# P-CRITICAL: a reservoir autoregulation plasticity rule for neuromorphic hardware

Ismael Balafrej, Fabien Alibart, Jean Rouat

# P-CRITICAL: a reservoir autoregulation plasticity rule for neuromorphic hardware

NEUROMORPHIC
Computing and Engineering

OPEN ACCESS

RECEIVED
17 September 2021

REVISED
14 February 2022

ACCEPTED FOR PUBLICATION
7 April 2022

PUBLISHED
28 April 2022



PAPER

# P-CRITICAL: a reservoir autoregulation plasticity rule for neuromorphic hardware

**Ismael Balafrej**[1,*], **Fabien Alibart**[2,3] and **Jean Rouat**[1]

[1] NECOTIS Research Lab, Institut interdisciplinaire d'innovation technologique (3IT), Université de Sherbrooke, Canada
[2] Laboratoire Nanotechnologies et Nanosystèmes, Institut interdisciplinaire d'innovation technologique (3IT), Université de Sherbrooke, Canada
[3] Institut d'électronique, de microélectronique et de nanotechnologie (IEMN), Université de Lille, France
[*] Author to whom any correspondence should be addressed.

**E-mail:** ismael.balafrej@usherbrooke.ca, fabien.alibart@usherbrooke.ca and jean.rouat@usherbrooke.ca

**Keywords:** reservoir computing, plasticity, small-world topology, branching factor, Loihi, neuromorphic computing, criticality

## Abstract

Backpropagation algorithms on recurrent artificial neural networks require an unfolding of accumulated states over time. These states must be kept in memory for an undefined period of time which is task-dependent and costly for edge devices. This paper uses the reservoir computing paradigm where an untrained recurrent pool of neurons is used as a preprocessor for temporally structured inputs and with a limited number of training data samples. These so-called reservoirs usually require either extensive fine-tuning or neuroplasticity. We propose a new local and unsupervised plasticity rule named P-CRITICAL designed for automatic reservoir tuning that translates well to physical and digital neuromorphic processors. The spiking neuronal architecture implementation is simulated on the Loihi research chip from Intel and on a conventional CPU. Comparisons on state-of-the-art machine learning datasets are given. Improved performance on visual and auditory tasks are observed. There is no need to *a priori* tune the reservoir when switching between tasks, making this approach suitable for physical implementations. Furthermore, such plastic behaviour of the reservoir is a key to end-to-end energy-efficient neuromorphic-based machine learning on edge devices.

## 1. Introduction

Reservoir computing (RC), brought by both Jaeger [1] and Maass *et al* [2] respectively as the echo state network (ESN) and the liquid state machine (LSM), have been successful on various complex tasks. Initially formulated from computational neurosciences and machine learning approaches, the concept of reservoir has attracted a large interest from neuromorphic engineers, and in particular from the emerging nanodevices community [3]. First, the apparent simple three-layer structure of the reservoir (input layer, reservoir and output layer) is very attractive from a hardware implementation perspective. Indeed, only the last layer is trained with supervised methods, making this architecture efficient in training time while reducing the complexity of weight learning to a single perceptron problem. Secondly, the reservoir layer in itself is used to project input signals onto a large dimensional space with some fading memory effect, which should ensure spatiotemporal separability of data. This second aspect also appears very interesting for emerging hardware implementation since nanodevices present most of the time complex dynamics that could be used for temporal reservoir engineering and are prone to high-density integration, thus allowing for the implementation of large dimensionality physical space.

### 1.1. Physical reservoir limitations

The development of physical reservoirs (i.e., based on physical hardware substrates) are now facing drawbacks such as scalability of the reservoir to complex problems (i.e., requiring to increase drastically the dimensionality) and ability of reservoirs to be adapted to various problems (i.e., finding a generic hardware implementation applicable to various tasks). Moreover, novel reservoir implementations with emerging nanodevices, which rely on intrinsic device physics for reproducing neuronal and synaptic behaviour, are limited in the

tuning capability of the various network parameters. Solving these major issues requires to bridge in more detail the fundamentals of RC with the constraints and opportunities bear by the hardware implementation and physics of nanodevices. Additionally, implementations based on emerging technologies often face undesired behaviour, such as device-to-device variability. Ideally, a reservoir computer should be either immune or enhanced by these variations.

### 1.2. Software-based solutions

Similarly, in neurocomputing, the major cost of the reservoir architecture is that the reservoir's connectivity, or topology, alongside its weights must be chosen carefully during initialization. Failure to do so results in a huge variability in the performance of the network. The topology choice and the various parameters of the network constitute a search space and therefore, optimization of these parameters is often mandatory to achieve an adequate performance.

To maximize the performance obtainable by a reservoir, this search space must be navigated using a heuristic. There exist three categories of heuristics to evaluate the effectiveness of a reservoir. The first and most straightforward category is simply post training accuracy. The second type of heuristic consists of *a posteriori* methods [4–7] that require some simulation time without the need for training the network on a dataset, e.g., computing the separation distance of closely inputs using the Lyapunov exponent [4]. Lastly, the *a priori* category encompasses methods that can create reservoirs without the need of simulation [8, 9] by algorithmically creating the reservoir from a mathematical definition. The reservoir's parameters can then be searched [10–14] and compared using one of these heuristic. More resources spent on optimization typically yield better reservoir performance. Yet, this idea conflicts with the premise of having fast trainable networks, both for software and hardware implementations of reservoir networks.

### 1.3. P-CRITICAL for hardware-aware reservoir computing

We propose in this work a hardware-friendly solution that helps to define the three main basic ingredients of a reservoir: (i) topology of the reservoir (number of neurons and connectivity map), (ii) neurons parameters and (iii) synaptic weight definition.

First, we present a novel topology initialization scheme that belongs in the *a priori* category of heuristics. Our topology is optimized in the hyperparameter space of small-world graphs, with respect to the average eigenvalues spectrum of connectomes [15]. This new technique offers a valuable performance boost at a low computational cost while remaining task independent. The fixed task-independent network topology can be embedded into any hardware device, in lieu of complex reconfigurable topology using addressable neuron/synapse as is done in digital or mixed digital/analog neuromorphic devices.

Second, we present P-CRITICAL, a plasticity rule for neuromorphic applications that removes most of the need for reservoir parameter tuning, such as the number of neurons, their parameters, and a bad set of initial synaptic weights. This simple plasticity rule is based on the concept of criticality, as used in [16–18], with a focus on neuromorphic hardware. Unlike hyperparameter optimization using heuristics, P-CRITICAL tries to continuously adapt the weights inside a reservoir such that the performance of the reservoir becomes less dependent on the initial values of the parameters or the behaviours of the underlying devices. Moreover, P-CRITICAL uses the same equations for neurons and for adaptation, which can simplify future hardware implementations. We benchmarked P-CRITICAL on well-known machine learning classification tasks and observed increased performance when compared to LSMs with fixed random weights.

We emphasize the importance of the many hardware constraints on the design and choices behind our approach. In this work, some of these constraints are set by the design choices of a modern neuromorphic processor, namely Intel's Loihi research test chip [19]. The major limitation is the inability of drastically changing the neuron model inside the neurocores, which we circumvent by combining neurons in a novel way. Moreover, resources are better distributed on Loihi when the network is sparse and not fully connected, for both energy and time efficiency. While these limitations exist on Loihi, they are typical of most neuromorphic hardware architectures where neuronal and synaptic memory is limited or with fixed analog or physical neurons and synapses. Additionally, our approach further showcases the applicability of reservoirs for other physical devices as we constraints ourselves with realistic hardware limitations.

We contribute to the research effort, which consists in editing the weights of a reservoir network using synaptic plasticity, and provide a model that is more hardware-friendly. In summary, the key contributions of this paper are:

(a) A new scheme to initialize the network connectivity (decide which neurons connect to which neurons) of a reservoir that is task-independent and sparse.

(b) Followed by P-CRITICAL, a plasticity rule which dynamically modifies the values of the connection weights to maintain the reservoir at an operating point which is optimal for the targeted task while adapting to the fact that the characteristics of the neurons (RC, threshold, etc) are unknown and can be decided by external factors (i.e., from a target hardware device).

We present results from a CPU/GPU implementation using PyTorch [20] and with Intel's Loihi.

## 2. Related works

The ESN and the LSM are mostly distinguished by their respective neuron models. ESNs use perceptron-like neurons with non-linear activation functions such as sigmoid or tanh [1], while LSMs use biologically inspired spiking neurons, often leaky-integrate-and-fire (LIF) or integrate-and-fire (IF) neurons [2]. These architectures comprise three layers: an input layer $W_{\mathrm{I}}$, a reservoir layer $W_{\mathrm{R}}$ and an output layer $W_{\mathrm{O}}$. Both $W_{\mathrm{I}}$ and $W_{\mathrm{R}}$ are not trained with supervised methods such as backpropagation, although the many parameters of these layers and even in the neuron model is often optimized with one of the aforementioned heuristics. Many LSM users implement a non-spiking readout layer $W_{\mathrm{O}}$ using machine learning methods [13, 21–24] or even $n$-layers formal neural networks [25]. In this work, we use a single formal layer with a softmax activation function as the output layer as we focus on the reservoir component of the LSM.

### 2.1. Optimization of the reservoir

While $W_{\mathrm{R}}$ is not trained, many authors include unsupervised neuroplasticity rules [22, 23, 26–28] as a way to either keep biological realism or to provide a higher computational performance. Similarly, several studies looked at the initialization of $W_{\mathrm{R}}$ in combination with various topologies such as small-world [27, 29–31] and scale-free networks [31, 32]. Furthermore, by using an orthogonal matrix for $W_{\mathrm{R}}$ in the context of ESNs, Hajnal and Lrincz [9] reduce the dimension of the reservoir hyperparameters search space, yielding an increased probability of finding a valid reservoir configuration. Our work uses state of the art topological and plastic reservoir enhancements to increase the computational power of our network with a focus on efficiency and hardware implementation.

#### *2.1.1. Echo state property*

Reservoirs are typically prone to two major problems as $W_{\mathrm{R}}$ is recurrent: the explosion or the fading of the internal states during recursion. In both cases, the information will be lost, albeit the explosion problem is worst as it can create uncontrollable noise (similar to chaotic behaviour [33]). That explosion can be solved for the ESN if $W_{\mathrm{R}}$ is diagonally Schur stable; this is known as the echo state property (ESP) [8]. As explained in Yildiz *et al* [8], a simple recipe for generating $W_{\mathrm{R}}$ that satisfies the ESP is to create a positive random matrix $W$, scale it down using the spectral radius $\rho$ of $W$, and convert any ratio of connections into inhibitory connections by changing the sign of the weights $w^{ij}$, considering $W = (w_{ij})$. This is, of course, equivalent to creating a random matrix and scaling it with $\rho(|w_{ij}|)$:

$$W_{\mathrm{R}} := \frac{W_{\mathrm{R}}}{\rho(|w_{ij}^{\mathrm{R}}|)} \tag{1}$$

Although equivalents of equation (1) are widely used in the ESN literature, these methods do not translate to LSMs [25, 34] as they are insufficient as the sole *a priori* heuristic for reservoir performance [5, 34, 35]. We will further demonstrate the inability of the ESP to tune LSMs in section 4.2.1 in comparison to our plastic approach.

#### *2.1.2. Dynamical optimization*

Other metrics have been explored to quantify performance based on *a posteriori* dynamic analysis such as the Lyapunov exponent [4], the average state entropy [5], the dynamic profile of the Jacobian of $W_{\mathrm{R}}$ [6] or the approximate state space model [7]. These methods allow for a more guided search on a reservoir's parameters. The most common approach is to use an evolutionary based search algorithm [10–13]. Similarly, Tian *et al* [14] showed improvements with a neural architecture search designed for LSM. Unfortunately, iterating over some extensive search space goes against the rationale of low-energy NC since simulation-based hyperparameter search is an energy-intensive operation. This is especially true when we consider that most optimization schemes are task-dependent. Indeed, the scaling of the weights inside the reservoir must somewhat match the amplitude of the input for adequate memory fading in the context of LSMs. If $W_{\mathrm{R}}$ is fixed and tuned to account for high frequency spiking inputs, the reservoir will not respond correctly for sparser input activity and the information will die out quickly. For the ESN, one typically normalizes the input to maximize what is inserted

in the reservoir. Unlike the ESN, spike trains with binary spikes have a fixed amplitude and normalizing the frequency is not possible for real-time systems without prior knowledge of the task. A possible solution is a plastic reservoir, since it can minimize the necessity of hyperparameter optimization for different input frequencies.

The reservoir should adapt its own parameters without requiring information about its output-layer performance nor an iterative search space. Hebbian STDP-like plasticity rules do not help in this regard as they aim to increase correlation of spike-timing, which can still happen during chaotic behaviour. There are few proposed models that help by tuning what is called the branching factor $\sigma$ of $W_R$. For $n_{pre}$, the number of pre-synaptic spikes for a neuron and $n_{post}$, the number of post-synaptic spikes for the same neuron:

$$\sigma = \frac{n_{pre}}{n_{post}} \tag{2}$$

We define $\bar{\sigma}$ as the mean branching factor of the neurons in $W_R$. There are three defined regimes for $\bar{\sigma}$ known as subcritical when $\bar{\sigma} < 1$, critical when $\bar{\sigma} = 1$ and supercritical when $\bar{\sigma} > 1$ [36]. For the latter, the aforementioned problem of unconstrained activity rises up. These regimes are similar to what can be expressed by the ESP [8] or the Lyapunov exponent [4]. A $\bar{\sigma}$ slightly below 1.0 can offer sufficient fading memory properties for a reservoir while being close to reproducing *in vivo* spike avalanches [37]. As such, a few models have been proposed for locally tuning $\sigma$ in a reservoir of spiking neurons [16–18]. Kello and Mayberry's [16] algorithm is memory-less in the sense that the branching factor is only considered in consecutive time steps (at $t$ and $t+1$). Stepp *et al* [18] tunes biologically inspired STDP rules to exhibit criticality behaviours. It is believed that branching factor tuning algorithms may increase computational power by bringing a network to the edge-of-chaos [16–18, 36]. Regardless, the reservoir dynamics must be adapted to $\bar{\sigma} \approx 1 - \epsilon$—or slightly subcritical—in order to maintain readable states with adequate memory decay.

To sum up, it remains computationally expensive to adjust the hyperparameters inside the reservoir, and optimization may be restrictive or even not possible depending on the target hardware. Most search methods are limited and reservoirs often require task-specific optimizations; some type of self-adaptation is therefore necessary. As illustrated in this work, branching-factor algorithms could be the answer to that problem.

### 2.2. Eigenvalues spectrum

Ideally, the reservoir's connectivity—or topology—should be as sparse as possible to reduce the computational cost. Choosing a topology that can work with any given task is still an open problem.

Many authors empirically verified that biologically inspired topologies perform better than their completely random counterparts. Manevitz and Hazan [31] showed that scale-free networks are more robust to noisy neuron models in the context of RC. Similarly, [27, 29, 30, 32, 38] presented improvements on various tasks with either small-world or scale-free topologies. Wijesinghe *et al* [39] introduced the concept of liquid ensembles, which can be thought as a small-world topology with disjoint inner networks. This idea allowed them to speedup computation while still observing the increased performance of small-worlds.

Recent studies [15, 40] looked at the patterns in the eigenvalues spectrum, or rather in the probabilistic distribution of the eigenvalues of the normalized Laplacian of unweighted and undirected biological connectomes. In mammalian brains, this distribution seems to be consistent intraspecies, and bifurcations from said distribution are linked to improperly developed brains [40]. In summary, years of evolution seem to lead to a fairly consistent topology. We propose to use the available data for adjusting the hyperparameters of a topology, thus using a computationally inexpensive and convenient way of choosing the topology that does not rely on simulated dynamics. Moreover, a task-independent topology can translate well to hardware implementations as the routing can be embedded instead of using more expensive dynamic routing options or fully connected crossbar-like architectures.

## 3. Proposed approach

First, we present a topology optimization scheme in section 3.1. We then present P-CRITICAL, a synaptic plasticity rule in section 3.3.

### 3.1. Eigenvalues spectrum inspired topology

Topology is an important aspect of a reservoir. It is known that some topologies are better than others for various tasks [27, 29–32, 38, 39]. But even within a choice of topology, some hyperparameters have to be searched. We therefore propose a simple method of choosing an adequate set of topology parameters that is task independent and biologically inspired. This aspect is particularly important when considering the added complexity of topological optimization in physical implementations. Our topology is a small-world with a distance-based connectivity. We first create a three-dimensional Cartesian grid lattice of $\{x, y, z\}$ positioned neurons. Each neuron is equally separated by distance $s$ with its neighbours. Furthermore, every group of $j$

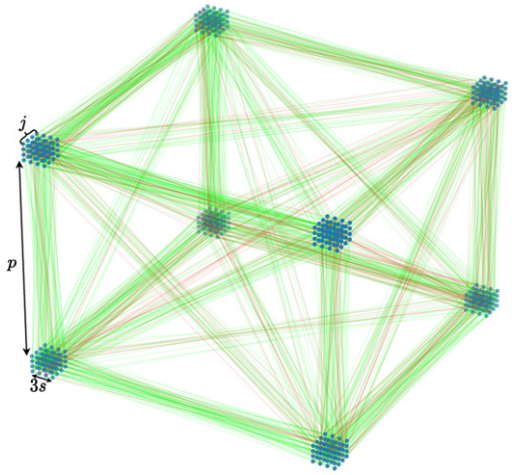


**Figure 1.** Tridimensional representation of the small-world topology with excitatory connections in green and inhibitory connections in red. 20% of the neurons are chosen to be inhibitory with all their outgoing connections being inhibitory. Once the neurons are positioned, all pairs of neurons can create a connection with a Euclidean distance-based probability, except with themselves. The mini-reservoirs of size $j^3 = 64$ neurons are separated by distance $p$. Each neuron is separated by distance $s$ with its neighbours.

neurons in all orthogonal directions are separated by distance $p$. The tridimensional distances and connectivity ratios are illustrated in figure 1. This results in $\frac{n}{j^3}$ mini-reservoirs of size $j^3$ neurons, assuming $W_R \in R^{n,n}$ and $n \equiv 0 \bmod j^3$. In order to achieve non-cubic mini-reservoirs, $j$ can also be represented as a vector $\mathbf{j} \in R^3$ where $j_k$ is the number of neurons $j$ used for the $k$th dimension.

The undirected adjacency matrix $A$, with $a_{ij} = a_{ji} = 1$ if neuron $i$ is connected to neuron $j$ and $a_{ij} = 0$ otherwise, can be generated by randomly connecting neurons based on their Euclidean distance $D$, as done in [2]. The probability $P$ of connection between neuron $a$ and neuron $b$ is given by:

$$P = C \cdot e^{-\frac{D(a,b)}{\lambda}},$$

where $C$ is the maximum connection probability and $\lambda$ is a control parameter which we refer to as an Euclidean distance divisor.

By looking at the eigenvalues spectrum of the macaque as presented in Lange *et al* [15], we manually tuned the parameters of a small-world topology to minimize the Kullback–Leibler divergence with the topology's eigen spectrum. We obtained a good approximation with values $s = 40$, $p = 1460$, $C = 0.11$ and $\lambda = 635$. From a topological perspective, our reservoir will yield a more similar macroscopic structure to what is seen in the brain. Therefore, this method uses millions of years of evolution in connectomes to enhance our reservoir's topology.

### 3.2. Input and output layers

The input matrix $W_I$ is simply a permutation matrix[4] multiplied by a constant weight $w_{ij}^I$. The value is chosen such that every pre-synaptic spike in the input layer causes a post-synaptic spike in the reservoir. By doing so, we remove any need to consider the input weights distribution. We therefore consider the input neurons to be within the reservoir and plastic connections can act immediately. 1 to $n$ connections can be created by changing $n - 1$ zeros into ones in each row of $W_I$ before the permutation operation[5].

For classification tasks, we bin the reservoir spikes from all the neurons into $R_{output}$ by counting the spikes in multiple fixed slices of time and train a weight matrix $W_O$. $W_O$ is trained with backpropagation using PyTorch's cross entropy loss function. We use a batch normalization layer $b_n$ [41] in between the reservoir output and the single layer classifier. Accuracies are calculated from labels $y$ with $\sum(y = \text{argmax}(b_n(R_{output})W_O))$.

[4] A permutation matrix can be created by randomly permuting the rows of the identity matrix. This is equivalent to connecting each input neuron to a unique reservoir neuron (one-for-one).

[5] Only a $10k$ samples subset of N-MNIST was used ($+10k$ in testing).

### 3.3. Branching-factor adaptation of reservoirs with weight updates

#### 3.3.1. Prior work

Brodeur and Rouat [17] introduced a neuron ensemble with a plasticity rule called CRITICAL that adapts the synaptic weights in a time-dependent manner. The locality of this plasticity rule incorporates new recorded states from the pre and post-synaptic neurons, referred to as pre and post-synaptic contributions or $c_{\text{out}}$ and $c_{\text{in}}$ respectively. These two extra variables are stored inside each neuron and are updated on pre and post-synaptic spikes. Let $n_i$ be a pre-synaptic neuron spiking at time $t_{n_i}$ and $n_j$ be a post-synaptic neuron spiking at time $t_{n_j}$, then the CRITICAL plasticity is:

$$\frac{\mathrm{d}c_{\text{in}}}{\mathrm{d}t} = w_{ij} \cdot \delta(t - t_{n_i}) \tag{3}$$

$$\frac{\mathrm{d}c_{\text{out}}}{\mathrm{d}t} = \frac{w_{ij}}{c_{\text{in}}}\delta(t - t_{n_j})\mathrm{e}^{\frac{-\Delta t_{ij}}{\tau}} \tag{4}$$

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \mathrm{lr}(\sigma - c_{\text{out}})\delta(t - t_{n_i}), \tag{5}$$

where lr is a learning rate and $\sigma$ is the target branching factor. $\delta$ represents the Dirac delta function with $\delta(t = 0) = 1$ otherwise $\delta(t \neq 0) = 0$. In equation (4), $\Delta t_{ij} = |t_{n_j} - t_{n_i}|$ contributes to the weight evolution with a Hebbian STDP-like window by taking into account the relative timing between the pre and post neurons. $c_{\text{in}}$ and $c_{\text{out}}$ are local variables to the neurons, but shared in between the synaptic connections of these neurons. Finally, $c_{\text{in}}$ and $c_{\text{out}}$ are reset during post and pre-synaptic spikes respectively.

Unfortunately, neuromorphic chips, such as Loihi, typically work with a fixed neuron model since it is embedded in the circuitry. Adding new states in the neuron model is therefore not possible.

#### 3.3.2. Regulation neurons

We were able to translate the intended behaviour of CRITICAL by recreating the adaptation with regulation neurons or $n'_i$. Each regulation neuron $n'_i$ is associated with a neuron $n_i$ and integrates its post-synaptic activity using transposed connections from $W_{\text{R}}^{\text{T}}$, i.e., projecting $n_j$ to $n'_i$. When the branching factor of neuron $n_i$ is above the targeted branching factor, $n'_i$ fires, causing a depreciation of all synaptic connections with $n_i$ as the presynaptic neuron. This design choice comes at the cost of doubling the number of neurons inside the reservoir. Furthermore, the nature of the unsigned and unweighted spike communication between the regulation neurons $n'_i$ and the synaptic connections of $n_i$ on Loihi forces a single direction in the branching factor estimation i.e. a synapse can receive information that the branching factor is too high or too low, but not both. An overview of this concept can be visualized in figure 2 and equation (6). We name this model P-CRITICAL, since every reservoir neuron now comes in pair with its regulation neuron. This regulation neuron concept also simplifies future physical or hardware implementations of reservoirs, as the neuron block can be reused as-is.

$$\frac{\mathrm{d}w_{ij}}{\mathrm{d}t} = \beta - \alpha\delta(t - t_{n'_i})\mathrm{e}^{\frac{-\Delta t_{ij}}{\tau}} \tag{6}$$

$\alpha$ and $\beta$ are two learning rates for depression and potentiation. The neuron $n'_i$ estimates the output contribution of neuron $n_i$, similar to $c_{\text{out}}$ in the critical rule. When the regulation neuron $n'_i$ spikes, the outgoing connections of $n_i$ are reduced by a ratio of $\alpha$. $w$ is constantly increasing by a small factor $\beta$ to provide a way for the reservoir to come back from sub-criticality if ever the depression caused by $\alpha$ is too strong, e.g. during a change in input spike frequency. Consequently, it is important to keep the relation $\alpha > \beta > 0$. Additionally, a decaying exponential is added to enforce causality between pre and post-synaptic spikes and weight changes, similar to short-term facilitation rules with constant $\tau$ in Hebbian-like plasticity or in the CRITICAL equation (4). The exponential can be approximated using a numerically decaying synaptic trace.

#### 3.3.3. Validation of the model

The P-CRITICAL concept is first simulated using the PyTorch [20] python library. The reservoir is initialized with a $W_{\text{R}}$ where the connectivity is decided with a distanced-based small-world topology, similar to [17]. We then connect the reservoir to the regulation reservoir using weight tensor $W_{\text{R}}^{\text{T}}$—the transpose of $W_{\text{R}}$. By doing so, every post-synaptic neuron of $n_i$ is projecting to $n'_i$. Neurons are chosen to be inhibitory or excitatory at initialization, such that all their outgoing weights are either positive or negative. Inhibitory connections do not have the same definition of branching factor, since they cannot create post-synaptic spikes. Therefore, inhibitory connections are not learned. For all simulated reservoirs, 20% of the neurons are randomly chosen inhibitory. For synapses coming from excitatory neurons, the initial weights are sampled from a uniform random distribution in range $[0.2, 0.5[$ and $[-0.3, -0.1[$ for synapses coming from inhibitory neurons.

Regulation neurons $n'_i$ have the same LIF model as neurons $n_i$. The voltage threshold $v\text{th}$ of the regulation neuron can be decreased or increased further to target a different branching factor for the reservoir. Finally, the amplitude of all the weights are clipped between zero and one after plasticity $W_{\text{R}} = \text{sign}(W_{\text{R}}) \circ \text{clip}(|W_{\text{R}}|, 0, 1)$ with $\circ$ as the Hadamard product.

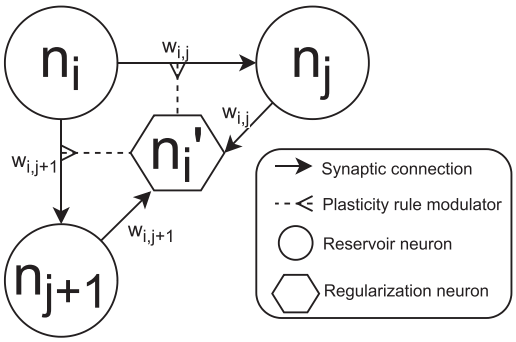


**Figure 2.** In this new adaptation rule, every neuron $n_i$ projecting to neurons $n_j$ and $n_{j+1}$ is associated with its regulation neuron $n_i'$. $n_i'$ approximates the branching factor of $n_i$ and generates spikes to decrease the weights of all outgoing connections of $n_i$ when super-criticality is reached.

By design, we can translate P-CRITICAL easily on Intel's Loihi using their on-chip local learning rules. All weights are scaled from PyTorch's 23 bits mantissas (32 bits floating point implementation) to Loihi's 8 bits (+1 sign bit). As such, we converted the $\pm\,[0, 1\,[$ possible weight range to $\pm\,[0, 256\,[$. The only remaining constraint is that $\alpha$ and $\beta$ must be chosen to minimally affect the least significant bit of all weights while maintaining $\alpha > \beta$. This constraint adds a slight noise in the convergence of the weights for Loihi. The significance of this noise is discussed further in section 4.1.

## 4. Experiments and results

We first tested our method empirically to validate the behaviour of the plasticity rule. We then tested our method against two well-known datasets of the machine learning and spiking neural network community: N-MNIST [42] and N-TIDIGITS [43]. Both of these datasets were created using event-based sensors from previously recorded data. N-MNIST comes from the saccadic presentation of the well-known handwritten digit recognition dataset MNIST to the event-based camera *ATIS sensor* [44]. N-TIDIGITS was recorded from TIDIGITS, an audio representation of spoken digits, using the spiking silicon cochlea sensor *CochleaAMS1b* [43, 45]. Successful training on these datasets could be a significant step to end to end training of low energy event-based hardware. As mentioned, the main goal of P-CRITICAL is to tune a reservoir to the input spike train representation as to offer stability. Many publications present optimized reservoir parameters for the task at hand. We demonstrate that P-CRITICAL can compensate for untuned sets of initial parameters and to some extent an initialization that is not suited for a specific task. This reduces the total training time, as the hyperparameters of the network do not require fine-tuning. Fixed initial parameters are also common in physical neuron implementations, as the neurons may be difficult to parameterize and have some untunable variability. All parameters for the various experiments are attached in appendix A and the code is available online at https://github.com/NECOTIS/PCRITICAL.

### 4.1. Validity of the model

To test the behaviour of the P-CRITICAL model, we connected 170 input neurons with a Poisson-distributed spiking activity to a reservoir of 512 neurons. The input neurons are connected in a one-to-one fashion to the reservoir. We vary the random input frequency from 10 to 50 Hertz. We aim for the reservoir to have a mean branching factor $\bar{\sigma} = 1$. The small-world topology constant $j = 4$ (figure 1). Results are shown in figure 3. As expected, the weights of the reservoir converge according to the input frequency to maintain a steady activity in the reservoir. For higher-frequency inputs, the average weight should be smaller to maintain a constant activity and bigger for a smaller input frequency. Similarly, we re-created the experiment on the Loihi chip and, as expected, we obtained similar results with the P-CRITICAL rule.

We compared the real branching factor with the theoretical and targeted branching factor similarly as Stepp *et al* [18]. We first subtract the input spike train, mapped to the reservoir's dimension by $W_{\text{I}}$, from the reservoir's spike train. This way, we ensure that the branching factor computation methods only consider the self-induced activity within the reservoir. We compute a local branching factor estimation where every neuron's post-synaptic activity is summed and divided by its pre-synaptic activity in terms of spike count. We then average this value for every excitatory neurons in the reservoir. This topology-aware method will overshoot slightly the global branching factor as post-synaptic spikes can be counted multiple times by pre-synaptic neurons. We then estimate the global branching factor with the total number of spikes at $t + 1$ divided by the

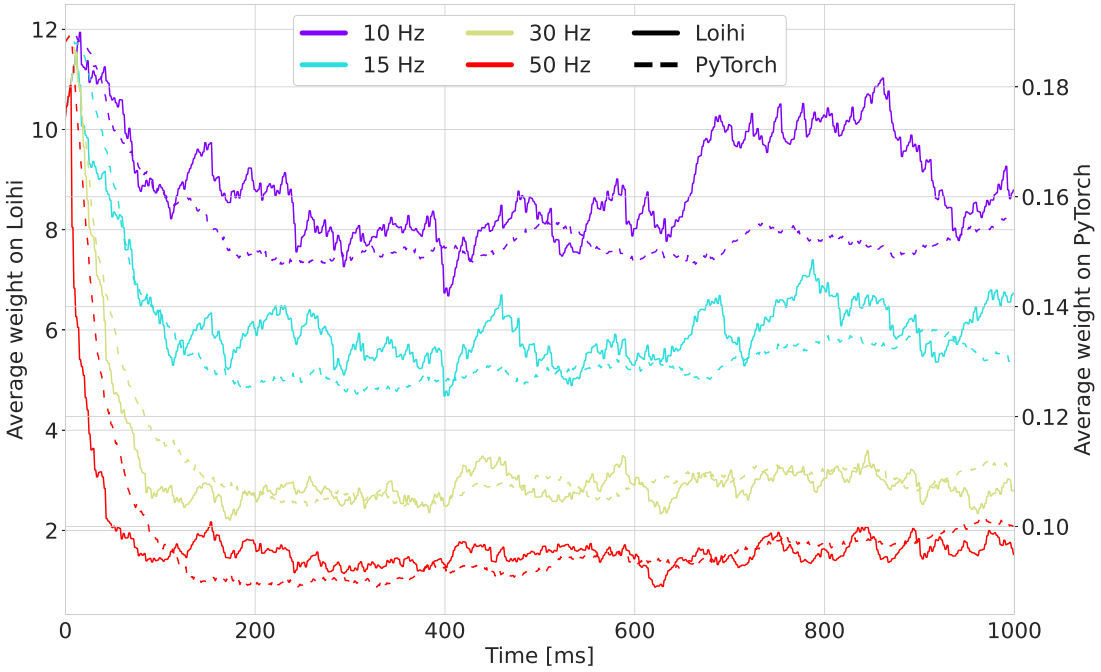


**Figure 3.** Autoregulation of the weights using the P-CRITICAL plasticity rule with random input and a target branching factor of 1. The average weight is shown as a function of time. There are 170 neurons in the 1 s input spike train with a one-to-one connectivity to the 512 reservoir neurons. The input spike train was sampled from a Poisson distribution of labelled frequencies. It is observed that P-CRITICAL is regulating the weights in function of the spiking input frequency. Results are presented for both Loihi and PyTorch implementations.

total number of spikes at time $t$ for excitatory neurons. For these tests, we use 5 s of continuous activity randomly sampled from the N-TIDIGITS dataset. After a small adaptation period, both methods revealed a fairly consistent branching factor of 1 with the P-CRITICAL plasticity rule. Spike activity and branching factor estimations are given in figure 4. The branching factor estimations are filtered for visualization purposes with:

$$\mathrm{BF}(t) = \mathrm{BF}_{\mathrm{measured}}(t) * \mathcal{N}(\sigma = 0.7),$$

where $*$ is the convolution operation between the measured branching factor (BF) and a 1D Gaussian kernel $\mathcal{N}(\sigma = 0.7)$.

Finally, we also compare the time-binned spike counts from the self-induced activity in a Poincaré plot in figure 5. We once again used 5 s of activity from the N-TIDIGITS datasets where features were randomly connected to two reservoir neurons each. We removed the first 2.5 s of the spike train to be sure that the reservoir had converged to the target branching factor of one. We then compute the spike counts using 5 ms bins and plot these counts for consecutive time periods. We compare this with a model of slope one, which represents a $\bar{\sigma} = 1$. We observe that the P-CRITICAL enabled reservoir can adequately maintain a branching factor of one.

### 4.2. Real-world tasks

We then compare randomly initialized reservoirs with P-CRITICAL on N-TIDIGITS and N-MNIST. All experiments are averaged over 5 executions using different random seeds and the standard deviation is presented. We also use identical LIF parameters (leak constants, membrane potential thresholds) for both experiments as we would expect in a generic reservoir-based NC chip, even though they come from different sensory representations. The learning rates of P-CRITICAL, the initial random distribution of the weights, the time-binning constants and the network topology are also fixed. All parameters are given in appendix A.

#### *4.2.1. Speaker-independent audio digit classification*

For the N-TIDIGITS classification task, we used a 512 neurons reservoir with small-world topology and constant $j$ set to 4. For comparison, we run the same sets of experiments with no plasticity and no tuning of the initial parameters and we also use the spectral radius $\rho$ normalization from equation (1). We run our model for 10 epochs, and we use a batch size of 32 samples when training the output layer. As mentioned, the output layer is a single feed-forward matrix $W_{\mathrm{O}}$ with softmax activation function. Only the single digit samples of the dataset were used for training. At each epoch, the data passes through the untrained input and reservoir

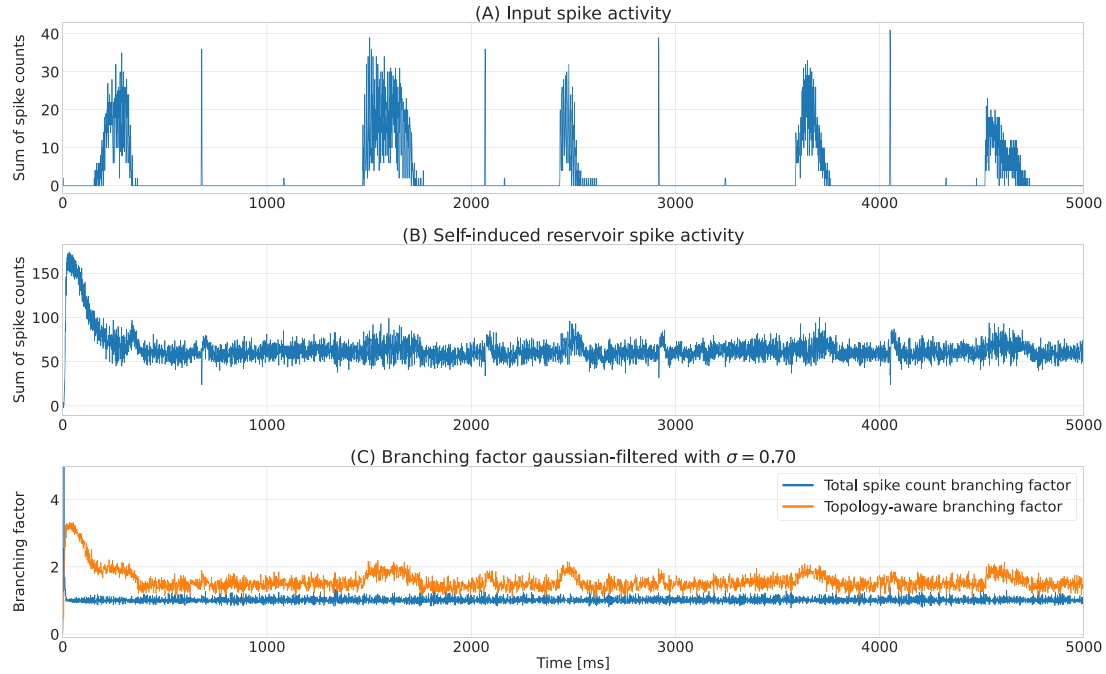


**Figure 4.** Spike activity for 5 s of randomly chosen samples from the N-TIDIGITS dataset. Plot (A) shows the sum of all spike counts after the input layer, as received by the 512 neurons reservoir. Plot (B) shows the sum of spike counts for the reservoir neurons with the input spike counts subtracted, therefore showing the self-induced activity inside the reservoir. Plot (C) shows the branching factor, estimated over the self-induced reservoir activity with two different methods as done in [18]: the total spike count and a topology-aware method. This branching factor estimation is Gaussian filtered over time with constant $\sigma = 0.7$. The total spike count method consists of dividing the total spike count at $t + 1$ by the total spike count at $t$. The topology-aware method works similarly, but spike counts are computed for every neuron and the resulting branching factors are averaged over all neurons in the reservoir. Both estimation methods are presented for excitatory neurons only.

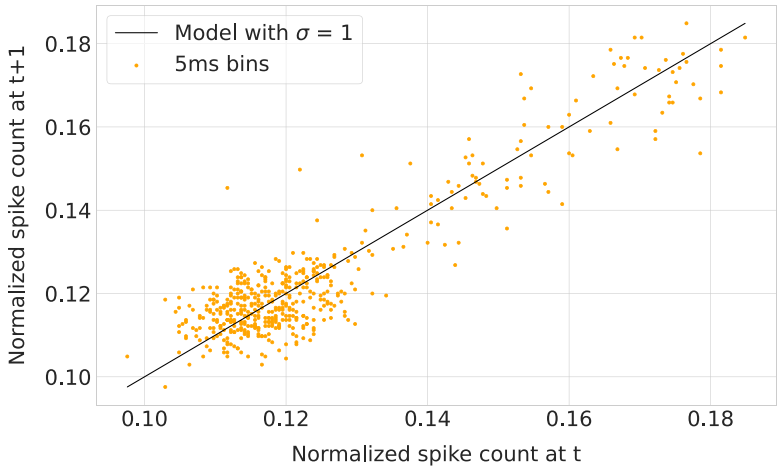


**Figure 5.** Poincaré visualization of the spike count time-binned with $T_{\text{bins}} = 5$ milliseconds. A slope of one was added as a comparison for a model with $\bar{\sigma} = 1$. The spikes come from the self-induced reservoir activity after presentation of the N-TIDIGITS dataset. We first tuned the reservoir over 2500 milliseconds of continuous input. We then simulate the model for another 2500 milliseconds, which resulted in the presented spike counts. We used a small-world reservoir of 512 neurons with constant $j = 4$.

layers. After a batch of data has passed through, we use the Adam [46] optimizer with a learning of $10^{-3}$ to learn the weights of the output layer. As expected, reservoirs that were tuned using spectral radius normalization outperformed random reservoirs. We observe, however, an increased accuracy on the test set with all experiments where P-CRITICAL plasticity was enabled. We obtained with P-CRITICAL an average accuracy of $71.26 \pm 0.92\%$ (figure 6) with the five different random seeds after 10 iterations of training each. Using the optimized eigenvalues spectrum offered all reservoirs running on the CPU with or without P-CRITICAL plasticity a mean accuracy boost of 16.77% on N-TIDIGITS at no task-specific optimization cost. When executing the exact same reservoir experiment on the Loihi research chip, we observed a 64.1% accuracy when running for 20 epochs and using a weight decay of $10^{-2}$.

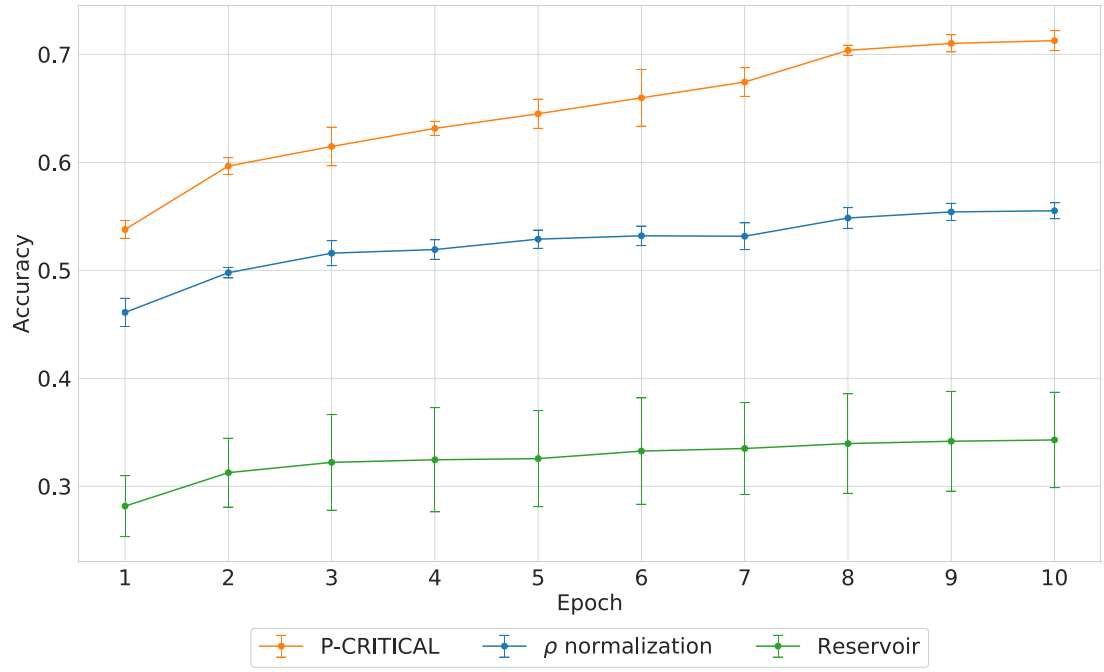


**Figure 6.** Test accuracy as a function of the number of epochs for the N-TIDIGITS classification task. No reservoir hyperparameters were tuned for this task. Classification results are sampled five times for each method with different initial random parameters of the same distributions. All initialization constants are presented in appendix A. The spectral radius $\rho$ normalization of the weights has an expected improved accuracy versus a completely random initialization. However, P-CRITICAL enabled reservoirs surpassed all other methods.

We compared the same reservoir topology but with the original CRITICAL plasticity rule [17] running on the Brian2 simulator [47] and obtained an accuracy of $63.48 \pm 0.86\%$ on the task. We note, however, that the neuron model in the CRITICAL reservoir has an optional adaptative threshold that did not affect performance. This adaptative threshold is modeled as a local threshold to each neuron, with an incremental value of 0.1 at every spike and an exponential decay back to its rest value of 1.0.

This work is one of the first LSM-based reported accuracy for the N-TIDIGITS dataset. In comparison, [48] obtained an accuracy of 87.65% using a deep non-spiking CNN, while [43] obtained 86.4% for a similar network and 82.82% using a non-spiking GRU RNN. All presented spiking methods uses a time-binning readout to convert the spikes from each audio sample into real-valued vectors for classification.

*4.2.2. Handwritten digit classification*

For N-MNIST, we use a 8640 neurons reservoir (with $\mathbf{j} = \begin{bmatrix}4 & 4 & 3\end{bmatrix}$) with PyTorch. Once again, we make no assumption on the difficulty of the task when choosing the number of reservoir neurons, as we focus solely on increasing the dimensionality of the problem. Specifically for N-MNIST, a lot of background neurons are inactive during the experiment and the reservoir size could have been reduced to what is more common in the literature which, however, would have biased our choice of parameters to the task at hand. We were unable to outperform other unoptimized reservoirs from the literature when using more neurons without a plasticity rule, as unoptimized reservoirs are very dependent on the initial state and parameters which are random. This emphasis the prior need in literature for many of the reservoir optimization schemes. Moreover, many physical reservoir implementations possess a constant number of neurons, which requires the reservoir to adapt to the task and not vice versa (i.e., by optimizing the number of neurons).

Only the ON polarity of the input spike trains as available in the N-MNIST dataset was kept. For the readout layer, we used the Adam optimizer with amsgrad [52], a learning rate of $1e-5$ and a batch size of 10. We observe a $95.22 \pm 0.09\%$ accuracy on the test data. As N-MNIST is more substantial in the amount of data, only 1 epoch through the whole dataset was necessary to learn the readout layer and achieve these results.

We conducted a second faster experiment with only 1156 neurons in the reservoir. To do so, the 3D input spike train of shape $34 \times 34 \times$ time was split into sub-spike-trains, or quadrants, of shape $17 \times 17 \times$ time. We refer to this second experiment as the quadrant method. Figure 7 demonstrates this idea using a MNIST digit. We ran the experiment on the Loihi chip with P-CRITICAL and obtained an accuracy of 88.61%. The quadrant method allows for a smaller-sized input and reservoir (1156 instead of 8640 neurons), which was needed to mitigate the required communication bandwidth of the Loihi chip. We compare our method with other mostly unsupervised approaches on N-MNIST in table 1.

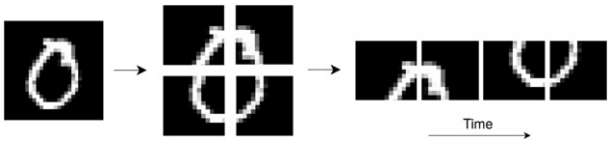


**Figure 7.** Quadrant representation of a MNIST digit. While this work uses the N-MNIST dataset, a MNIST digit is used for visualization purposes. Using a smaller representation of the digit in both height and width reduces the size of the reservoir and therefore the required bandwidth in and out of the Loihi research chip.

**Table 1.** Comparison of several models benchmarked on N-MNIST. All except [49] are based on the LSM architecture. These models combine preprocessing layers with either plastic adaptation or fixed-weights combined with a fully trained readout layer(s). Some software optimized reservoirs methods are shown as a comparison, although this work compares itself in the realm of unoptimized reservoirs and outperform contenders in this category.

| Model | Reservoir size | Details | Accuracy (%) |
|---|---|---|---|
| Iranmehr *et al* [50] | 625 | Unoptimized reservoir | 91.48 |
| Iranmehr *et al* [50] | 625 | Optimized reservoir | 92.56 |
| Iranmehr *et al* [50] | 625 | Optimized reservoir with a 120 neurons hidden fully connected layer | **98.38** |
| Guo *et al* [51] | 1000 | This work focus on input compression for smaller reservoirs | 91.67 |
| Thiele *et al* [49] | — | This work uses an unsupervised STDP trained CNN | 95.77 |
| CRITICAL (this work) | 8640 | Task-independent unoptimized reservoir with CRITICAL plasticity | **96.17** |
| P-CRITICAL (this work) | 8640 | Task-independent unoptimized reservoir with P-CRITICAL plasticity | 95.22 |
| P-CRITICAL (this work, on-chip) | 1156 | Task-independent unoptimized reservoir with the quadrant method | 88.61 |

The input layer and the reservoir are mapped directly in the neurocores of Loihi using the Nx SDK version 0.9.5. Time-binning is done on one of the available on-chip $\times 86$ processor before being transmitted to another computer for readout-layer classification and training.

The experiment was recreated using instead the original CRITICAL plasticity rule which inspired this work, with all the exact same parameters. An accuracy of $96.17 \pm 0.08\%$ on the test dataset was obtained. P-CRITICAL was able to achieve comparable results as CRITICAL, and surpass other unoptimized reservoir-like methods. As mentioned, no task-specific hyperparameters optimization was done and the number of neurons was selected on the assumption that the size of the reservoir had to be larger than the number of input features. This constraint greatly reduces the total training time of our reservoir approach, while providing guidelines for future physical reservoir implementation who may have limitations in the fine-tuning of network parameters with added physical variability.

### 4.3. Neuromorphic efficiency

We benchmarked P-CRITICAL with a network of 64 input neurons connected to a 512 neurons reservoir. The input neurons possess a current bias causing a fixed 40 Hz spiking input. This set-up is comparable to the number of spikes in the N-TIDIGITS experiment, without the added computational and energy cost of I/O. A 512 neurons reservoir with P-CRITICAL only takes about 2 to 3 neurocores on Loihi depending on the connectivity, out of a possible 128 cores per chip. The reservoir is running on 2 neurocores on Loihi. We compare the same network running on a chosen power-efficient CPU: an Intel i7-9750H. The CPU ran the PyTorch implementation. Such a network takes on average 0.88 ms per timestep to run on PyTorch. This model is therefore 1.13 times faster than our simulated timestep of 1 ms on PyTorch. In comparison, the same reservoir takes 17.52 $\mu$s on Loihi. We also benchmarked power efficiency for both implementations. The PyTorch version consumes 46 W of dynamical power. In comparison, the Loihi implementation only takes 17.3 mW. Table 2 shows a breakdown of energy and time consumption of both implementations.

We then scaled the reservoir to all 128 cores of a single Loihi chip ($64 \times 512$ neurons reservoirs). The required time to simulate was only increased to 19.75 $\mu$s because of the parallel nature of the chip. All efficiency experiments used the Nx SDK version 0.9.5 on a Nahuku 32 board ncl-ext-ghrd-01 with power probing.

**Table 2.** Comparison of energy consumption and speed of P-CRITICAL implementations for a 512 neurons reservoir. The CPU's efficiency was measured using Intel SoC Watch on Linux with kernel version 5.4.0-7634, Python 3.8.1 and PyTorch 1.4.0. Loihi's efficiency was measured using the power probes of the Nx SDK version 0.9.5 on Nahuku 32 board ncl-ext-ghrd-01. Spikes were generated using input neurons with a bias current to simulate a 40 Hz input frequency on Loihi to avoid I/O latency. Intel's Loihi research chip has shown major improvements in both power and time efficiency for P-CRITICAL when compared to a conventional CPU.

| | Power (mW) | | | | |
|---|---|---|---|---|---|
| | Static (Idle) | Dynamic | Total | $\frac{\text{Time}}{\text{Timestep}}$ ($\mu$s) | $\frac{\text{Energy}}{\text{Timestep}}$ ($\mu$J) |
| Loihi neurocores | $0.91 \pm 0.10$ | $18.3 \pm 0.1$ | $19.2 \pm 0.2$ | 17.52 | $0.336 \pm 0.004$ |
| Intel i7-9750H | $5380 \pm 40$ | $46\,000 \pm 2000$ | $51\,000 \pm 2000$ | 880 | $45\,000 \pm 2000$ |

## 5. Discussion

In terms of efficiency, running our model on Loihi is 50 times faster and three orders of magnitude more power efficient than our PyTorch implementation running on CPU. Indeed, the asynchronous nature of Loihi and the event-based communication enable a fast and efficient simulation of spiking neural networks. Loihi, like most neuromorphic processors, is limited in the neuronal model. This design choice allows an efficient implementation of spiking neural networks. In is therefore valuable to design algorithms such as P-CRITICAL with this hardware design choice in mind.

Another interesting aspect of our reservoir approach is the near independence of scaling the number of neurons to the simulation time on Loihi. Indeed, the neurons can be parallelized to a vast number of cores easily. The small-world topology choice that we made is efficient for this kind of architecture as it is more likely for neurons to communicate with their close-by neighbours, yet every neuron still has access to the whole networks in a few number of hops. Optimizing the parameters of this small-world topology with available brain data was simple yet effective. This new approach could help reduce the hardware design choices in supporting many types of topologies. Indeed, the neuromorphic processor would no longer have to support many reservoir topologies for different tasks, but rather a single one that works with many tasks.

While this new plasticity model did not stem from biological replication, we note an interesting relation between the engineering requirements of the regulation neurons and astrocytes in the brain. Our work showed that regulation mechanisms are easier to implement as their own entity in neuromorphic processors, suggesting that the brain could have developed astrocytes for similar reasons. In terms of physical reservoir implementation, reusing neuron blocks as-is for the adaptation of the network also helps reduce the implementation complexity. As many emergent nano-devices showcase neuron-like behaviour, the same devices could be reused as reservoir-regulating astrocytes, as it is done digitally in our work.

## 6. Conclusion

We proposed P-CRITICAL, a local plasticity rule that uses a mechanism that includes the use of astrocyte-like neurons and automatically adapts the neural network to operating points that are close to criticality [16–18]. Indeed, it is known that the control of this dynamic allows the reservoir to be more stable while having a faster and more efficient response. Without this type of control, a randomly configured reservoir might not work properly (chaotic, no activity, etc).

P-CRITICAL achieved its goal by tuning the branching factor of various reservoirs. The plasticity rule was able to offer a stable activity when connected to various raw input spike trains. As figure 4 demonstrated, even with a sparse input, the reservoir can maintain a fairly constant activity. By doing so, the reservoir will not suffer from sub or super criticality. Furthermore, this branching factor model should allow edge of chaos behaviour, maximizing the computing power and memory retention of the reservoir [16]. The plasticity rule was able to increase the test accuracy of initially unoptimized reservoirs for various high-level tasks coming from different sensory inputs that were captured with event-based sensors. We aim for P-CRITICAL to extend current RC methods such that they can be implemented on a neuromorphic processor and offer low-power edge devices the ability to train without requiring extensive computation or cloud server access.

RC is a good alternative to RNNs for faster training times. Plasticity-enabled reservoirs are well suited for neuromorphic engineering applications, as both information transmission and learning is sparse. This new model was compared between a CPU following the von Neumann architecture and the Loihi neuromorphic research chip. Both in time and power efficiency, Loihi was able to outperform its counterpart by many orders of magnitude.

**Table A1.** Current-leaky-integrate-and-fire generic constants.

| Symbol | PyTorch | Loihi | Description |
|---|---|---|---|
| $\tau_v$ | 30 ms | | Membrane potential decay constant |
| $\tau_i$ | 5 ms | | Membrane current decay constant |
| $v_{reset}$ | 0 | | Membrane reset voltage |
| $v_{threshold}$ | 1.0 | 256 | Membrane threshold voltage |
| $T_{refractory}$ | 2 ms | | Refractory period |

Furthermore, we created a new topology optimization scheme that is task independent and based on the eigenvalues spectrum of connectomes. This approach is a simple way of tuning the topology-related hyperparameters while avoiding fully connected reservoirs. We observed that the approach enhanced the performance of the reservoir by a significant margin when compared to an arbitrary choice of hyperparameters. Further work is needed to quantify the performance gains, both theoretically and empirically.

The independence of the network parameters in our LSM model allows the network to be scaled more easily, as the total training time is drastically reduced since hyperparameters optimization is unnecessary. Other physical or analog devices can be targeted with this approach, as any variability coming from the devices can be incorporated into the network behaviour and self-autocorrected by the plasticity rule. The parameter independence of reservoirs is not a subject studied thoroughly, even more so in physical implementation of reservoirs, yet this aspect is crucial for their relevance. Moreover, the task-independent topology can be embedded into a physical device as it requires no additional optimization. This contrasts to more generic digital and reprogrammable routing of the network that is done in devices such a Loihi. It is also sparser than other approaches, such as memristor memory arrays, that are analogous to fully connected reservoirs. Task-independent physical reservoirs can be implemented more efficiently when reprogrammability is unnecessary, such that a self-adapting reservoir could be suitable for very low energy and close-to-sensor applications.

In conclusion, we presented P-CRITICAL, a plasticity rule created for the autoregulation of reservoirs that tunes the branching factor to a target value. The plasticity rule was designed and adapted from recent literature [17] with Intel's Loihi as a target platform. With the hardware constraints in mind, we developed a plasticity rule able to successfully increase the computational power of reservoirs in LSMs. This approach will give future hardware implementation guidelines to create more efficient reservoirs. This will increase the impact of hardware reservoirs on real functional applications. We believe that this will be a key component for end-to-end energy-efficient machine learning algorithms on edge devices. In future works, we hope to combine our reservoir-plasticity method with state-of-the-art LSM readout layers [53] that can account for spike dynamics.

## Acknowledgments

The authors would like to thank Intel for giving us access to the Loihi chip, for travel grants and guidance, along with Hydro-Québec, Université de Sherbrooke and Compute Canada. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), funding reference number 559730. Fabien Alibart acknowledges support from IONOS-ERC project #773228. We thank the Fonds de recherche du Québec—nature et technologies (FRQNT) for the Chistera UNICO project #287330. We also thank the reviewers and the developers of all the mentioned python libraries. Finally, we thank Simon Brodeur for his valuable insights on the CRITICAL plasticity rule.

## Data availability statement

No new data were created or analysed in this study.

## Appendix A. Parameters

All simulations were executed with a numeric differential step size $dt = 1$ ms in both PyTorch and Loihi. Although similar in most cases, both PyTorch and Loihi values are presented (Tables A1–A4).

**Table A2.** Small-world topology constants.

| Symbol | PyTorch | Loihi | Description |
|---|---|---|---|
| $s$ | 40 | | Distance between neurons |
| $p$ | 1460 | | Distance increment between small-worlds |
| $C$ | 0.11 | | Maximum probability connection |
| $\lambda$ | 635 | | Euclidean distance divisor constant |
| $W_R^E \sim$ | [0.2, 0.5[ | [51.2, 128[ | Uniform distribution range of excitatory weights |
| $W_R^I \sim$ | [0.1, 0.3[ | [25.6, 75.8[ | Uniform distribution range of inhibitory weights |

**Table A3.** P-CRITICAL constants.

| Symbol | PyTorch | Loihi | Description |
|---|---|---|---|
| $\alpha$ | $1 \times 10^{-2}$ | 2 | Learning rate |
| $\beta$ | $1 \times 10^{-5}$ | 0.25 | Increment constant |
| $\tau_v'$ | 5 ms | | Membrane potential decay constant for regulation neurons |
| $\tau_i'$ | 0 ms | | Membrane current decay constant for regulation neurons |

**Table A4.** Time-binned read-out layer constants.

| Symbol | PyTorch | Loihi | Description |
|---|---|---|---|
| $T_{bins}$ | 60 ms | | Size of the time bins |

## ORCID iDs

Ismael Balafrej https://orcid.org/0000-0001-6730-0794
Fabien Alibart https://orcid.org/0000-0002-9591-220X
Jean Rouat https://orcid.org/0000-0002-9306-426X

## References

[1] Jaeger H 2001 The echo state approach to analysing and training recurrent neural networks-with an erratum note, Bonn, Germany: German National Research Center for Information Technology GMD *Technical Report* **vol 148** 13
[2] Maass W, Natschläger T and Markram H 2002 Real-time computing without stable states: a new framework for neural computation based on perturbations *Neural Comput.* **14** 2531–60
[3] Tanaka G, Yamane T, Héroux J B, Nakane R, Kanazawa N, Takeda S, Numata H, Nakano D and Hirose A 2019 Recent advances in physical reservoir computing: a review *Neural Netw.* **115** 100–23
[4] Legenstein R and Maass W 2007 Edge of chaos and prediction of computational performance for neural circuit models *Neural Netw.* **20** 323–34
[5] Oztuik M C, Xu D and Principe J C 2007 Analysis and design of echo state networks *Neural Comput.* **19** 111–38
[6] Verstraeten D and Schrauwen B 2009 On the quantification of dynamics in reservoir computing *Artificial Neural Networks—ICANN 2009* (Berlin: Springer) pp 985–94
[7] Gorad A, Saraswat V and Ganguly U 2019 Predicting performance using approximate state space model for liquid state machines *2019 Int. Joint Conf. Neural Networks (IJCNN), 2019-July* (Institute of Electrical and Electronics Engineers Inc.) pp 1–8
[8] Yildiz I B, Jaeger H and Kiebel S J 2012 Re-visiting the echo state property *Neural Netw.* **35** 1–9
[9] Hajnal M A and Lőrincz A 2006 Critical echo state networks *Artificial Neural Networks—ICANN 2006, 4131 LNCS* (Berlin: Springer) pp 658–67
[10] Roeschies B and Igel C 2009 Structure optimization of reservoir networks *Logic J. IGPL* **18** 635–69
[11] Ferreira A A and Ludermir T B 2009 Genetic algorithm for reservoir computing optimization *Proc. Int. Joint Conf. Neural Networks* pp 811–5
[12] Ju H, Xu J-X, Chong E and VanDongen A M J 2013 Effects of synaptic connectivity on liquid state machine performance *Neural Netw.* **38** 39–51
[13] Reynolds J J M, Plank J S and Schuman C D 2019 Intelligent reservoir generation for liquid state machines using evolutionary optimization *Proc. Int. Joint Conf. Neural Networks, 2019-July* (Institute of Electrical and Electronics Engineers Inc.)
[14] Tian S, Qu L, Hu K, Li N, Wang L and Xu W 2020 A neural architecture search based framework for liquid state machine design (arXiv:2004.07864 [cs.NE])
[15] de Lange S C, van den Heuvel M P and de Reus M A 2016 The role of symmetry in neural networks and their Laplacian spectra *Neuroimage* **141** 357–65

[16] Kello C T and Mayberry M R 2010 Critical branching neural computation *Proc. Int. Joint Conf. Neural Networks* pp 1–7
[17] Brodeur S and Rouat J 2012 Regulation toward self-organized criticality in a recurrent spiking neural reservoir *Artificial Neural Networks and Machine Learning—ICANN 2012* (*Ser. Lecture Notes in Computer Science, 7552 LNCS*) (Berlin: Springer) pp 547–54
[18] Stepp N, Plenz D and Srinivasa N 2015 Synaptic plasticity enables adaptive self-tuning critical networks *PLoS Comput. Biol.* **11** 1–28
[19] Davies M *et al* 2018 Loihi: a neuromorphic manycore processor with on-chip learning *IEEE Micro* **38** 82–99
[20] Paszke A *et al* 2019 PyTorch: an imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems* ed H Wallach, H Larochelle, A Beygelzimer, F d'Alch-Buc, E Fox and R Garnett vol 32
[21] Soures N, Hays L and Kudithipudi D 2017 Robustness of a memristor based liquid state machine *Proc. Int. Joint Conf. Neural Networks, 2017-May* pp 2414–20
[22] Moinnereau M-A, Brienne T, Brodeur S, Rouat J, Whittingstall K and Plourde E 2018 Classification of auditory stimuli from EEG signals with a regulated recurrent neural network reservoir (arXiv:1804.10322 [cs,eess])
[23] Luo S, Guan H, Li X, Xue F and Zhou H 2018 Improving liquid state machine in temporal pattern classification *2018 15th Int. Conf. Control, Automation, Robotics and Vision, ICARCV 2018* pp 88–91
[24] Soures N and Kudithipudi D 2019 Deep liquid state machines with neural plasticity for video activity recognition *Front. Neurosci.* **13** 686
[25] Tieck J C V, Pogančić M V, Kaiser J, Roennau A, Gewaltig M-O and Dillmann R 2018 Learning continuous muscle control for a multi-joint arm by extending proximal policy optimization with a liquid state machine *27th Int. Conf. Artificial Neural Networks* vol 11139 (Rhodes, Greece–October 2018) ed V Kurková, Y Manolopoulos, B Hammer, L S Iliadis and I Maglogiannis pp 211–21
[26] Liu Y, Zhang W and Li P 2019 Enabling non-Hebbian learning in recurrent spiking neural processors with hardware-friendly on-chip intrinsic plasticity *IEEE J. Emerg. Sel. Top. Circuits Syst.* **9** 465–74 (Institute of Electrical and Electronics Engineers Inc.)
[27] Xue F, Li Q, Zhou H and Li X 2017 Reservoir computing with both neuronal intrinsic plasticity and multi-clustered structure *Cogn. Comput.* **9** 400–10
[28] Jin Y, Liu Y and Li P 2016 SSO-LSM: a sparse and self-organizing architecture for liquid state machine based neural processors *Proc. 2016 IEEE/ACM Int. Symp. Nanoscale Architectures, NANOARCH 2016* pp 55–60
[29] Kawai Y, Tokuno T, Park J and Asada M 2018/2017 Echo in a small-world reservoir: time-series prediction using an economical recurrent neural network *2017 Joint IEEE Int. Conf. Development and Learning and Epigenetic Robotics (ICDL-EpiRob)* pp 126–31
[30] Kawai Y, Park J and Asada M 2019 A small-world topology enhances the echo state property and signal propagation in reservoir computing *Neural Netw.* **112** 15–23
[31] Manevitz L and Hazan H 2010 Stability and topology in reservoir computing *Advances in Soft Computing* ed G Sidorov, A Hernandez Aguirre and C A Reyes Garcia 6438 LNAI (Berlin: Springer) pp 245–56
[32] Deng Z and Zhang Y 2007 Collective behavior of a small-world recurrent neural system with scale-free distribution *IEEE Trans. Neural Netw.* **18** 1364–75
[33] Bertschinger N and Natschläger T 2004 Real-time computation at the edge of chaos in recurrent neural networks *Neural Comput.* **16** 1413–36
[34] Verstraeten D, Schrauwen B, D'Haene M and Stroobandt D 2007 An experimental unification of reservoir computing methods *Neural Netw.* **20** 391–403
[35] Alexandre L A, Embrechts M J and Linton J 2009 Benchmarking reservoir computing on time-independent classification tasks *Proc. Int. Joint Conf. Neural Networks* pp 89–93
[36] Beggs J M 2007 The criticality hypothesis: how local cortical networks might optimize information processing *Phil. Trans. R. Soc. A* **366** 329–43
[37] Priesemann V, Wibral M, Valderrama M, Prpper R, Le Van Quyen M, Geisel T, Triesch J, Nikoli D and Munk M H J 2014 Spike avalanches *in vivo* suggest a driven, slightly subcritical brain state *Front. Syst. Neurosci.* **8** 108
[38] Davey N, Calcraft L and Adams R 2006 High capacity, small world associative memory models *Connect. Sci.* **18** 247–64
[39] Wijesinghe P, Srinivasan G, Panda P and Roy K 2019 Analysis of liquid ensembles for enhancing the performance and accuracy of liquid state machines *Front. Neurosci.* **13** 504
[40] Wang M B, Owen J P, Mukherjee P and Raj A 2017 Brain network eigenmodes provide a robust and compact representation of the structural connectome in health and disease *PLoS Comput. Biol.* **13** e1005550 ed S Jbabdi
[41] Ioffe S and Szegedy C 2015 Batch normalization: accelerating deep network training by reducing internal covariate shift *32nd Int. Conf. Machine Learning, ICML 2015*
[42] Orchard G, Jayawant A, Cohen G K and Thakor N 2015 Converting static image datasets to spiking neuromorphic datasets using saccades *Front. Neurosci.* **9** 437
[43] Anumula J, Neil D, Delbruck T and Liu S-C 2018 Feature representations for neuromorphic audio spike streams *Front. Neurosci.* **12** 23
[44] Posch C, Matolin D and Wohlgenannt R 2011 A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS *IEEE J. Solid-State Circuits* **46** 259
[45] Chan V, Liu S C, van Schaik A and AER E A R 2007 A matched silicon cochlea pair with address event representation interface *IEEE Trans. Circuits Syst.* I **54** 48
[46] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980 [cs.LG])
[47] Stimberg M, Brette R and Goodman D F 2019 Brian 2, an intuitive and efficient neural simulator *Elife* **8** e47314
[48] Neil D and Liu S-C 2016 Effective sensor fusion with event-based sensors and deep network architectures *2016 IEEE Int. Symp. Circuits and Systems (ISCAS)* pp 2282–5
[49] Thiele J C, Bichler O and Dupret A 2018 A timescale Invariant STDP-based spiking deep network for unsupervised online feature Extraction from event-based sensor data *2018 Int. Joint Conf. Neural Networks (IJCNN), 2018-July* (Piscataway, NJ: IEEE) pp 1–8
[50] Iranmehr E, Shouraki S B, Faraji M M, Bagheri N and Linares-Barranco B 2019 Bio-inspired evolutionary model of spiking neural networks in ionic liquid space *Front. Neurosci.* **13** 1–18
[51] Guo S, Qu L, Wang L, Tian S, Li S and Xu W 2020 Exploration of input patterns for enhancing the performance of liquid state machines (arXiv:2004.02540 [cs.CV])
[52] Reddi S J, Kale S and Kumar S 2018 On the convergence of Adam and beyond *6th Int. Conf. Learning Representations, ICLR 2018—Conf. Track Proc.*
[53] Zhang Y, Li P, Jin Y and Choe Y 2015 A digital liquid state machine with biologically inspired learning and its application to speech recognition *IEEE Trans. Neural Netw. Learning Syst.* **26** 2635–49