Biologically-plausible hierarchical chunking on mixed-signal neuromorphic hardware

Atilla Schreiber[1], **Shuchen Wu**[2], **Chenxi Wu**[1], **Giacomo Indiveri**[1], **Eric Schulz**[2]
1. Institute of Neuroinformatics, UNI–ETH Zurich, Switzerland
2. Max Planck Institute for Biological Cybernetics, Tübingen, Germany

Abstract

Humans seamlessly group perceptual sequences in units of chunks, parsed and memorized as separate entities. Chunking is a computational principle essential for memory compression, structural decomposition, and predictive processing. How can this ability be accomplished in a neural system? On an algorithmic level, computational models such as the Hierarchical Chunking Model (HCM) propose grouping frequently occurring proximal observational units as chunks. Chunks, once learned, are stored as separate entities in memory, ready for reuse and recombination. In doing so, the HCM learns an interpretable and hierarchical representation that resembles human chunk learning, without the need for gradient based training. In this work, we propose a biologically plausible and highly efficient implementation of the HCM with spiking neurons: the neuromorphic HCM (nHCM). When parsing through perceptual sequences, the nHCM uses sparsely connected spiking neurons to construct hierarchical chunk representations. Simulation on a standard computer showed remarkable improvement of nHCM in speed, power consumption, and memory usage compared to its original counterpart. Taking it one step further, we validate the model on mixed-signal neuromorphic hardware DYNAP-SE 2, which uses analog spiking neurons in an event-driven way to imitate biological computation. The transistors in the neural cores are run in sub-threshold, reducing energy requirements by more than one thousand times. We verified this implementation's robust computing properties, overcoming the analog circuits' heterogeneity, variability, and low precision. This work demonstrates cognitively plausible sequence learning in energy-efficient dedicated neural computing electronic processing systems.

1 Introduction

When parsing through perceptual sequences, humans tend to segregate the stream of perceptual input into units of chunks [11]. These segregated units are stored in memory, ready for reuse and recombination. This ability, termed chunking, enables us, for example, to learn to play piano by piecing together phrases, or excel at a foreign language via flexibly combining the previously acquired words. Chunking has been suggested to improve memory compression, prediction, planning, generalization, and transfer across diverse psychological domains [13, 2, 6].

Cognitive models such as the hierarchical chunking model (HCM) propose using chunks as building blocks to recursively learn and acquire more complex representations [20]. The chunk learning algorithm processes the ability to transfer representation and resembles human sequence learning [21]. While it remains unclear how chunking works in the brain, the neuromorphic HCM (nHCM) provides a potential implementation of human chunking in biological neurons.

Neuromorphic hardware is a novel computing paradigm with high potential. The asynchronous mixed-signal neuromorphic processor DYNAP-SE 2 [12, 16] is comprised of analog neuron and

37th First Workshop on Machine Learning with New Compute Paradigms at NeurIPS 2023(MLNPCP 2023).

synapse circuits that can be interconnected and configured for computation. The neuron circuits operate in the sub-threshold regime, leading transistors that are orders of magnitudes more energy-efficiency than state-of-the-art hardware [12], which are best suited for power-limited applications such as edge-computing or robotics [1]. Different from rate-based neurons in software simulations, the analog neurons in the neuromorphic chip make use of the temporal information of the spikes from connected neurons and are inherently parallel, overcoming the von Neumann bottleneck by combining processing with memory storage [14]. However, due to device-mismatch in the analog circuit components, obtaining robust and reliable computations at the system level, for arbitrary applications remains an open challenge [22].

In this work, we adapt and simplify the original HCM algorithm following a neuromorphic programming paradigm, thereby harnessing the computational power efficiency of neuromorphic hardware and the interpretability of cognitive models, while stepping toward a biologically plausible sequence chunking implementation. In contrast to traditional recurrent neural network approaches using long short-term memory [7] or more recently transformers [19] the HCM does not require backpropagation [17]. While gradient-based training was successfully implemented on digital neuromorphic processors[4, 10], the component variability caused by analog circuits makes such approaches unfeasible. In addition, deep neural networks are not interpretable which causes key shortcomings [8, 5] and concerns regarding safety and trustworthiness [18, 15, 3]. The HCM on the other hand learns the structure directly and represents it symbolically and hierarchically. The nHCM expands on this, by constructing the entire network with identical small neuron circuits each representing a chunk wired into a hierarchy. We illustrate the adaptation of HCM to neuromorphic design philosophy implemented on a computer and on a neuromorphic processor, followed by a set of evaluations to assess the power and computational efficacy the implementation affords.

2 Methods

2.1 Overview

The original HCM assumes that independent samples of chunks constructed the observed sequence. The goal of the algorithm is to learn these chunks, which are sequence segments comprising one or multiple symbols, and use the chunks to parse the sequence. For this, the algorithm starts with chunks being the atomic sequential units. When parsing, it notes the frequency of each chunk and each chunk transition. After parsing, a χ^2 test of independence picks the pair of chunks to be concatenated into a new chunk. This process is repeated until the independence test is no longer passed or a satisfactory number of chunks has been reached. If the assumptions are met, the algorithm is guaranteed to converge.

We have implemented sequence parsing in chunks on the neuromorphic chip DYNAPSE-2 [12] and with a neuron-emulating data structure on a traditional computer. As illustrated in Fig. 1A, nHCM also parses a perceptual sequence in units of chunks. To adapt HCM to a neural population to become nHCM, we first adjust the algorithm to a heuristic version and propose new chunks by sampling. In this way, nHCM learns chunks online. These chunks are manifested in a hierarchy, as illustrated in Fig. 1B. At a neural level, nHCM uses a neuron chain to encode the temporal information of each chunk and a disinhibition mechanism to ensure the activation of higher-order chunks dependent upon the relative timing of its subordinate chunks, while inhibiting its subordinates in favor of the better explanation. New neurons are recruited and integrated upon the formation of a new chunk. On the neuromorphic processor, a combination of delay, coincidence detection and cross-inhibition is used, whereas, on a computer, the implementation is done with a queue. We first introduce sampling in section 2.2, followed by delay, coincidence detection, and cross-inhibition from section 2.3.1 to 2.3.3.

2.2 Sampling

Instead of an independence test, nHCM uses a sampling method to propose new chunk pairs to concatenate into a new chunk. To do so, every time a new chunk is parsed, it is sampled probabilistically in combination with the previous chunk. If the same pair of chunks gets sampled k-times in a row, it will be combined into a new chunk and stored in long-term memory. Specifically, for each chunk transition the nHCM parses, there is a certain possibility or sampling rate (0.05 by default) to save that chunk transition. If this chunk transition is recorded k (k=2,3, ...) times in a row, the

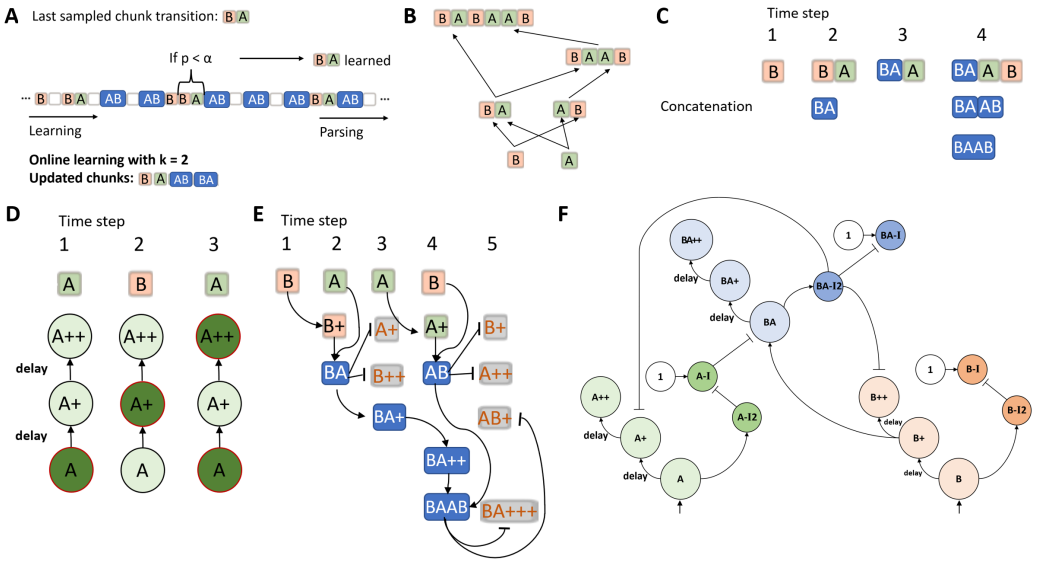

Figure 1: **nHCM implementation on a computer and a neuromorphic processor** (A) Online learning while parsing the sequence with $k = 2$. Sampling occurs some time steps after the parsing, once the neuron chains have reached their output neurons. (B) Example hierarchy. Adapted from [20]. (C) Queue implementation on a computer. Whenever an input is parsed, as many concatenation steps as possible of the rightmost chunks are done. (D) Illustration of plus-neuron population activation of a chunk A during the parsing of sequence ABA. The signal travels one neuron per time step. (E) Run time illustration of activation and inhibition of neurons of chunks A, B, AB, BA, BAAB during the sequence BAAB. Inhibited neurons are grayed out and the excitatory connections are left out. (F) Neuronal disinhibition circuit of chunks A, B, and BA. 1 indicates constitutive activation of inhibitory interneurons. Subsequent activation of population B and then A will lead to disinhibition and activation of AB neurons. Feedback from AB neurons suppresses downstream neurons of chunk population A and B. On chip neurons can be excitatory and inhibitory at the same time, removing the need for I2 neurons

parsing ends, and we add the chunk transition as a new chunk before starting the next iteration. We decided to implement at least two identical samples in a row so that a chunk transition is learned on the probability squared of occurrence. This asymmetrically enforces learning of more frequent transitions.

The sampling method replaces the frequency collection and independence test of the original algorithm. This adaptation is necessary to transition the algorithm to a neuromorphic processor. It slightly decreases the accuracy to trade off with online learning during the model run-time. The effect of removing the independence test depends on the structure of the underlying sequence and is further discussed in the section 4.1.

2.3 Neuromorphic Processor

2.3.1 Delay

A delay mechanism underlies chunk detection on the neuromorphic chip. As the hardware is event-driven and running in parallel and all information is represented in spikes, precise timings are essential to execute any successful algorithm. To encode temporal information after each chunk has been detected, several delay neurons are assigned to each chunk, each symbolizing a distinct time step after the moment of chunk detection. Fig. 1D illustrates a neuron population with delayed synapses. When chunk A activates, the primary neuron fires, and one of its synapses is delayed to activate neuron A+ one timestep later. This repeats for A++ and subsequent plus iterations until the information of chunk A is no longer necessary.

To ensure proper activation, parent chunks are only activated by the corresponding plus neurons and use coincidence detection to determine whether the two neurons have fired at the same time. For example, a chunk AB would fire if the chunk A has been delayed by exactly one interval, and neuron A+ is active at the time of the arrival of chunk B, whereas the chunk $A - BB$ would receive inputs only from A++ and BB. The neuron chain also solves the issue of charge accumulation that would otherwise lead to chunk activation by repeats of the first input alone, as elaborated in 6.3. This mechanism is not necessary on a computer and was instead solved using a queue, discussed in section 2.4.

2.3.2 Coincidence detection

Coincidence detection is an essential part of the algorithm, as neuromorphic hardware is inherently noisy due to manufacturing imperfections of analog circuits. This results in constitutively active neurons, large mismatch, variable delay from synapses, and differences in spike amplitude and activation thresholds, which require robust and error-mitigating algorithms. For a bio-plausible coincidence detection method, we picked a disinhibition approach, which has been suggested to facilitate precise firing windows in the brain [9]. A newly implemented chunk population includes an interneuron from the second child population, that constitutively inhibits the new chunk, and only once that interneuron has inhibited itself, can an input activate the population. This asymmetric architecture eliminates most false positives, which are the most harmful for our model as a chunk neuron firing too frequently strongly influences subsequent learning behavior. False negatives are less harmful, as the chunk can be relearned, with a different set of neurons replacing the malfunctioning ones. The disinhibition has proven to be more easily tuned and resistant to mismatch than an AMPA/NMDA method. As illustrated in Fig. 1F, we picked the first input to be the activator and the second input for the disinhibition. As the second input always stems from the first neuron in the delay chain, assigning it for disinhibition minimizes the number of neurons needed in the topology. We implemented the architecture on the chip to allow for coincidence detection between 4 and 250 milliseconds intervals. Faster interval times are increasingly hard to achieve due to the inherent excitation time that a neuron takes to charge and fire. Slower speeds are limited mostly by mismatch and their viability for practical applications. The final implementation of the nHCM uses an interval time of 10 ms.

2.3.3 Learning and cross-inhibition

Only the last instance of the plus-delay chain counts as an activation of the chunk for learning purposes. As a parent chunk will fire immediately after its children have fired, it will activate before the next plus neuron in the children's delay chain will activate.
The parent neuron is wired to inhibit the next plus neuron of its children in case of activation. This way, the plus-delay chain of the children will be broken, and the last instance of the chain, which counts as the final activation, will never be activated. If we have a sequence ABC with chunks $\{A, B, C, A - B, B - C, AB - C\}$, the chunk AB will stop the chunk B from being relayed another time step and activate BC. However, ABC can still be activated.
This takes advantage of the fact that only proximal chunks get learned as new chunks, so a new signal can never lead to multiple chunk activations. A runtime example is illustrated in Fig. 1E, and a complete three-chunk circuit is illustrated in Fig. 1F.

2.4 Computer

2.4.1 Queue

When implementing the design on traditional computers, no considerations for coincidence detection and delay are necessary. Instead, a queue is used, that keeps track of the most recent chunk activations. Whenever a new entry is parsed, it gets compared with the entry before that in the queue. If they form a known chunk together, they concatenate. This process is repeated until the chunks no longer concatenate, as illustrated in Fig. 1C. The queue size is set to be one larger than the hierarchy depth. When the size is exceeded, the oldest chunk gets removed, and the transition between the removed and the new oldest chunk is probabilistically sampled. We also implemented a deterministic method where all chunks are recorded and the most frequent transition is selected.

3 Results

3.1 Computer Implementation

The design principles implemented on a computer using a deterministic version and a probabilistic variant that is similar to the neuromorphic implementation were compared to the original model. An evaluation of bits needed to encode the sequence during the learning process on a handcrafted sequence Fig. 2A and on a generated hierarchy Fig. 2B shows that the nHCM in some cases compresses more quickly, due to learning the most frequent chunks first. In our experiments over 30 trials with a sequence length of 5000 symbols, Fig. 2C, the original, rational model took an average of 9.52 $\pm$ 0.81 seconds. The deterministic version took 0.22 $\pm$ 0.03 and the probabilistic 0.07 $\pm$ 0.02 seconds, giving us a significant improvement for each group comparison (two-sample t-test $<$ 0.05). The adaptations lead to a speed-up of about two magnitudes to the original version. To compare accuracy between models, a sequence is generated from the learned hierarchy, which is then partitioned into chunks using the ground truth hierarchy. The frequency distribution of the ground truth hierarchy with probabilities based on the generated sequence and the original sequence are then evaluated on their similarity using the Kullback-Leibler (KL) divergence. The rational (original) model achieved a KL divergence of 0.71 $\pm$ 0.14, the deterministic version of 0.75 $\pm$ 0.17, and the probabilistic of 1.12 $\pm$ 0.28. No significant difference was observed between the rational and deterministic version (two-sample t-test $p = 0.335$) but between the probabilistic and the other two (two-sample t-test $p \leq 0.05$). The models were tested on sequences generated by the method developed in the original paper [20].

3.2 Neuromorphic processor implementation

The nHCM on the neuromorphic processor achieves comparable results in the experiments (Fig. 2D). As the hardware has a fixed time constant, no speed comparisons were made. During 3 trials with a sequence length of 500 the rational model showed a mean KL divergence of 0.051 $\pm$ 0.02 and the neuromorphic processor implementation 0.398 $\pm$ 0.269. While the performance of the neuromorphic model is significantly worse (one-tailed two-sample t-test $p \leq 0.05$), we show that it can work in theory. Core issues stem from the hardware itself, and with further improvement to the technology and fine-tuning of the parameters we expect an improvement to similar values as the probabilistic version. Example outputs can be seen in Fig. 2E, displaying the hardware mismatch and noise.

4 Discussion

4.1 Removing the independence test

Removing the independence test, changes the overall objective of the algorithm. The nHCM will find the most frequent chunks that make up the sequence, while the original algorithm will only find chunks that appear more often than their components would suggest. The preferable method depends on the sequence and the objective. We constructed a way to reintroduce the independence test and implemented it on a computer. To achieve this, the sampling rate of a transition is set by the weight of the synapse from the final plus neuron leading to the sampling circuit. This weight is reduced whenever the neuron is active while simultaneously increasing all other chunks by a fraction of the same amount. This normalizes the chunk sampling and as it is rooted in the frequency of component chunks, transitions that appear more often than by random chance have an increased probability of sampling. This biases the learning circuit to discover underlying chunks.

4.2 Hierarchy depth

Even though we did not observe this with our experiments, we theorize that the algorithm might eventually learn the deepest possible hierarchy given enough time and data due to its probabilistic nature. This might lead to an underlying hierarchy $\{$AB, CD, ABCD$\}$ being learned as $\{$A-B,AB-C,ABC-D, C-D$\}$ even if the more concise chunking is available, as the chunks ABC might show up without the concluding D. Once the chunks ABC is learned, it will break the chunk AB-CD as it will be chunked into ABC first. A deeper hierarchy will lead to more neurons needed and higher-order chunking will lose precision due to accumulation of hardware irregularities. If observed, this should also be solved by reintroducing the independence test.

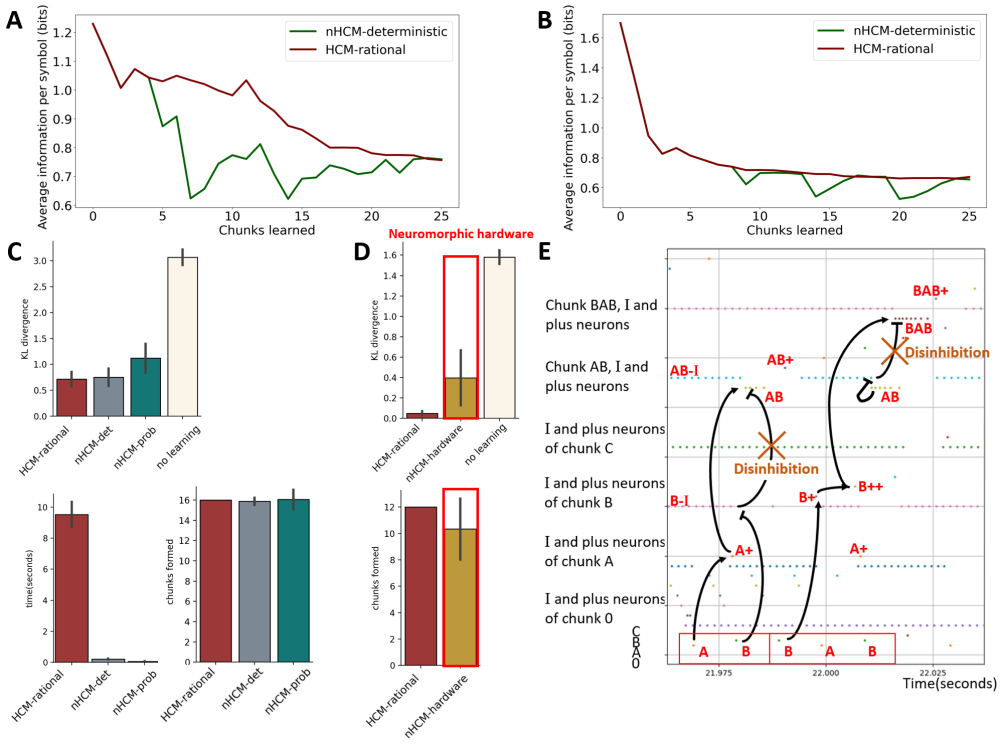

Figure 2: **nHCM results on a computer and neuromorphic processor compared to the HCM** (A,B) Decline in bits needed to encode a sequence of length 5000 using the rational (original) HCM and the deterministic nHCM. (A) on a hand-made ground truth hierarchy ("transfer original") (B) on a generative model with 12 complex chunks and 4 atomic chunks. (C) Experiments with rational HCM, deterministic version of nHCM and probabilistic version with online learning on a computer (Chunks 16, Sequence length 5000, n = 30). (Top) KL divergence when compared to no learning (Bottom left) Speed comparison. (Bottom right) Number of chunks formed. (D) Experiments with rational HCM on a computer and nHCM on the neuromorphic processor (Chunks 12, Sequence length 500, n = 3). (Top) KL divergence compared to no learning. (Bottom) Number of chunks formed. (E) Spike train output of the nHCM on the neuromorphic processor showing chunking of inputs AB and BAB. Primary activation of atomic elements at the bottom. Chunks are ordered from bottom to top as primary activation, constitutive inhibition, and downstream delayed neurons with temporal information. I2 interneurons not included.

4.3 Future Work

While the algorithm works as a proof of concept, it has not yet been applied to a real-world problem. We see future work focusing mainly on making the time-components more flexible and applying the nHCM to bio-signal processing in energy-limited environments. The nHCM on standard computers, should be evaluated on its capability to evaluate large quantities of real data.

5 Conclusion

In this paper, we investigate a bio-plausible implementation of the HCM on standard computers and a neuromorphic processor. We show that replacing the frequency collection and independence test with a sampling method transforms the algorithm into a heuristic version that increases the computation speed significantly while retaining high accuracy. We also demonstrate the nHCM running on a neuromorphic processor, successfully taking advantage of the inherent parallel computation and energy efficiency. Finally, we demonstrate that neuroplausible design encourages working principles that are intrinsically efficient and robust to component failure and noise.

References

[1] Chiara Bartolozzi, Giacomo Indiveri, and Elisa Donati. Embodied neuromorphic intelligence. Nature Communications, 13(1):1024, Feb 2022.

[2] Timothy F. Brady, Talia Konkle, and George A. Alvarez. Compression in Visual Working Memory: Using Statistical Regularities to Form More Efficient Memory Representations. Journal of Experimental Psychology: General, 138(4), 2009.

[3] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608, 2017.

[4] Steve K Esser, Rathinakumar Appuswamy, Paul Merolla, John V Arthur, and Dharmendra S Modha. Backpropagation for energy-efficient neuromorphic computing. Advances in neural information processing systems, 28, 2015.

[5] Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. Cognition, 28(1-2):3–71, 1988.

[6] Fernand Gobet, Peter C.R. Lane, Steve Croker, Peter C.H. Cheng, Gary Jones, Iain Oliver, and Julian M. Pine. Chunking mechanisms in human learning. Trends in Cognitive Sciences, 5(6), 2001.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[8] Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In International conference on machine learning, pages 2873–2882. PMLR, 2018.

[9] Johannes J. Letzkus, Steffen B.E. Wolff, and Andreas Lüthi. Disinhibition, a circuit mechanism for associative learning and memory. Neuron, 88(2):264–276, 2015.

[10] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. Science, 345(6197):668–673, 2014.

[11] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 101(2):343–352, 1956.

[12] Saber Moradi, Qiao Ning, Fabio Stefanini, and Giacomo Indiveri. A scalable multi-core architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). CoRR, abs/1708.04198, 2017.

[13] A. Newell and Paul Rosenbloom. Mechanisms of skill acquisition and the law of practice. Cognitive Skills and Their Acquisition, Vol. 1, 08 1993.

[14] Bipin Rajendran, Abu Sebastian, Michael Schmuker, Narayan Srinivasa, and Evangelos Eleftheriou. Low-power neuromorphic hardware for signal processing applications: A review of architectural and system-level design approaches. IEEE Signal Processing Magazine, 36(6):97–110, 2019.

[15] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144, 2016.

[16] Ole Richter, Chenxi Wu, Adrian M. Whatley, German Köstinger, Carsten Nielsen, Ning Qiao, and Giacomo Indiveri. Dynap-se2: a scalable multi-core dynamic neuromorphic asynchronous spiking neural network processor, 2023.

[17] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. nature, 323(6088):533–536, 1986.

[18] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296, 2017.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[20] Shuchen Wu, Noemi Elteto, Ishita Dasgupta, and Eric Schulz. Learning structure from the ground up—hierarchical representation learning by chunking. In Thirty-Sixth Conference on Neural Information Processing Systems, 2022.

[21] Shuchen Wu, Noémi Éltető, Ishita Dasgupta, and Eric Schulz. Chunking as a rational solution to the speed–accuracy trade-off in a serial reaction time task. Scientific Reports, 13(1):7680, May 2023.

[22] Dmitrii Zendrikov, Sergio Solinas, and Giacomo Indiveri. Brain-inspired methods for achieving robust computation in heterogeneous mixed-signal neuromorphic processing systems. Neuromorphic Computing and Engineering, 3(3):034002, July 2023.

6 Supplementary material

6.1 Sampling rate

The probability we used for sampling, is based on the assumption that a certain hierarchy is governing the sequence. The sampling rate needs to be low enough, to escape local dependencies and ensure that chunks can be learned. Else, if there are large chunks without repeats, we will not be able to encounter a constituent twice in a row, as we sample again before we leave the local chunk. If we know that the chunks that govern the sequence are very large, the sampling rate has to be adjusted to be lower. For a sequence governed by small chunks, we could increase the sampling rate to speed up the learning process, but caution is advised, as we usually cannot be certain of the chunk size, and by its probabilistic nature, even the lowest sampling rate introduces a bias. We recommend a sampling rate that has a maximum probability of 0.05 to sample twice within the largest chunk size. We say p_s is our sampling rate and use C_{max} as the largest chunk we expect in our underlying hierarchy. The number of samplings within the largest chunk follows a binomial distribution, and the condition that this number is smaller than 2 is F, where F(x; N, p) is the CDF of a binomial distribution. and can be calculated with the formula (1):

$$(1 - p_s)^{C_{max}} + C_{max} p_s (1 - p_s)^{C_{max}-1} > 0.95 \tag{1}$$

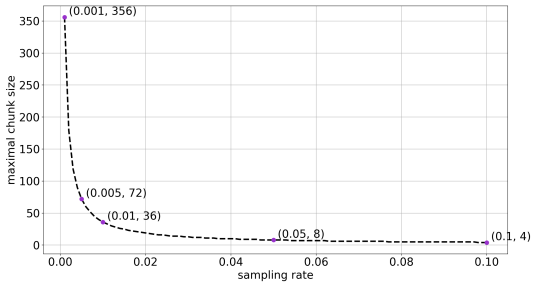

Figure 3: The maximum chunk size the algorithm can parse without sampling twice within a chunk in correlation with the sampling rate.

As we sample chunk transitions, we should add plus one to the maximum length of chunks, as the sampling takes place on two proximal entities at once. Thus for the base rate of 0.05, the probability of sampling a chunk twice is lower than 5% for chunks up to the size of 8. For a sampling rate of 0.01, the maximum chunk length increases to 36.

6.2 DYNAP-SE 2

The DYNAP-SE 2 is a the neuromorphic chip that was used for the nHCM. It is made up of 4 cores, each with a total of 1024 neurons. Each neuron has a fan-in of maximally 64 Synapses, meaning that it can only receive signals from 64 synapses in total. the fan-out is much larger and a neuron can have outgoing synapses to all other neurons. The neurons work event-driven without a clock governing the system. As the chip is comprised only of neurons and synapses, no conventional memory exists. To store information, one has to implement neurons and synapses to encode this information. This is especially limiting for arithmetic functions and requires entirely different approaches to implement algorithms. This design principle drove our design to end up in the architecture used.

The number of synapses available per neuron, limit the number of chunks that an individual chunk can activate. A neuron has a total of 64 synapses. A total amount of 8 synapses are used to activate the next instance in the chain, 4 to inhibit each child respectively and 4 are used per new chunk created. A child of same temporality can thus activate a maximum of 12 parent chunks. This maximum is present for each instance of the plus-chain individually, but as the right chunk always uses its first chain-instance to activate the parent, a chunk will be limited to around 24 parent chunks. Multiple synapses are used simultaneously to average out chip-inherent mismatch.

6.3 Delay

Alternatives to the delay approach could have been AMPA/NMDA or a simple charge addition system, but would need a non-zero voltage in a neuron one interval after the first input has fired, to enable interaction with the second input. This makes an accumulation of charge solely by repeats of the left input possible. This could cause potential misfiring if the underlying sequence contains high repeats, no matter how small the voltage of the first input is. Another reason why the delay approach was chosen, is that it allows for implementing chunks that have the same child neuron twice. For example, the chunk $B - B$ should only fire when BB appears in the sequence but not with a single B. With the delay approach, we do not need to distinguish between left and right input as all other methods would have to. Instead, we can wire an activation resulting from the simultaneous activation of B+ and B

This method also guarantees that the wiring stays simple and that underlying child chunks can be inhibited. The chip constrains the delay to be no longer than the interval time, as a chunk cannot start firing again before the last firing is complete. This means that if the delay were longer than the interval time, the sequence AA would only activate A+ once, for the first A. This results in a slight mismatch between the delay or plus instances of neurons and the next interval. This will accumulate for all plus neurons, but does not traverse chunk hierarchies as the higher order chunks will fire immediately after the second input, and the second input is never delayed. Due to this delay accumulation, the primary input needs to have a low time constant Tau to bridge the gap between the primary and secondary input. The time constant Tau, determining the voltage decay needs to be kept low for the primary input, to bridge the mismatch caused by the delay. This leads to multiple spikes from the parent chunk but does not alter any subsequent firings and is reduced to one spike again after the first plus instance. Another issue is that the inhibition of the parent neuron needs to be deactivated extremely quickly by the disinhibition, to allow for an immediate response. Further tuning of this process is necessary, especially for long delays