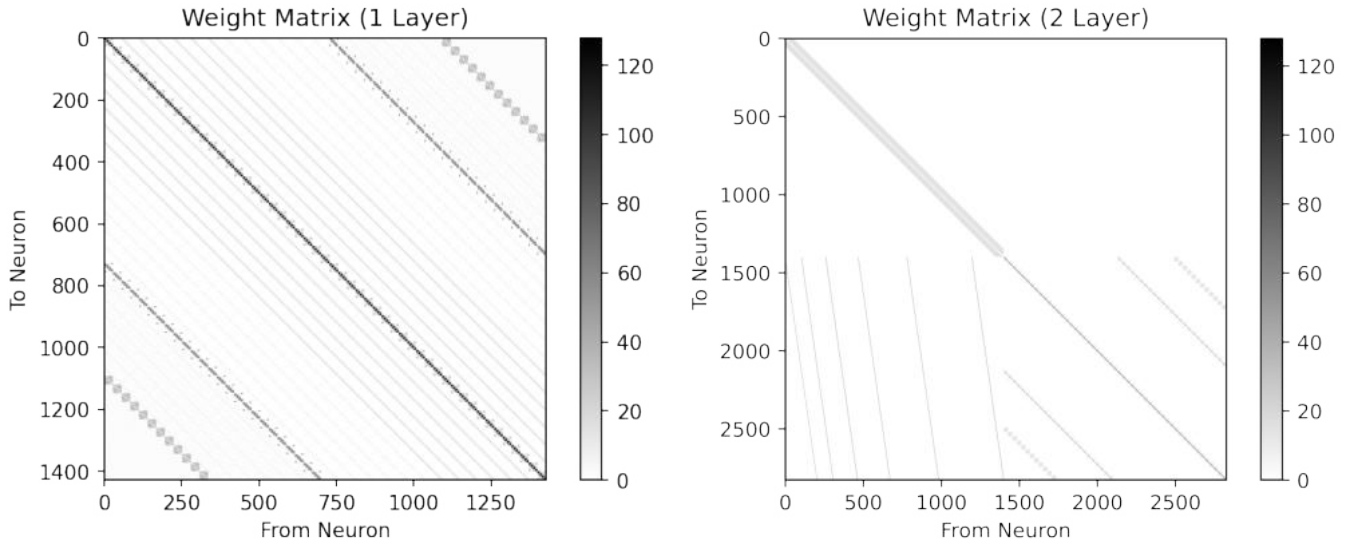

Fig. 6: (left) A 1-layer SLCA solving for 357 complex channels (51 time $\times$ 7 Doppler $\times$ 4 phase bins) requires 540k nonzero synaptic weights. (right) A 2-layer SLCA solving the same problem requires only 70k nonzero weights (a 7.6x improvement), despite requiring an additional 1400 neurons for the first layer (neurons 1-1400). The We were able to scale the 2-layer SLCA to solve for 1470 complex channels (210 time $\times$ 7 Doppler $\times$ 4 phase bins.)

where $n = n_A \times n_B$ and D_{Aj}, and D_{Bj} are constructed from frequencies f_{Aj} and f_{Bj} as in (2), such that:

$$\left[e^{j2\pi f_{B1}}, \cdots, e^{j2\pi f_{Bn_B}}\right] \otimes \left[e^{j2\pi f_{A1}}, \cdots, e^{j2\pi f_{An_A}}\right] = \left[e^{j2\pi f_1}, \cdots, e^{j2\pi f_n}\right]$$

and $\forall j : 2|f_{Aj}| < 1/l_{\phi B}$. In order to maximize p (and minimize m), this means making $l_{\phi B} \ll l_\phi$, thereby allowing more distinct frequencies f_{Aj}.

One candidate decomposition is $\phi_A = \phi'_A \otimes [1, \mathbf{0}_{l_{\phi_B}-1}]^{\mathrm{T}}$, which both makes A sparse and yields $\phi = \phi'_A \otimes \phi_B$.

In this configuration, the number of interneuronal connections for the first convolutional LCA (with term $B^{\mathrm{T}}B - I$) is $2n_B^2(l_y - l_{\phi B})l_{\phi B}$, and for the second (with term $A^{\mathrm{T}}A - I$) is $2n_B n_A^2(y - l_\phi)l_{\phi A'}$. Furthermore, there must be connections between the two layers (term A) with $n_B(l_y - l_{\phi_B})n_A l_{\phi A'}$ elements, for a total of:

$$n_B\left((l_y - l_{\phi B})(2ml_{\phi B} + n_A l_{\phi A'}) + 2n_A^2(y - l_\phi)l_{\phi A'}\right) . \tag{13}$$

If we make $l_{\phi B} \approx l_{\phi A'} \approx \sqrt{l_\phi}$, then this reduces to:

$$((2n_B^2 + n_A)(l_y - \sqrt{l_\phi}) + 2n_B n_A^2(l_y - l_\phi))\sqrt{l_\phi} \tag{14}$$

This represents a reduction of interneuronal connections by $< (\frac{1}{n_A^2} + \frac{1}{n_B})\frac{2\sqrt{l_\phi}}{l_y - l_\phi}$ relative to the non-convolutional LCA, or $< (\frac{1}{n_A^2} + \frac{1}{n_B})\frac{2}{\sqrt{l_\phi}}$ relative to the single-layer convolutional LCA (for conservative assumption of $l_y - \sqrt{l_\phi} < 2(l_y - l_\phi)$).

IV. Implementation and Evaluation Methods

A. Complex SLCA Hardware Implementation

The true benefits of algorithms like the SLCA comes from implementing them on neuromorphic hardware, with orders of magnitude power and latency reductions [14] relative to CPU or GPU-based implementations (such as FISTA).

The last decade has shown a scaling of neuromorphic hardware accelerated SLCA architectures, from outputs with miniscule dimensionality [10], to those with hundreds or even thousands of output elements [5], [16].

In this work, we used the Intel Loihi chip presented in [5]. We successfully compiled the SLCA via two methods: remotely via the Loihi servers that Intel makes available to the Intel Neuromorphic Research Community, and locally on the Kapoho Bay USB stick, which Intel allowed us to use. The results in this paper were primarily generated on the remote Loihi servers.

The Loihi chip implements Current-Based Leaky Integrate and Fire neurons (CUBA-LIF) neurons rather than the simple integrate and fire neurons assumed in [11]. The most important distinction is that the CUBA-LIF neurons operate in discrete time steps, and have integer valued states u and v. Thereore we used a discrete-time SLCA (adapted from templates previously used in [5]):

$$\begin{aligned}
u[t'] =& \lfloor c_1 \Re\{\Phi^* y\} - c_1\lambda \rfloor \\
&- \lfloor c_2 \Re\{(\Phi^*\Phi - I)s[t'-1]\} \rfloor \\
&+ \lfloor (1 - 1/T_u)u_{R+}[t'-1] \rfloor \\
v_{R+}[t'] =& \begin{cases} u_{R+}[t'] + v_{R+}[t'-1] & \text{if } v_{R+}[t'-1] < c_2 T_u \\ u_{R+}[t'] & \text{otherwise} \end{cases} \\
s_{R+}[t'] =& \begin{cases} 1 & \text{if } v_{R+}[t'-1] < c_2 T_u \\ 0 & \text{otherwise} \end{cases}
\end{aligned}$$

where $\lfloor \cdot \rfloor$ represents the floor function (i.e. conversion to an integer), $T_u = 2$ is the current decay time, and c_1 and c_2 are integer scaling factors that minimize the quantization errors.

The imaginary and negative neurons have similar dynamics, but replace $\Re$ function with $\Im$ functions, and/or invert the inputs their values respectively.

B. Waveform generation

We wrote a Python script with Intel's nxSDK library [5] in order to create a channel estimation simulation that would feed a signal into a 2-layer convolutional complex-valued spiking LCA.

We created an input channel with temporal resolution $1/5$, where 1 is the receiver's temporal resolution, and additive white gaussian noise $\nu \sim \mathcal{N}(0, \sigma^2)$, $\sigma = 0.15$.

The transmitted waveform was a repeated Barker signal, with:

$$\phi_B = [1\text{ -1 1 -1 1 1 -1 -1 1 1 1 1 1}]/\sqrt{13}\,, \tag{15}$$

$$\phi'_A = [1\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1]/\sqrt{6} \tag{16}$$

We had four different test setups:

- A single channel model
- Two temporally separated models with same amplitude
- Two temporally overlapping models with same amplitude
- Two temporally overlapping models with amplitude differential

We used (2) to generate the signal $y(t)$. For the single channel model, we used a_i as an independent variable, $\tau_i \in [0,\, l_y - l_\phi]$, $f_i \in [-3/l_y,\, 3/l_y]$ and $\theta_i \in [0,\, 2\pi]$. For the separated channels model we set $a_2 = a_1$, and set $\tau_2 - \tau_1 = 3.5 l_{\phi B}$. For the overlapping models with same amplitude, we set $a_1 = a_2$ $\tau_2 - \tau_1 = 0.5 l_{\phi B}$. For the amplitude differential model we set $a_1 = 2a_2$.

We implemented a SLCA with 5880 output neurons, corresponding to 210 time bins $\times$ 7 Doppler bins $\times$ 4 phase bins.

We ran the SLCA for $T_w = 128$ time steps, and summed the spikes in each time-Doppler bin:

$$|\hat{x}| = \sqrt{\sum_{t'=0}^{T_w} (s_{R+}[t'] - s_{R-}[t'])^2 + (s_{I+}[t'] - s_{I-}[t'])^2} \tag{17}$$

As a baseline comparison, we implemented a matched filter as a SLCA without the lateral inhibitory terms $(\Omega^*\Omega - I)$ or negative offset λ.

V. Results

In each scenario, we swept the amplitude from 1 to 45, where the spike rates of the neurons saturated just under 50% (64 spikes in 128 time steps). We measured three different properties of the filter's outputs:

- The number of spikes in the bins corresponding to channels truly present in the signal. Since the true time and Doppler offset often do not perfectly line up with a particular basis, we also include the spikes in the immediately adjacent bins.
- The total number of spikes in all the bins with the same time offset as the true channels. The bases for these bins overlapped relatively heavily with the true bins, leading to a relatively higher rate of false spikes.
- The maximum number of spikes in any other incorrect set of adjacent bins.

TABLE I: Temporal smearing in channel estimation. Averaged at amplitude $= 40$ over 10 Monte Carlo runs.

	SLCA	Matched Filter
Single Channel	0.9	5.6
Separate Channels	0.7	6.0
Overlapping Channels	0.5	6.4
Unequal Channels	0.4	4.4

As seen in Figure 7, the SLCA was able to produce cleaner and more precise results than the matched filter in all cases. The SLCA's spike response in the bins representing the true channels mimicked the soft-threshold behavior expected in the LCA (4), over an order of magnitude of sensitivity, for each individual true channel.

The SLCA's response in the true channels consistently outperformed that of incorrect channels, and was consistently at least double the total number of spikes in all channels with correct time offset, but incorrect Doppler offset. This means that the 2-layer SLCA could always easily pull out the true bins from any false ones, even when the channels were in close proximity (see Figure 3), or if the amplitude of one was twice the amplitude of the other.

Contrast this behavior with the matched filter. The overcomplete basis means that the matched filter produces significant ambiguity in the channel estimation, especially when multiple true channels are present. This produces extra spikes in channels both adjacent, and non adjacent to the true channels. The high amount of ambiguity makes the true channels non-distinctive in many cases, as illustrated in Figure 2 and Figure 3.

We used a 'temporal smearing' metric to summarize overall performance. Temporal smearing represents the total number of spikes in channels with incorrect time offsets divided by the spikes in the correct channel. As can be seen in Table I, the SLCA is a clear winner over the linear matched filter, with smear an order of magnitude lower in the most complex scenarios.

VI. Discussion and Future Work

Sparse estimation in an overcomplete basis is a longstanding problem in signal processing. Matched filters have long been recognized as a computationally cheap, tractable solution that can be applied in real time, at the cost of imposing a substantial ambiguity function on the output estimate. More complex linear solvers, such as the Moore-Penrose pseudoinverse, remove much of this ambiguity, but still do not take advantage of the sparse statistics in the underlying basis.

A number of digital solvers have emerged [15], [18] that use iterative techniques to converge on solutions to sparse approximation objectives like BPDN or LASSO [19]. However, for high-dimensional problems, the convergence can take hundreds of iterations.

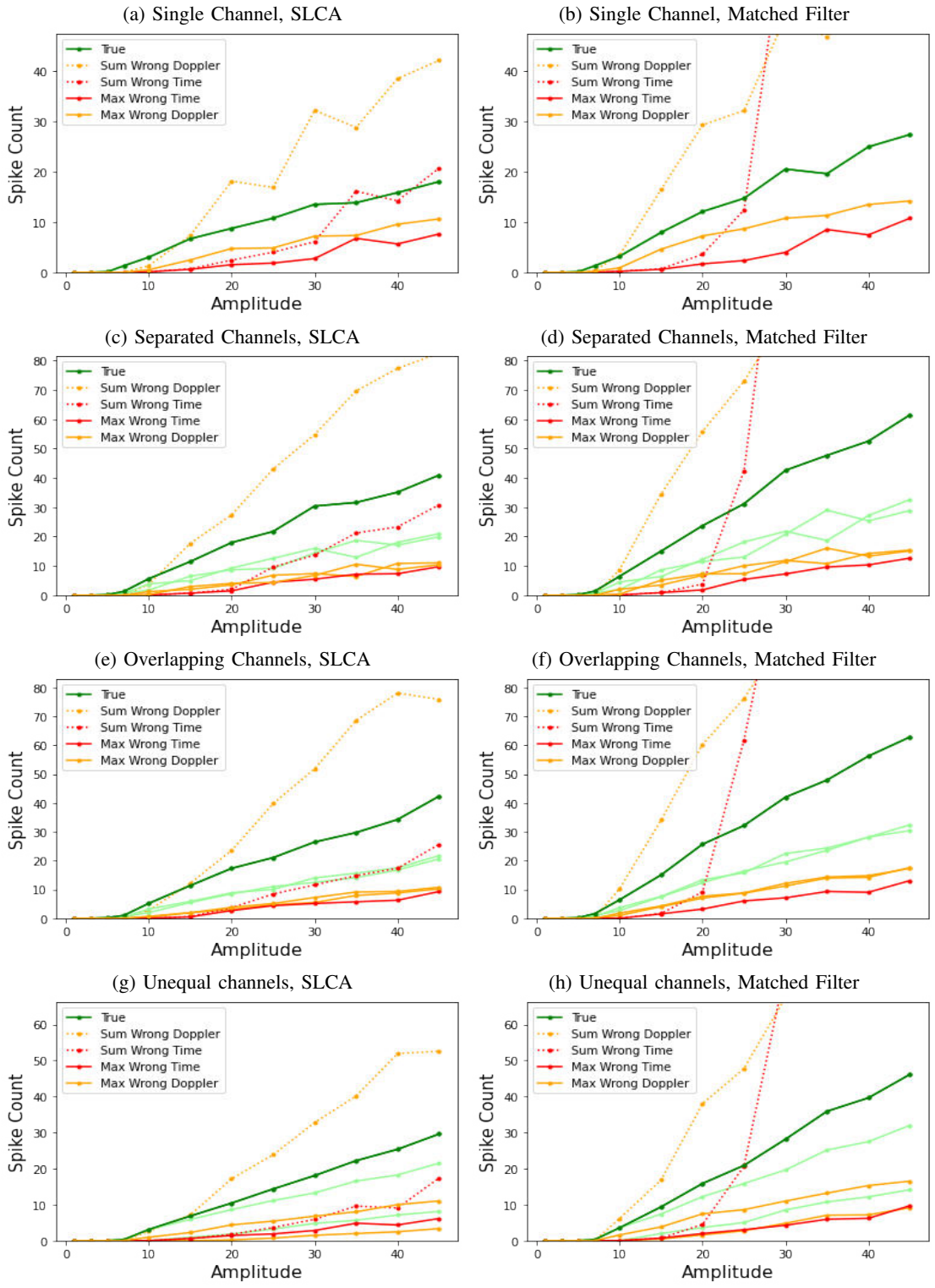

Fig. 7: Performance of the 2-layer SLCA (left) vs matched filters (right) for channel estimation under scenarios described in Sec. IV-B, averaged over 10 Monte Carlo trials. The dark green lines represent the total number of spikes in the true channels (light green shows individual true channels). Dashed orange lines count all spikes in channels with correct temporal offset, but wrong Doppler, while solid red lines count all spikes in channels with the wrong temporal offset. Dashed lines count the spikes in the worst single channel.

[14] demonstrated that neuromorphic implementations – specifically the Spiking LCA – can outcompete these digital solvers in both speed and energy. Furthermore, these advantages grow as the problem scales. However, a naive implementation of the SLCA sees $O(N^2)$ scaling in the number of synapses, as well as linear scaling in the fan-in and fan-out for each neuron. SNN synapses are very light weight (each one is just an adder), but the quadratic scaling and near all-to-all connectivity quickly exhaust the resources on a single chip.

By applying the multi-layer SLCA architecture introduced in this work, we can reduce the synapse scaling to $O(N\sqrt{N})$ and fan-in/fan-out to $O(\sqrt{N})$. As $N \gg 100$ the resource savings become vital to at scale deployment. [16] was able to achieve similar scaling properties, but this relied on specific properties of a continuously repeating Alltop sequence [17]. The 2-layer SLCA is far more flexible, and can be applied to a much larger class of waveforms $\phi = \phi_B \otimes \phi_{A'}$.

This multi-layer approach can be applied to any sparse coding problem where the generative model is decomposable $\Omega \approx BA$. This will be especially productive when $B^T B$ has few nonzero elements, and $A = I \otimes A'$, as in the channel estimation problem explored here. We leave for future work characterization of which matrices Ω can be so decomposed. Another potential extension is research into architectures with more than two layers.

SNN approaches like the SLCA do impose one important restriction: their outputs have a dynamic range equal to T_w. Digital solvers will generally converge to floating point precision exponentially; the error will scale $\propto e^{-t'}$, whereas SNN precision scales $\propto \frac{1}{t'}$. More advanced hardware, like Loihi 2, can allow for longer computational windows; [20] recently demonstrated $T_w = 256$, for a $N = 784$ LCA with throughput of 1000s of solutions per second.

For applications with sufficiently stringent dynamic range and throughput requirements, the SLCA may not be sufficient by itself. However, it can identify the nonzero elements quickly. Once the sparse components have been downselected, a Moore-Penrose psuedoinverse could be run to provide more precise estimate on the remaining elements.

VII. Acknowledgements

The authors would like to acknowledge Georgia Tech Research Institute for funding this research, and Intel for granting access to the Loihi chipset, the nxSDK software library, and the convolutional SLCA template.

References

[1] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *International journal of neural systems*, vol. 19, no. 04, pp. 295–308, 2009.

[2] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[3] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural networks*, vol. 111, pp. 47–63, 2019.

[4] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[5] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[6] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM review*, vol. 43, no. 1, pp. 129–159, 2001.

[7] B. A. Olshausen and D. J. Field, "Sparse coding of sensory inputs," *Current opinion in neurobiology*, vol. 14, no. 4, pp. 481–487, 2004.

[8] A. Templeton, T. Conerly, J. Marcus, J. Lindsey, T. Bricken, B. Chen, A. Pearce, C. Citro, E. Ameisen, A. Jones, H. Cunningham, N. L. Turner, C. McDougall, M. MacDiarmid, C. D. Freeman, T. R. Sumers, E. Rees, J. Batson, A. Jermyn, S. Carter, C. Olah, and T. Henighan, "Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet," *Transformer Circuits Thread*, 2024.

[9] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural computation*, vol. 20, no. 10, pp. 2526–2563, 2008.

[10] S. Shapero, C. Rozell, and P. Hasler, "Configurable hardware integrate and fire neurons for sparse approximation," *Neural Networks*, vol. 45, pp. 134–143, 2013.

[11] S. Shapero, M. Zhu, J. Hasler, and C. Rozell, "Optimal sparse approximation with integrate and fire neurons," *International journal of neural systems*, vol. 24, no. 05, p. 1440001, 2014.

[12] G. Turin, "An introduction to matched filters," *IRE transactions on Information theory*, vol. 6, no. 3, pp. 311–329, 1960.

[13] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes rendus. Mathematique*, vol. 346, no. 9-10, pp. 589–592, 2008.

[14] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.

[15] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[16] P. L. Brown, M. O'Shaughnessy, C. Rozell, J. Romberg, and M. Flynn, "A 17.8-ms/s compressed sensing radar accelerator using a spiking neural network," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 3, pp. 834–843, 2020.

[17] W. Alltop, "Complex sequences with low periodic correlations (corresp.)," *IEEE Transactions on Information Theory*, vol. 26, no. 3, pp. 350–354, 1980.

[18] M. Zibulevsky and M. Elad, "L1-l2 optimization in signal and image processing," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 76–88, 2010.

[19] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

[20] G. Parpart, S. Risbud, G. Kenyon, and Y. Watkins, "Implementing and benchmarking the locally competitive algorithm on the loihi 2 neuromorphic processor," in *Proceedings of the 2023 International Conference on Neuromorphic Systems*, pp. 1–6, 2023.