STEAM: Spike Time Encoded Addressable Memory

Extended Abstract

John V. Monaco
Naval Postgraduate School
Monterey, CA
vinnie.monaco@nps.edu

Manuel M. Vindiola
U.S. Army Research Laboratory
Aberdeen, MD
manuel.m.vindiola.civ@mail.mil

Ryad Benosman
Carnegie Mellon University
Pittsburgh, PA
rbenosma@andrew.cmu.edu

KEYWORDS
spiking neural network, temporal encoding, symbolic processing

1 INTRODUCTION
We introduce Spike Time Encoded Addressable Memory (STEAM), a method for persistent and addressable memory in spiking neural networks. STEAM is built using principles from the Spike Time Interval Computation Kernel (STICK) framework [3], which encodes numeric values as the time interval between spike events and relies on precise timing and synchrony to perform event-driven computation. We implement addressable memory through the composition of networks that separately encapsulate flow control and persistence. With the ability to superimpose values encoded by time intervals, these structures lead to linear-sized spiking neural networks that sort and search in linear time. STEAM can be implemented entirely using basic leaky integrate and fire (LIF) neurons.

2 MEMORY AND SUPERIMPOSITION
Registers play a critical role in digital computers. In a neural architecture, mechanisms that implement re-writable and volatile memory may provide analogous functionality by quickly storing and retrieving values needed for computation and providing the basis for the ability to perform operations over variables [4].

In STICK, values are encoded as the time interval between two spikes [3], and neural networks are constructed to perform operations on the temporally-encoded values. We use an encoding function \( f(\cdot) \) and decoding function \( f^{-1}(\cdot) \) to provide a linear map from the domain \( \mathbb{R} \) to a time interval \( t_{\text{min}} \leq t \leq t_{\text{max}} \), where the encoded interval \( \Delta t \) is bounded between \( t_{\text{min}} \) and \( t_{\text{max}} \). For neuron model dynamics and figure notation, we refer to [3].

2.1 Scalars
Temporally-encoded values are transient by nature, i.e., a time interval expires after a finite duration. In such a scheme, memory networks are crucial for storing and synchronizing values. In a neural architecture, memory may generally be achieved by setting the membrane potential of a neuron proportional to a numeric value to store. In this sense, memory can be viewed as a spatiotemporal conversion: a value remains serialized as a physical quantity until it is later recalled, upon which it is converted from space (a physical quantity) to time (interval between spikes).

The ability to store and recall a single scalar value is achieved by the memory network shown in Figure 1. The network behaves much like a timer, where the value of the timer is set by two input spikes to the store neuron. The difference in time between the two input spikes is stored in the membrane potential of the acc neuron. This difference can be recalled by inputting a single spike to the recall neuron, upon which a single spike is later emitted by the output neuron. This behavior is shown in the chronogram in Figure 1 where the network stores and recalls the value 0.3.

2.2 Vectors and sets
Using the memory network as a building block, we introduce two memory structures: a vector, which stores an ordered sequence of scalars, and a set, which stores an unordered set of scalars.

Shown in Figure 2 (top), the sequential memory network is composed of \( N \) memory networks connected in serial. Each memory location is a copy of the memory network described above. The input neuron recall is connected to the first memory location, mem0. A single spike to recall causes a chain reaction as each memory network simultaneously outputs the value it stored and recalls the next memory location. A network that stores \( N \) values will output \( 2N \) spikes when recalled. The spacing between values, i.e., delay between the 2nd spike of value \( x_i \) and the 1st spike of value \( x_{i+1} \), is controlled by the synaptic delay between the output of mem \( i \) and recall of mem \( i+1 \).

As an example, the chronogram of a network that stores \( N = 3 \) values is shown in Figure 3 (left), where values \( x_0 = 0.25, x_1 = 0.15 \), and \( x_2 = 0.35 \). Since values are output in sequence, the total time...
to recall the contents of a vector is \((N - 1) \delta + \sum_{i=0}^{N-1} f(x_i)\) for \(N\) values and \(\delta\) spacing delay between values.

A central principle of encoding data as time is that values can be superimposed, whereby several values are projected onto the same interval. A single spike marks the beginning of a set of superimposed values, and each subsequent spikes marks the end of an interval, i.e., a single value. Superposition provides a compact representation for multiple values, requiring only \(N + 1\) spikes to represent \(N\) values. The principle of superposition can be leveraged to build spiking networks that sort and search in linear time.

Encompassing this principle is the superimposed memory network, shown in Figure 2 (bottom). Unlike sequential memory, an input spike to recall causes every value to be recalled in parallel and piped to a single output, resulting in the superimposition of all values in memory. Since values are recalled in parallel, if there are two values of equal value in different locations, they will appear as a single value at the output. In this regard, superimposed memory can indicate only set membership and does not indicate the location or number of equalities of each value.

The behavior of the superimposed memory network and resulting output is shown in Figure 3 (right). Generally, the total time to recall the contents of superimposed memory is proportional to the largest value, \(\max x_i\), since this will be the largest interval.

3 APPLICATION: SORTING

Sorting may be achieved by combining three components: superimposed memory, a routing network (not described here), and sequential memory. Sorting with a spiking neural network exploits the natural ordering of timed events, akin to the ordering of integers in a non-comparison sort. The superimposed memory stores the input values to be sorted and the sequential memory stores the output values in ascending order. As the input values are recalled in parallel, the output values are also stored in parallel. Since values are superimposed, this operation completes in time proportional to the largest value.

4 CONCLUSIONS AND FUTURE WORK

Alongside the use of neuromorphic chips as accelerators for deep neural networks, there is a growing interest in leveraging neural architectures to perform symbolic processing [1, 2, 5, 6]. Our work doesn’t aim to provide a biologically plausible or even biologically inspired memory, but instead, a practical framework for addressable memory on neuromorphic devices. We see this work as a step towards leveraging emerging neural-inspired architectures for symbolic processing tasks for which machine learning is currently not well suited and towards the integration of symbolic and sub-symbolic processing on a single architecture.

In future work, we aim to further build the repertoire of spiking networks that process temporally-encoded values. We plan to demonstrate how massive parallelism and event-driven computation may lead to efficient solutions to classical computer science problems such as sorting and searching and to design spiking neural networks that perform variable binding and instruction execution in a temporal encoding scheme.

REFERENCES