frontiers | Frontiers in Computational Neuroscience

TYPE Original Research
PUBLISHED 16 November 2022
DOI 10.3389/fncom.2022.1037976

OPEN ACCESS

EDITED BY
Yuqi Han,
Tsinghua University, China

REVIEWED BY
Rachmad Vidya Wicaksana Putra,
Vienna University of Technology, Austria
Chenglong Zou,
Peking University, China
Oliver Rhodes,
The University of Manchester, United Kingdom
Federico Corradi,
Eindhoven University of Technology, Netherlands

*CORRESPONDENCE
Nicolas Skatchkovsky
nicolas.skatchkovsky@kcl.ac.uk

RECEIVED 06 September 2022
ACCEPTED 26 October 2022
PUBLISHED 16 November 2022

CITATION
Skatchkovsky N, Jang H and Simeone O (2022) Bayesian continual learning *via* spiking neural networks. *Front. Comput. Neurosci.* 16:1037976. doi: 10.3389/fncom.2022.1037976

Bayesian continual learning *via* spiking neural networks

Nicolas Skatchkovsky[1]*, Hyeryung Jang[2] and Osvaldo Simeone[1]

[1]King's Communication, Learning and Information Processing (KCLIP) Lab, Department of Engineering, King's College London, London, United Kingdom, [2]Department of Artificial Intelligence, Dongguk University, Seoul, South Korea

Among the main features of biological intelligence are energy efficiency, capacity for continual adaptation, and risk management *via* uncertainty quantification. Neuromorphic engineering has been thus far mostly driven by the goal of implementing energy-efficient machines that take inspiration from the time-based computing paradigm of biological brains. In this paper, we take steps toward the design of neuromorphic systems that are capable of adaptation to changing learning tasks, while producing well-calibrated uncertainty quantification estimates. To this end, we derive online learning rules for spiking neural networks (SNNs) within a Bayesian continual learning framework. In it, each synaptic weight is represented by parameters that quantify the current epistemic uncertainty resulting from prior knowledge and observed data. The proposed online rules update the distribution parameters in a streaming fashion as data are observed. We instantiate the proposed approach for both real-valued and binary synaptic weights. Experimental results using Intel's Lava platform show the merits of Bayesian over frequentist learning in terms of capacity for adaptation and uncertainty quantification.

KEYWORDS

spiking neural networks, Bayesian learning, neuromorphic learning, neuromorphic hardware, artificial intelligence

1. Introduction

Recent advances in machine learning and artificial intelligence systems have been largely driven by a pursuit of accuracy *via* resource-intensive pattern recognition algorithms run in a *train-and-then-deploy* fashion. In stark contrast, neuroscience paints a picture of intelligence that revolves around *continual adaptation*, *uncertainty quantification*, and resource budgeting (allostasis) for the parsimonious processing of *event-driven* information (Doya et al., 2007; Friston, 2010; Feldman Barrett, 2021; Hawkins, 2021). Taking inspiration from neuroscience, over the last decade, neuromorphic engineering has pursued the goal of implementing energy-efficient machines that process information *with* time *via* sparse inter-neuron binary signals—or *spikes* (Davies et al., 2021). The main aim of this paper is to introduce algorithmic solutions to endow neuromorphic models, namely spiking neural networks (SNNs), with the capacity for adaptation to changing learning tasks, while ensuring the reliable quantification of uncertainty of the model's decisions.

1.1. Managing uncertainty *via* Bayesian learning

Training algorithms for SNNs have been overwhelmingly derived by following the *frequentist* approach which consists in minimizing the training loss with respect to the model parameter vector (Shrestha and Orchard, 2018; Zenke and Ganguli, 2018; Bellec et al., 2020; Kaiser et al., 2020). This is partly motivated by the dominance of frequentist learning, and associated software tools, in the literature on deep learning for conventional artificial neural networks (ANNs). Frequentist learning is well justified when enough data are available to make the training loss a good empirical approximation of the underlying population loss (Clayton, 2021). When this condition is not satisfied, while the model's average accuracy may be satisfactory on test data, the decisions made by the trained model can be badly calibrated, often resulting in overconfident predictions (Nguyen et al., 2015; Guo et al., 2017). The problem is particularly significant for decisions made on test data that differ significantly from the data observed during training—a common occurrence for applications such as self-driving vehicles. Furthermore, the inability of frequentist learning to account for uncertainty limits its capacity to adapt to new tasks while retaining the capacity to operate on previous tasks (Ebrahimi et al., 2020).

The main cause of the poor calibration of frequentist learning is the selection of a single parameter vector, which disregards any uncertainty on the best model to use for a certain task due to the availability of limited data. A more principled approach that has the potential to properly account for such *epistemic uncertainty*, i.e., for uncertainty related to the availability of limited data, is given by *Bayesian learning* (Jaynes, 2003) and by its generalized form known as *information risk minimization* (see, e.g., Zhang, 2006; Guedj, 2019; Knoblauch et al., 2019; Jose and Simeone, 2021; Simeone, 2022). Bayesian learning maintains a *distribution* over the model parameter vector that represents the partial information available to the learner. This way, Bayesian models can provide well-calibrated decisions, which quantify accurately the associated degree of uncertainty and can be used to detect out-of-distribution inputs (Daxberger and Hernández-Lobato, 2019). In the self-driving example provided earlier, the vehicle may hand back control to the driver when the certainty of its decision is below a certain threshold.

Bayesian reasoning is at the core of the *Bayesian brain* hypothesis in neuroscience, according to which biological brains constantly update an internal model of the world in an attempt to minimize their information-theoretic surprise. This hypothesis is formalized by the free energy principle, which measures surprise in terms of a variational free energy (Friston, 2012). In this context, synaptic plasticity has been hypothesized to be well-modeled as Bayesian learning, which keeps track of the distributions of synaptic weights over time (Aitchison et al., 2021).

In the present paper, we propose (generalized) Bayesian learning rules for SNNs with binary and real-valued synaptic weights that can adapt over time to changing learning tasks.

1.2. Related work

Bayesian learning, and its application to deep ANNs, typically labeled as *Bayesian deep learning*, is receiving increasing attention in the literature. We refer to the following work for a recent overview (Wang and Yeung, 2020). Natural gradient descent rule known as the *Bayesian learning rule* was introduced in Khan and Lin (2017), then applied in Meng et al. (2020) to train binary ANNs, and to a variety of other scenarios in Khan and Rue (2021). Khan and Rue (2021) demonstrates that the Bayesian learning rule recovers many state-of-the-art machine learning algorithms in a principled fashion. We also point to the Kreutzer et al. (2020) that explores the use of natural gradient descent for frequentist learning in spiking neurons.

As mentioned, the choice of a Bayesian learning framework is in line with the importance of the Bayesian brain hypothesis in computational neurosciences (Friston, 2012). The recent Aitchison et al. (2021) explores a Bayesian paradigm to model biological synapses as an explanation of the capacity of the brain to perform learning in the presence of noisy observations. A Bayesian approach to neural plasticity was previously proposed for synaptic sampling, by modeling synaptic plasticity as sampling from a posterior distribution (Kappel et al., 2015). Apart from the conference version (Jang et al., 2021) of the present work, this paper is the first to explore the definition of Bayesian learning and Bayesian continual learning rules for general SNNs adopting the standard spike response model [SRM, see, e.g., (Gerstner and Kistler, 2002)].

Continual learning is a key area of machine learning research, which is partly motivated by the goal of understanding how biological brains maintain previously acquired skills while adding new capabilities. Unlike traditional machine learning, whereby one performs training based on a single data source, in continual learning, several datasets, corresponding to different tasks, are sequentially presented to the learner. A challenge in continual learning is the ability of the learning algorithm to perform competitively on previous tasks after training on the subsequently observed datasets. In this context, *catastrophic forgetting* indicates the situation in which performance drops sharply on previously encountered tasks after learning new ones. Many continual learning techniques follow the principle of preserving synaptic connections that are deemed important to perform well on previously learned tasks *via* a regularization of the learning objective (Kirkpatrick et al., 2017; Zenke et al., 2017). Bayesian approaches have also been proposed for this purpose, whereby priors are selected as the posterior evaluated on the previous task to prevent the new posterior distribution from deviating too much from

learned states. Biological mechanisms are explicitly leveraged in works such as Laborieux et al. (2021) and Soures et al. (2021), which combine a variety of neural mechanisms to obtain state-of-the-art performance for SNNs on standard continual learning benchmarks. Putra and Shafique (2022) also proposes a continual learning algorithm for SNNs in an unsupervised scenario by assuming limited precision for the weights. In the present paper, we demonstrate how Bayesian learning allows obtaining similar biologically inspired features by following a principled objective grounded in information risk minimization.

Traditionally, training of SNNs has relied on biologically realistic Hebbian rules, among which spike-timing dependent plasticity (STDP) is the most popular. STDP modulates the synaptic weight between two neurons based on the firing times of both neurons. A long-term potentiation (i.e., an increase in the weight) of the synapse occurs when the pre-synaptic neuron spikes right before the post-synaptic neuron, while long-term depression (i.e., a decrease in the weight) of a synapse happens when the pre-synaptic neuron spikes after the post-synaptic neuron. STDP implements a form of unsupervised learning, and can be leveraged to perform tasks such as clustering, while also supporting continual learning (Vaila et al., 2019).

Supervised learning based on the minimization of the training loss is challenging in SNNs due to the activation function of spiking neurons, the derivative of which is always zero, except at the spike time, where it is not differentiable. Modern training algorithms (Zenke and Ganguli, 2018; Bellec et al., 2020; Kaiser et al., 2020) overcome this difficulty through the use of *surrogate gradients*, i.e., by replacing the true derivative with that of a well-defined differentiable function (Neftci et al., 2019). An alternative approach, reviewed in Jang et al. (2019), is to view the SNN as a probabilistic model whose likelihood can be directly differentiated. Further extensions of the probabilistic modeling approach and associated training rules are presented in Jang and Simeone (2022) and Jang et al. (2020b).

An application of Bayesian principles to SNNs has first been proposed in the conference version of this paper (Jang et al., 2021). Jang et al. (2021) focuses on SNNs with binary synaptic weights and offline learning, presenting limited experimental results. In contrast, the current paper provides all the necessary background, including frequentist learning; it covers frequentist and Bayesian continual learning; and it provides extensive experimental results on a variety of tasks.

1.3. Main contributions

In this work, we derive online learning rules for SNNs within a Bayesian continual learning framework. In it, each synaptic weight is represented by parameters that quantify the current epistemic uncertainty associated with prior knowledge and data observed thus far. Bayesian methods are key to handling uncertainty over time, providing the model knowledge of what is to be retained, and what can be forgotten (Ebrahimi et al., 2020). The main contributions are as follows.

i) We introduce general frameworks for the definition of single-task and continual Bayesian learning problems for SNNs that are based on information risk minimization and variational inference. Following the desiderata formulated in Farquhar and Gal (2019a), we focus on the standard formulation of continual learning in which there exist clear demarcations between subsequent tasks, but the learner is unaware of the identity of the current task. For example, in the typical example of an autonomous vehicle navigating in several environments, the vehicle may be aware that it is encountering a new terrain, while being a priori unaware of the type of new terrain. Furthermore, the model is not modified between tasks, and tasks may be encountered more than once;

ii) We instantiate the general Bayesian learning frameworks for SNNs with real-valued synapses. To this end, we adopt a Gaussian variational distribution for the synaptic weights, and demonstrate learning rules that can adapt the parameters of the weight distributions online. This choice of variational posterior has been previously explored for ANNs, and can yield state-of-the-art performance on real-life datasets (Osawa et al., 2019);

iii) We then introduce Bayesian single-task and continual learning rules for SNNs with binary weights, with the main goal of supporting more efficient hardware implementations (Courbariaux et al., 2016; Rastegari et al., 2016), including platforms based on beyond-CMOS memristors (Mehonic et al., 2020);

iv) Through experiments on both synthetic and real neuromorphic datasets, we demonstrate the advantage of the Bayesian learning paradigm in terms of accuracy and calibration for both single-task and continual learning. As neuromorphic algorithms are designed to be run on dedicated hardware, we run the experiments using Intel's Lava software emulator platform (Intel Corporation, 2021), accounting for the limited precision of synaptic weights in hardware.

2. Methods

We first introduce the adopted SNN model, namely the standard spike response model (SRM), before giving a short overview of frequentist, Bayesian, continual, and biologically inspired learning. We then detail learning rules for offline and continual frequentist learning, and derive associated online Bayesian learning rules.

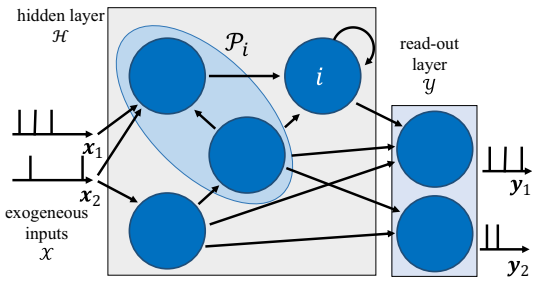

FIGURE 1
Illustration of the internal architecture of an SNN. The behavior of neurons in the read-out layer is guided by the training data, while that of neurons in the hidden layer is adjusted to fit the data. The blue shaded area represents the set of pre-synaptic neurons $\mathcal{P}_i$ to neuron i.

2.1. SNN model

2.1.1. Spike response model

The architecture of an SNN is defined by a network of spiking neurons connected over an arbitrary graph, which possibly includes (directed) cycles. As illustrated in Figure 1, the directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is described by a set $\mathcal{N}$ of nodes, representing the neurons, and by a set $\mathcal{E}$ of directed edges $i \to j$ with $i \neq j \in \mathcal{N}$, representing synaptic connections.

Focusing on a discrete-time implementation, each spiking neuron $i \in \mathcal{N}$ produces a binary value $s_{i,t} \in \{0, 1\}$ at discrete time $t = 1, 2, \ldots$, with "1" denoting the firing of a spike. We collect in an $|\mathcal{N}| \times 1$ vector $\boldsymbol{s}_t = (s_{i,t} : i \in \mathcal{N})$ the spikes emitted by all neurons $\mathcal{N}$ at time t, and denote by $\boldsymbol{s}^t = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t)$ the spike sequences of all neurons up to time t. Without loss of generality, we consider time-sequences of length T, and write $\boldsymbol{s} := \boldsymbol{s}^T$. Each neuron i receives input spike signals $\{s_{j,t}\}_{j\in\mathcal{P}_i} = \boldsymbol{s}_{\mathcal{P}_i,t}$ at time t from the set $\mathcal{P}_i = \{j \in \mathcal{N} : (j \to i) \in \mathcal{E}\}$ of parent, or pre-synaptic, neurons, which are connected to neuron i *via* directed links in the graph $\mathcal{G}$. With some abuse of notations, this set is taken to include also exogeneous input signals.

Each neuron i maintains a scalar analog state variable $u_{i,t}$, known as the *membrane potential*. Mathematically, neuron i outputs a binary signal $s_{i,t}$, or spike, at time t when the membrane potential $u_{i,t}$ is above a threshold ϑ, i.e.,

$$s_{i,t} = \Theta(u_{i,t} - \vartheta), \tag{1}$$

with $\Theta(\cdot)$ being the Heaviside step function and ϑ being the fixed firing threshold. Following the standard discrete-time SRM (Gerstner and Kistler, 2002), the membrane potential $u_{i,t}$ is obtained by summing filtered contributions from pre-synaptic neurons in set $\mathcal{P}_i$ and from the neuron's own output. In particular, the membrane potential evolves as

$$u_{i,t} = \sum_{j\in\mathcal{P}_i} w_{ij}(\alpha_t * s_{j,t}) - \beta_t * s_{i,t}, \tag{2}$$

where w_{ij} is a learnable synaptic weight from pre-synaptic neuron $j \in \mathcal{P}_i$ to post-synaptic neuron i; and we collect in vector $\boldsymbol{w} = \{\boldsymbol{w}_i\}_{i\in\mathcal{N}}$ the model parameters, with $\boldsymbol{w}_i := \{w_{ij}\}_{j\in\mathcal{P}_i}$ being the synaptic weights for each neuron i. We have denoted as α_t and β_t the spike responses of synapses and somas, respectively; while $*$ denotes the convolution operator $f_t * g_t = \sum_{\delta>0} f_\delta g_{t-\delta}$. When implemented with autoregressive filters, the SRM is equivalent to leaky integrate-and-fire (LIF) neuron model (Gerstner and Kistler, 2002; Kaiser et al., 2020). The techniques developed in this work can be directly generalized to other, more complex, neuron models, such as resonate-and-fire (Izhikevich, 2001), but we leave an investigation of this point to future work.

2.1.2. Real-valued and binary-valued synapses

In this paper, we will consider two implementations of the SRM introduced in the previous subsection. In the first, the synaptic weights in vector $\boldsymbol{w}$ are real-valued, i.e., $w_{ij} \in \mathbb{R}$, with possibly limited resolution, as dictated by deployment on neuromorphic hardware (see Section 3). In contrast, in the second implementation, the weights are binary, i.e., $w_{ij} \in \{+1, -1\}$. The advantages of models with binary-valued synapses, which we call binary SNNs, include a reduced complexity for the computation of the membrane potential $u_{i,t}$ in Equation (2). Furthermore, binary SNNs are particularly well suited for implementations on chips with nanoscale components that provide discrete conductance levels for the synapses (Mehonic et al., 2020). In this regard, we note that the methods described in this paper can be generalized to models with weights having any discrete number of values.

2.2. Frequentist vs. Bayesian learning

With traditional frequentist learning, the vector of synaptic weights $\boldsymbol{w}$ is optimized by minimizing a training loss. The training loss is adopted as a proxy for the population loss, i.e., for the loss averaged over the true, unknown, distribution of the data. Therefore, frequentist learning disregards the inherent uncertainty caused by the availability of limited training data, which causes the training loss to be a potentially inaccurate estimate of the population loss. As a result, frequentist learning is known to potentially yield poorly calibrated, and overconfident decisions for ANNs (Nguyen et al., 2015).

In contrast, as seen in Figure 2A, Bayesian learning optimizes over a distribution $q(\boldsymbol{w})$ in the space of the synaptic weight vector $\boldsymbol{w}$. The distribution $q(\boldsymbol{w})$ captures the *epistemic* uncertainty induced by the lack of knowledge of the true

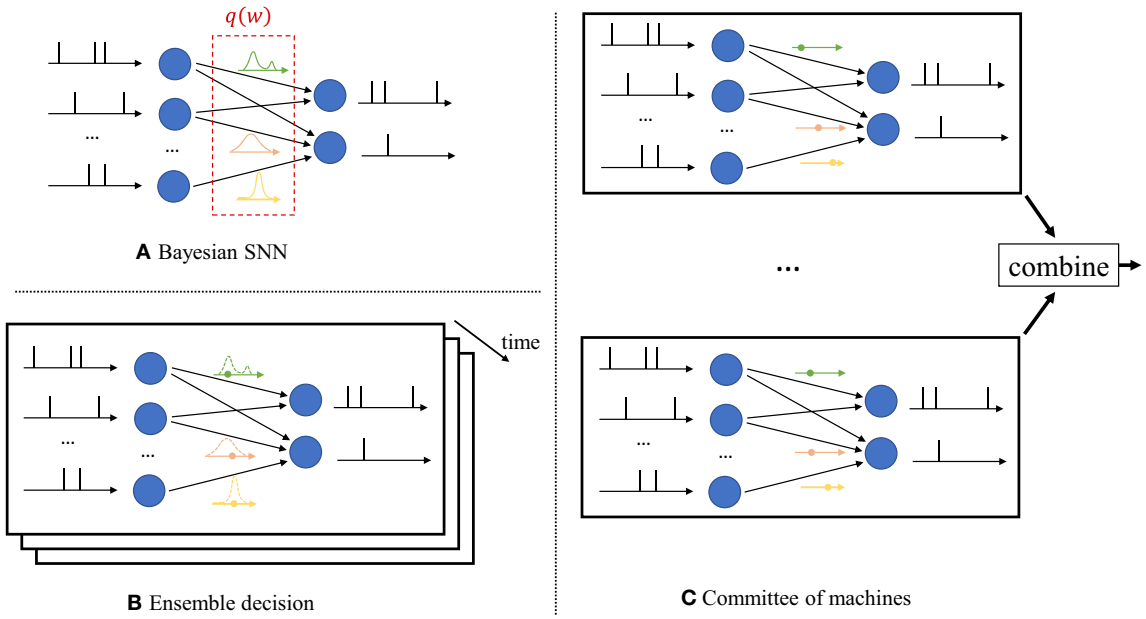

FIGURE 2
Illustration of Bayesian learning in an SNN. **(A)** In a Bayesian SNN, the synaptic weights $\boldsymbol{w}$ are assigned a joint distribution $q(\boldsymbol{w})$, often simplified as a product distribution across weights. **(B)** An ensemble decision is obtained by sampling several times from the distribution $q(\boldsymbol{w})$, and by averaging the predictions of the sampled models. Sampling is done independently for each new input. **(C)** In a committee machine, the weights for several models are drawn only once from $q(\boldsymbol{w})$, and the same models are run in parallel for any new input.

distribution of the data. This is done by assigning similar values of $q(\boldsymbol{w})$ to model parameters that fit equally well the data, while also being consistent with prior knowledge. As a consequence, Bayesian learning is known to produce better calibrated decisions, i.e., decisions whose associated confidence better reflects the actual accuracy of the decision (Guo et al., 2017). Furthermore, models trained *via* Bayesian learning can better detect out-of-distribution data, i.e., data that is not covered by the distribution of the training set (Daxberger and Hernández-Lobato, 2019; Kristiadi et al., 2020).

Once distribution $q(\boldsymbol{w})$ is optimized *via* Bayesian learning, at inference time a decision on any new test input is made by averaging the decisions of multiple models, with each being drawn from the distribution $q(\boldsymbol{w})$. The average over multiple models can be realized in one of two ways.

i) Ensemble predictor: Given a test input, as seen in Figure 2B, one draws a new synaptic weight vector several times from the distribution $q(\boldsymbol{w})$, and an *ensemble* decision is obtained by averaging the decisions produced by running the SNN with each sampled weight vector;

ii) Committee machine: Alternatively, one can sample a number of realizations from the distribution $q(\boldsymbol{w})$ that are kept fixed and reused for all test inputs. This solution foregoes the sampling step at inference time as illustrated in Figure 2C. However, the approach generally requires a larger memory to store all samples $\boldsymbol{w}$ to be used for inference, while the ensemble predictor can make decisions using different weight vectors $\boldsymbol{w} \sim q(\boldsymbol{w})$ sequentially over time.

2.3. Offline vs. continual learning

Offline learning denotes the typical situation where the system is presented with a single training dataset $\mathcal{D}$, which is used to measure a training loss. In offline learning, optimization of the training loss is carried out once and for all, resulting in a synaptic weight vector $\boldsymbol{w}$ or in a distribution $q(\boldsymbol{w})$ for frequentist or Bayesian learning, respectively. Offline learning is hence, by construction, unable to adapt to changing conditions, and it is deemed to be a poor representation of how intelligence works in biological organisms (Kudithipudi and Aguilar-Simon, 2022).

In continual learning, the system is sequentially presented datasets $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \ldots$ corresponding to distinct, but related, learning tasks, where each task is selected, possibly with replacement, from a pool of tasks, and its identity is unknown to the system. For each task k, the system is given a training set $\mathcal{D}^{(k)}$, and its goal is to learn to make predictions that generalize well on the new task, while causing minimal loss of accuracy on previous tasks $1, \ldots, k-1$. In frequentist continual learning, the model parameter vector $\boldsymbol{w}$ is updated as data from successive tasks is collected. Conversely, in Bayesian continual learning, the distribution $q(\boldsymbol{w})$ is updated over time as illustrated in Figure 3. The updates should be sufficient to address the needs of the

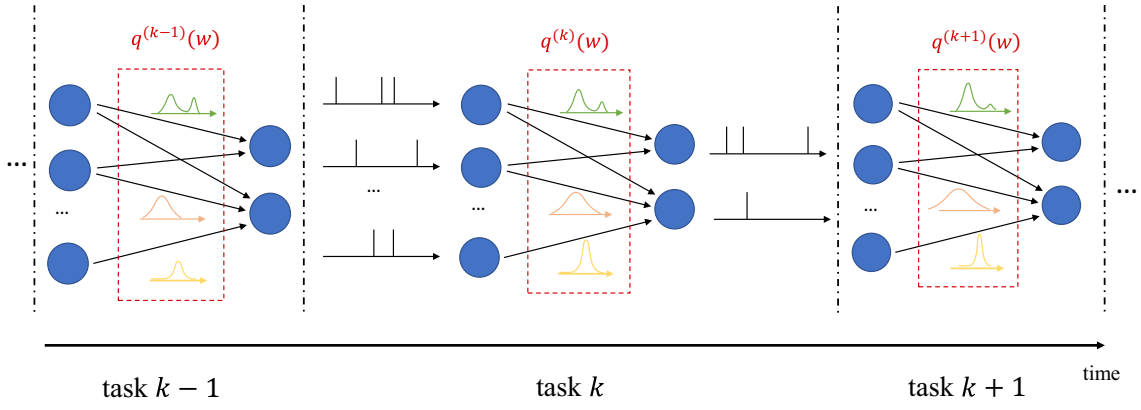

FIGURE 3
Illustration of Bayesian continual learning: the system is successively presented with similar, but different, tasks. Bayesian learning allows the model to retain information about previously learned information.

new task, while not disrupting performance on previous tasks, operating on a *stability-plasticity trade-off*.

2.4. Biological principles of learning

Many existing works on continual learning draw their inspiration from the mechanisms underlying the capability of biological brains to carry out life-long learning (Soures et al., 2021; Kudithipudi and Aguilar-Simon, 2022). Learning is believed to be achieved in biological systems by modulating the strength of synaptic links. In this process, a variety of mechanisms are at work to establish short-to intermediate-term and long-term memory for the acquisition of new information over time (Kandel et al., 2014). These mechanisms operate at different time and spatial scales.

One of the best understood mechanisms, *long-term potentiation*, contributes to the management of long-term memory through the consolidation of synaptic connections (Morris, 2003; Malenka and Bear, 2004). Once established, these are rendered resistant to disruption by changing their capacity to change *via metaplasticity* (Abraham and Bear, 1996; Finnie and Nader, 2012). As a related mechanism, return to a base state is ensured after exposition to small, noisy changes by *heterosynaptic plasticity*, which plays a key role in ensuring the stability of neural systems (Chistiakova et al., 2014). *Neuromodulation* operates at the scale of neural populations to respond to particular events registered by the brain (Marder, 2012). Finally, *episodic replay* plays a key role in the maintenance of long-term memory, by allowing biological brains to re-activate signals seen during previous active periods when inactive (i.e., sleeping) (Kudithipudi and Aguilar-Simon, 2022).

2.5. Frequentist offline learning

We now review frequentist offline training algorithms for SNNs, under the SRM model described in Section 2.1.1. This will provide the necessary background for Bayesian learning and its continual version, described in Sections 2.6 and 2.8, respectively.

2.5.1. Empirical risk minimization

To start, as illustrated in Figure 1, we divide the set $\mathcal{N}$ of neurons of the SNN into two subsets $\mathcal{Y}$ and $\mathcal{H}$ with $\mathcal{N} = \mathcal{Y} \cup \mathcal{H}$: a set of read-out, or output, neurons $\mathcal{Y}$ and a set of hidden neurons $\mathcal{H}$. The set of exogeneous inputs is defined as $\mathcal{X}$. We focus on supervised learning, in which a dataset $\mathcal{D}$ is given by $|\mathcal{D}|$ pairs $(\boldsymbol{x}, \boldsymbol{y})$ of signals generated from an unknown distribution $p(\boldsymbol{x}, \boldsymbol{y})$, with $\boldsymbol{x}$ being exogeneous input signals, one for each element of the set $\mathcal{X}$, and $\boldsymbol{y}$ the corresponding desired output signals. Both $\boldsymbol{x}$ and $\boldsymbol{y}$ are vector sequences of length T, with $\boldsymbol{x}$ comprising $|\mathcal{X}|$ signals, and $\boldsymbol{y}$ including $|\mathcal{Y}|$ signals. Each output samples $y_{m,t}$ in $\boldsymbol{y}$ dictates the desired behavior of the mth neuron in the read-out set $\mathcal{Y}$. The sequences in $\boldsymbol{x}$ and $\boldsymbol{y}$ can generally take arbitrary real values (see Section 3 for specific examples).

In frequentist learning, the goal is to minimize the training loss over the parameter vector $\boldsymbol{w}$ using the training dataset $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{y})\}$. To elaborate, we define the loss $\mathcal{L}_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{w})$ measured with respect to a data $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$ as the error between the reference signals $\boldsymbol{y}$ and the output spiking signals produced by the SNN with parameters $\boldsymbol{w}$, given the input $\boldsymbol{x}$. Accordingly, the loss is written as a sum over time instants $t = 1, \dots, T$ and over the $|\mathcal{Y}|$ read-out neurons as

$$\mathcal{L}_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{w}) = \sum_{t=1}^{T} \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w}) = \sum_{t=1}^{T} \sum_{m \in \mathcal{Y}} L\big(y_{m,t}, f_m(\boldsymbol{w}, \boldsymbol{x}^t)\big), \quad (3)$$

where function $L\big(y_{m,t}, f_m(\boldsymbol{w}, \boldsymbol{x}^t)\big)$ is a local loss measure comparing the target output $y_{m,t}$ of neuron m at time t and the actual output $f_m(\boldsymbol{w}, \boldsymbol{x}^t)$ of the same neuron. The notations $f_m(\boldsymbol{w}, \boldsymbol{x}^t)$ and $\mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})$ are used as a reminder that the output of the SNN and the corresponding loss at time t generally depend on the input $\boldsymbol{x}^t$ up to time t, and on the target output $\boldsymbol{y}_t$ at time t. Specifically, the notation $f_m(\boldsymbol{w}, \boldsymbol{x}^t)$ makes it clear that the output of neuron $m \in \mathcal{Y}$ is produced with the model parameters $\boldsymbol{w}$ from exogeneous input $\boldsymbol{x}^t$, consisting of all input samples up to time t, using the SRM (Equations 1, 2).

The training loss $\mathcal{L}_{\mathcal{D}}(\boldsymbol{w})$ is an empirical estimate of the population loss based on the data samples in the training dataset $\mathcal{D}$, and is given as

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{w}) = \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}} \mathcal{L}_{\boldsymbol{x}, \boldsymbol{y}}(\boldsymbol{w}). \tag{4}$$

Frequentist learning addresses the empirical risk minimization (ERM) problem

$$\min_{\boldsymbol{w}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{w}). \tag{5}$$

Problem (Equation 5) cannot be directly solved using standard gradient-based methods since: *(i)* the spiking mechanism (Equation 1) is not differentiable in $\boldsymbol{w}$ due to the presence of the threshold function $\Theta(\cdot)$; and *(ii)* in the case of binary SNNs, the domain of the weight vector $\boldsymbol{w}$ is the discrete set of binary values.

To tackle the former problem, as detailed in Section 2.5.2, surrogate gradients (SG) methods replace the derivative of the threshold function $\Theta(\cdot)$ in Equation (1) with a suitable differentiable approximation (Neftci et al., 2019). In a similar manner, for the latter issue, optimization over binary weights is conventionally done *via* the straight-through estimator (STE) (Bengio et al., 2013; Jang et al., 2021), which is covered in Section 2.5.3.

2.5.2. Surrogate gradient

As discussed in the previous subsections, the gradient $\nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{x}, \boldsymbol{y}}(\boldsymbol{w})$ is typically evaluated *via* SG methods. SG techniques approximate the Heaviside function $\Theta(\cdot)$ in Equation (1) when computing the gradient $\nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{x}, \boldsymbol{y}}(\boldsymbol{w})$. Specifically, the derivative $\Theta'(\cdot)$ is replaced with the derivative of a differentiable surrogate function, such as rectifier or sigmoid. For example, with a sigmoid surrogate, given by function $\sigma(x) = (1 + e^{-x})^{-1}$, we have $\partial s_{i,t} / \partial u_{i,t} \approx \sigma'(u_{i,t} - \vartheta)$, with derivative $\sigma'(x) = \sigma(x)(1 - \sigma(x))$. Using the loss decomposition in Equation (3), the partial derivative of the training loss $\mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})$ at each time instant t with respect to a synaptic weight w_{ij} can be accordingly approximated as

$$\frac{\partial \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} \approx \underbrace{\sum_{m \in \mathcal{Y}} \frac{\partial L(y_{m,t}, f_{m,t})}{\partial s_{i,t}}}_{e_{i,t}} \cdot \underbrace{\frac{\partial s_{i,t}}{u_{i,t}}}_{\sigma'(u_{i,t} - \vartheta)} \cdot \underbrace{\frac{\partial u_{i,t}}{\partial w_{ij}}}_{\alpha_t * s_{j,t}}, \tag{6}$$

where the first term $e_{i,t}$ is the derivative of the loss at time t with respect to the output $s_{i,t}$ of post-synaptic neuron i at time t; and the third term can be directly computed from Equation (2) as the filtered pre-synaptic trace of neuron j. For simplicity of notation, we have defined $f_{m,t} := f_m(\boldsymbol{w}, \boldsymbol{x}^t)$ and omitted the explicit dependence of $s_{i,t}$ and $u_{i,t}$ on exogeneous inputs $\boldsymbol{x}^t$ and synaptic weights $\boldsymbol{w}$. The second term is the source of the approximation, as the derivative of the threshold function $\Theta'(\cdot)$ from Equation (1), which is zero almost everywhere, is replaced using the derivative of the sigmoid function.

At every time instant $t = 1, \dots, T$, using Equation (6), the online update is obtained *via* stochastic gradient descent (SGD) as

$$w_{ij,t+1} \leftarrow w_{ij,t} - \eta \cdot \frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{B}} \frac{\partial \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w}_t)}{\partial w_{ij}}, \tag{7}$$

where $\eta > 0$ is a learning rate, and $\mathcal{B} \subseteq \mathcal{D}$ is a mini-batch of examples $(\boldsymbol{x}, \boldsymbol{y})$ from the training dataset. Note that the sequential implementation of the update (Equation 7) over time t requires running a number of copies of the SNN model equal to the size of the mini-batch $\mathcal{B}$. In fact, each input $\boldsymbol{x}$, with $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{B}$, generally causes the spiking neurons to follow distinct trajectories in the space of the membrane potentials. Henceforth, when referring to online learning rules, we will implicitly assume that parallel executions of the SNN are possible when the mini-batch size is larger than 1.

The weight update in the direction of the negative gradients in Equation (7) implements a standard *three-factor* rule. Three-factor rules generalize two-factor Hebbian updates such as STDP (Gerstner et al., 2018), and can be implemented on hardware with similar complexity (Zenke and Ganguli, 2018; Kaiser et al., 2020; Stewart et al., 2020). In fact, the partial derivative (Equation 6) can be written as

$$\frac{\partial \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} = \underbrace{e_{i,t}}_{\text{error signal}} \cdot \underbrace{\sigma'(u_{i,t} - \vartheta)}_{\text{post}_{i,t}} \cdot \underbrace{\big(\alpha_t * s_{j,t}\big)}_{\text{pre}_{j,t}}, \tag{8}$$

where we distinguish three terms. The first is the per-neuron error signal $e_{i,t}$, which can be in principle computed *via* backpropagation through time (Huh and Sejnowski, 2018). In practice, this term is approximated, e.g. *via* local signals (Bellec et al., 2020), or *via* random projections (Kaiser et al., 2020). The latter technique has previously been likened to the biological mechanisms behind short-term memory (Zou et al., 2022). We will discuss a specific implementation in Section 3.2. The second contribution is given by the local post-synaptic term $\sigma'(u_{i,t} - \vartheta)$, which measures the

current sensitivity to changes in the membrane potential of the neuron i. Finally, the last term is the local pre-synaptic trace $\alpha_t * s_{j,t}$ that depends on the activity of the neuron j.

2.5.3. Straight-through estimator

As mentioned in Section 2.5.1, optimization over binary weights can be carried out using STE (Bengio et al., 2013; Jang et al., 2021), which maintains latent, real-valued weights to compute gradients during training. Binary weights, obtained *via* quantization of the real-valued latent weights, are used as the next iterate. To elaborate, in addition to the binary weight vector $\boldsymbol{w} \in \{+1, -1\}^{|\boldsymbol{w}|}$, we define the real-valued weight vector $\boldsymbol{w}^{\mathrm{r}} \in \mathbb{R}^{|\boldsymbol{w}|\times 1}$. We use $|\boldsymbol{w}|$ to denote the size of vector $\boldsymbol{w}$. With STE, gradients are estimated by differentiating over the real-valued latent weights $\boldsymbol{w}^{\mathrm{r}}$, instead of discrete binary weights $\boldsymbol{w}$, to compute the gradient $\nabla_{\boldsymbol{w}^{\mathrm{r}}} \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w}^{\mathrm{r}})|_{\boldsymbol{w}^{\mathrm{r}}=\boldsymbol{w}}$. The technique can be naturally combined with the SG method, detailed in Section 2.5.2, to obtain the gradients with respect to the real-valued latent weights.

The training algorithm proceeds iteratively by selecting a mini-batch $\mathcal{B}$ of examples $(\boldsymbol{x}, \boldsymbol{y})$ from the training dataset $\mathcal{D}$ at each iteration as in Equation (7). Accordingly, the real-valued latent weight vector $\boldsymbol{w}^{\mathrm{r}}$ is updated *via* online SGD as

$$w^{\mathrm{r}}_{ij,t+1} \leftarrow w^{\mathrm{r}}_{ij,t} - \eta \cdot \frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}} \frac{\partial \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w}^{\mathrm{r}}_t)}{\partial w^{\mathrm{r}}_{ij,t}}\bigg|_{w^{\mathrm{r}}_{ij,t}=w_{ij,t}}, \tag{9}$$

and the next iterate for the binary weights $\boldsymbol{w}$ is obtained by quantization as

$$w_{ij,t+1} = \mathrm{sign}\big(w^{\mathrm{r}}_{ij,t+1}\big), \tag{10}$$

where the sign function is defined as $\mathrm{sign}(x) = +1$ for $x \geq 0$ and $\mathrm{sign}(x) = -1$ for $x < 0$.

2.6. Bayesian offline learning

In this section, we describe the formulation of Bayesian offline learning, and then develop two Bayesian training algorithms for SNNs with real-valued and binary synaptic weights.

2.6.1. Information risk minimization

Bayesian learning formulates the training problem as the optimization of a probability distribution $q(\boldsymbol{w})$ in the space of synaptic weights, which is referred to as the *variational posterior*. To this end, the ERM problem (Equation 5) is replaced by the information risk minimization (IRM) problem

$$\min_{q(\boldsymbol{w})} \Big\{ \mathcal{F}\big(q(\boldsymbol{w})\big) = \mathbb{E}_{q(\boldsymbol{w})}\Big[\mathcal{L}_{\mathcal{D}}(\boldsymbol{w})\Big] + \rho \cdot \mathrm{KL}\big(q(\boldsymbol{w})||p(\boldsymbol{w})\big)\Big\}, \tag{11}$$

where $\rho > 0$ is a "temperature" constant, $p(\boldsymbol{w})$ is an arbitrary prior distribution over synaptic weights, and $\mathrm{KL}(\cdot||\cdot)$ is the Kullback-Leibler divergence

$$\mathrm{KL}(q(\boldsymbol{w})||p(\boldsymbol{w})) = \mathbb{E}_{q(\boldsymbol{w})}\left[\log \frac{q(\boldsymbol{w})}{p(\boldsymbol{w})}\right]. \tag{12}$$

The objective function in IRM problem (Equation 11) is known as (variational) free energy (Jose and Simeone, 2021).

The problem of minimizing the free energy in Equation (11) must strike a balance between fitting the data—i.e., minimizing the first term—and not deviating too much from the reference behavior defined by prior $p(\boldsymbol{w})$—i.e., keeping the second term small. Note that with $\rho = 0$, the IRM problem (Equation 11) reduces to the ERM problem (Equation 5) in the sense that the optimal solution of the IRM problem with $\rho = 0$ is a distribution concentrated at the solution of the ERM problem (assuming that the latter is unique). The KL divergence term in Equation (11) is hence essential to Bayesian learning, and it is formally justified as a regularizing penalty that accounts for epistemic uncertainty due to the presence of limited data in the context of PAC Bayes theory (Zhang, 2006). It can also be used as a model of bounded rationality accounting for the complexity of information processing (Jose and Simeone, 2021).

If no constraints are imposed on the variational posterior $q(\boldsymbol{w})$, the optimal solution of Equation (11) is given by the *Gibbs posterior*

$$q^{\star}(\boldsymbol{w}) = \frac{p(\boldsymbol{w}) \exp\big(-\mathcal{L}_{\mathcal{D}}(\boldsymbol{w})/\rho\big)}{\mathbb{E}_{p(\boldsymbol{w})}\Big[\exp\big(-\mathcal{L}_{\mathcal{D}}(\boldsymbol{w})/\rho\big)\Big]}. \tag{13}$$

Due to the intractability of the normalizing constant in Equation (13), we adopt a mean-field variational inference (VI) approximation that limits the optimization domain for problem (Equation 11) to a class of factorized distributions (see, e.g., Angelino et al., 2016; Simeone, 2022). More specifically, we focus on Gaussian and Bernoulli variational approximations, targeting SNN models with real-valued and binary synaptic weights, respectively, which are detailed in the rest of this section.

2.6.2. Gaussian mean-field variational inference

In this subsection, we derive a Gaussian mean-field VI algorithm that approximately solves the IRM problem (Equation 11) by assuming variational posteriors of the form $q(\boldsymbol{w}) =$

Input: dataset $\mathcal{D}$, learning rate η, temperature parameter ρ, prior $(\boldsymbol{m}_0, \boldsymbol{p}_0)$
Output: learned parameters pair $(\boldsymbol{m}, \boldsymbol{p})$
initialize parameters $(\boldsymbol{m}_1, \boldsymbol{p}_1)$
repeat
  select mini-batch $\mathcal{B} \subseteq \mathcal{D}$
  for each time-step $t = 1, \dots, T$ **do**
    sample weights $\boldsymbol{w}$ as $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_t, \boldsymbol{P}_t^{-1})$.
    for each $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{B}$ **do**
      compute the gradient $\nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})$ locally at each synapse using SG (see Section 2.5.2).
    end for
    update the mean and precision parameters $(m_{ij,t}, p_{ij,t})$ for all synapses $(i,j) \in \mathcal{E}$ as

$$p_{ij,t+1} \leftarrow (1 - \eta\rho) \cdot p_{ij,t} + \eta \cdot \left[\frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}} \left(\frac{\partial \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} \right)^2 + \rho \cdot p_{ij,0} \right]$$

$$m_{ij,t+1} \leftarrow m_{ij,t} - \eta \cdot p_{ij,t+1}^{-1} \cdot \left[\frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}} \frac{\partial \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} - \rho \cdot p_{ij,0} \cdot \left(m_{ij,0} - m_{ij,t} \right) \right].$$

  end for
  set $(\boldsymbol{m}_1, \boldsymbol{p}_1) = (\boldsymbol{m}_T, \boldsymbol{p}_T)$
until convergence

Algorithm 1. Bayesian offline learning with real-valued synapses.

$\mathcal{N}(\boldsymbol{w}|\boldsymbol{m}, \boldsymbol{P}^{-1})$, where $\boldsymbol{m}$ is a mean vector and $\boldsymbol{P}$ is a precision diagonal matrix with positive vector $\boldsymbol{p}$ on the main diagonal. For the $|\boldsymbol{w}| \times 1$ weight vector $\boldsymbol{w}$, the distribution of the parameters $\boldsymbol{w}$ is defined by the $|\boldsymbol{w}| \times 1$ mean vector $\boldsymbol{m}$ and $|\boldsymbol{w}| \times 1$ precision vector $\boldsymbol{p} = \{p_{ij}\}_{(i,j)\in\mathcal{E}}$ with $p_{ij} > 0$ for all $(i,j) \in \mathcal{E}$. This variational model is well suited for real-valued synapses, which can be practically realized to the fixed precision allowed by the hardware implementation (Davies et al., 2018). We choose the prior $p(\boldsymbol{w})$ as the Gaussian distribution $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_0, \boldsymbol{P}_0^{-1})$ with mean $\boldsymbol{m}_0$ and precision matrix $\boldsymbol{P}_0$ with positive diagonal vector $\boldsymbol{p}_0$.

We tackle the IRM problem (Equation 11) with respect to the so-called *variational parameters* $(\boldsymbol{m}, \boldsymbol{p})$ of the Gaussian variational posterior $q(\boldsymbol{w})$ *via* the Bayesian learning rule (Khan and Rue, 2021). The Bayesian learning rule is derived by applying *natural gradient descent* to the variational free energy $\mathcal{F}(q(\boldsymbol{w}))$ in Equation (11). The derivation leverages the fact that the distribution $q(\boldsymbol{w})$ is an exponential-family distribution with natural parameters $\boldsymbol{\lambda} = (\boldsymbol{P}\boldsymbol{m}, -1/2\boldsymbol{P})$, sufficient statistics $\mathbf{T} = (\boldsymbol{w}, \boldsymbol{w}\boldsymbol{w}^T)$ and mean parameters $\boldsymbol{\mu} = (\boldsymbol{m}, \boldsymbol{P}^{-1} + \boldsymbol{m}\boldsymbol{m}^T)$. Updates to the mean $\boldsymbol{m}_t$ and precision $\boldsymbol{p}_t$ parameters at iteration t can be obtained as Osawa et al. (2019) and Khan and Rue (2021).

$$p_{ij,t+1} \leftarrow (1 - \eta\rho) \cdot p_{ij,t} + \eta \cdot \mathbb{E}_{q_t(\boldsymbol{w})} \left[\frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}} \left(\frac{\partial \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} \right)^2 + \rho \cdot p_{ij,0} \right] \tag{14}$$

$$m_{ij,t+1} \leftarrow m_{ij,t} - \eta \cdot p_{ij,t+1}^{-1} \cdot \mathbb{E}_{q_t(\boldsymbol{w})} \left[\frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}} \frac{\partial \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} - \rho \cdot p_{ij,0} \cdot \left(m_{ij,0} - m_{ij,t} \right) \right] \tag{15}$$

where $\eta > 0$ is a learning rate; $\mathcal{B} \subseteq \mathcal{D}$ is a mini-batch of examples $(\boldsymbol{x}, \boldsymbol{y})$ from the training dataset; and $q_t(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_t, \boldsymbol{P}_t^{-1})$ is the variational posterior at iteration t with $\boldsymbol{m}_t$ and $\boldsymbol{p}_t$.

In practice, the updates (Equations 14, 15) are estimated by evaluating the expectation over distribution $q_t(\boldsymbol{w})$ *via* one or more randomly drawn samples $\boldsymbol{w} \sim q_t(\boldsymbol{w})$. Furthermore, the gradients $\nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})$ can be approximated using the online SG method described in Section 2.5.2. The overall training algorithm proceeds iteratively by selecting a mini-batch $\mathcal{B} \subseteq \mathcal{D}$ of examples $(\boldsymbol{x}, \boldsymbol{y})$ from the training dataset at each iteration, and is summarized in Algorithm 1. Note that, as mentioned in Section 2.5.2, the implementation of a rule operating with mini-batches requires running $|\mathcal{B}|$ SNN models in parallel, where $|\mathcal{B}|$ is the cardinality of the mini-batch. When this is not possible, the rule can be applied with mini-batches of size $|\mathcal{B}| = 1$.

2.6.3. Bernoulli mean-field variational inference

In this subsection, we turn to the case of binary synaptic weights $w_{ij} \in \{+1, -1\}$. For this setting, we adopt the variational posterior $q(\boldsymbol{w}) = \text{Bern}(\boldsymbol{w}|\boldsymbol{p})$, with

$$q(\boldsymbol{w}) = \prod_{i \in \mathcal{N}} \prod_{j \in \mathcal{P}_i} p_{ij}^{\frac{1+w_{ij}}{2}} (1 - p_{ij})^{\frac{1-w_{ij}}{2}}, \tag{16}$$

where the $|\boldsymbol{w}| \times 1$ vector $\boldsymbol{p} = \{\{p_{ij}\}_{j\in\mathcal{P}_i}\}_{i\in\mathcal{N}}$ defines the variational posterior, with p_{ij} being the probability that synaptic weights w_{ij} equals $+1$.

The variational posterior (Equation 16) can be reparameterized in terms of the mean parameters $\boldsymbol{\mu} = \{\{\mu_{ij}\}_{j\in\mathcal{P}_i}\}_{i\in\mathcal{N}}$ as

$$q(\boldsymbol{w}) = \text{Bern}\left(\boldsymbol{w} \middle| \frac{\boldsymbol{\mu} + \mathbf{1}}{2}\right) \tag{17}$$

by setting $p_{ij} = (\mu_{ij} + 1)/2$, where $\mathbf{1}$ is the all-ones vector. It can also be expressed in terms of the logits, or natural parameters, $\boldsymbol{w}^{\text{r}} = \{\{w_{ij}^{\text{r}}\}_{j\in\mathcal{P}_i}\}_{i\in\mathcal{N}}$ as $q(\boldsymbol{w}) = \text{Bern}(\boldsymbol{w}|\sigma(2\boldsymbol{w}^{\text{r}}))$ by setting

$$w_{ij}^{\text{r}} = \frac{1}{2} \log \left(\frac{p_{ij}}{1 - p_{ij}} \right) = \frac{1}{2} \log \left(\frac{1 + \mu_{ij}}{1 - \mu_{ij}} \right), \tag{18}$$

Input: dataset $\mathcal{D}$, learning rate η, temperature parameter ρ, GS trick parameter τ, logits $\boldsymbol{w}_0^{\mathrm{r}}$ of prior distribution
Output: learned real-valued weights $\boldsymbol{w}^{\mathrm{r}}$
initialize real-valued weights $\boldsymbol{w}_1^{\mathrm{r}}$
repeat
  select mini-batch $\mathcal{B} \subseteq \mathcal{D}$
  for each time-step $t = 1, \dots, T$ **do**
    sample relaxed binary weights as

$$w_{ij} = \tanh\left(\frac{w_{ij,t}^{\mathrm{r}} + \delta_{ij}}{\tau}\right),$$

    with $\delta_{ij} = \frac{1}{2}\log\frac{\epsilon_{ij}}{1-\epsilon_{ij}}$ and $\epsilon_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0,1)$ for all $(i,j) \in \mathcal{E}$.
    for each $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{B}$ **do**
      compute the gradient $\nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{x}^t, \boldsymbol{y}_t}(\boldsymbol{w})$ locally at each synapse using SG (see Section 2.5.2).
    end for
    update the real-valued weights $w_{ij,t}^{\mathrm{r}}$ for all synapses $(i,j) \in \mathcal{E}$ as

$$w_{ij,t+1}^{\mathrm{r}} \leftarrow (1-\eta\rho)\cdot w_{ij,t}^{\mathrm{r}} - \eta \cdot \Big[\frac{1-w_{ij}^2}{\tau\left(1-\tanh^2(w_{ij,t}^{\mathrm{r}})\right)} \cdot \frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}} \frac{\partial \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} - \rho \cdot w_{ij,0}^{\mathrm{r}}\Big].$$

  end for
  set $\boldsymbol{w}_1^{\mathrm{r}} = \boldsymbol{w}_T^{\mathrm{r}}$
until convergence

Algorithm 2. Bayesian offline learning with binary-valued synapses.

for all $(i,j) \in \mathcal{E}$. The notation $\boldsymbol{w}^{\mathrm{r}}$ has been introduced to suggest a relationship with the STE method described in Section 2.5.3, as defined below. We assume that the prior distribution $p(\boldsymbol{w})$ also follows a mean-field Bernoulli distribution of the form $p(\boldsymbol{w}) = \mathrm{Bern}(\boldsymbol{w}|\sigma(2\boldsymbol{w}_0^{\mathrm{r}}))$, for some vector of $\boldsymbol{w}_0^{\mathrm{r}}$ logits. For example, setting $\boldsymbol{w}_0^{\mathrm{r}} = \boldsymbol{0}$ indicates that the binary weights are equally likely to be either $+1$ or -1 a priori.

In a manner similar to the case of Gaussian VI developed in the previous subsection, we apply natural gradient descent to minimize the variational free energy in Equation (11) with respect to the variational parameters $\boldsymbol{w}^{\mathrm{r}}$ defining the variational posterior $q(\boldsymbol{w})$. Following Meng et al. (2020), and applying the online SGD rule detailed in Section 2.5.2, this yields the update

$$w_{ij,t+1}^{\mathrm{r}} \leftarrow (1-\eta\rho)\cdot w_{ij,t}^{\mathrm{r}} - \eta \cdot \left[\frac{\partial}{\partial \mu_{ij,t}} \mathbb{E}_{q_t(\boldsymbol{w})}\left[\frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}} \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})\right] - \rho \cdot w_{ij,0}^{\mathrm{r}}\right], \quad (19)$$

where $\eta > 0$ is a learning rate and $q_t(\boldsymbol{w})$ the variational posterior with $\boldsymbol{w}_t^{\mathrm{r}}$ and $\boldsymbol{\mu}_t$ related through (Equation 18). Note that the gradient in Equation 19 is with respect to the mean parameters $\boldsymbol{\mu}_t$.

In order to estimate the gradient in Equation 19, we leverage the *reparameterization* trick *via* the *Gumbel-Softmax* (GS) distribution (Jang et al., 2016; Meng et al., 2020). Accordingly, we first obtain a sample $\boldsymbol{w}$ that is approximately distributed according to $q_t(\boldsymbol{w}) = \mathrm{Bern}\big(\boldsymbol{w}|\sigma(2\boldsymbol{w}_t^{\mathrm{r}})\big)$. This is done by drawing a vector $\boldsymbol{\delta} = \{\{\delta_{ij}\}_{j\in\mathcal{P}_i}\}_{i\in\mathcal{N}}$ of i.i.d. Gumbel variables, and computing

$$w_{ij} = \tanh\left(\frac{w_{ij,t}^{\mathrm{r}} + \delta_{ij}}{\tau}\right), \quad (20)$$

where $\tau > 0$ is a parameter. When τ in Equation (20) tends to zero, the $\tanh(\cdot)$ function tends to the $\mathrm{sign}(\cdot)$ function, and the vector $\boldsymbol{w}$ follows distribution $q_t(\boldsymbol{w})$ (Meng et al., 2020). To generate $\boldsymbol{\delta}$, one can set $\delta_{ij} = \frac{1}{2}\log\left(\frac{\epsilon_{ij}}{1-\epsilon_{ij}}\right)$, with $\epsilon_{ij} \sim \mathcal{U}(0,1)$ being i.i.d. samples.

With this sample, for each example $(\boldsymbol{x}, \boldsymbol{y})$, we then obtain an approximately unbiased estimate of the gradient in Equation (19) by using the following approximation

$$\begin{aligned}
&\frac{\partial}{\partial \mu_{ij,t}} \mathbb{E}_{q_t(\boldsymbol{w})}\left[\mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})\right] \\
&\overset{(a)}{\approx} \mathbb{E}_{p(\boldsymbol{\delta})}\left[\frac{\partial \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})}{\partial \mu_{ij,t}}\bigg|_{\boldsymbol{w}=\tanh\left(\frac{w_t^{\mathrm{r}}+\delta}{\tau}\right)}\right] \\
&\overset{(b)}{=} \mathbb{E}_{p(\boldsymbol{\delta})}\left[\frac{\partial \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} \cdot \frac{\partial}{\partial \mu_{ij,t}} \tanh\left(\frac{w_{ij,t}^{\mathrm{r}} + \delta_{ij}}{\tau}\right)\right] \\
&= \mathbb{E}_{p(\boldsymbol{\delta})}\left[\frac{\partial \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} \cdot \frac{1-w_{ij}^2}{\tau\left(1-\tanh^2(w_{ij,t}^{\mathrm{r}})\right)}\right], \quad (21)
\end{aligned}$$

where the approximate equality (a) is exact when $\tau \to 0$ and the equality (b) follows the chain rule. We note that the gradient $\nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})$ can be computed as detailed in Section 2.5.2.

As summarized in Algorithm 2, the resulting rule proceeds iteratively by selecting a mini-batch $\mathcal{B}$ of examples $(\boldsymbol{x}, \boldsymbol{y})$ from the training dataset $\mathcal{D}$ at each iteration. Using the samples w_{ij} from Equation (20), we obtain at every time-step t the estimate of the gradient (Equation 19) as

$$\frac{\partial}{\partial \mu_{ij,t}} \mathbb{E}_{q_t(\boldsymbol{w})}\left[\mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})\right] \approx \frac{1-w_{ij}^2}{\tau\left(1-\tanh^2(w_{ij,t}^{\mathrm{r}})\right)} \cdot \frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}} \frac{\partial \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} - \rho \cdot w_{ij,0}^{\mathrm{r}}. \quad (22)$$

This is unbiased when the limit $\tau \to 0$ holds.

2.7. Frequentist continual learning

We now consider a continual learning setting, in which the system is sequentially presented datasets $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \ldots$ corresponding to distinct, but related, learning tasks. Applying a frequentist approach, at every subsequent task k, the system minimizes a new objective based on dataset $\mathcal{D}^{(k)}$ in order to update the model parameter vector $\boldsymbol{w}$, where we have used superscript (k) to denote the quantities corresponding to the kth task. We first describe an algorithm based on coresets and regularization (Farquhar and Gal, 2019b). Then, we briefly review a recently proposed biologically inspired rule.

2.7.1. Regularization-based continual learning

In a similar manner to Equation (4), let us first define as

$$\mathcal{L}_{\mathcal{D}^{(k)}}(\boldsymbol{w}) = \frac{1}{|\mathcal{D}^{(k)}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}^{(k)}} \mathcal{L}_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{w}) \tag{23}$$

the training loss evaluated on dataset $\mathcal{D}^{(k)}$ for the kth task. A general formulation of the continual learning problem in a frequentist framework is then obtained as the minimum of the objective

$$\mathcal{L}_{\mathcal{D}^{(k)}}(\boldsymbol{w}) + \sum_{k'=1}^{k-1} \mathcal{L}_{\mathcal{C}^{(k')}}(\boldsymbol{w}) + \alpha \cdot R\big(\boldsymbol{w}, \{\boldsymbol{w}^{(k')}\}_{k'=1}^{k-1}\big), \tag{24}$$

where $\mathcal{L}_{\mathcal{C}^{(k')}}(\boldsymbol{w})$ is the training loss evaluated on a *coreset*, that is, a subset $\mathcal{C}^{(k')} \subset \mathcal{D}^{(k')}$ of examples randomly selected from a previous task $k' < k$ and maintained for use when future tasks are encountered; $\alpha \geq 0$ determines the strength of the regularization; and $R(\boldsymbol{w}, \{\boldsymbol{w}^{(k')}\}_{k'=1}^{k-1})$ is a regularization function aimed at preventing the current weights from differing too much from previously learned weights $\{\boldsymbol{w}^{(k')}\}_{k'=1}^{k-1}$, hence mitigating the problem of catastrophic forgetting (Parisi et al., 2019).

A popular choice for the regularization function, yielding the Elastic Weight Consolidation (EWC) method, proposes to estimate the relative importance of synapses for previous tasks *via* the Fisher information matrices (FIM) computed on datasets $k' < k$ (Kirkpatrick et al., 2017). This corresponds to the choice of the regularizer

$$R\big(\boldsymbol{w}, \{\boldsymbol{w}^{(k')}\}_{k'=1}^{k-1}\big) = \sum_{k'=1}^{k-1} (\boldsymbol{w} - \boldsymbol{w}^{(k')})^T F^{(k')}(\boldsymbol{w}^{(k')})(\boldsymbol{w} - \boldsymbol{w}^{(k')}), \tag{25}$$

where $F^{(k)}(\boldsymbol{w}) = \mathrm{diag}\big(\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}^{(k)}} (\nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{w}))^2\big)$ is an approximation of the FIM estimated on dataset $\mathcal{D}^{(k)}$. The square operation in vector $(\nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{w}))^2$ is evaluated element-wise. Intuitively, a larger value of an entry in the diagonal of the matrix $F^{(k)}(\boldsymbol{w})$ indicates that the corresponding entry of the vector $\boldsymbol{w}$ plays a significant role for the kth task.

2.7.2. Biologically inspired continual learning

The authors of Soures et al. (2021) introduce a biologically inspired, frequentist, continual learning rule for SNNs, which we briefly review here. The approach operates online in discrete time t, and implements the mechanisms described in Section 2.4. It considers a leaky integrate-and-fire (LIF) neuron model. The LIF is a special case of the SRM (Equations 1, 2) in which the synaptic response α implemented as the *alpha-function* spike response $\alpha_t = \exp(-t/\tau_{\mathrm{mem}}) - \exp(-t/\tau_{\mathrm{syn}})$ and the exponentially decaying feedback filter $\beta_t = -\exp(-t/\tau_{\mathrm{ref}})$ for $t \geq 1$ with some positive constants $\tau_{\mathrm{mem}}, \tau_{\mathrm{syn}}$, and τ_{ref}. This choice enables an autoregressive update of the membrane potential (Jang et al., 2020a; Kaiser et al., 2020).

A metaplasticity parameter ν_{ij} is introduced for each synapse $(i,j) \in \mathcal{E}$ that determines the degree to which the synapse is prone to change. This quantity is increased by a fixed step $\Delta\nu$ as

$$\nu_{ij,t+1} \leftarrow \nu_{ij,t} + \Delta\nu \tag{26}$$

when the pre- and post-synaptic neurons spiking rates, i.e., the spiking rate of neuron i and j, respectively, pass a predetermined threshold. Furthermore, each synapse $(i,j) \in \mathcal{E}$ maintains a reference weight w_{ij}^{ref} to mimic heterosynaptic plasticity by adjusting the weight updates to drive synaptic weights toward this resting state. It is updated over time as

$$w_{ij,t+1}^{\mathrm{ref}} \leftarrow w_{ij,t}^{\mathrm{ref}} + \kappa \cdot \Big(w_{ij,t} - w_{ij,t}^{\mathrm{ref}}\Big), \tag{27}$$

where $\kappa > 0$, and serves as a reference to implement heterosynaptic plasticity.

With these definitions, the update of each synaptic weight $\boldsymbol{w}$ is computed according to the online learning rule

$$w_{ij,t+1} \leftarrow w_{ij,t} - \exp\big(-|\nu_{ij} \cdot w_{ij,t}|\big) \Big(\eta \cdot e_{i,t} \cdot s_{j,t} \cdot \sigma'(u_{i,t} - \vartheta) + \gamma \cdot (w_{ij,t} - w_{ij,t}^{\mathrm{ref}}) \cdot s_{i,t}\Big), \tag{28}$$

where η and γ are respectively learning and decay rates, and $e_{i,t}$ is an error signal from neuron i (see Soures et al., 2021 for details). The rule (Equation 28) takes a form similar to that of three-factor rules (Equation 8), with the term $e_{i,t} \cdot s_{j,t} \cdot \sigma'(u_{i,t} - \vartheta)$ evaluating the product of error, post-synaptic, and pre-synaptic signals. The update (Equation 28) implements metaplasticity *via* the term $\exp\big(-|\nu_{ij} \cdot w_{ij,t}|\big)$ that decreases the magnitude of the updates during the training procedure for active synapses. It also accounts for heterosynaptic plasticity thanks to the term $(w_{ij,t} - w_{ij,t}^{\mathrm{ref}})$, which drives the updates toward learned "resting" weight $w_{ij,t}^{\mathrm{ref}}$ when the pre-synaptic neuron is active.

2.8. Bayesian continual learning

In this section, we generalize the Bayesian formulation seen in Section 2.6 from the offline setting to continual learning.

2.8.1. Bayesian continual learning

To allow the adaptation to task k without catastrophic forgetting, we consider the problem (Farquhar and Gal, 2019a).

$$\min_{q^{(k)}(\boldsymbol{w})} \mathcal{F}^{(k)}\left(q^{(k)}(\boldsymbol{w})\right) \tag{29}$$

of minimizing the free energy metric

$$\mathcal{F}^{(k)}\left(q^{(k)}(\boldsymbol{w})\right) = \mathbb{E}_{q^{(k)}(\boldsymbol{w})}\left[\mathcal{L}_{\mathcal{D}^{(k)}}(\boldsymbol{w}) + \sum_{k'=1}^{k-1} \mathcal{L}_{\mathcal{C}^{(k')}}(\boldsymbol{w})\right] + \rho \cdot \mathrm{KL}\left(q^{(k)}(\boldsymbol{w}) || q^{(k-1)}(\boldsymbol{w})\right), \tag{30}$$

which combines the IRM formulation (Equation 11) with the use of coresets. Minimizing the free energy objective (Equation 30) must strike a balance between fitting the new training data $\mathcal{D}^{(k)}$, as well as the coresets $\{\mathcal{C}^{(k')}\}_{k'=1}^{k-1}$ from the previous tasks, while not deviating too much from previously learned distribution $q^{(k-1)}(\boldsymbol{w})$. Comparing (Equation 30) with the free energy (Equation 11), we observe that the distribution $q^{(k-1)}(\boldsymbol{w})$ plays the role of prior for the current task k.

2.8.2. Continual gaussian mean-field variational inference

Similarly to the approach for offline learning described in Section 2.6, we first assume a Gaussian variational posterior $q(\boldsymbol{w})$, and address the problem (Equation 30) *via* natural gradient descent. To this end, we adopt the variational posterior $q^{(k)}(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}^{(k)}, (\boldsymbol{P}^{(k)})^{-1})$, with mean vector $\boldsymbol{m}^{(k)}$ and diagonal precision matrix $\boldsymbol{P}^{(k)}$ with positive diagonal vector $\boldsymbol{p}^{(k)}$ of size $|\boldsymbol{w}| \times 1$ for every task k. We choose the prior $p(\boldsymbol{w})$ for dataset $\mathcal{D}_1$ as the Gaussian distribution $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_0, \boldsymbol{P}_0^{-1})$ with positive diagonal vector $\boldsymbol{p}_0$ of size $|\boldsymbol{w}| \times 1$. Applying the Bayesian learning rule (Khan and Rue, 2021) as in Section 2.6.2, updates to the mean and precision parameters can be obtained *via* online SGD as

$$p_{ij,t+1}^{(k)} \leftarrow (1-\eta\rho)\cdot p_{ij,t}^{(k)} + \eta \cdot \mathbb{E}_{q_t^{(k)}(\boldsymbol{w})} \left[\frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}} \left(\frac{\partial \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}}\right)^2 + \rho \cdot p_{ij}^{(k-1)}\right] \tag{31}$$

$$m_{ij,t+1}^{(k)} \leftarrow m_{ij,t}^{(k)} - \eta \cdot (p_{ij,t+1}^{(k)})^{-1} \cdot \mathbb{E}_{q_t^{(k)}(\boldsymbol{w})} \left[\frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}} \frac{\partial \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} - \rho \cdot p_{ij}^{(k-1)} \cdot \left(m_{ij}^{(k-1)} - m_{ij,t}^{(k)}\right)\right], \tag{32}$$

where mini-batch $\mathcal{B}$ is now selected at random from both dataset $\mathcal{D}^{(k)}$ and coresets from previous tasks, i.e., $\mathcal{B} \subseteq \mathcal{D}^{(k)} \cup_{k'=1}^{k} \mathcal{C}^{(k')}$. The rule can be directly derived by following the steps detailed in Section 2.6.2, and using for prior at every task k the mean $\boldsymbol{m}^{(k-1)}$ and precision $\boldsymbol{P}^{(k-1)}$ obtained at the end of training on the previous task.

2.8.3. On the biological plausibility of the Bayesian learning rule

The continual learning rule (Equations 31, 32) exhibits some of the mechanisms thought to enable memory retention in biological brains as described in Section 2.4. In particular, synaptic consolidation and metaplasticity for each synapse $(i,j) \in \mathcal{E}$ are modeled by the precision p_{ij}. In fact, a larger precision $p_{ij,t+1}$ effectively reduces the step size $1/p_{ij,t+1}$ of the synaptic weight update (Equation 32). This is a similar mechanism to the metaplasticity parameter $v_{ij,t}$ introduced in the rule (Equation 28). Furthermore, by Equation 31, the precision p_{ij} is increased to a degree that depends on the relevance of the synapse $(i,j) \in \mathcal{E}$ as measured by the estimated FIM $(\partial \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})/\partial w_{ij})^2$ for the current mini-batch $\mathcal{B}$ of examples.

Heterosynaptic plasticity, which drives the updates toward previously learned and resting states to prevent catastrophic forgetting, is obtained from first principles *via* the IRM formulation with a KL regularization term, rather than from the addition of the reference weight $\boldsymbol{w}_{\mathrm{ref}}$ in the previous work (Soures et al., 2021). This mechanism drives the updates of the precision $p_{ij,t+1}^{(k)}$ and mean parameter $m_{ij,t+1}^{(k)}$ toward the corresponding parameters of the variational posterior obtained at the previous task, namely $p_{ij}^{(k-1)}$ and $m_{ij}^{(k-1)}$.

Finally, the use of coresets implements a form of replay, or reactivation, in biological brains (Buhry et al., 2011).

2.8.4. Continual bernoulli mean-field variational inference

We now consider continual learning with a Bernoulli mean-field variational posterior, and force the synaptic weight w_{ij} to be binary, i.e., $w_{ij} \in \{+1, -1\}$. Following Equation (16), the posterior is of the form $q^{(k)}(\boldsymbol{w}) = \mathrm{Bern}\left(\boldsymbol{w}|\boldsymbol{p}^{(k)}\right)$.

We leverage the Gumbel-softmax trick, and use the reparametrization in terms of the natural parameters at task k

$$w_{ij}^{\mathrm{r},(k)} = \frac{1}{2}\log\left(\frac{1+\mu_{ij}^{(k)}}{1-\mu_{ij}^{(k)}}\right). \tag{33}$$

We then apply the Bayesian learning rule, and, following the results obtained in the offline learning case of Section 2.6.3, we obtain the learning rule at task k as

$$w_{ij,t+1}^{\mathrm{r},(k)} \leftarrow (1-\eta\rho)\cdot w_{ij,t}^{\mathrm{r},(k)} - \eta \cdot \left[\mathbb{E}_{p(\boldsymbol{\delta})}\left[\frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}} \frac{\partial \mathcal{L}_{\boldsymbol{x}^t,\boldsymbol{y}_t}(\boldsymbol{w})}{\partial w_{ij}} \cdot \frac{1-w_{ij}^2}{\tau\left(1-\tanh^2\left(w_{ij,t}^{\mathrm{r},(k)}\right)\right)}\right] - \rho \cdot w_{ij}^{\mathrm{r},(k-1)}\right], \tag{34}$$

where we denote as $\boldsymbol{w}^{\mathrm{r},(k-1)}$ the logits obtained at the end of the previous task $k-1$, and mini-batch $\mathcal{B}$ is selected at random as $\mathcal{B} \subseteq \mathcal{D}^{(k)} \cup_{k'=1}^{k} \mathcal{C}^{(k')}$.

3. Experiments

In this section, we compare the performance of frequentist and Bayesian learning schemes in a variety of experiments, using both synthetic and real neuromorphic datasets. All experiments consist of classification tasks with C classes. In each such task, we are given a dataset $\mathcal{D}'$ consisting of spiking inputs $\boldsymbol{x}$ and label $c_{\boldsymbol{x}} \in \{0, 1, \dots, C-1\}$. Each pair $(\boldsymbol{x}, c_{\boldsymbol{x}})$ is converted into a pair of spiking signals $(\boldsymbol{x}, \boldsymbol{y})$ to obtain the dataset $\mathcal{D}$. To do this, the target signals $\boldsymbol{y}$ are such that each sample $\boldsymbol{y}_t$ is the $C \times 1$ one-hot encoding vector of label $c_{\boldsymbol{x}}$ for all time-steps $t = 1, \dots, T$.

3.1. Datasets

3.1.1. Two-moons dataset

We first consider an offline 2D binary classification task on the two-moons dataset (Scikit-Learn library, 2020). Training is done on 200 examples per class with added noise with standard deviation $\sigma = 0.1$ as proposed in Meng et al. (2020) for 100 epochs. The inputs $\boldsymbol{x}$ are obtained *via* population encoding following (Jang et al., 2020a) over $T = 100$ time-steps and *via* 10 neurons.

3.1.2. DVS-gestures

Next, we consider a real-world neuromorphic dataset for offline classification, namely the DVS-Gestures dataset (Amir et al., 2017). The dataset comprises 11 classes of hand movements, captured with a DVS camera. Movements are recorded from 30 different persons under 5 lighting conditions. To evaluate the calibration of Bayesian learning algorithms, we obtain in- and out-of-distribution dataset $\mathcal{D}_{\text{id}}$ and $\mathcal{D}_{\text{ood}}$ by partitioning the dataset by users and lighting conditions. We selected the first 15 users for the training set, while the remaining 15 users are used for testing. The first 4 lighting conditions are used for in-distribution testing; and the one left out from the training set is used for out-of-distribution testing. Images are of size $128 \times 128 \times 2$, and preprocessed following (Amir et al., 2017) to yield inputs of size $32 \times 32 \times 2$, with sequences of length 500 ms for training and 1,500 ms for testing, with a sampling rate of 10 ms.

3.1.3. Split-MNIST and MNIST-DVS

For continual learning, we first conduct experiments on the 5-ways split-MNIST dataset (Farquhar and Gal, 2019a; Soures et al., 2021). Examples from the MNIST dataset, of size 28×28 pixels, are hence rate-encoded over $T = 50$ time-steps (Jang et al., 2020a), and training examples drawn from subsets of two classes are successively presented to the system for training. The order of the pairs is selected as $\{0, 1\}$, then $\{2, 3\}$, and so on. We restrict here our study to rate encoding, although the proposed methods are applicable to any spike encoding scheme. In a similar way, we also consider a neuromorphic continual learning setting based on the neuromorphic counterpart to the MNIST dataset, namely, the MNIST-DVS dataset (Serrano-Gotarredona and Linares-Barranco, 2015). Following the preprocessing adopted in Skatchkovsky et al. (2020a,b, 2021), we cropped images spatially to 26×26 pixels, capturing the active part of the image, and temporally to a duration of 2 s. For each pixel, positive and negative events are encoded as (unsigned) spikes over two different input channels, and the input $\boldsymbol{x}$ is of size 1,352 spiking signals. Uniform downsampling over time is then carried out to restrict the length to $T = 80$ time-samples. The training dataset is composed of 900 examples per class, and the test dataset contains 100 examples per class. For continual learning, classes are presented to the network in pairs by following the lexicographical order, i.e., the classes $\{0, 1\}$ are presented first, then $\{2, 3\}$, and so on.

3.2. Implementation

All schemes are implemented using the SG technique DECOLLE (Kaiser et al., 2020) to compute the gradients. In DECOLLE, the SNN is organized into L layers, with the first $L-1$ layers encompassing the hidden neurons in set $\mathcal{H}$, and the Lth layer containing the read-out neurons in set $\mathcal{Y}$. To evaluate the partial derivative (Equation 8), we need to specify how to compute error signals $e_{i,t}$ for each neuron $i \in \mathcal{N}$. To this end, at each time t, the spiking outputs $\boldsymbol{s}_t^{(l)}$ of each layer $l \in \{1, \dots, L\}$ are used to compute local per-layer errors

$$L(y_{m,t}, \boldsymbol{s}_t^{(l)}) = -y_{m,t} \cdot \log\Big(\text{Softmax}_m\big(\boldsymbol{B}^{(l)} \boldsymbol{s}_t^{(l)}\big)\Big), \tag{35}$$

where $\boldsymbol{B}^{(l)} \in \mathbb{R}^{C \times |l|}$ are random, fixed weights, $|l|$ is the cardinality of layer l, and $\text{Softmax}_m(\boldsymbol{a}) = \exp(a_m) / \sum_{1 \le n \le C} \exp(a_n)$ is the ith element of the softmax of vector $\boldsymbol{a}$ with length C. The local losses (Equation 35) at every time-step t are then used to compute the error signals $e_{i,t}$ in Equation (8) for every neuron $i \in l$ as

$$e_{i,t} = \sum_{m \in \mathcal{Y}} \frac{\partial L(y_{m,t}, \boldsymbol{s}_t^{(l)})}{\partial s_{i,t}}. \tag{36}$$

While the algorithms introduced in this work are valid for any SNN architecture as highlighted in Figure 1, DECOLLE is limited to feedforward layered architectures, which we hence adopt for our experiments (Kaiser et al., 2020). Furthermore, we consider autoregressive filters for the spike responses of synapses α_t and somas β_t in the membrane potential (Equation 2), as discussed in Section 2.1.1.

Results have been obtained by using Intel's Lava software framework (Intel Corporation, 2021), under Loihi-compatible

fixed-point precision (Davies et al., 2018)[1]. We use as benchmark the frequentist algorithms detailed in Sections 2.5, 2.7, for which gradients are as described in the previous paragraph. For Bayesian learning with real-valued (fixed-precision) synapses, we set the threshold of each neuron as $\vartheta = 64$; while for binary synapses the threshold ϑ is selected as the square-root of the fan-in of the corresponding layer.

Implementation of the proposed methods on hardware is left for future work. While Loihi supports the injection of Gaussian noise to the membrane potential of the neurons (Davies et al., 2018), it does not provide mechanisms for the sampling of the model parameters. In contrast, recent work (Dalgaty et al., 2021) has proposed leveraging the inherent noise of nanoscale devices in order to implement Bayesian inference.

3.3. Performance measures

Apart from the test accuracy, performance metrics include calibration measures, namely reliability diagrams and expected calibration error (ECE), which are described next. We note that, as the hardware implementation of Bayesian SNNs is currently an open problem (see Section 3.2), we are unable to provide measurements in terms of energy expenditure and computation time. As a general remark, as discussed in Section 2.2, Bayesian learning requires a larger memory to store all samples for the weights distribution to be used for inference using a committee machine implementation, while an ensemble predictor implementation increases inference latency.

3.3.1. Confidence levels

For frequentist learning, predictive probabilities are obtained from a single pass through the network with parameter vector $\boldsymbol{w}$ as

$$p(c_{\boldsymbol{x}} = k \mid \boldsymbol{x}, \boldsymbol{w}) = \frac{1}{T}\sum_{t=1}^{T} \text{Softmax}_k\big(\boldsymbol{B}^{(L)} f(\boldsymbol{w}, \boldsymbol{x}^t)\big), \tag{37}$$

where $f(\boldsymbol{w}, \boldsymbol{x}^t)$ is the output of read-out layer L for weights $\boldsymbol{w}$, as detailed in the previous subsection.

In contrast, for Bayesian learning, decisions and confidence levels are obtained by drawing N_S samples $\{\boldsymbol{w}_s\}_{s=1}^{N_S}$ from the distribution $q(\boldsymbol{w})$, and by averaging the read-out outputs of the model to obtain the probability assigned to each class as

$$p\big(c_{\boldsymbol{x}} = k \mid \boldsymbol{x}, \{\boldsymbol{w}_s\}_{s=1}^{N_S}\big) = \frac{1}{N_S}\frac{1}{T}\sum_{s=1}^{N_S}\sum_{t=1}^{T} \text{Softmax}_k\big(\boldsymbol{B}^{(L)} f(\boldsymbol{w}_s, \boldsymbol{x}^t)\big). \tag{38}$$

1 Our implementation can be found at: https://github.com/kclip/bayesian-snn.

Unless mentioned otherwise, the predictions (Equation 38) are obtained by using the committee machine approach, and hence the weights $\{\boldsymbol{w}_s\}_{s=1}^{N_S}$ are kept fixed for all test inputs $\boldsymbol{x}$ (see Section 2.2). All results presented are averaged over three repetitions of the experiments and 10 draws from the posterior distribution $q(\boldsymbol{w})$, i.e., we set $N_S = 10$ in all experiments.

For Bayesian learning, the hard prediction of the model is hence obtained as

$$c^*_{\boldsymbol{x}} = \underset{1\le k\le C}{\text{argmax}}\, p\big(c_{\boldsymbol{x}} = k \mid \boldsymbol{x}, \{\boldsymbol{w}_s\}_{s=1}^{N_S}\big), \tag{39}$$

corresponding to the predictive probability

$$p\big(c^*_{\boldsymbol{x}} \mid \boldsymbol{x}, \{\boldsymbol{w}_s\}_{s=1}^{N_S}\big) = \max_{1\le k\le C} p\big(c_{\boldsymbol{x}} = k \mid \boldsymbol{x}, \{\boldsymbol{w}_s\}_{s=1}^{N_S}\big). \tag{40}$$

The probability (Equation 40) can be interpreted as the confidence of the model in making decisions (Equation 39).

A model is considered to be well calibrated when there is no mismatch between confidence level $p\big(c^*_{\boldsymbol{x}} \mid \boldsymbol{x}, \{\boldsymbol{w}_s\}_{s=1}^{N_S}\big)$ and the actual probability for the model to correctly classify input $\boldsymbol{x}$ (Guo et al., 2017). Definitions (Equations 39, 40) can be straightforwardly adapted to the frequentist case by replacing the average over draws $\{\boldsymbol{w}_s\}_{s=1}^{N_S}$ with a single parameter vector $\boldsymbol{w}$.

3.3.2. Reliability diagrams

Reliability diagrams plot the actual probability of correct detection as a function of the confidence level (Equation 40). This is done by first dividing the probability interval $[0, 1]$ into M intervals of equal length, and then evaluating the average accuracy and confidence for all inputs $\boldsymbol{x}$ in each mth interval $(\frac{m-1}{M}, \frac{m}{M}]$, also referred to as mth *bin*. We denote as $\mathcal{B}_m$ the subset of examples whose associated confidence level $p\big(c^*_{\boldsymbol{x}} \mid \boldsymbol{x}, \{\boldsymbol{w}_s\}_{s=1}^{N_S}\big)$ lies within bin m, that is, Guo et al. (2017)

$$\mathcal{B}_m = \left\{\boldsymbol{x} \in \mathcal{D} \;\middle|\; p\big(c^*_{\boldsymbol{x}} \mid \boldsymbol{x}, \{\boldsymbol{w}_s\}_{s=1}^{N_S}\big) \in \left(\frac{m-1}{M}, \frac{m}{M}\right]\right\}. \tag{41}$$

The average empirical accuracy of the predictor within bin m is defined as

$$\text{acc}(\mathcal{B}_m) = \frac{1}{|\mathcal{B}_m|}\sum_{\boldsymbol{x}\in\mathcal{B}_m} \mathbf{1}(c^*_{\boldsymbol{x}} = c_{\boldsymbol{x}}), \tag{42}$$

with $\mathbf{1}(\cdot)$ being the indicator function; while the average empirical confidence of the predictor for bin m is defined as

$$\text{conf}(\mathcal{B}_m) = \frac{1}{|\mathcal{B}_m|}\sum_{\boldsymbol{x}\in\mathcal{B}_m} p\big(c^*_{\boldsymbol{x}} \mid \boldsymbol{x}, \{\boldsymbol{w}_s\}_{s=1}^{N_S}\big). \tag{43}$$

Reliability diagrams plot the per-bin accuracy $\text{acc}(\mathcal{B}_m)$ vs. confidence level $\text{conf}(\mathcal{B}_m)$ across all bins m. A model is said to be perfectly calibrated when, for all bins m, the

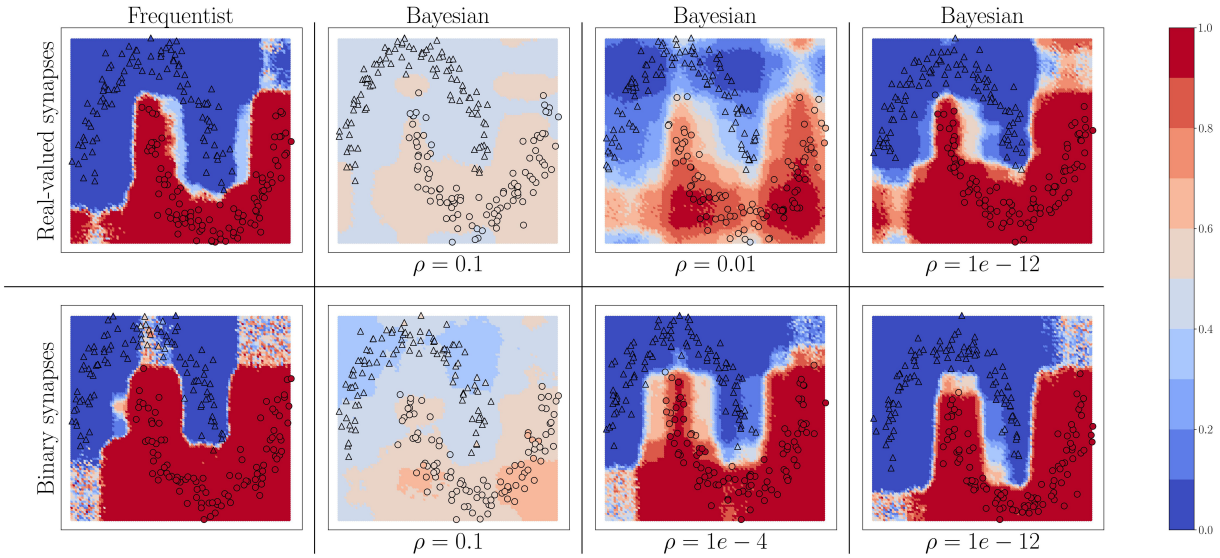

FIGURE 4
Predictive probabilities (Equation 40) evaluated on the two-moons dataset after training with different values of the temperature ρ in Equation (11) for Bayesian learning. **Top row:** Real-valued synapses; **Bottom row:** Binary synapses.

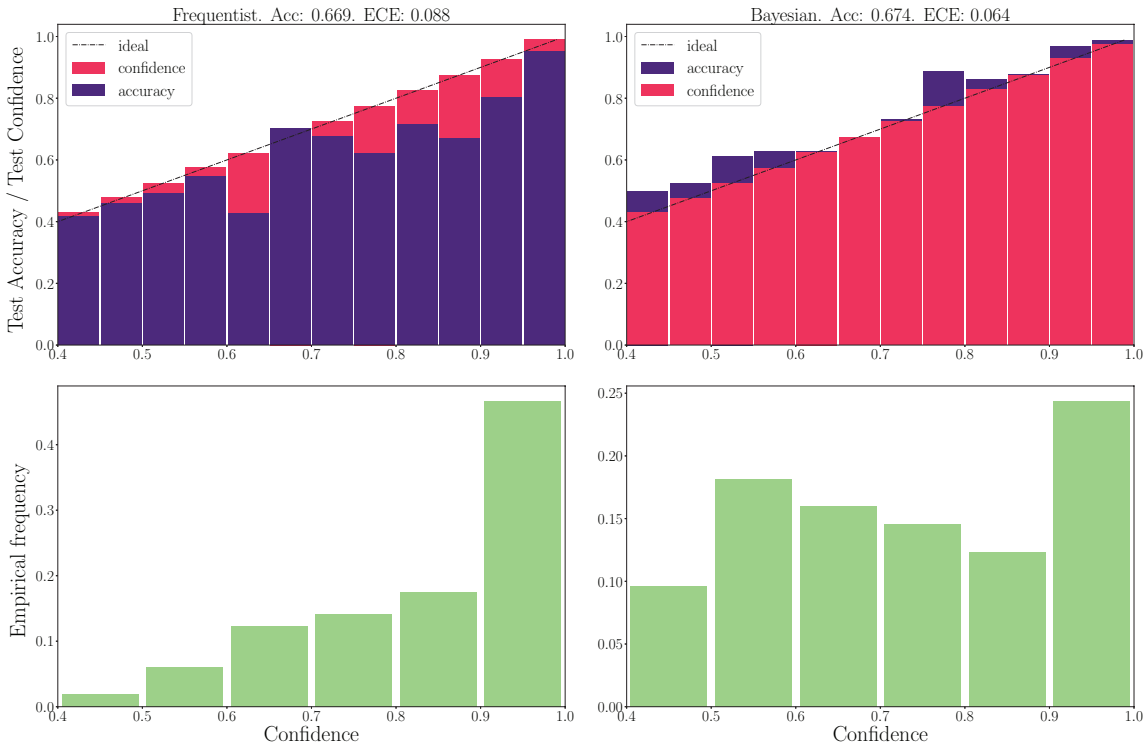

FIGURE 5
Top: Reliability diagrams (for in-distribution data) with real-valued synapses for the DVS-Gestures dataset. **Bottom**: Corresponding empirical confidence histograms for in-distribution data.

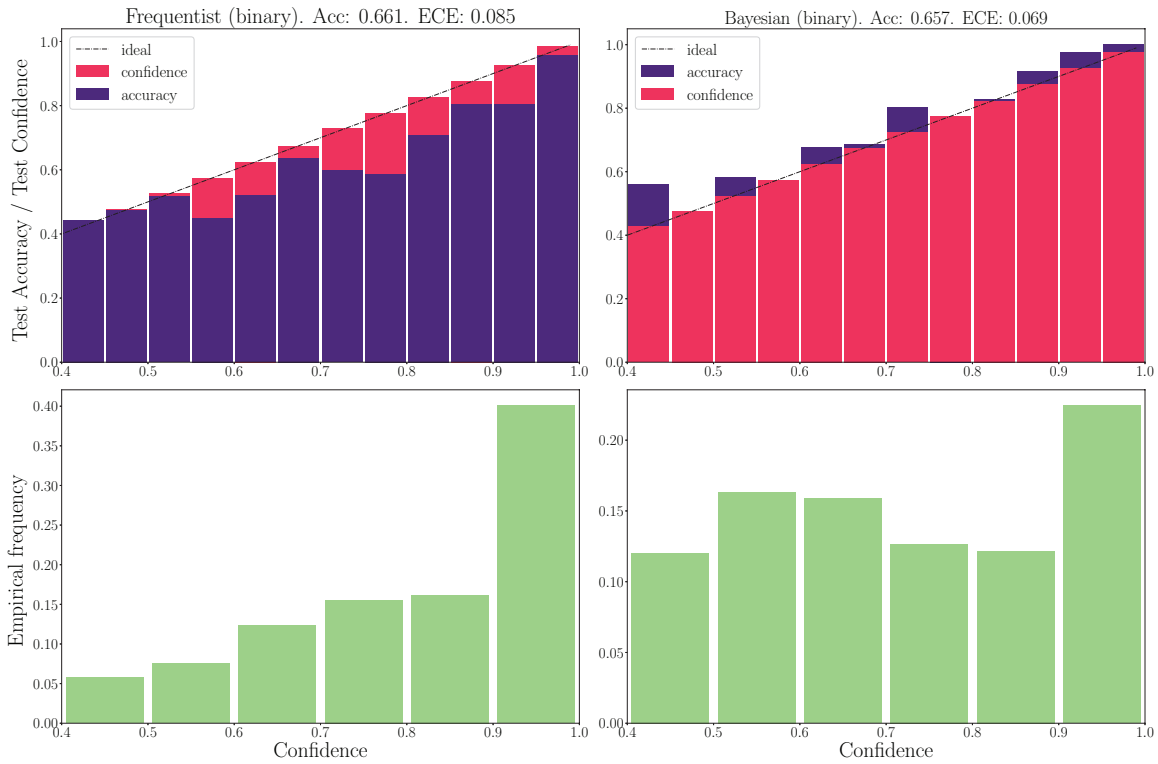

FIGURE 6
Top: Reliability diagrams (for in-distribution data) with binary synapses for the DVS-Gestures dataset. **Bottom**: Corresponding empirical confidence histograms for in-distribution data.

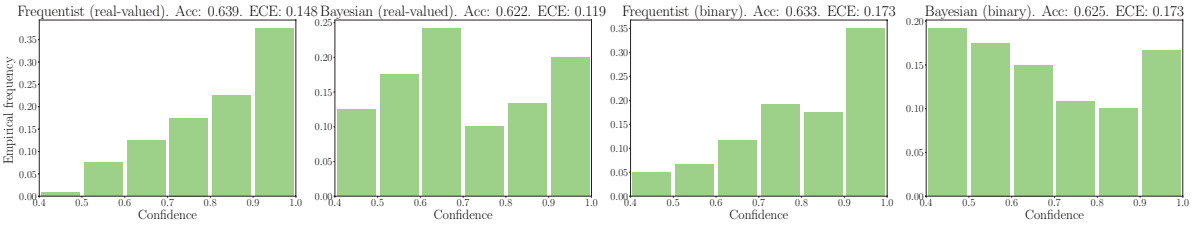

FIGURE 7
Out-of-distribution empirical confidence histograms for SNNs with real-valued and binary synapses on the DVS-Gestures dataset.

equality $\text{acc}(\mathcal{B}_m) = \text{conf}(\mathcal{B}_m)$ holds. If in the mth bin, the empirical accuracy and empirical confidence are different, the predictor is considered to be over-confident when the inequality $\text{acc}(\mathcal{B}_m) < \text{conf}(\mathcal{B}_m)$ holds, and under-confident when the reverse inequality $\text{acc}(\mathcal{B}_m) > \text{conf}(\mathcal{B}_m)$ holds.

3.3.3. Expected calibration error

While reliability diagrams offer a fine-grained description of calibration, the ECE provides a scalar measure of the global miscalibration of the model. This is done by computing the average

difference between per-bin confidence and accuracy as Guo et al. (2017).

$$\mathrm{ECE} = \frac{1}{|\mathcal{D}|}\sum_{m=1}^{M} |\mathcal{B}_m| \left|\mathrm{conf}(\mathcal{B}_m) - \mathrm{acc}(\mathcal{B}_m)\right|. \tag{44}$$

Models with a lower ECE are considered to be better calibrated.

3.3.4. Out-of-distribution empirical confidence

Reliability diagrams and ECE assume that the test data follows the same distribution as the training data. A well-calibrated model is also expected to assign lower probabilities to out-of-distribution data, i.e., data that does not follow the training distribution (DeGroot and Fienberg, 1983). To gauge the capacity of a model to recognize out-of-distribution data, a common approach is to plot the histogram of the predictive probabilities $\left\{p\left(c^*_{\boldsymbol{x}}|\boldsymbol{x}, \{\boldsymbol{w}_s\}_{s=1}^{N_S}\right)\right\}_{\boldsymbol{x}\in\mathcal{D}_{\mathrm{ood}}}$ evaluated on a dataset $\mathcal{D}_{\mathrm{ood}}$ of out-of-distribution examples (DeGroot and Fienberg, 1983; Daxberger and Hernández-Lobato, 2019). Such examples may correspond, as discussed, to examples recorded in different lighting conditions with a neuromorphic camera.

3.4. Offline learning

3.4.1. Two-moons dataset

We start by considering the two-moons dataset. For this experiment, the SNN comprises two fully connected layers with 256 neurons each. Bayesian learning is implemented with different values of the temperature parameter ρ in the free energy (Equation 11). In Figure 4, triangles indicate training points for a class "0," while circles indicate training points for a class "1." The color intensity represents the predictive probabilities (Equation 37) for frequentist learning and Equation (38) for Bayesian learning: the more intense the color, the higher the prediction confidence determined by the model. Bayesian learning is observed to provide better calibrated predictions, that are more uncertain outside the input regions covered by training data points.

For both real-valued and binary synapses, the temperature parameter ρ has an important role to play in preventing overfitting and underfitting of the training data, while also enabling uncertainty quantification. When the parameter ρ is too large, the model cannot fit the data correctly, resulting in inaccurate predictions; while when ρ is too small, the training data is fit more tightly, leading to a poor representation of the prediction uncertainty outside the training set. A well-chosen value of ρ strikes the best trade-off between faithfully fitting the training data and allowing for uncertainty quantification. Frequentist algorithms, obtained in the limit when $\rho \to 0$, yield the most over-confident estimates.

3.4.2. DVS-gestures

We now turn to the DVS-Gestures dataset, for which we plot the performance for real-valued and binary-valued SNNs, in terms of accuracy, reliability diagrams (DeGroot and Fienberg, 1983), and ECE (Guo et al., 2017) in Figures 5, 6. In all cases, the SNNs have two fully connected layers comprising 4,096 neurons each, and they are trained for 200 epochs. The architecture was chosen to highlight the benefits of Bayesian learning over frequentist learning in regimes characterized by epistemic uncertainty, and it was not optimized for maximal accuracy. The figures confirm that Bayesian SNNs generally produce better calibrated outcomes. In fact, reliability diagrams (top rows) demonstrate that frequentist learning algorithms produce overconfident decisions, while Bayesian learning outputs decisions whose confidence levels match well the test accuracies. This improvement is reflected, for models with real-valued synapses (with fixed precision), in a lower ECE of 0.064, as compared to 0.088 for frequentist SNNs; while, for binary SNNs, the reduction in ECE is from 0.085 for frequentist learning, to 0.069 for Bayesian learning. This benefit may come at the cost of a slight decrease in terms of accuracy, which is only observed here for binary synapses. The bottom parts of Figures 5, 6 also show that frequentist learning tends to output high-confidence decisions with a larger probability.

We now turn to evaluate the performance in terms of robustness to out-of-distribution data. As explained in Section 3.3, to this end, we evaluate the histogram of the confidence levels produced by frequentist and Bayesian learning, as shown in Figure 7. From the figure, it is remarked that Bayesian learning correctly provides low confidence levels on out-of-distribution data, while frequentist learning outputs decisions with confidence levels similar to the case of in-distribution data, which are shown in Figures 5, 6.

This point is further illustrated in Figure 8 by showing the three largest probabilities assigned by the different models on selected examples, considering real-valued synapses in the top row and binary synapses in the bottom row. In the left column, we observe that, when both models predict the wrong class, Bayesian SNNs tend to do so with a lower level of certainty, and typically rank the correct class higher than their frequentist counterparts. Specifically, in the examples shown, Bayesian models with both real-valued and binary synapses rank the correct class second, while the frequentist models rank it third. Furthermore, as seen in the middle column, in a number of cases, the Bayesian models manage to predict the correct class, while the frequentist models predict a wrong class with high certainty. Finally, in the right column, we show that even when frequentist models predict the correct class and Bayesian models fail to do so, they still assign lower probabilities (i.e., $< 50\%$) to the predicted class.

A key advantage of SNNs is the possibility to obtain intermediate decisions during the observation of the T

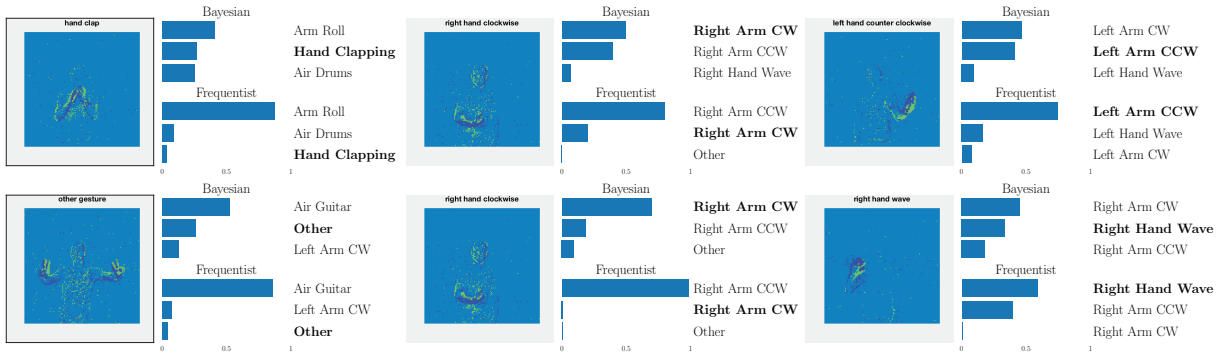

FIGURE 8
Top three classes predicted by both Bayesian and frequentist models on selected examples. **Top:** real-valued synapses. **Bottom:** binary synapses. The correct class is indicated in bold font.

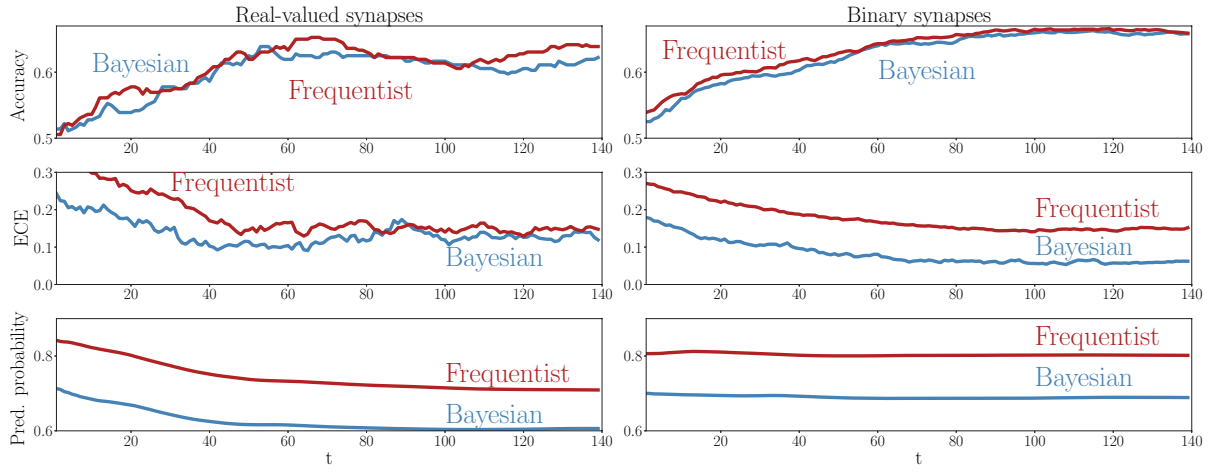

FIGURE 9
Evolution of the accuracy **(top)**, ECE **(middle)**, and predictive probabilities **(bottom)** during the presentation of out-of-distribution test examples for the DVS-Gestures dataset. The horizontal axis represents the time instants t within the presentation of each test example. **Left**: Real-valued synapses. **Right**: Binary synapses.

samples of a test example. To elaborate on this aspect, Figure 9 reports the evolution of the mean test accuracy, ECE, and predictive probabilities (Equations 38, 37) for all examples in the out-of-distribution dataset as a function of the discrete time-steps $t = 1, 2, \ldots, T$. Although both Bayesian and frequentist methods show similar improvements in accuracy over time, frequentist algorithms remain poorly calibrated, even after the observation of many time samples. The bottom plots show that frequentist learning tends to be more confident in its decisions, especially when a few samples t have been observed. On the contrary, Bayesian algorithms offer better calibration and confidence estimates, even when only part of the input signal $\boldsymbol{x}$ has been observed.

TABLE 1 **Final average test accuracy and ECE on the split-MNIST dataset (real-valued synapses).**

Model	Accuracy	ECE
TACOS (Soures et al., 2021) (Full Precision)	83.45 ± 0.55%	N/A
Frequentist (Kirkpatrick et al., 2017)	77.19 ± 0.65%	0.39 ± 0.01
Bayesian committee machine	**85.44 ± 0.16%**	**0.36 ± 0.01**
Bayesian ensemble decision	85.03 ± 0.54%	**0.36 ± 0.01**

3.5. Continual learning

We now turn to continual learning benchmarks. Starting with the rate encoded MNIST dataset, we use coresets

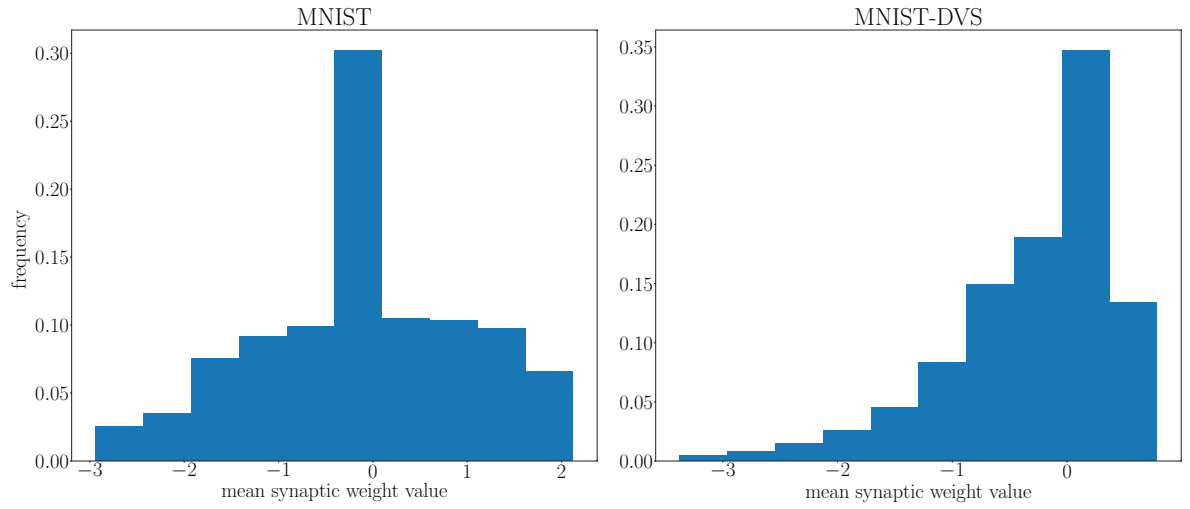

FIGURE 10
Distribution of the mean parameter $\boldsymbol{m}$ at the end of training on the MNIST and MNIST-DVS datasets.

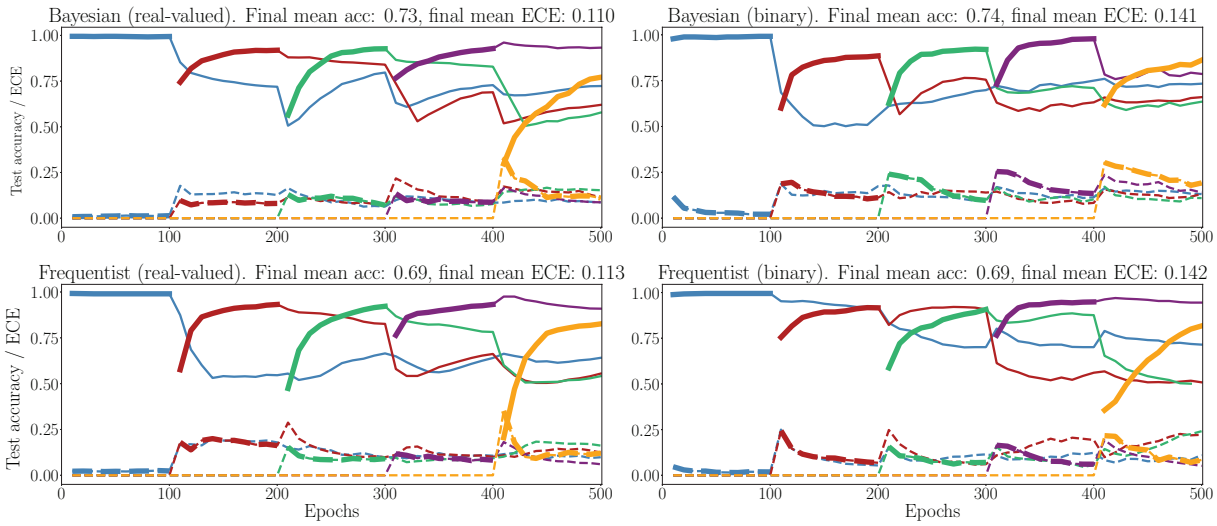

FIGURE 11
Evolution of the average test accuracies and ECE on all tasks of the split-MNIST-DVS across training epochs, with Gaussian and Bernoulli variational posteriors, and frequentist schemes for both real-valued and binary synapses. Continuous lines: test accuracy, dotted lines: ECE, bold: current task. Blue: {0, 1}; Red: {2, 3}; Green: {4, 5}; Purple:{6, 7}; Yellow: {8, 9}.

representing 7.5% of randomly selected training examples for each class. To establish a fair comparison with the protocol adopted in Soures et al. (2021), we train SNNs comprising a single layer with 400 neurons for one epoch on each subtask. This choice was found to be advantageous for Bayesian techniques—a result that may be related to the known asymptotic behavior of Bayesian neural networks as non-parametric models (Neal, 1996). In Table 1, we show the average accuracy over all tasks at the end of training on the last task, as well as the average ECE at that point for real-valued synapses, enabling a comparison with Soures et al. (2021). Bayesian continual learning is seen to achieve the best accuracy and calibration across all the methods studied here, including the solution introduced in Soures et al. (2021). The latter incurs a 2.5× memory overhead as compared to standard frequentist methods. Considering that we performed training

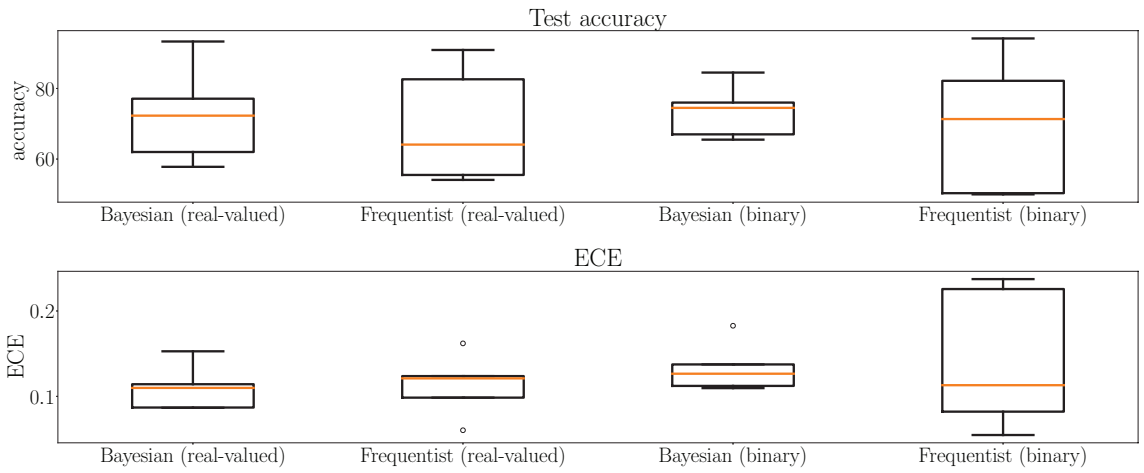

FIGURE 12
Box plots for final test accuracy **(top)** and ECE **(bottom)** across all tasks for Bayesian and frequentist learning with real-valued and binary synapses after the 500th epoch of training on the MNIST-DVS dataset (see Figure 11). The horizontal bar represents the median value across the tasks, while the box extends from the first to the third quartile. The whiskers extend from the box by 1.5 times the inter-quartile range. Circles represent outliers.

using the 8-bit precision imposed by the neuromorphic chip Loihi, our solution outperforms the state-of-the-art with a 5× memory consumption improvement. This saving can be leveraged, e.g., to store several samples of the weights for a committee machine implementation.

Next, for the MNIST-DVS dataset (Serrano-Gotarredona and Linares-Barranco, 2015), we use coresets representing 10% of randomly selected training examples for each class, and implement multilayer SNNs with $2,048 - 4,096 - 4,096 - 2,048 - 1024$ neurons per layer, that we train on each subtask for 100 epochs. This task requires a larger architecture and longer training time to allow for the processing of the richer spatio-temporal information recorded by neuromorphic cameras, as compared to the spatial information from static image datasets, such as MNIST, encoded into spikes *via* rate encoding (Jang et al., 2020a).

We highlight the requirement for a larger architecture on the MNIST-DVS dataset in Figure 10 by comparing the distribution of the mean parameter $\boldsymbol{m}$ at the end of training on the MNIST and MNIST-DVS datasets. For the larger network trained on the MNIST-DVS dataset, 83.5% of the mean parameters are non-zero, a larger proportion than that of the network trained on the MNIST dataset, for which only 80.1% of the mean weight parameters are non-zero. This demonstrates that the larger number of weights used for this task is important for the network to perform well.

In Figure 11, we show the evolution of the test accuracy and ECE on all tasks, represented with lines of different colors, during training. The performance on the current task is shown as a thicker line. We consider frequentist and Bayesian learning, with both real-valued and binary synapses. With Bayesian learning, the test accuracy on previous tasks does not decrease excessively when learning a new task, which shows the capacity of the technique to tackle catastrophic forgetting. Also, the ECE across all tasks is seen to remain more stable for Bayesian learning as compared to the frequentist benchmarks. For both real-valued and binary synapses, the final average accuracy and ECE across all tasks show the superiority of Bayesian over frequentist learning.

This point is further elaborated in Figure 12, which shows test accuracy and ECE on all tasks at the final epoch—the 500th—in Figure 12. Bayesian learning can be seen to offer a better test accuracy and ECE on average across tasks, as well as a lower dispersion among tasks.

4. Conclusion

In this work, we have introduced a Bayesian learning framework for SNNs with both real-valued and binary-valued synapses. Bayesian learning is particularly well suited for applications characterized by limited data—a situation that is likely to be encountered in use cases of neuromorphic computing such as edge intelligence. We have demonstrated the benefits of Bayesian learning in terms of calibration metrics that gauge the effectiveness of uncertainty quantification over a variety of offline and continual learning. We have also argued that the proposed rules exhibit mechanisms resembling those that enable lifelong learning in biological brains from a theoretically motivated information risk minimization framework. While this work focused on variational inference Bayesian learning methods, future research may explore Monte-Carlo based solutions. Finally, we recall the importance of investigating solutions for hardware design, adopting either

ensemble predictors or committees of machines. As an example, consider ensemble predictions based on binary synapses. An implementation based on digital hardware would need to store the real-valued parameters of the parameter vector distribution, and to sample from the distribution using auxiliary circuitry, which incurs energy and memory overheads. Alternatively, one could leverage the inherent stochasticity of analog hardware for sampling (Dalgaty et al., 2021), a line of research that we reserve for future work.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

OS first proposed to train SNNs *via* Bayesian learning. HJ derived the rule for binary synapses. NS extended it to real-valued synapses and designed and implemented the experiments. NS, HJ, and OS wrote the text. All authors contributed to the article and approved the submitted version.

Funding

This study received funding from Intel Labs through the Intel Neuromorphic Research Community (INRC). The work of HJ was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A10663288). The work of OS has also been supported by the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme (Grant Agreement No. 725731) and by an Open Fellowship of the EPSRC. The funders were not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fncom.2022.1037976/full#supplementary-material

References

Abraham, W. C., and Bear, M. F. (1996). Metaplasticity: the plasticity of synaptic plasticity. *Trends Neurosci*. 19, 126–130. doi: 10.1016/S0166-2236(96)80018-X

Aitchison, L., Jegminat, J., Menendez, J. A., Pfister, J.-P., Pouget, A., and Latham, P. E. (2021). Synaptic plasticity as Bayesian inference. *Nat. Neurosci.* 24, 565–571. doi: 10.1038/s41593-021-00809-5

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., et al. (2017). "A low power, fully event-based gesture recognition system," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI: IEEE), 7243–7252.

Angelino, E., Johnson, M., and Adams, R. (2016). Patterns of scalable Bayesian inference. *Foundat. Trends Mach. Learn.* 9, 119–247. doi: 10.1561/9781680832198

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, Darjan Legenstein, R., and Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nat. Commun.* 11, 3625. doi: 10.1038/s41467-020-17236-y

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*. doi: 10.48550/arXiv.1308.3432

Buhry, L., Azizi, A. H., and Cheng, S. (2011). Reactivation, replay, and preplay: how it might all fit together. *Neural Plast.* 2011, 1–11. doi: 10.1155/2011/203462

Chistiakova, M., Bannon, N. M., Bazhenov, M., and Volgushev, M. (2014). Heterosynaptic plasticity: multiple mechanisms and multiple roles. *Neuroscientist* 20, 483–498. doi: 10.1177/1073858414529829

Clayton, A. (2021). *Bernoulli's Fallacy*. New York, NY: Columbia University Press.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*. doi: 10.48550/arXiv.1602.02830

Dalgaty, T., Vianello, E., and Querlioz, D. (2021). "Harnessing intrinsic memristor randomness with bayesian neural networks," in *2021 International Conference on IC Design and Technology* (Dresden: ICICDT), 1–4.

Davies, M., Srinivasa, N., Lin, T. H., Chinya, G., Cao, Y., Harsha, S., et al. (2018). Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359

Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G. A. F., Joshi, P., et al. (2021). Advancing neuromorphic computing with loihi: a survey of results and outlook. *Proc. IEEE* 109, 911–934. doi: 10.1109/JPROC.2021.3067593

Daxberger, E. A., and Hernández-Lobato, J. M. (2019). Bayesian variational autoencoders for unsupervised out-of-distribution detection. *arXiv preprint 1912.05651*, abs/1912.05651. doi: 10.48550/arXiv.1912.05651

DeGroot, M. H., and Fienberg, S. E. (1983). The comparison and evaluation of forecasters. *J. R. Stat Soc.*32, 12–22. doi: 10.2307/2987588

Doya, K., Ishii, S., Pouget, A., and Rao, R. P. N. (Eds.). (2007). *Bayesian Brain. Computational Neuroscience Series*. London: MIT Press.

Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. (2020). Uncertainty-guided continual learning with Bayesian neural networks. *arXiv preprint 1906.02425*. doi: 10.48550/arXiv.1906.02425

Farquhar, S., and Gal, Y. (2019a). Towards robust evaluations of continual learning. *arXiv preprint 1805.09733*. doi: 10.48550/arXiv.1805.09733

Farquhar, S., and Gal, Y. (2019b). A unifying bayesian view of continual learning. *arXiv preprint 1902.06494*. doi: 10.48550/arXiv.1902.06494

Feldman Barrett, L. (Ed.). (2021). *Seven and a Half Lessons About the Brain*. London: Picador.

Finnie, P. S., and Nader, K. (2012). The role of metaplasticity mechanisms in regulating memory destabilization and reconsolidation. *Neurosci. Biobeh. Rev*. 36, 1667–1707. doi: 10.1016/j.neubiorev.2012.03.008

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nat. Rev. Neurosci*. 11, 127–138. doi: 10.1038/nrn2787

Friston, K. (2012). The history of the future of the bayesian brain. *Neuroimage* 62, 1230–1233. doi: 10.1016/j.neuroimage.2011.10.004

Gerstner, W., and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge: Cambridge University Press.

Gerstner, W., Lehmann, M., Liakoni, V., Corneil, D., and Brea, J. (2018). Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Front. Neural Circ*. 12, 53. doi: 10.3389/fncir.2018.00053

Guedj, B. (2019). A primer on PAC-bayesian learning. doi: 10.48550/arXiv.1901.05353

Guo, S., Yu, Z., Deng, F., Hu, X., and Chen, F. (2017). Hierarchical Bayesian inference and learning in spiking neural networks. *IEEE Trans. Cybern*. 49, 133–145. doi: 10.1109/TCYB.2017.2768554

Hawkins, J. (2021). *A Thousand Brains: A New Theory of Intelligence* (Hachette).

Huh, D., and Sejnowski, T. J. (2018). "Gradient descent for spiking neural networks," in *Advances in Neural Information Processing Systems, Vol. 31* (Montreal, QC).

Intel Corporation (2021). *Lava Software Framework*. Available online at: https://lava-nc.org/

Izhikevich, E. M. (2001). Resonate-and-fire neurons. *Neural Netw*. 14, 883–894. doi: 10.1016/S0893-6080(01)00078-8

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*. doi: 10.48550/arXiv.1611.01144

Jang, H., and Simeone, O. (2022). Multisample online learning for probabilistic spiking neural networks. *IEEE Trans. Neural Net. Learn. Syst*. 33, 2034–2044. doi: 10.1109/TNNLS.2022.3144296

Jang, H., Simeone, O., Gardner, B., and Grüning, A. (2019). An introduction to probabilistic spiking neural networks: probabilistic models, learning rules, and applications. *IEEE Sig. Proc. Mag*. 36, 64–77. doi: 10.1109/MSP.2019.2935234

Jang, H., Skatchkovsky, N., and Simeone, O. (2020a). Spiking neural networks-parts I, II, and III. doi: 10.48550/arXiv.2010.14217

Jang, H., Skatchkovsky, N., and Simeone, O. (2020b). VOWEL: a local online learning rule for recurrent networks of probabilistic spiking winner-take-all circuits. *arXiv preprint arXiv:2004.09416*. doi: 10.48550/arXiv.2004.09416

Jang, H., Skatchkovsky, N., and Simeone, O. (2021). "BiSNN: training spiking neural networks with binary weights via bayesian learning," in *2021 IEEE Data Science and Learning Workshop (DSLW)* (Toronto, ON: IEEE), 1–6.

Jaynes, E. T. (2003). *Probability Theory*. Cambridge: Cambridge University Press.

Jose, S. T., and Simeone, O. (2021). Free energy minimization: a unified framework for modeling, inference, learning, and optimization [Lecture Notes]. *IEEE Signal Proc. Mag*. 38, 120–125. doi: 10.1109/MSP.2020.3041414

Kaiser, J., Mostafa, H., and Neftci, E. (2020). Synaptic plasticity dynamics for deep continuous local learning (DECOLLE). *Front. Neurosci*. 14, 424. doi: 10.3389/fnins.2020.00424

Kandel, E. R., Dudai, Y., and Mayford, M. R. (2014). The molecular and systems biology of memory. *Cell* 157, 163–186. doi: 10.1016/j.cell.2014.03.001

Kappel, D., Habenschuss, S., Legenstein, R., and Maass, W. (2015). Network plasticity as bayesian inference. *PLoS Comput. Biol*. 11, e1004485. doi: 10.1371/journal.pcbi.1004485

Khan, M., and Lin, W. (2017). "Conjugate-computation variational inference converting variational inference in non-conjugate models to inferences in conjugate models," in *2017 International Conference on Artificial Intelligence and Statistics* (Ft. Lauderdale, FL), 878–887.

Khan, M. E., and Rue, H. (2021). The bayesian learning rule. *arXiv preprint 2107.04562*. doi: 10.48550/arXiv.2107.04562

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. U.S.A*. 114, 3521–3526. doi: 10.1073/pnas.1611835114

Knoblauch, J., Jewson, J., and Damoulas, T. (2019). Generalized variational inference. *arXiv preprint arXiv:1904.02063*. doi: 10.48550/arXiv.1904.02063

Kreutzer, E., Petrovici, M. A., and Senn, W. (2020). "Natural gradient learning for spiking neurons," in *Proceedings of the Neuro-inspired Computational Elements Workshop*, 1–3.

Kristiadi, A., Hein, M., and Hennig, P. (2020). "Being bayesian, even just a bit, fixes overconfidence in ReLU networks," in *2020 International Conferences on Machine Learning*, 5436–5446.

Kudithipudi, D., and Aguilar-Simon, M. B. J. E. A. (2022). Biological underpinnings for lifelong learning machines. *Nat. Mach. Intell*. 4, 196–210. doi: 10.1038/s42256-022-00452-0

Laborieux, A., Ernoult, M., Hirtzlin, T., and Querlioz, D. (2021). Synaptic metaplasticity in binarized neural networks. *Nat. Commun*. 12, 2549. doi: 10.1038/s41467-021-22768-y

Malenka, R. C., and Bear, M. F. (2004). LTP and LTD: an embarrassment of riches. *Neuron* 44, 5–21. doi: 10.1016/j.neuron.2004.09.012

Marder, E. (2012). Neuromodulation of neuronal circuits: back to the future. *Neuron* 76, 1–11. doi: 10.1016/j.neuron.2012.09.010

Mehonic, A., Sebastian, A., Rajendran, B., Simeone, O., Vasilaki, E., and Kenyon, A. J. (2020). Memristors–from in-memory computing, deep learning acceleration, and spiking neural networks to the future of neuromorphic and bio-inspired computing. *Adv. Intell. Syst*. 2, 2000085. doi: 10.1002/aisy.202000085

Meng, X., Bachmann, R., and Khan, M. E. (2020). Training binary neural networks using the bayesian learning rule. *arXiv preprint arXiv:2002.10778*. doi: 10.48550/arXiv.2002.10778

Morris, R. G. M. (2003). Long-term potentiation and memory. *Physiol. Rev*. 358, 643–647. doi: 10.1098/rstb.2002.1230

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. New York, NY: Springer.

Neftci, E., Mostafa, H., and Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. *IEEE Sig. Proc. Mag*. 36, 51–63. doi: 10.1109/MSP.2019.2931595

Nguyen, A., Yosinski, J., and Clune, J. (2015). "Deep neural networks are easily fooled: high confidence predictions for unrecognizable images," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, MA: IEEE), 427–436.

Osawa, K., Swaroop, S., Jain, A., Eschenhagen, R., Turner, R. E., Yokota, R., et al. (2019). Practical deep learning with bayesian principles. *arXiv preprint 1906.02506*. doi: 10.48550/arXiv.1906.02506

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: a review. *Neural Netw*. 113, 54–71. doi: 10.1016/j.neunet.2019.01.012

Putra, R. V. W., and Shafique, M. (2022). lpSpikeCon: enabling low-precision spiking neural network processing for efficient unsupervised continual learning on autonomous agents. *arXiv preprint 2205.12295*. doi: 10.48550/arXiv.2205.12295

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). "XNOR-Net: imagenet classification using binary convolutional neural networks," in *Proceedings of European Conference on Computer Vision* (Amsterdam: Springer), 525–542.

Scikit-Learn library (2020). *Two Moons Dataset*. Available online at: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

Serrano-Gotarredona, T., and Linares-Barranco, B. (2015). Poker-DVS and MNIST-DVS. Their history, how they were made, and other details. *Front. Neurosci*. 9, 481. doi: 10.3389/fnins.2015.00481

Shrestha, S. B., and Orchard, G. (2018). "SLAYER: spike layer error reassignment in time," in *Advances in Neural Information Processing Systems, Vol. 31* (Montreal, QC).

Simeone, O. (2022). *Machine Learning for Engineers*. Cambridge: Cambridge University Press.

Skatchkovsky, N., Jang, H., and Simeone, O. (2020a). "End-to-end learning of neuromorphic wireless systems for low-power edge artificial intelligence," in *Asilomar Conference on Signals, Systems, and Computers* (Pacific Grove, CA).

Skatchkovsky, N., Jang, H., and Simeone, O. (2020b). "Federated neuromorphic learning of spiking neural networks for low-power edge intelligence," in *ICASSP*

2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Barcelona: IEEE), 8524–8528.

Skatchkovsky, N., Simeone, O., and Jang, H. (2021). "Learning to time-decode in spiking neural networks through the information bottleneck," in *Advances in Neural Information Processing Systems*, 17049–17059.

Soures, N., Helfer, P., Daram, A., Pandit, T., and Kudithipudi, D. (2021). "Tacos: task agnostic continual learning in spiking neural networks," in *ICML Workshop*.

Stewart, K., Orchard, G., Shrestha, S. B., and Neftci, E. (2020). "Live demonstration: on-chip few-shot learning with surrogate gradient descent on a neuromorphic processor," in *2020 2nd IEEE Int. Conf. on Artificial Intelligence Circuits and Systems (AICAS)* (Genova: IEEE), 128–128.

Vaila, R., Chiasson, J. N., and Saxena, V. (2019). Deep convolutional spiking neural networks for image classification. *arXiv preprint 1903.12272*. doi: 10.48550/arXiv.1903.12272

Wang, H., and Yeung, D.-Y. (2020). A survey on bayesian deep learning. *ACM Comput. Surv.* 53, 1–37. doi: 10.1145/3409383

Zenke, F., and Ganguli, S. (2018). SuperSpike: supervised learning in multilayer spiking neural networks. *Neural Comput.* 30, 1514–1541. doi: 10.1162/neco_a_01086

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. *arXiv preprint 1703.04200*. doi: 10.48550/arXiv.1703.04200

Zhang, T. (2006). Information-theoretic upper and lower bounds for statistical estimation. *IEEE Trans. Inf. Theory* 52, 1307–1321. doi: 10.1109/TIT.2005.864439

Zou, Z., Alimohamadi, H., Zakeri, A., Imani, F., Kim, Y., Najafi, M. H., et al. (2022). Memory-inspired spiking hyperdimensional network for robust online learning. *Sci. Rep.* 12, 7641. doi: 10.1038/s41598-022-11073-3