# Compressed Counting with Spiking Neurons (Extended Abstract)*

Yael Hitron Merav Parter†‡

**Abstract**

We consider the task of measuring time with probabilistic threshold gates implemented by bio-inspired spiking neurons. In the model of *spiking neural networks*, network evolves in discrete rounds, where in each round, neurons fire in pulses in response to a sufficiently high membrane potential. This potential is induced by spikes from neighboring neurons that fired in the previous round, which can have either an excitatory or inhibitory effect.

Discovering the underlying mechanisms by which the brain perceives the duration of time is one of the largest open enigma in computational neuro-science. To gain a better algorithmic understanding onto these processes, we introduce the *neural timer* problem. In this problem, one is given a time parameter $t$, an input neuron $x$, and an output neuron $y$. It is then required to design a minimum sized neural network (measured by the number of auxiliary neurons) in which every spike from $x$ in a given round $i$, makes the output $y$ fire for the subsequent $t$ consecutive rounds.

We first consider a deterministic implementation of a neural timer and show that $\Theta(\log t)$ (deterministic) threshold gates are both sufficient and necessary. This raised the question of whether randomness can be leveraged to reduce the number of neurons. We answer this question in the affirmative by considering neural timers with spiking neurons where the neuron $y$ is required to fire for $t$ consecutive rounds with probability at least $1-\delta$, and should stop firing after at most $2t$ rounds with probability $1-\delta$ for some input parameter $\delta \in (0,1)$. Our key result is a construction of a neural timer with $O(\log\log 1/\delta)$ spiking neurons. Interestingly, this construction uses only *one* spiking neuron, while the remaining neurons can be deterministic threshold gates. We complement this construction with a matching lower bound of $\Omega(\min\{\log\log 1/\delta, \log t\})$ neurons. This provides the first separation between deterministic and randomized constructions in the setting of spiking neural networks.

Finally, we demonstrate the usefulness of compressed counting networks for *synchronizing* neural networks. In the spirit of distributed synchronizers [Awerbuch-Peleg, FOCS'90], we provide a general transformation (or simulation) that can take any synchronized network solution and simulate it in an asynchronous setting (where edges have arbitrary response latencies) while incurring a small overhead w.r.t the number of neurons and computation time.

*The full version appears on arXiv:1902.10369

†Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. Emails: {yael.hitron,merav.parter}@weizmann.ac.il.

‡Supported in part by the BSF-NSF grants.

# 1 Introduction

Understanding the mechanisms by which brain experiences time is one of the major research objectives in neuroscience [MHM13, ATGM14, FSJ+15]. Humans measure time using a global clock based on standardized units of minutes, days and years. In contrast, the brain perceives time using specialized neural clocks that define their own time units. Living organisms have various other implementations of biological clocks, a notable example is the circadian clock that gets synchronized with the rhythms of a day.

In this paper we consider the algorithmic aspects of measuring *time* in a simple yet biologically plausible model of *stochastic spiking neural networks* (SNN) [Maa96, Maa97], in which neurons fire in discrete pulses, in response to a sufficiently high membrane potential. This model is believed to capture the spiking behavior observed in real neural networks, and has recently received quite a lot of attention in the algorithmic community [LMP17a, LMP17b, LMP17c, LM18, LMPV18, PV19, CCL19]. In contrast to the common approach in computational neuroscience and machine learning, the focus here is not on general computation ability or broad learning tasks, but rather on specific algorithmic implementation and analysis.

**Spiking Neural Networks (SNN).** The SNN network is represented by a directed weighted graph $G = (V, A, W)$, with a special set of neurons $X \subset V$ called *inputs* that have no incoming edges, and a subset of *output* neurons[1] $Y \subset V$. The neurons in the network can be either deterministic threshold gates or probabilistic threshold gates. As observed in biological networks, and departing from many artificial network models, neurons are either strictly inhibitory (all outgoing edge weights are negative) or excitatory (all outgoing edge weights are positive). The network evolves in discrete, synchronous *rounds* as a Markov chain, where the firing probability of every neuron in round $\tau$ depends on the firing status of its neighbors in the preceding round $\tau - 1$. For probabilistic threshold gates this firing is modeled using a standard sigmoid function. For a complete description of the model see [HP19]. Observe that an SNN network is in fact, a *distributed network*, every neuron responds to the firing spikes of its *neighbors*, while having no global information on the entire network.

**Remark.** In the setting of SNN, unlike classical distributed algorithms (e.g., LOCAL or CONGEST), the algorithm is fully specified by the *structure* of the network. That is, for a given network, its dynamic is fully determined by the model. Hence, the key complexity measure here is the size of the network measured by the number of auxiliary neurons[2]. For certain problems, we also care for the tradeoff between the size and the computation time.

# 2 Measuring Time with Spiking Neural Networks

We consider the algorithmic challenges of measuring time using networks of threshold gates and probabilistic threshold gates. We introduce the *neural timer* problem defined as follows:

> Given an input neuron $x$, an output neuron $y$, and a time parameter $t$, it is required to design a small neural network such that any firing of $x$ in a given round invokes the firing of $y$ for exactly the next $t$ rounds.

In other words, it is required to design a succinct timer, activated by the firing of its input neuron, that alerts when exactly $t$ rounds have passed.

A trivial solution with $t$ auxiliary neurons can be obtained by taking a directed chain of length $t$ (Fig. 1): the head of the chain has an incoming edge from the input $x$, the output $y$ has incoming edges from the input $x$, and all the other $t$ neurons on the chain. All these neurons are simple $OR$-gates, they fire in round $\tau$ if at least one of their incoming neighbor fired in round $\tau - 1$. Starting with the firing of $x$ in round 0, in each round $i$, exactly one neuron, namely the

[1] In contrast to the definition of *circuits*, we do allow output neurons to have outgoing edges and self loops. The requirement will be that the value of the output neurons converges over time to the desired solution.

[2] I.e., neurons that are not the input or the output neurons.

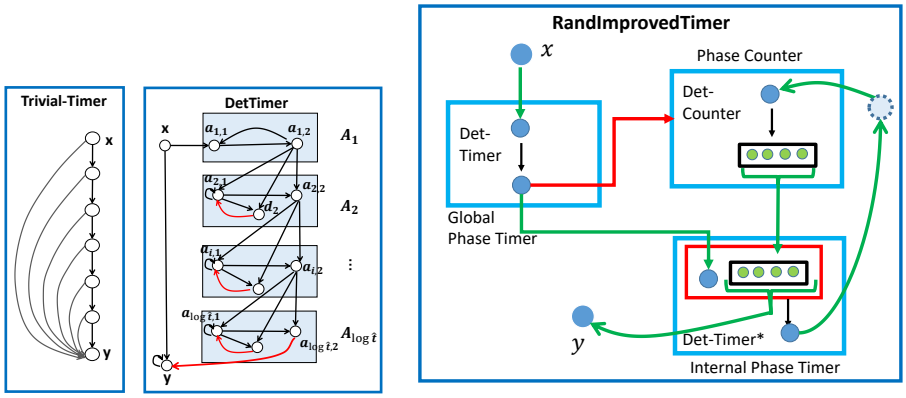


Figure 1: Illustration of timer networks with time parameter $t$. Left: The naïve timer with $\Theta(t)$ neurons. Mid: deterministic timer with $\Theta(\log t)$ neurons. Right: randomized timer with $O(\log\log 1/\delta))$ neurons, using the DetTimer modules with parameter $t' = \log 1/\delta$.

$i^{th}$ neuron on the chain fires, which makes $y$ keep on firing for exactly $t$ rounds until the chain fades out. In this basic solution, the network spends one neuron that counts $+1$ and dies. It is noteworthy that the neurons in our model are very simple, they do not have any memory, and thus cannot keep track of the firing history. They can only base their firing decisions on the firing of their neighbors in the *previous* round.

With such a minimal model of computation, it is therefore intriguing to ask how to beat this linear dependency (of network size) in the time parameter $t$. Can we count to ten using only two (memory-less) neurons? We answer this question in the affirmative, and show that even with just simple deterministic threshold gates, we can measure time up to $t$ rounds using only $O(\log t)$ neurons. It is easy to see that this bound is tight when using deterministic neurons (even when allowing some approximation). The reason is that $o(\log t)$ neurons encode strictly less than $t$ distinct configurations, thus in a sequence of $t$ rounds, there must be a configuration that re-occurs, hence locking the system into a state in which $y$ fires forever.

**Theorem 1** (Deterministic Timers)**.** *For every input time parameter $t \in \mathbb{N}_{>0}$, (1) there exists a deterministic neural timer network $\mathcal{N}$ with $O(\log t)$ deterministic threshold gates, (2) any deterministic neural timer requires $\Omega(\log t)$ neurons.*

This timer can be easily adapted to the related problem of *counting*, where the network should output the number of spikes (by the input $x$) within a time window of $t$ rounds.

**Does Randomness Help in Time Estimation?** Neural computation in general, and neural spike responses in particular, are inherently stochastic [Lin09]. One of our broader scope agenda is to understand the power and limitations of randomness in neural networks. Does neural computation become *easier* or *harder* due to the stochastic behavior of the neurons?

We define a randomized version of the neural timer problem that allows some slackness both in the approximation of the time, as well as allowing a small error probability. For a given error probability $\delta \in (0,1)$, the output $y$ should fire for at least $t$ rounds, and must stop firing after at most $2t$ rounds[3] with probability at least $1-\delta$. It turns out that this randomized variant leads to a considerably improved solution for $\delta = 2^{-O(t)}$:

**Theorem 2** (Upper Bound for Randomized Timers)**.** *For every time parameter $t \in \mathbb{N}_{>0}$, and error probability $\delta \in (0,1)$, there exists a probabilistic neural timer network $\mathcal{N}$ with $O(\min\{\log\log 1/\delta, \log t\})$ deterministic threshold gates plus* additional *random spiking neuron.*

Our starting point is a simple network with $O(\log 1/\delta)$ neurons, each firing independently with probability $1-1/t$. The key observation for improving the size bound into $O(\log\log 1/\delta)$ is to use the *time axis*: we will use a *single* neuron to generate random samples over time,

[3]Taking $2t$ is arbitrary here, and any other constant would work as well.

rather than having *many* random neurons generating these samples in a *single* round. The deterministic neural counter network with time parameter of $O(\log 1/\delta)$ is used as a building block in order to gather the firing statistics of a single spiking neuron. In light of the $\Omega(\log t)$ lower bound for deterministic networks, we get the first separation between deterministic and randomized solutions for error probability $\delta = \omega(1/2^t)$. This shows that randomness can help, but up to a limit: Once the allowed error probability is exponentially small in $t$, the deterministic solution is the best possible. Perhaps surprisingly, we show that this behavior is tight:

**Theorem 3** (Lower Bound for Randomized Timers)**.** *Any SNN network for the neural timer problem with time parameter $t$, and error $\delta \in (0,1)$ must use $\Omega(\min\{\log\log 1/\delta, \log t\})$ neurons.*

**Neural Counters.** Spiking neurons are believed to encode information via their firing rates. This underlies the *rate coding* scheme [Adr26, TM97, GKMH97] in which the spike-count of the neuron in a given span of time is interpreted as a *letter* in a larger alphabet. In a network of memory-less spiking neurons, it is not so clear how to implement this rate dependent behavior. How can a neuron convey a complicated message over time if its neighboring neurons remember only its recent spike? This challenge is formalized by the following neural counter problem: Given an input neuron $x$, a time parameter $t$, and $\Theta(\log t)$ output neurons represented by a vector $\bar{y}$, it is required to design a neural network such that the output vector $\bar{y}$ holds the binary representation of the number of times that $x$ fired in a sequence of $t$ rounds. As we already mentioned this problem is very much related to the neural timer problem and can be solved using $O(\log t)$ neurons. Can we do better?

The problem of maintaining a *counter* using a small amount of space has received a lot of attention in the *dynamic streaming* community. The well-known Morris algorithm [Mor78, Fla85] maintains an approximate counter for $t$ counts using only $\log\log t$ bits. The high-level idea of this algorithm is to increase the counter with probability of $1/2^{C'}$ where $C'$ is the current read of the counter. The counter then holds the exponent of the number of counts. By following ideas of [Fla85], carefully adapted to the neural setting, we show:

**Theorem 4** (Approximate Counting)**.** *For every time parameter $t$, and $\delta \in (0,1)$, there exists a randomized construction of approximate counting network using $O(\log\log t + \log(1/\delta))$ deterministic threshold gates plus an additional* single *random spiking neuron, that computes an $O(1)$ (multiplicative) approximation for the number of input spikes in $t$ rounds with probability $1-\delta$.*

We note that unlike the deterministic construction of timers that could be easily adopted to the problem of neural counting, our optimized randomized timers with $O(\log\log 1/\delta)$ neurons cannot be adopted into an approximate counter network. We therefore solve the latter by adopting Morris algorithm to the neural setting.

**Broader Scope: Lessons From Dynamic Streaming Algorithms.** We believe that approximate counting problem provides just one indication for the potential relation between succinct neural networks and dynamic streaming algorithms. In both settings, the goal is to gather statistics (e.g., over time) using a small amount of space. In the setting of neural network there are additional difficulties that do not show up in the streaming setting. E.g., it is also required to obtain fast *update time*, as illustrated in our solution to the approximate counting problem.

## 3 Neural Synchronizers

The standard model of spiking neural networks assumes that all edges (synapses) in the network have a uniform response latency. That is, the electrical signal is passed from the presynaptic neuron to the postsynaptic neuron within a fixed time unit which we call a *round*. However, in real biological networks, the response latency of synapses can vary considerably depending on the biological properties of the synapse, as well as on the distance between the neighboring neurons. This results in an asynchronous setting in which different edges have distinct response time. We formalize a simple model of spiking neurons in the asynchronous setting, in which the given neural network also specifies a *response latency* function $\ell : A \to \mathbb{R}_{\geq 1}$ that determines the

number of rounds it takes for the signal to propagate over the edge. Inspired by the synchronizers of Awerbuch and Peleg [AP90], and using the above mentioned compressed timer and counter modules, we present a general simulation methodology (a.k.a synchronizers) that takes a network $\mathcal{N}_{\mathsf{sync}}$ that solves the problem in the synchronized setting, and transform it into an "analogous" network $\mathcal{N}_{\mathsf{async}}$ that solves the same problem in the asynchronous setting.

The basic building blocks of this transformation is the neural time component adapted to the asynchronous setting. The cost of the transformation is measured by the overhead in the number of neurons and in the computation time. Using our neural timers leads to a small overhead in the number of neurons.

**Theorem 5** (Synchronizer, Informal)**.** *There exists a synchronizer that given a network $\mathcal{N}_{\mathsf{sync}}$ with $n$ neurons and maximum response latency[4] $L$, constructs a network $\mathcal{N}_{\mathsf{async}}$ that has an "analogous" execution in the asynchronous setting with a total number of $O(n+L\log L)$ neurons and a time overhead of $O(L^3)$.*

We note that although the construction is inspired by the work of Awerbuch and Peleg [AP90], due to the large differences between these models, the precise formulation and implementation of our synchronizers are quite different. The most notable difference between the distributed and neural setting is the issue of memory: in the distributed setting, nodes can aggregate the incoming messages and respond when all required messages have arrived. In strike contrast, our neurons can only respond (by either firing or not firing) to signals arrived in the *previous* round, and all signals from previous rounds cannot be locally stored. For this reason and unlike [AP90], we must assume a bound on the largest edge latency. In particular, we show that the size overhead of the transformed network $\mathcal{N}_{\mathsf{async}}$ must depend, at least logarithmically, on the value of the largest latency $L$.

**Observation 1.** *The size overhead of any synchronization scheme is $\Omega(\log L)$.*

This provably illustrates the difference in the overhead of synchronization between general distributed networks and neural networks. We leave the problem of tightening this lower bound (or upper bound) as an interesting open problem.

**Related Work on Neural Syncronization.** To the best of our knowledge, there are two main previous theoretical work on asynchronous neural networks. Maass [Maa94] considered a quite elaborated model for deterministic neural networks with *arbitrary* response *functions* for the edges, along with latencies that can be chosen by the network designer. Within this generalized framework, he presented a coarse description of a synchronization scheme that consists of various time modules (e.g., initiation and delay modules). Our work complements the scheme of [Maa94] in the simplified SNN model by providing a rigorous implementation and analysis for size and time overhead. Khun et al. [KSPS10] analyzed the synchronous and asynchronous behavior under the stochastic neural network model of DeVille and Peskin [DP08]. Their model and framework is quite different from ours, and does not aim at building synchronizers.

## References


[Adr26] Edgar D Adrian. The impulses produced by sensory nerve endings. *The Journal of physiology*, 61(1):49–72, 1926.

[AP90] Baruch Awerbuch and David Peleg. Network synchronization with polylogarithmic overhead. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 514–522, 1990.

[ATGM14] Melissa J Allman, Sundeep Teki, Timothy D Griffiths, and Warren H Meck. Properties of the internal clock: first-and second-order principles of subjective time. *Annual review of psychology*, 65:743–771, 2014.


[4] I.e., $L$ correspond to the *length* of the longest round.

[CCL19] Chi-Ning Chou, Kai-Min Chung, and Chi-Jen Lu. On the algorithmic power of spiking neural networks. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 26:1–26:20, 2019.

[DP08] RE Lee DeVille and Charles S Peskin. Synchrony and asynchrony in a fully stochastic neural network. *Bulletin of mathematical biology*, 70(6):1608–1633, 2008.

[Fla85] Philippe Flajolet. Approximate counting: A detailed analysis. *BIT*, 25(1):113–134, 1985.

[FSJ+15] Gerald T Finnerty, Michael N Shadlen, Mehrdad Jazayeri, Anna C Nobre, and Dean V Buonomano. Time in cortical circuits. *Journal of Neuroscience*, 35(41):13912–13916, 2015.

[GKMH97] Wulfram Gerstner, Andreas K Kreiter, Henry Markram, and Andreas VM Herz. Neural codes: firing rates and beyond. *Proceedings of the National Academy of Sciences*, 94(24):12740–12741, 1997.

[HP19] Yael Hitron and Merav Parter. Counting to ten with two fingers: Compressed counting with spiking neurons, 2019.

[KSPS10] Fabian Kuhn, Joel Spencer, Konstantinos Panagiotou, and Angelika Steger. Synchrony and asynchrony in neural networks. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms*, pages 949–964. SIAM, 2010.

[Lin09] Benjamin Lindner. Some unsolved problems relating to noise in biological systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(01):P01008, 2009.

[LM18] Nancy Lynch and Cameron Musco. A basic compositional model for spiking neural networks. *arXiv preprint arXiv:1808.03884*, 2018.

[LMP17a] Nancy Lynch, Cameron Musco, and Merav Parter. Computational tradeoffs in biological neural networks: Self-stabilizing winner-take-all networks. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS)*, 2017.

[LMP17b] Nancy Lynch, Cameron Musco, and Merav Parter. Spiking neural networks: An algorithmic perspective. In *5th Workshop on Biological Distributed Algorithms (BDA 2017)*, July 2017.

[LMP17c] Nancy A. Lynch, Cameron Musco, and Merav Parter. Neuro-ram unit with applications to similarity testing and compression in spiking neural networks. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 33:1–33:16, 2017.

[LMPV18] Robert A. Legenstein, Wolfgang Maass, Christos H. Papadimitriou, and Santosh Srinivas Vempala. Long term memory and the densest k-subgraph problem. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 57:1–57:15, 2018.

[Maa94] Wolfgang Maass. Lower bounds for the computational power of networks of spiking neurons. *Electronic Colloquium on Computational Complexity (ECCC)*, 1(19), 1994.

[Maa96] Wolfgang Maass. On the computational power of noisy spiking neurons. In *Advances in Neural Information Processing Systems 8 (NIPS)*, 1996.

[Maa97] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[MHM13] Hugo Merchant, Deborah L Harrington, and Warren H Meck. Neural basis of the perception and estimation of time. *Annual review of neuroscience*, 36:313–336, 2013.

[Mor78] Robert Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, 1978.

[PV19] Christos H. Papadimitriou and Santosh S. Vempala. Random projection in the brain and computation with assemblies of neurons. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 57:1–57:19, 2019.

[TM97] Misha V Tsodyks and Henry Markram. The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proceedings of the national academy of sciences*, 94(2):719–723, 1997.