Exact Spike Timing Computational Model of Convolutional Associative Memories

Igor Peric, Felix Schneider, Cameron H. Price, Stefan Ulbrich, Arne Roennau,
Marius Zoellner, Ruediger Dillmann
FZI Forschungszentrum Informatik, Karlsruhe, Germany
Email: {peric, fschneid, price, sulbrich, roennau, zoellner, dillmann}@fzi.de

Abstract—In this paper we propose the use of exact spike timing signalling for a spiking neural network model of associative memory in the mammalian cortex. We extend the spike timing function engineering framework STICK with a computational block for performing addition with overflow (modulus). This computational block is used to implement a Circular Holographic Reduced Representation, a specific instance of Vector Symbolic Architectures, which is a candidate for an associative memory framework described by holonomic brain theory. We provide illustrations of spiking dynamics during various computational stages contain elements of the stochastic spike synchrony observed in cortical EEG recordings, as well as clearly separable computational phases suitable for functional analysis. We show that the framework is able to very accurately memorize and recall associated symbolic identifiers under ideal conditions (with no noise), maintaining very accurate and predictable spike timing. Furthermore, we test the framework in the noisy environment which is known to exist in cortical tissue. We do this by injecting Poisson spike trains into each neuron, approximating presence of unpredictable, stochastic environment in which actual compute module would reside in biological system. Besides reporting accuracy analysis for this case, we propose computational requirements (constraints) needed for the exact spike timing framework to work in such noisy conditions.

Keywords—associative memory, neural computation, precise spike timing, signal processing

I. Introduction

DESPITE investing numerous decades of effort, many scientific disciplines still haven't identified the underlying mechanisms used by the brain to perform various computational tasks. These tasks include, but are not limited to, memory representation, quick symbol manipulation, long-term storage and recall.

Holonomic brain theory [1][2] suggest that representations used by the brain are highly dispersed (ideally uniformly) throughout the whole representational space. In other words, each symbolic representation consists of current set of neuron activation across the whole brain. Lead by this theory, cognitive neuroscientists have created series of computational models that rely on this theory. One of the most popular ones is Holographic Reduced Representations (HRR) [3][4][5], briefly explained in chapter III.

The Neural Engineering Framework (NEF) [6][7] is a general-purpose function approximation framework that uses spiking computational units and synaptic connections between them to encode input-output transformations. Since every function approximation block in NEF consist of two layers and all-to-all connectivity between neurons in them, the number of synapses grows exponentially with the number of neurons

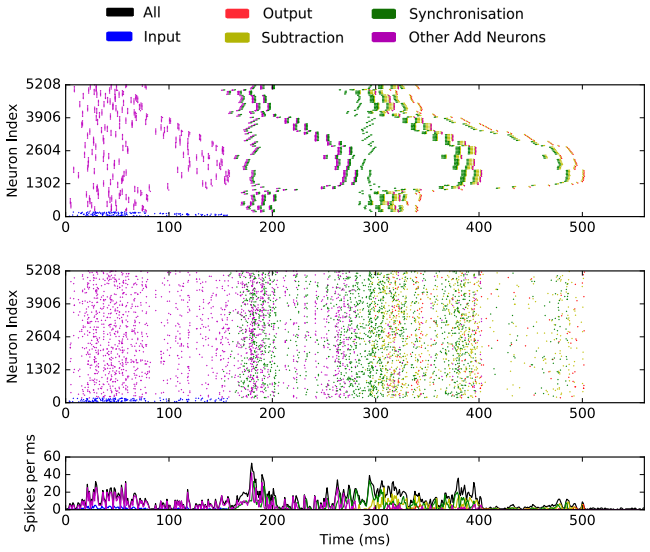

Fig. 1: Spiking chronograph of a neural population performing convolution with Gaussian kernel, effectively doing smoothing of input signal. Color codes represent different stages of processing, associating spikes with their computational function. Top plot shows ordered spike trains, as constructed by our framework. Middle plot shows same activity with randomly shuffled neurons (vertical axis) for visualization purposes only, resulting in a more confusing graph in which individual spikes' function becomes hardly distinguishable. These kind of graphs are a very common outcome of a cortical spiking activity recording. Bottom plot shows total population activity over time. Details of this network are given in sections II-B and IV-A

used. It abstracts away underlying spiking dynamics and considers only firing rates of individual neurons in the decoding process, thus not fully utilizing the computational capabilities (advantages) of discrete spiking signaling. This paper explores the alternative approach, described in chapter I-A, which uses neuron model dynamics as the computational substrate and membrane voltages for storing values.

The authors of NEF have been able to approximate HRR operations in spiking neural networks and named this approximation Semantic Pointer Architecture (SPA) [8]. They used SPA to build a large scale cognitive architecture *Spaun* [9],

a model capable of performing 10 cognitive, perceptual and motor tasks. Their simulation had 2.5 million neurons, required 24 GB of RAM due to the large number of synapses and took 3 hours to simulate 1 second of activity.

Another popular function approximation framework for spiking neural networks are Liquid State Machines (LSM) [10]. This framework assumes random connectivity of a $reservoir$ of spiking computational units with input and output subpopulations. Readout weights are trained with Linear Least Squares (LLS) optimization, producing the desired input-output mappings. To the best of our knowledge, there are no cognitive architecture models built using the LSM framework yet.

The structure of this paper is as follows. Section I-A gives a brief overview of the STICK framework, needed for the comprehension of the following sections. Section II describes our contributions to the STICK framework - design of a new computational block for performing addition with overflow (section II-A) and proposal for using the existing linear addition block for performing convolutions in time domain (section II-B). Section III introduces basic properties of Holographic Reduced Representations (HRR) and its variant (Circular HRR), which are two candidates for symbolic processing in cognitive architecture models. In sections III-D and III-E we give the comparison of these two methods running the same cognitive task in STICK framework. Lastly, section IV we show results of three experiments: performing convolution on 1D signals, modelling associative visual scene memory using CHRR and effects of adding noise on the inner computational dynamics and accuracy.

A. The STICK Framework

All computations presented in this paper were performed using the STICK framework using the NEST simulator[11]. STICK refers to "Spike Timing Interval Computational Kernel", a framework for general purpose computation in neurons utilizing, precise timings, delays, and synchrony [12]. A full description of the framework can be found in [12], but a brief overview is provided here. The basic computational unit is a non-leaky integrate and fire neuron. The neuron dynamics can be described by the following equations:

$$\begin{aligned} \tau_m \cdot \frac{dV}{dt} &= g_e + \mathit{gate} \cdot g_f \\ \frac{dg_e}{dt} &= 0 \\ \tau_f \cdot \frac{dg_f}{dt} &= -g_f \end{aligned} \tag{1}$$

The state variables of the model are V (membrane voltage), g_e (constant input current), g_f (exponential decaying postsynaptic current) and *gate* (switch for postsynaptic current, only changed by synaptic events). The parameters to the model are τ_m (membrane time constant), τ_f (synaptic time constant), V_t (threshold voltage) and V_{reset} (resting potential)[1]. When the membrane voltage reaches the threshold voltage, the neuron fires along all outgoing synapses, resets V to V_{reset}, and all other state variables to 0.[2] This corresponds to a non-leaky integrate-and-fire neuron with exponential decaying post-synaptic current.

STICK differentiates between four different synaptic connections, each affecting one of the four state variables: V-synapses directly modify the membrane voltage, g_e-synapses modify the constant input current, g_f-synapses modify the exponential input current and *gate*-synapses set the *gate* variable to turn exponential input current on and off.

For calculations with real numbers, values between 0 and 1 are encoded the time between two successive spikes via the encoding function

$$\Delta t = f(x) = T_{\text{min}} + xT_{\text{cod}} \tag{2}$$

where x is the value to be encoded and T_{min} and T_{cod} are the time intervals representing minimum and maximum delay between encoding par of spikes, respectively. The choice of T_{cod} is very important for the performance of the network. Primarily it affects the resolution of the value representation. Choosing a smaller value while keeping the time resolution of the neural simulator constant results in a more coarse representation. It also affects the speed of many operations in STICK, as they are often tied to $T_{\text{max}} = T_{\text{min}} + T_{\text{cod}}$, the longest possible duration of an encoded input. In most of the experiments presented in this paper we used the coding constants $T_{\text{min}} = 10\text{ms}$ & $T_{\text{max}} = 100\text{ms}$, as in [12]. To represent signed values we use pairs of two neurons, one to represent positive (or zero) input and one to represent negative input.

We note that unless otherwise noted the particular constants used in this paper are exactly as in [12], with the exception of the there mentioned T_{neu} (neuron delay), which by restriction of the nest simulator was set to 0 for all experiments in this paper.

The STICK framework includes the following operators, realised by neurons: Many of the above capabilities are used

Block name	*Description*
Memory	Storing any representable value for later recall
Signal Synchronisation	Taking in a given number of inputs to be fed forward only once all signals have been received.
Minimum and Maximum	Identifying and returning the minimum/maximum of a number of inputs.
Subtraction	Returning the difference between two operands.
Logarithm	A block with output is proportional to the natural logarithm of the input.
Exponential	A block with output proportional to exp(*input*).
Multiplication	Multiplies two operands.
Integration	Performs integration of the input over time.

TABLE I: List of blocks implemented in original STICK framework.

[1]The values for these parameters are the same as in the original paper: $\tau_m = 100\text{ms}, \tau_f = 20\text{ms}, V_t = 10\text{mV}, V_{\text{reset}} = 0\text{mV}, T_{\text{syn}} = 1\text{ms}, T_{\text{min}} = 10\text{ms}, T_{\text{cod}} = 100\text{ms}$. The exception is T_{neu}, which is 0ms in NEST.

[2]This is a defining factor of STICK and means that input current is not preserved across

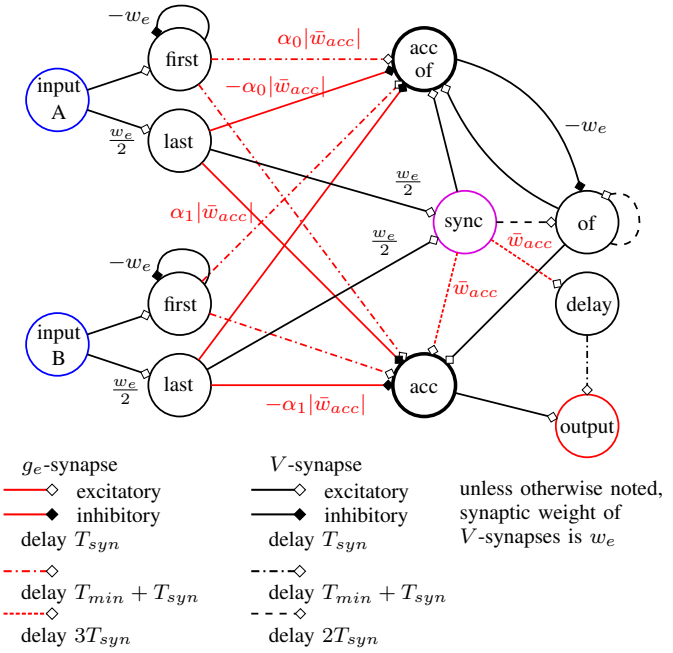

Fig. 2: Diagram illustrating overflow addition computational block. This network performs a weighted addition which overflows back to zero if the sum exceeds one. The neurons drawn in bold have a threshold voltage of $2V_t$, twice that of the other neurons. The notable change to the regular linear combination block is the `acc_of` neuron, which is used to check for an overflow. The `of` neuron is used to make sure the accumulator is reset at the end of the calculation, as well as inverting the signal from `acc_of` (from spike on overflow to spike on no overflow).

in the experiments presented in this paper. For details of these networks we refer the reader to [12].

II. Additions To Stick Framework

A. Modulo Arithmetic

As discussed in sections III-C and III-E, to perform the experiments presented in section III-F in a framework (such as STICK), it is necessary to be able to perform modulo addition. Intuitively, implementing an addition with modulo should be something that neurons are inherently capable of, as a neuron's internal mechanics have a built-in "overflow" when the membrane potential exceeds the threshold voltage, it returns to the resting potential. However, when this happens, information about previous input currents is lost, so it proved infeasible to use this property. We instead propose an arrangement that is shown in Fig. 2. The arrangement shown here is limited to two inputs, but can be trivially expanded to support more. More importantly, it is limited to one overflow, i.e. the weighted sum of the inputs does not exceed two.

B. Convolution with a Fixed Kernel: As a use case of linear combination blocks

In the field of 1D signal processing an important operation is the circular convolution of an arbitrary vector with a fixed kernel. This operation is defined as follows: Let $\vec{k}$ be a vector of odd length $|k|$ called the kernel, the circular convolution $\vec{v} * \vec{k}$ of these two vectors is given by:

$$(\vec{v} * \vec{k})_i = \sum_{j=0}^{|\vec{v}|-1} x_{j-i} \vec{k'}_{j-\alpha} \tag{3}$$

where $\vec{k'}$ is the kernel extended by appending 0s until it is of length $|\vec{v}|$, all indices are taken modulo $|\vec{v}|$, and $\alpha = (|\vec{k}|-1)/2$

Authors in [12] showed how the STICK framework can be used to form a weighted linear combination of an arbitrary number of inputs. Given kernel $\vec{k}$ we can consider a STICK block which produces the weighted linear combination of its inputs with weights being the values of the kernel. For a diagram of this block refer to Lagorce 2015[12]. We consider the block being applied to the neuron attached to the center input of the block, that is input $\alpha = (|\vec{k}| - 1)/2$. Given a row of N neurons, each representing a number in the STICK framework we can consider this row to be representing a vector $\vec{x}$ and we can apply N kernel blocks to the N neurons. The output of these N kernel blocks is then the circular convolution of the vector $\vec{x}$ with the kernel $\vec{k}$. Fig. 3 describes this principle.

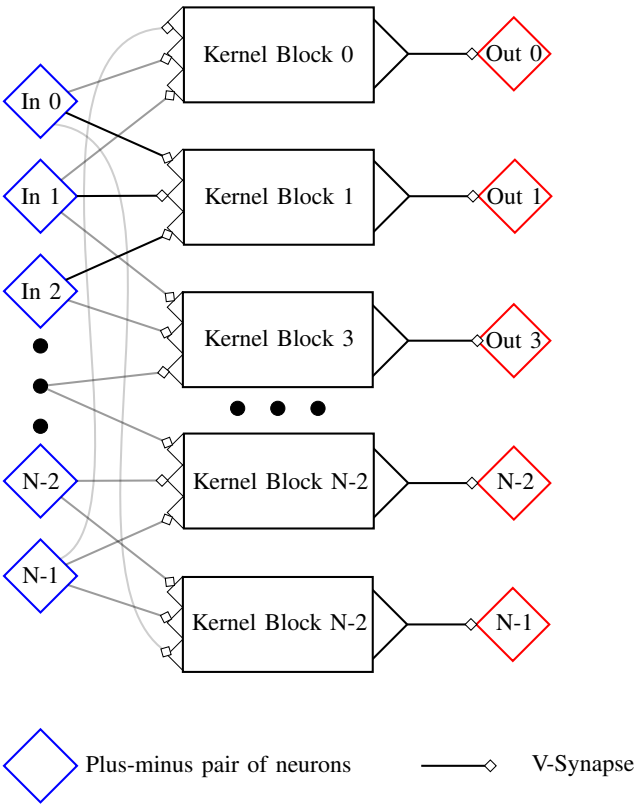

Fig. 3: An example of the fixed kernel convolution block for a kernel of length 3. We note that a connection between a plus-minus pair of neurons is actually a pair of synapses, connecting the plus neuron to the plus neuron and the minus neuron to the minus neuron.

III. Holographic Reduced Representations

A. Associative Memory Systems

Authors in [9] have shown that many common cognitive tasks can be solved by an associative memory system. Suppose we have some representation for memory, and we call the space of memorable objects (or their representations) $\mathbb{M}$. If given $a, b, c, d \in \mathbb{M}$ we can perform operations of binding $\otimes$, dissociation $\triangleright$, and superposition $\oplus$, then we can use this representation to form an associative memory system. The association and dissociation operations are defined such that $a \otimes b = x \implies a \triangleright x \approx b \quad \& \quad b \triangleright x \approx a$, which is to say, two elements can be bound together to form a new element, and either of the original elements can be recovered from the new element by probing it with the other. The superposition operation is defined such that given $(a \otimes b) \oplus (c \otimes d) = z$, we can probe z with any of a, b, c, d to recover the element paired to it; for example, probing the above z with a results in $a \triangleright z \approx b$. Binding is commutative and associative so that an arbitrary number of elements can be bound together, and then probed by an element to recover all other elements. Similarly superposition should also allow for storing an arbitrary number of bindings in a single superposition state.

B. Classical HRRs

It can be shown [4][3] that given a vector space of dimension D over the real line $\mathbb{R}$, the operations of circular convolution, vector addition, and correlation (or involution which is a good approximation in high dimensional spaces) satisfy the relations for the binding, superposition, and dissociation operations, required for associative memory systems. The similarity of two vectors can be measured using the dot product. This formalism is known as "Holographic Reduced Representations" (HRRs)[4]. In this case the number of memorable objects that can be represented in a working memory depends on the dimension of the real space considered; in particular higher dimensional spaces can be used to form larger working memories. Note that by "working memory" we mean that the processes of binding and superposition operations can be done with a reasonable amount of vectors, and decoding will produce a result that is closer to the target vector than any other vector used to represent an object in this particular memory representation.

HRRs compare favourably to many associative memory models as the space of representations is closed under the memory operations. In systems which are not closed under these operations, performing multiple operations leads to dimensional explosion which is unsatisfactory from a computational standpoint.

C. Circular HRRs

A significant problem of HRRs is that given a vector space of dimension D, convolution and involution require $\mathcal{O}(D^2)$ operations. According to the convolution theorem, the circular convolution of two vectors x, y is equivalent to the inverse Fourier transform of element-wise product of the Fourier transforms of x and y. The use of fast Fourier transform algorithms can then allow circular convolution calculations in $\mathcal{O}(D \log D)$ operations.

This suggests that working with a vector field over the complex plane may allow one to avoid computationally expensive convolutions in favour of vector addition. Plate [5] showed that by working in a vector field of unitary vectors over the complex plane (where the methods of HRRs apply) and representing the elements of the vectors (which are complex numbers of absolute value 1), as angles $\Theta \in [-\pi, \pi]$, a system of memory equivalent to HRRs can be constructed using alternative operations. Plate also provided the following table of equivalences between operations on the traditional vector representation and the new "Circular HRR" (CHRR) representation.

Object or Operation	Representation in HRRs	Representation in CHRRs
Memory Element	Vector with elements in $(0, 1)$	Vector with elements in $(-\pi, \pi]$
Binding	Convolution	Modulo 2π element-wise sum
Dissociation	Involution	Modulo 2π element-wise negation
Superposition	Vector addition	Angle of vector sum of complex representation[3]
Similarity	Dot Product	Mean of cosine of element wise angle differences

The operations for CHRRs are much better suited to the STICK framework that their equivalent in HRRs.

CHRRs are mathematically equivalent to HRRs and therefore all the properties of associative memory systems discussed in sections III-A and III-B

D. Example Classical HRR Calculations in Neural Networks

Of the operations required for classical HRRs, convolution is the most problematic for computation in STICK. Convolution of two arbitrary vectors requires $\mathcal{O}(n^2)$ operations (multiplications and linear combinations) which can be performed in STICK using the relevant blocks from Lagorce [12]. For this reason in investigating the feasibility of implementing HRRs in STICK we have focused on implementing the binding operation.

We have constructed and run a network to compute the convolution of two vectors of 10 dimensions. The network consists of 2480 neurons and on our reference machine takes 284ms to simulate the 600ms of in-simulation time required to perform the calculation, meaning it runs about twice as fast as real-time. This calculation is exact, within the accuracy of our representation. However, the amount of neurons required rises quadratically with the number of dimensions of the inputs. It is possible to reduce the number of required neurons by increasing simulation time, reusing the blocks that perform the calculations and feeding the input sequentially. Nevertheless, as HRRs require a high-dimensional vector space representation the number of operations (and in the case of STICK,

Methdod	*Dim*	*Neurons*	*Time*	*In-simulation time*
HRR Convolution	10	2480	290ms	600ms
HRR Convolution	20	9160	1597ms	600ms
HRR Convolution	40	35120	6923	600ms
CHRR Binding	10	160	22ms	600ms
CHRR Binding	150	2400	276	600ms
CHRR Binding	500	8000	1363	600ms
CHRR superposition	10	192	22ms	600ms
CHRR superposition	150	2852	360ms	600ms
CHRR superposition	500	9502	1335ms	600ms

TABLE II: Comparison of metrics between HRR and CHRR with STICK

neurons) required for convolution is problematic.

E. Example Circular HRR Calculations in Neural Networks

As stated in table II the operations needed implement for CHRRs are: Modulo-2π addition for binding and deriving the angle of a sum of complex numbers that are themselves represented as an angle for superposition. In contrast to classical HRRs, binding is the simpler operation in this case. For the special case of exactly two operands, the angle of the sum of two complex numbers is the average of the angles of the two numbers. This requires can be realised with the existing linear combination block. The implementation of modulo addition has been described in section II-A.

For this first implementation of circular HRRs, we have focused on implementing the necessary operations for exactly two operands. Expanding the processing blocks to a more general case is a possible avenue for future research.

We have performed binding for a range of vector lengths. Table II provides information about the amount of neurons and simulation times required to perform these calculations. We also performed superposition for a range of vector lengths which are also included in Table II. We note that superposition is marginally more computationally expensive than binding in the CHRR scheme.

F. Comparison of Methods

Table II shows the number of neurons and the computation time for the key operations in both the HRR and CHRR schemes using the STICK framework. This also shows that performing both super position and binding operations in the CHRR framework, require far fewer neurons than performing convolution in the HRR framework by more than an order of magnitude for vectors of length 10. In fact, performing binding in HRRs with vectors of length 40 requires 6 times as much simulation time as performing binding of CHRR vectors of length 500 demonstrating that for the high dimensional vectors required for an effective memory system, CHRRs are the only feasible option. For this reason we have chosen to use only the CHRR to perform the more advanced experiments in section IV of this paper.

IV. Experiments

A. Signal Processing in Time Domain

Using the methods described in section II-B we have performed the smoothing and discrete differentiation (or "edge detection") of a signal using the kernels [0.09 0.24 0.33 0.24 0.09], and [-0.27, -0.45, 0, 0.45, 0.27] respectively. The results of this are shown in figure 7. From this, we suggest that signal processing with an arbitrary kernel can be successfully performed within the STICK framework.

We have also included spike timing diagrams for these networks. These diagrams include a scatter plot of spike times with neurons labelled in an ordered manner (neuron with label i is a pre-synaptic neuron to some neurons labelled by indices slightly larger than i), and in an unordered manner close to as would be observed in a real cortical spiking activity recording. We note that in the diagrams with systematic neuron labels, the signal being fed to the network can clearly be seen. This is because in the stick framework the inter-spike timing for a single neuron is larger when the value being represented by a neuron is larger.

The output neuron inter-spike timings differ significantly between the two networks. In the smoothing network, the scatter plot retains the shape of the signal as expected. In the edge detection network, the inter-spike timing for the subtraction and output neurons is generally very small except in the region where the signal changes rapidly, as one would expected.

For both of these networks the signal processed was represented by 93 ample points, and the total number of neurons was 5208. The number of neurons required scales linearly with the number of sample points in the signal and linearly with the length of the kernel. In fact the total number of neurons n required for processing a signal of length x with a kernel of length y is given by

$$n = (36 + 4y)x \tag{4}$$

We note that restriction of the kernel or the signal can allow for the use of fewer neurons (for example in cases where only positive values can be represented, or some elements of the kernel are 0).

As discussed in section I-A, by reducing the precision over the range of representable values, networks in the STICK frame work can perform operations in much less time. In particular, we repeated the above experiments with the originally proposed STICK parameters changed so that $\tau_{\text{syn}} = 5.0,\ T_{\text{min}} = 2.0,\ T_{\text{cod}} = 2.0$. The results of these experiments are presented in figure **??**. Under these parameters the network is still able to correctly perform edge detection, however, signal smoothing is significantly impacted. These changes reduced the network run time by a factor of 10, from ≈ 500ms to ≈ 50ms. Rapid edge detection is highly significant to real-time image processing and further research in this area will be carried out.

B. Visual Scene Memory

In section III-E, we demonstrated that the STICk framework could be used to perform the operations required to construct a circular holographic reduced representation. We have also performed somewhat complex experiments to test the performance and accuracy of nested operations in CHRRs. For four vectors a, b, c, d, we calculated $(a \times b) \oplus (c \times d)$, which is to say

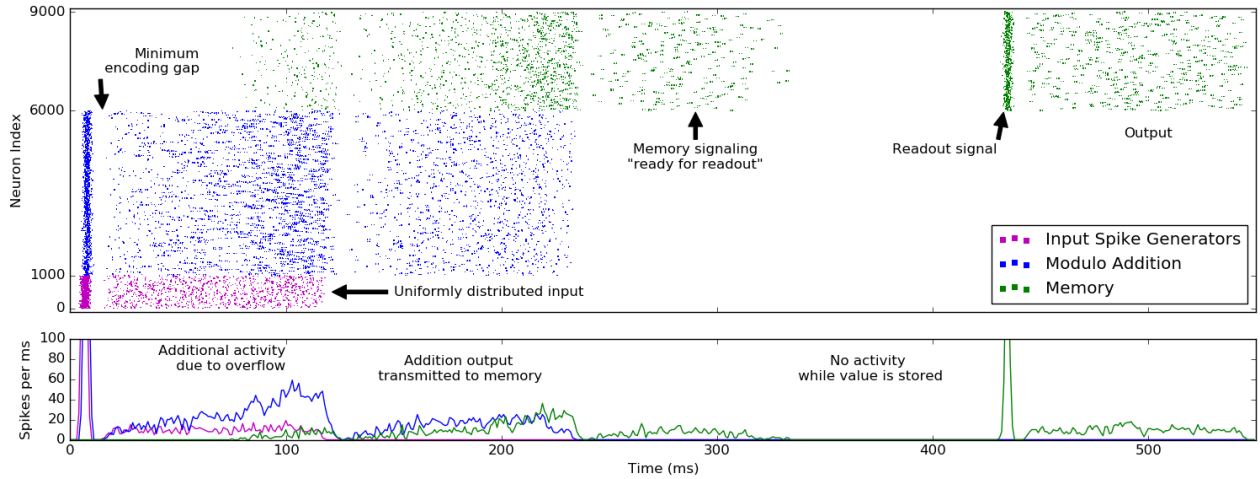

Fig. 4: Spike time diagram of a 500-dimensional binding operation (association). The network consists of three populations: The input spike generators, the modulo addition blocks and the memory blocks.

Methdod	*Dim*	*Neurons*	*Time*	*In-simulation time*
CHRR memory	64	2698	886ms	1800ms
CHRR memory	128	5386	4635ms	1800ms
CHRR memory (feed forward)	64	3330	918s	600ms
CHRR memory (feed forward)	128	6658	2025ms	600ms

TABLE III: Comparison networks used for CHRR visual memory experiment

the memory vector resulting from binding a and b as well as c and d, respectively, then superposing the results. This is an abstraction of a common experiment where a and c represent objects in space and b and d the represent locations in space. By binding them, we associate objects with their locations. From this memory vector, we can now probe to find either an object's location (by probing with a or c), or finding an object at a known location (by probing with b or d). We expect that, if we probe the above memory vector with b, the result should be significantly more similar to a than to any of the other three vectors.

For this experiment, we opted to construct a network that would reuse the computation blocks. This reduces the number of neurons that are required, at the cost of longer in-simulation time, as the inputs have to be provided sequentially. Fig 11 shows a schematic of the network. This simulation was done on vectors with 64 dimensions, requiring 2698 neurons. The simulation took 886ms to simulate 1800ms of in-simulation time. In table III, we also report details of a network that performed the same experiment without reusing blocks; we refer to this as the "feed forward" version of the network.

Performing computations in STICK networks introduces a small error to the computation (due to the resolution of the simulation compared to the range of the encoding function) which is noticeable under composition of several operations. However HRRs are resistant to such noise by design, as vectors chosen to represent objects must be significantly dissimilar from one another, and therefore small amounts of noise will still allow for identification of the chosen memorable object. We therefore find the effect of STICK introduced errors to be negligibly small in the context of HRR calculations as demonstrated in figure 12 which compares the results of this experiment performed using STICK and a native python implementation of the same experiment.

As expected, the error encountered in this experiment was larger than the error in the single operation experiments run in section III-D. Figure 12 shows the similarity measure for the vector produced by probing the memory vector with b and four base vectors. As expected this vector is significantly more similar to a than to any other vector. This demonstrates that the CHRR implementation in STICK is a feasible memory system.

C. Effect of Noise

The STICK framework is designed to be valid in a deterministic, noiseless environment with precise spike timings. In the previous experiments, we have provided such an environment. However this environment is unlikely to exist in any biological neural system. Therefore we have repeated the experiment from section III-D of calculating a single superposition of two CHRR vectors, with Poisson noise introduced to the network. This was done by by connecting each neuron to a Poisson spike source with an average firing rate of 0.1Hz, simulating spontaneous activity. A spike from a source will cause the target neuron to fire. Effectively, each neuron has a Poisson firing process added to its normal dynamics. As expected, STICK performance decreased notably. Random spiking activity disrupts the internal state of the STICK operators, causing unexpected behaviour, often completely inhibiting meaningful calculations. Further, for STICK networks to function for sequential inputs the neurons within the network must be reset to their initial state after the calculation; STICK blocks are all designed such that after producing an output, all neurons will have been reset to their initial state. As a result, errors in the network accumulate, rather than normalise over time.

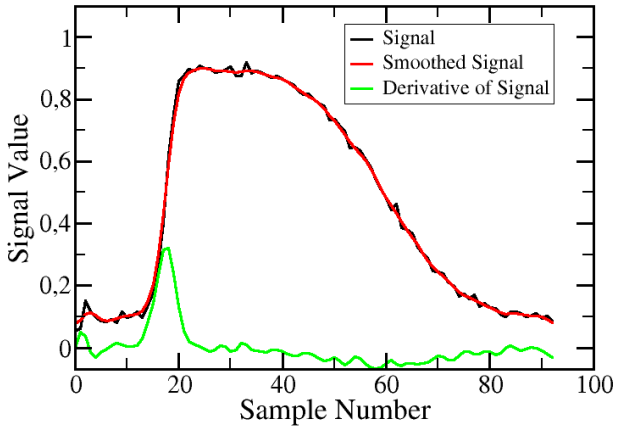

Fig. 5: With originally proposed STICK parameters.

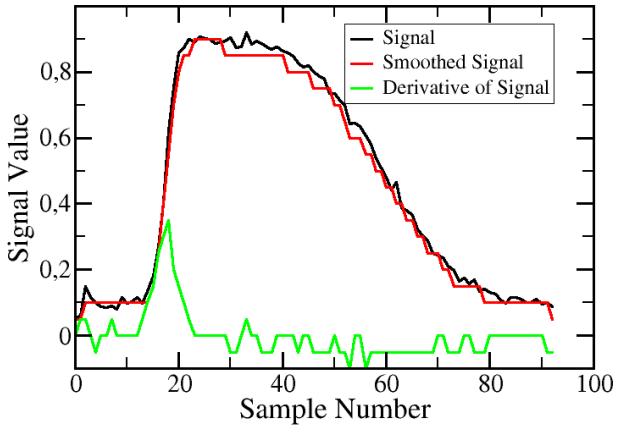

Fig. 6: With reduced STICK parameters

Fig. 7: The above plots show the results of signal processing performed in the STICK framework as discussed in section IV-A, and compare the effects of the STICK parameter changes discussed therein.

In our CHRR superposition experiment, the operation is performed independently on each dimension of the input vectors. It is therefore possible to view each dimension as an independent process, with a final result obtained if we have produced a valid output for each dimension. In order to evaluate the results, we must first account for invalid readouts, i. e. cases where the readout neuron only fires once (a valid number representation requires two spikes) or the spiking interval exceeds T_{cod} or is less than T_{min}. We cannot interpret the outputs from these neurons in any meaningful way, so they are omitted in the following evaluation. In order to still receive an output for each dimension, and to investigate the effect of error on successive operations, we repeat the input to the network, executing the calculation multiple times. Figure 13 shows the number of invalid results obtained in each iteration. After several iterations, the number of invalid results stabilises at around 100, or 20% of all dimensions.

While fig 13 already shows that invalid results become more frequent after the first few iterations, this is not the only type of error that occurs. In addition to producing completely unusable results, the noisy calculation can also produce valid, but incorrect results. Normally, CHRRs can, by their design, compensate for such noise. However, as the network is exposed to noise for longer and longer periods of time, the results become too inaccurate for CHRRs to operate as a functioning memory system. This is shown in figure 14. In this experiment, we average between the (valid) results of successive iterations. This does not lead to decreasing error, however, as the results become increasingly inaccurate with more iterations. The bottom plot shows the CHRR similarity measure between the calculated and expected result. While it is decreasing, even after 50 iterations, the result is not completely unusable. It is still significantly more similar to the true result than a random vector would be, and it is still dissimilar to the two vectors that were superposed. Nevertheless, it can be clearly seen that noise affects accuracy drastically.

We attempt an alternate approach by resetting the neurons after each iteration, as if they had spiked (i.e. resetting V, g_e, g_f and *gate*). We can now consider each iteration an independent evaluation of the experiment, as it is not affected by the previous iterations. Averaging over the results of each iteration does bring us closer to the true result in this case. This is shown in figure 15. We also make two observations about the error in this STICK network:

- Averaging over the iterations does not remove the error completely. This means that the mean of the error that is added to the output result is not zero; the error is systematic or biased.
- We do not converge to a similarity of 1.0. This is a result of the previous observation. Averaging the results of each iteration at best gives us $\mathit{TrueResult} + E[\mathit{error}]$, which is not equal to the true result if the error is biased.

Resetting the neurons in between iterations is not a part of the original (STICK) model. It could be seen as an extension to the model, adding another synapse type to perform the reset without causing the neuron to spike. Although we have not implemented this functionality as an extension to the model but rather performed the reset manually outside of it (in the neural simulator), our experiment suggests that functionality similar to this one is desirable for successful computation in time domain for relatively high presence of noise.

V. Discussion

A. Experiments

We have shown that this framework can be used to perform 1D signal processing in short times and with significant accuracy. We have shown that this framework can be used to implement a valid memory system of CHRRs and that this memory system provides feasible results.

B. Noise Resistance

We have analysed the accuracy of the framework under presence of external Poisson noise, expected to be present in

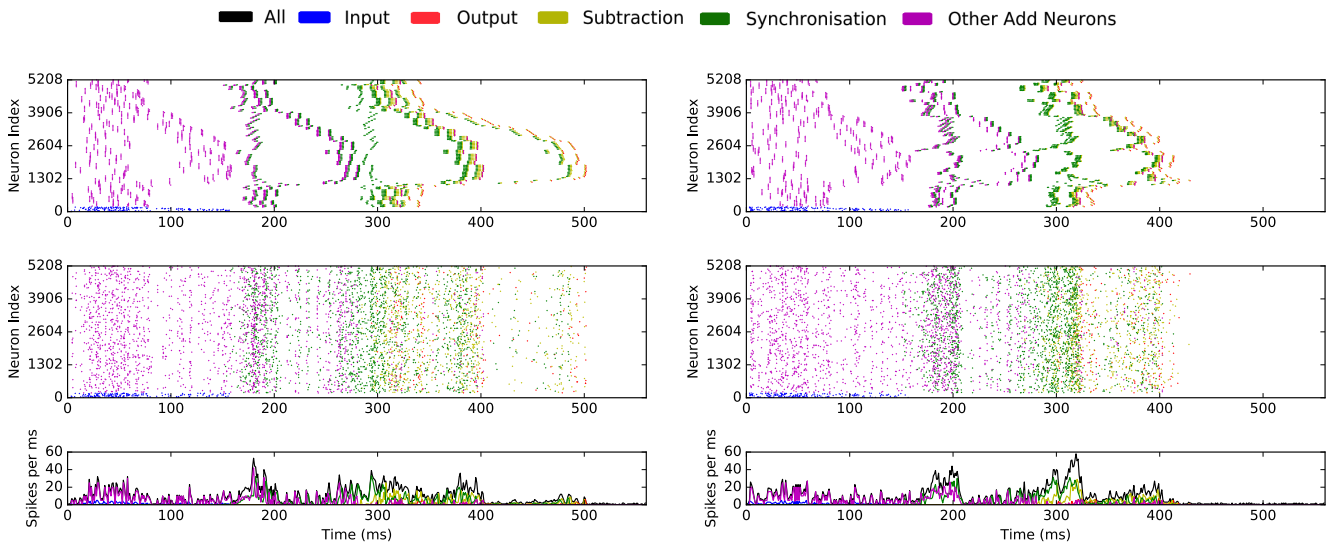

Fig. 8: Neuron spike time diagram for signal smoothing

Fig. 9: Neuron spike time diagram for edge detection

Fig. 10: Neuron spike timing plots showing scatter plots of the neuron spike times in an ordered manner where the signal being represented can be clearly seen, and in an unordered manner as would be seen in an cortical spiking activity recording. Also shown is the spike rate for the network and different neuron populations in the network. For further understanding of the role of each population, refer to section II-B and [12]

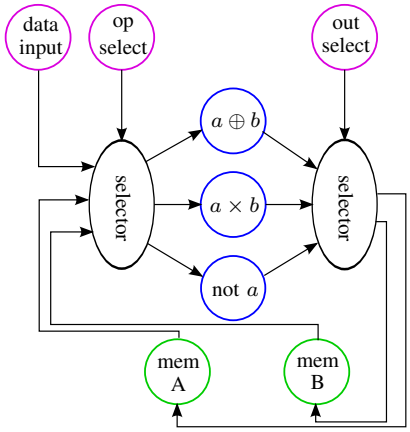

Fig. 11: Advanced STICK CHRR setup. Units in this graph represent blocks of neurons.

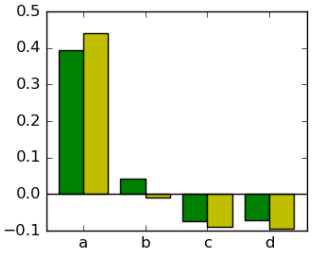

Fig. 12: CHRR similarity measure against the four starting vectors. Exact reference in green, results of our implementation in yellow.

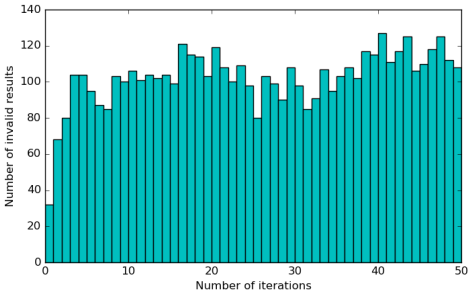

Fig. 13: When running the CHHR experiment in a noise environment, several output neurons do not produce meaningful (in the STICK framework) response. The above bar graph displays the number of output neurons (out of 500) which failed to produce meaningful result, and how this number increases as the network is run for repeated simulations.

a realistic environment. Our results show that recall accuracy decreases, as expected, but it does not render the proposed symbolic architecture useless. Furthermore, we propose a model of noise compensation mechanism which allows recall accuracy to improve over successive iterations. This suggests a STICK framework extension with a fifth synapse type which performs neuron reset without emitting a spike, which would be a useful addition to the framework from a computational standpoint.

C. Additions to the STICK framework

We have successfully extended the STICK framework with a new type of computational block performing addition with

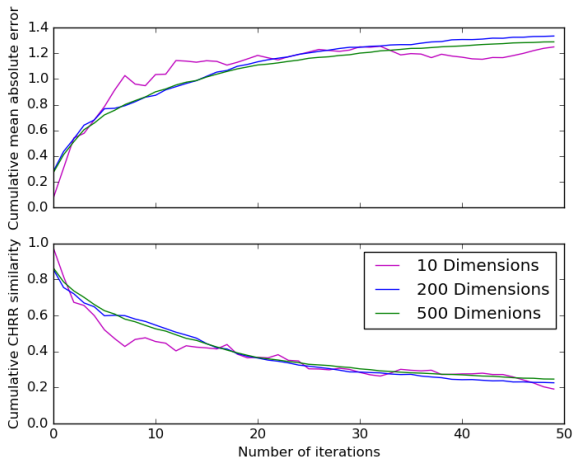

Fig. 14: Plot showing how the accuracy of the result from the noisy experiment described in sections IV-B and IV-C decreases with successive runs of the network

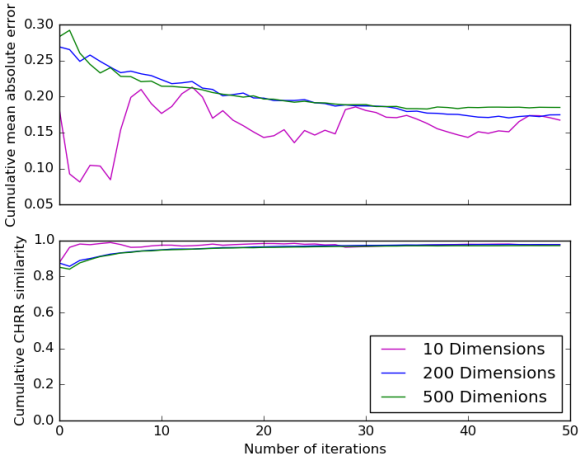

Fig. 15: Plot showing how the accuracy of the result from the noisy experiment described in sections IV-B and IV-C increases by averaging the result over independent trials of the experiment

overflow (modulus operator) in time domain. Although this extension in form of circular addition can be used in various applications, we have tested it in a specific type of holographic associative memories, called *Circular Holographic Reduced Representations (CHRR)*.

D. Comparison with Neural Engineering Framework

Direct comparisons with Neural Engineering Framework cannot be made because of the fundamentally different ways information is represented and manipulated in STICK. However it is safe to claim that this timing-based framework needs on average 20x less neurons and 1500x less synapses per operation than NEF equivalent (for example when performing addition). The reasons for this drastic improvement lies in two facts:

- NEF uses the firing rates of a population of neurons to represent a single value, whereas STICK uses the inter-spike timing of a single neuron
- NEF uses synaptic weights of all to all connections between populations to encode a function, while STICK uses neuron and synapse dynamics

When it comes to associative memories and cognitive architectures, NEF uses expensive convolution operation required for *Holographic Reduced Representations (HRR)*, whereas we use less expensive equivalent *CHRR*. Circular addition, required for CHHR to work, is highly nonlinear operation which is hard to be approximated effectively using linear approximation techniques used by NEF and this makes it difficult to implement CHRR in NEF.

E. Functional Analysis

Having an engineered computational models of associative memory and convolutional filtering enabled us various ways of visualizing internal network dynamics underlying actual operations. We were able to identify the spike times and populations of neurons that are in charge of a specific computation sub-step. Simple random scrambling of neuron ordering in spike plots resulted in spiking activity that a researcher would usually expect to see in a typical cortical spiking activity recordings. Our hope is that this framework will drive the development of spike train analysis tools like Elephant [13], used to discover patterns of synchrony in spike trains.

VI. Conclusions

We have successfully implemented the STICK function engineering framework [12] in the NEST neural simulator [11]. Using existing computational blocks of STICK we have built convolution circuit and have shown that it can be used to perform signal processing in time domain (smoothing with Gaussian kernel and edge detection with derivative kernel). We used this implementation of convolution circuit as a base operation to implement HRR model of associative memories.

Furthermore, we have implemented another model of associative memories - Circular HRRs (CHRR). In order to this, we had to extend STICK framework by proposing design of a new computational block for performing modulo arithmetics.

We have tested and compared these two models of associative memories and shown that CHRRs require smaller amount of neurons and scale better than equivalent HRR architectures of the same dimensionality, which makes them computationally less expensive and faster to simulate. Lastly, we have exposed these associative memory models to a small amount of background noise, modeling stochastic nature of cortical environment known to exist in mammalian cortices. Even though underlying computations are relying heavily on relative spike timing and each spike is expected to destroy the computation completely, we have found that recalling symbolic items from memory is still possible with slightly reduced accuracy. Furthermore, we have proposed computational

requirement for mitigating the effect of noisy environment, leading to increased accuracy of recall.

All of the work presented exploits advantages of using neural and synaptic dynamics as computational substrates. Models presented in this paper are one of, if not the very first ones, with this property in functional cognitive neuroscience.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 604102 (Human Brain Project).

REFERENCES

[1] K. H. Pribram., "Holonomic brain theory," *Scholarpedia*, vol. 2, no. 5, p. 2735, 2007.

[2] Karl H. Pribram, *Brain and Perception: Holonomy and Structure in Figural Processing (Distinguished Lecture Series)*. Psychology Press, 2013.

[3] Tony A. Plate, *Holographic Reduced Representation: Distributed Representation for Cognitive Structures (Lecture Notes)*. Center for the Study of Language and Inf, 2015.

[4] Tony A Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–641, 1995.

[5] T. A. Plate, "Distributed representations and nested compositional structure," Ph.D. dissertation, University of Toronto, 1994.

[6] Eliasmith C. and Anderson C.H., *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. Cambridge, MA: MIT Press, 2003.

[7] Eliasmith C., "A unified approach to building and controlling spiking attractor networks," *Neural computation*, vol. 7, pp. 1276–1314, 2005.

[8] Blouw, Peter and Solodkin, Eugene and Thagard, Paul and Eliasmith, Chris, "Concepts as semantic pointers: A framework and computational model," *Cognitive Science*, vol. 40, no. 5, pp. 1128–1162, 2016. [Online]. Available: http://dx.doi.org/10.1111/cogs.12265

[9] Eliasmith, Chris and Stewart, Terrence C. and Choo, Xuan and Bekolay, Trevor and DeWolf, Travis and Tang, Yichuan and Rasmussen, Daniel, "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–1205, 2012. [Online]. Available: http://science.sciencemag.org/content/338/6111/1202

[10] W. Maass, "Liquid state machines: Motivation, theory, and applications," *World Scientific Review*, 2010.

[11] H. Bos, A. Morrison, A. Peyser, J. Hahne, M. Helias, S. Kunkel, T. Ippen, J. M. Eppler, M. Schmidt, A. Seeholzer, M. Djurfeldt, S. Diaz, J. Morn, R. Deepu, T. Stocco, M. Deger, F. Michler, and H. E. Plesser, "Nest 2.10.0," Dec. 2015. [Online]. Available: http://dx.doi.org/10.5281/zenodo.44222

[12] X. Lagorce and R. Benosman, "Stick: Spike time interval computational kernel, a framework for general purpose computation using neurons, precise timing, delays, and synchrony," *Neural Computation*, vol. 27, no. 11, pp. 2261–2317, Sep 2015.

[13] . The NeuralEnsemble Initiative. (2016) Elaphant. [Online]. Available: http://neuralensemble.org/elephant/