entropy

MDPI

Article

Agreeing to Stop: Reliable Latency-Adaptive Decision Making via Ensembles of Spiking Neural Networks

Jiechen Chen, Sangwoo Park * and Osvaldo Simeone

KCLIP Laboratory—King's Communications, Learning and Information Processing Laboratory, Department of Engineering, King's College London, London WC2R 2LS, UK;
jiechen.chen@kcl.ac.uk (J.C.); osvaldo.simeone@kcl.ac.uk (O.S.)
* Correspondence: sangwoo.park@kcl.ac.uk

Abstract: Spiking neural networks (SNNs) are recurrent models that can leverage sparsity in input time series to efficiently carry out tasks such as classification. Additional efficiency gains can be obtained if decisions are taken as early as possible as a function of the complexity of the input time series. The decision on when to stop inference and produce a decision must rely on an estimate of the current accuracy of the decision. Prior work demonstrated the use of conformal prediction (CP) as a principled way to quantify uncertainty and support adaptive-latency decisions in SNNs. In this paper, we propose to enhance the uncertainty quantification capabilities of SNNs by implementing ensemble models for the purpose of improving the reliability of stopping decisions. Intuitively, an ensemble of multiple models can decide when to stop more reliably by selecting times at which most models agree that the current accuracy level is sufficient. The proposed method relies on different forms of information pooling from ensemble models and offers theoretical reliability guarantees. We specifically show that variational inference-based ensembles with p-variable pooling significantly reduce the average latency of state-of-the-art methods while maintaining reliability guarantees.

Keywords: spiking neural networks; conformal prediction; latency adaptivity; Bayesian learning

check for updates

Citation: Chen, J.; Park, S.; Simeone, O. Agreeing to Stop: Reliable Latency-Adaptive Decision Making via Ensembles of Spiking Neural Networks. *Entropy* **2024**, *26*, 126. https://doi.org/10.3390/e26020126

Academic Editor: Shuangming Yang

Received: 16 December 2023
Revised: 27 January 2024
Accepted: 30 January 2024
Published: 31 January 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Context: With the advent of large language models, sequence models are currently among the most studied machine learning techniques. Unlike methods based on conventional neural networks, such as transformers, spiking neural networks (SNNs) process time series with the prime objective of optimizing energy efficiency, particularly in the presence of sparse inputs [1–3]. The energy consumption of an SNN depends on the number of spikes generated internally by the constituent spiking neurons [4], and inference energy can be further reduced if decisions are taken as early as possible as a function of the complexity of the input time series [5].

In fact, in conventional SNN classifiers, decisions are typically made after processing the entire input sequence, leading to uniform inference latency levels across inputs [2]. However, the online operation of SNNs supports an alternative operating principle whereby inference latency is tailored to the difficulty of each example [5]. The decision on when to stop inference and produce a decision must rely on an estimate of the current accuracy of the decision, as stopping too early may cause unacceptable drops in accuracy. The latency-adaptive rule proposed in [5] uses the SNN's output confidence levels to estimate the true accuracy, while reference [6] determined the stopping time via a separate policy network.

SNN models, like their conventional neural network counterpart, tend to be poorly calibrated and thus produce overconfident decisions [7] (see also Figure 1 in [8]). As a consequence, the schemes in [5,6] do not offer any reliability guarantee at the stopping time. To address this problem, recent work [8] demonstrated the use of *conformal prediction* (CP) [9–12] as a principled way to quantify uncertainty and support adaptive-latency decisions in SNNs.

In the SpikeCP method introduced in [8], the SNN produces *set predictions* consisting of a subset of the set of all possible outputs. For instance, given an electroencephalography (EEG) or electrocardiography (ECG) time series as input, a set predictor determines a set of plausible conditions that a doctor may need to test for. Accordingly, for many applications, set predictors provide actionable information while also offering an inherent measure of uncertainty in the form of the size of the predicted set [9]. SpikeCP leverages the theoretical properties of CP to define reliable stopping rules based on the size of the predicted set.

Motivation: Predictive uncertainty can be decomposed into *aleatoric uncertainty*, which refers to the inherent randomness of the data-generation mechanism, and *epistemic uncertainty*, which arises due to the limited knowledge that can be extracted from a finite dataset [13,14]. While aleatoric uncertainty is captured by individual machine learning models like SNNs, epistemic uncertainty is typically accounted for by using *ensembles* of models. In particular, epistemic uncertainty is quantified by gauging the level of *disagreement* among the models in the ensemble [13,14]. By relying on conventional SNN models, SpikeCP does not attempt to quantify *epistemic uncertainty* and instead focuses only on aleatoric uncertainty quantification. The application of Bayesian learning and model *ensembling* as means to quantify epistemic uncertainty in SNNs was investigated in [15–17] and showed improvements in standard calibration metrics.

In this paper, we propose to enhance the uncertainty quantification capabilities of SpikeCP by implementing ensemble SNN models for the purpose of improving the reliability of stopping decisions. Intuitively, an ensemble of multiple models can decide when to stop more reliably by selecting times at which most models agree that the current accuracy level is sufficient. The proposed method relies on tailored information pooling strategies across the models in the ensemble that preserve the theoretical guarantees of CP and SpikeCP.

Main contributions: The main contributions of this work are summarized as follows.

- We propose a novel ensemble-based SNN model that can reliably decide when to stop in order to produce set predictions with coverage guarantees and with an average latency that is significantly lower than that of the state of the art.
- As shown in Table 1, we compare two ensembling strategies—*deep ensembles* (DE) [18,19] and Bayesian learning via *variational inference* (VI) [14,15]—and introduce two methods to efficiently combine the decisions from multiple models: namely, *confidence merging* (CM) and *p-variable merging* (PM). In both cases, the resulting set predictors satisfy theoretical reliability guarantees.
- Experiments show that VI-based ensembles with PM significantly reduce the average latency of state-of-the-art methods while maintaining reliability guarantees.

Organization: The remainder of the paper is organized as follows. Section 2 presents the problem, and Section 3 reviews the DC-SNN, while Section 4 introduces the proposed framework. Section 5 describes the experimental setting and results.

Table 1. Ensembling stategies and information pooling methods for SNN classifiers based on SpikeCP [8] studied in this paper

ensembling stategies	variational inference (VI)	deep ensembles (DE)
information pooling	confidence merging (CM)	p-variable merging (PM)

2. Problem Definition

In this paper, we study adaptive-latency multi-class classification for time series via SNNs [5,6,8]. As illustrated in Figure 1, unlike prior work [5,6,8], we propose to enhance the reliability of stopping decisions by explicitly accounting for epistemic uncertainty when deciding whether to stop or to continue processing the input. The end goal is to produce reliable set predictions with complexity and latency tailored to the difficulty of each example. In this section, we start by defining the problem and performance metrics.

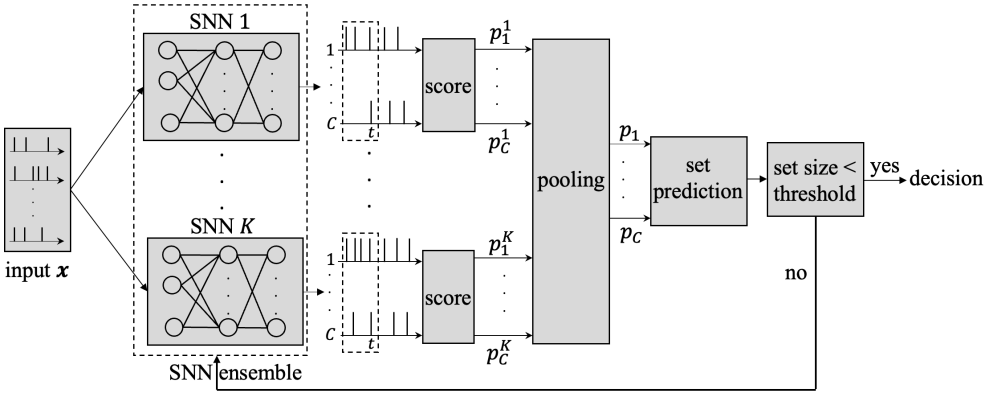

Figure 1. In the proposed system, an ensemble of K SNN models processes an input $\boldsymbol{x}$ and agrees on when to stop in order to make a classification decision. Each kth SNN model produces a score p_c^k for every candidate class $c = 1, ..., C$. The scores are combined to determine in an adaptive way whether to stop inference or to continue processing the input.

2.1. Multi-Class Classification with SNNs

We wish to classify a vector time series $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, ...$ with $N \times 1$ time samples $\boldsymbol{x}_t = [x_{t,1}, ..., x_{t,N}]$ into C classes using an SNN model. The entries of input vector $\boldsymbol{x}_t$ can be arbitrary, although typical SNN implementations assume binary inputs [20]. As shown in Figure 1, based on the time samples $\boldsymbol{x}^t = (\boldsymbol{x}_1, ..., \boldsymbol{x}_t)$ observed so far, at any time t, the C read-out neurons of the SNN produce the $C \times 1$ binary vector $\boldsymbol{y}_t = [y_{t,1}, ..., y_{t,C}]$, with entries equal to 1 representing spikes. Specifically, the SNN processes the input vector $\mathbf{x}_t$ at each time t to generate an output vector $\mathbf{y}_t$. The output $\mathbf{y}_t$ depends on the input history $\mathbf{x}^t$, effectively capturing the input's temporal dependencies and evolution over time.

Internally, an SNN model can be viewed as a recurrent neural network (RNN) with binary activations. Its operation is defined by a vector $\boldsymbol{\theta}$ of synaptic weights, which determines the response of each spiking neuron to incoming spikes. As in most existing implementations, we adopt a standard spike response model (SRM) [21] for the spiking neurons.

Carrying out decisions on the basis of the outputs of the C read-out neurons is typically achieved by *rate decoding* [22]. In rate decoding, at each time t, the SNN maintains a *spike count vector* $\boldsymbol{r}(\boldsymbol{x}^t) = [r_1(\boldsymbol{x}^t), ..., r_C(\boldsymbol{x}^t)]$ in which each cth entry

$$r_c(\boldsymbol{x}^t) = \sum_{t'=1}^{t} y_{t',c} \tag{1}$$

counts the number of spikes emitted so far by read-out neuron c. A normalized measure of *confidence* can then be obtained via the softmax function as [22]

$$f_c(\boldsymbol{x}^t) = e^{r_c(\boldsymbol{x}^t)} / \sum_{c'=1}^{C} e^{r_{c'}(\boldsymbol{x}^t)}, \tag{2}$$

for each class c. Conversely, the *loss* assigned by the SNN model to label c for input x^t is given by the *log-loss*

$$s_c(\boldsymbol{x}^t) = -\log f_c(\boldsymbol{x}^t). \tag{3}$$

The general goal of this work is to make reliable classification decisions at the earliest possible time t on the basis of the confidence levels (2) or, equivalently, of the losses (3) produced by SNN classifiers.

2.2. Ensemble Inference and Learning for SNNs

Conventional SNN models consist of a single SNN making decisions on the basis of the confidence levels (2), or (3), at a fixed time $t = T$. Neuroscience has long explored the connection between networks of spiking neurons and Bayesian reasoning [23], and the recent work [15] has explored the advantages of Bayesian learning and model ensembling in terms of uncertainty quantification for SNN classifiers. In this work, we leverage the enhanced uncertainty quantification capabilities of ensemble models to improve the reliability of adaptive-latency decision making via SNN models.

As illustrated in Figure 1, in the considered setting, K *pre-trained* SNN classifiers are used in parallel on an input sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, ...$ The operation of each kth SNN classifier is defined by a vector $\boldsymbol{\theta}^k$ of synaptic weights as explained in the previous subsection. We specifically consider two design methods for the ensembles: namely, *deep ensembles* (DE) [19] and *Bayesian learning* via *variational inference* (VI) [14].

In DE, the K models are obtained by running conventional SNN training methods based on a surrogate gradient [24] with K independent weight initializations, with each weight selected in an independent and identically distributed (i.i.d.) manner as Gaussian $\mathcal{N}(0, \sigma^2)$ variables for some fixed variance σ^2. In contrast, in VI, assuming an i.i.d. Gaussian prior distribution $\mathcal{N}(0, \sigma^2)$ for the model parameter vector $\boldsymbol{\theta}$, one optimizes over a variational posterior distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\zeta}^2)$ parameterized by mean vector $\boldsymbol{\mu}$ and diagonal covariance matrix with diagonal elements given by vector $\boldsymbol{\zeta}^2$. The optimization is done by using gradient descent via the reparameterization trick [15]. At inference time, the K models are generated by sampling the weight vectors $\boldsymbol{\theta}^k$ from the optimized distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\zeta}^2)$.

With DE, generating the K models in the ensemble requires retraining from scratch, whereas this can be done by simply drawing Gaussian variables in the case of VI. Therefore, with DE, the ensemble should be practically shared across many input test sequences, while for VI, it is possible to draw new ensembles more frequently—possibly even for each new input.

2.3. Set Prediction and Latency Adaptivity

As mentioned, we focus on latency-adaptive classifiers for which the time at which a decision is made is a function of the input $\boldsymbol{x}$ through the vector $\boldsymbol{f}(\boldsymbol{x}^t) = [f_1(\boldsymbol{x}^t), ..., f_C(\boldsymbol{x}^t)]$ of confidence levels (2) produced by the read-out neurons. Intuitively, when the model confidence is high enough, the classifier can produce a decision. We denote as $T_s(\boldsymbol{x})$ the time at which a decision is made for input $\boldsymbol{x}$. Furthermore, we allow the decision to be in the form of a set $\Gamma(\boldsymbol{x}) \subseteq \{1, ..., C\}$ of the set of C labels [9]. As mentioned in Section 1, set decisions provide actionable information in many applications of interest, such as for robotics, medical diagnosis, and language modeling, and they provide a measure of uncertainty via the predicted set's size $|\Gamma(\boldsymbol{x})|$ [9].

The performance of the classifier is measured in terms of reliability and latency. A predictive set $\Gamma(\boldsymbol{x})$ is said to be *reliable* if the probability that the correct label c is included in the set is no smaller than a predetermined target accuracy p_{targ}, i.e.,

$$\Pr(c \in \Gamma(\boldsymbol{x})) \geq p_{\text{targ}}, \tag{4}$$

where the probability is taken with respect to the distribution of the test example $(\boldsymbol{x}, c)$ as well as of the calibration data to be discussed next. The latency of the set prediction is defined as $\mathbb{E}[T_s(\boldsymbol{x})]$, where the expectation is taken over the same distribution as for (4).

The models are assumed to be pre-trained, and we assume we have access to a separate *calibration dataset*:

$$\mathcal{D}^{\text{cal}} = \{(\boldsymbol{x}[i], c[i])\}_{i=1}^{|\mathcal{D}^{\text{cal}}|}, \tag{5}$$

with $|\mathcal{D}^{\text{cal}}|$ examples $(\boldsymbol{x}[i], c[i])$ generated i.i.d. from the same distribution followed by the test example $(\boldsymbol{x}, c)$ [8,9]. As we will discuss in the next section, calibration data are

used to optimize the process of deciding when to stop so as to guarantee the reliability requirement (4).

3. Ensemble-Based Adaptive Point Classification via SNNs

In this section, we first review dynamic-confidence SNN (DC-SNN), a point predictor for latency-adaptive SNN classification [5], and then introduce the ensemble-based version.

3.1. DC-SNN

DC-SNN produces a decision at the first time t for which the maximum confidence level across all possible classes is larger than a fixed *target confidence level* $p_{\text{th}} \in (0,1)$. Accordingly, the stopping time is given by

$$T_s(\boldsymbol{x}) = \min_{t\in\{1,\ldots,T\}} t \text{ s.t. } \max_{c\in\mathcal{C}} f_c(\boldsymbol{x}^t) \geq p_{\text{th}} \tag{6}$$

if there is a time $t < T$ that satisfies the constraint and $T_s(\boldsymbol{x}) = T$ otherwise. The rationale for this approach is that, by (6), if $T_s(\boldsymbol{x}) < T$, the classifier has a confidence level no smaller than p_{th} on the decision

$$\hat{c}(\boldsymbol{x}) = \arg\max_{c\in\mathcal{C}} f_c(\boldsymbol{x}^{T_s(x)}). \tag{7}$$

If the SNN classifier is *well calibrated*, the confidence level coincides with the true accuracy of the decision given by the class $\arg\max_{c\in\mathcal{C}} f_c(\boldsymbol{x}^t)$ at all times t. Therefore, setting the target confidence level p_{th} to be equal to the target accuracy p_{targ}, i.e., $p_{\text{th}} = p_{\text{targ}}$, guarantees a zero or negative reliability gap for the adaptive decision (7) when $T_s(\boldsymbol{x}) < T$. However, the assumption of calibration is typically not valid. To address this problem, reference [5] introduced a solution based on the use of a calibration dataset.

Specifically, DC-SNN evaluates the empirical accuracy of the decision (7), i.e.,

$$\hat{\mathcal{A}}^{\text{cal}}(p_{\text{th}}) = \frac{1}{|\mathcal{D}^{\text{cal}}|} \sum_{i=1}^{|\mathcal{D}^{\text{cal}}|} \mathbb{1}(\hat{c}(\boldsymbol{x}[i]) = c[i]), \tag{8}$$

where $\mathbb{1}(\cdot)$ is the indicator function, for a grid of possible values of the target confidence level p_{th}. Then, it chooses either the minimum value p_{th} that ensures the inequality $\hat{\mathcal{A}}^{\text{cal}}(p_{\text{th}}) \geq p_{\text{targ}}$ so that the calibration accuracy exceeds the target accuracy level p_{targ} or the smallest value p_{th} that maximizes $\hat{\mathcal{A}}^{\text{cal}}(p_{\text{th}})$ if the constraint $\hat{\mathcal{A}}^{\text{cal}}(p_{\text{th}}) \geq p_{\text{targ}}$ cannot be met.

3.2. Ensemble-Based DC-SNN

Following Section 2.2, one can directly extend DC-SNN to implement approximate Bayesian learning by means of VI and DE methods. Accordingly, at inference time, a decision is made on the basis of K SNN models from a trained ensemble, which is fixed in the case of DE and randomly generated for VI. In this subsection, we briefly describe the decision procedure for a Bayesian version of DC-SNN.

Given some input $\boldsymbol{x}$, each kth model produces a confidence value $f_c^k(\boldsymbol{x}^t)$ for the pair $(\boldsymbol{x}^t, c)$. Implementing standard Bayesian model averaging, the confidence values $f_c^k(\boldsymbol{x}^t)$, $k = 1,\ldots,K$ for all models are then pooled by averaging as

$$f_c(\boldsymbol{x}^t) = \frac{1}{K}\sum_{k=1}^{K} f_c^k(\boldsymbol{x}^t). \tag{9}$$

The ensemble probability $f_c(\boldsymbol{x}^t)$ in (9) is finally applied in (6) and (7) to obtain the final decision.

4. Ensemble-Based Adaptive Set Classification via SNNs

In this section, we introduce *ensemble-based SpikeCP*, a novel framework for latency-adaptive classification that wraps around any pre-trained ensemble of SNN classifiers, including ensembles obtained via DE and VI. We propose two implementations corresponding to different ways of pooling information across the K models in the ensemble.

4.1. SpikeCP

We first review SpikeCP [8], which applies to a single SNN model, i.e., with $K = 1$. The presentation here, unlike in [8], adopts the language of p-variables (see, e.g., [12,25]) in order to facilitate the extension to ensemble models.

SpikeCP fixes a predetermined set of *checkpoint times* $\mathcal{T}_s \subseteq \{1, ..., T\}$ at which inference may stop to produce a decision. The information available to determine whether to stop or not is the losses $\{s_c(\boldsymbol{x}^t)\}_{c=1}^{C}$ in (3) for the current input $\boldsymbol{x}^t$ as well as the corresponding losses $s_{c[i]}(\boldsymbol{x}^t[i])$ for the calibration data points indexed by $i = 1, ..., |\mathcal{D}^{\text{cal}}|$. For each class c, SpikeCP computes the quantity

$$p_c(\boldsymbol{x}^t) = \frac{\sum_{i=1}^{|\mathcal{D}^{\text{cal}}|} \mathbb{1}(s_c(\boldsymbol{x}^t) \leq s_{c[i]}(\boldsymbol{x}^t[i])) + 1}{|\mathcal{D}^{\text{cal}}| + 1}, \tag{10}$$

where $\mathbb{1}(\cdot)$ equals 1 if the argument is true and 0 otherwise. The quantity (10) corresponds, approximately, to the fraction of calibration data points for which the loss is no smaller than the loss for label c when assigned to the current test input $\boldsymbol{x}^t$. The corrections by 1 for the numerator and denominator are required to guarantee the following property, which follows from the standard theory of CP ([26], Proposition 1).

Theorem 1. *Let $\mathcal{D}^{t,\text{cal}} = \{(\boldsymbol{x}^t[i], c[i])\}_{i=1}^{|\mathcal{D}^{\text{cal}}|}$ be the calibration dataset with samples up to time t, and define as $\mathcal{H}_c^t$ the hypothesis that the pair $(\boldsymbol{x}^t, c)$ and the calibration data $\mathcal{D}^{t,\text{cal}}$ are i.i.d. The quantity (10) is a p-variable for null hypothesis $\mathcal{H}_c^t$; i.e., we have the conditional probability*

$$\Pr(p_c(\boldsymbol{x}^t) \leq \alpha | \mathcal{H}_c^t) \leq \alpha, \tag{11}$$

for all $\alpha \in (0, 1)$, where the probability is taken over the distribution of test and calibration data.

At each checkpoint $t \in \mathcal{T}_s$, SpikeCP constructs a predictive set by including all classes c with a p-variable larger than threshold α

$$\Gamma(\boldsymbol{x}^t) = \{c \in \mathcal{C} : p_c(\boldsymbol{x}^t) > \alpha\}. \tag{12}$$

By (11), the probability that the set (12) does not include the true test label c is smaller or equal to α or, equivalently, ([26], Proposition 1)

$$\Pr(c \in \Gamma(\boldsymbol{x}^t)) \geq 1 - \alpha. \tag{13}$$

Accordingly, SpikeCP sets $\alpha = (1 - p_{\text{targ}})/|\mathcal{T}_s|$ to ensure that condition (13) is satisfied irrespective of which checkpoint is selected. As detailed in [8], this is a form of *Bonferroni correction* ([27], Appendix 2).

SpikeCP stops inference at the first time $T_s(\boldsymbol{x})$ for which the size of the predicted set is smaller than a target set size I_{th}, so the stopping time is given by

$$T_s(\boldsymbol{x}) = \min\{t \in \mathcal{T}_s : |\Gamma(\boldsymbol{x}^t)| \leq I_{\text{th}}\}. \tag{14}$$

The threshold I_{th} is a design choice that is dictated by the desired informativeness of the resulting set predictor. For any threshold I_{th}, by construction, SpikeCP satisfies the reliability property (4) ([8], Theorem 1).

4.2. Ensemble-Based SpikeCP with Confidence Merging

In the proposed ensemble-SNN architecture in Figure 1, each SNN classifier parameterized by $\boldsymbol{\theta}^k$, $k = 1, ..., K$ produces a generally different probability $f_c^k(\boldsymbol{x}^t)$ in (2) or, correspondingly, a different loss $s_c^k(\boldsymbol{x}^t)$ for each class c given an input $\boldsymbol{x}^t$. In this paper, we study and compare two combining mechanisms.

First, in order to produce a confidence level for each possible label c, the confidence levels output by the K models in the ensemble can be combined using the generalized mean [28]:

$$f_c(\boldsymbol{x}^t) = \left(\frac{1}{K} \sum_{k=1}^{K} \left(f_c^k(\boldsymbol{x}^t) \right)^r \right)^{1/r} \tag{15}$$

for some integer $r \in [-\infty, +\infty]$. When $r = 1$, the ensemble probability (15) reduces to standard model averaging (9). Other values of r may in practice be advantageous, e.g., to enhance robustness [29,30], with the maximum operation recovered for $r = \infty$ and the minimum operation obtained with $r = -\infty$.

The probability (15) is used to calculate the score via (3), which is then directly used in (10) and (12) to determine the set predictor. Note that the same combination in (15) is also applied to calibration data. By the same arguments as for SpikeCP, this approach guarantees the reliability condition (4) by setting $\alpha = (1 - p_{\text{targ}})/|\mathcal{T}_s|$.

4.3. Ensemble-Based SpikeCP with P-Variable Merging

Given the reliance of the predicted set (12) on p-variables, directly merging the confidence levels may be suboptimal [31]. Accordingly, in this subsection, we explore the idea of directly pooling the p-variables rather than combining confidence levels. To this end, we first calculate the losses for the calibration set by using the kth model as $\{s_{c[i]}^k(\boldsymbol{x}^t[i])\}_{i=1}^{|\mathcal{D}^{t,\text{cal}}|}$ for $k = 1, ..., K$. Then, for a test input $\boldsymbol{x}^t$, we evaluate the p-variable (10) for the kth model as

$$p_c^k(\boldsymbol{x}^t) = \frac{1 + \sum_{i=1}^{|\mathcal{D}^{\text{cal}}|} \mathbb{1}(s_c^k(\boldsymbol{x}^t) \leq s_{c[i]}^k(\boldsymbol{x}^t[i]))}{|\mathcal{D}^{\text{cal}}| + 1}. \tag{16}$$

The p-variables $\{p_c^k(\boldsymbol{x}^t)\}_{k=1}^K$ are then pooled by using any *p-merging* function $F(\cdot)$, as defined next.

Definition 1 ([32,33]). *A function $F : [0,1]^K \to [0,\infty)$ is said to be a p-merging function if, when the inputs are p-variables, the output is also a p-variable, i.e., we have the inequality*

$$\Pr(F(p_c{}^1(\boldsymbol{x}^t), ..., p_c{}^K(\boldsymbol{x}^t)) \leq \alpha') \leq \alpha', \text{ for all } \alpha' \in (0,1), \tag{17}$$

where the probability is taken over the joint distribution of the K input p-variables.

Using the merged p-value generated as

$$p_c(\boldsymbol{x}^t) = F(p_c{}^1(\boldsymbol{x}^t), ..., p_c{}^K(\boldsymbol{x}^t)) \tag{18}$$

for any p-merging function $F(\cdot)$, the predictive set can be constructed by following (12). By definition of the p-merging function, the resulting set predictor also satisfies the reliability condition (4).

We observe that while CM is also applicable to DC-SNN as per (9), PM is specific to SpikeCP, which relies on p-variables to construct the predicted set (12).

In the experiments reported in the next section, we focus on the class of p-merging functions of the form [33]

$$F(p^1, ..., p^K) = a_r \left(\frac{1}{K} \sum_{k=1}^{K} \left(p^k\right)^r \right)^{1/r}, \tag{19}$$

where a_r is a constant chosen so as to ensure (17) as specified in ([33], Table 1). For example, setting $r = -\infty$ and, correspondingly, $a_r = K$, yields the p-merging function $F(p^1, ..., p^K) = K \min(p^1, ..., p^K)$, while setting $r = \infty$ with $a_\infty = 1$ yields $F(p^1, ..., p^K) = \max(p^1, ..., p^K)$.

5. Experiments

For numerical evaluations, we consider the standard DVS128 Gesture dataset [34], MNIST-DVS dataset [35], and the CIFAR-10 dataset. The first dataset represents a video recognition task, and the latter two represent image classification tasks. The calibration dataset $\mathcal{D}^{\text{cal}}$ is obtained by randomly sampling $|\mathcal{D}^{\text{cal}}| = 50$ examples from the test set, with the rest used for training, which is done via the surrogate gradient method [24]. The length of the time series is $T = 80$ samples, and we fix the set of possible checkpoints as $\mathcal{T}_s = \{20, 40, 60, 80\}$ and the target set size to $I_{\text{th}} = 3$. The target accuracy p_{targ} is set to 0.9.

We compare the performance of ensemble-based SpikeCP using DE or VI equipped with confidence merging (CM) or p-variable merging (PM) and ensemble-based DC-SNN. For DE, we follow the standard random initialization made available by PyTorch, while for VI, we set the prior distribution variance to 0.03. The parameter r in (15) for CM is set to 1, yielding standard model averaging [15]; while r in (19) for PM is set to $r = 45$, with $a_r = K^{1/r}$ following ([33], Table 1) based on the numerical minimization of latency on a held-out dataset. The results are averaged over 50 different realizations of calibration and test datasets, and the number of ensemble K is set to 6. For fair comparison, we apply the stopping rule defined in Section 3 to obtain the stopping time and use a top-3 predictor to produce a set $\Gamma^d(\boldsymbol{x})$ for ensemble-based DC-SNN.

5.1. MNIST-DVS Dataset

The MNIST-DVS dataset contains time series recorded from a DVS camera that is shown moving handwritten digits from "0" to "9" on a screen. The dataset contains 8000 training examples as well as 2000 examples used for calibration and testing. For this experiment, we adopt a fully connected SNN with one hidden layer having 1000 neurons.

Figure 2 reports accuracy—$\Pr(c \in \Gamma^d(\boldsymbol{x}))$ for ensemble-based DC-SNN and $\Pr(c \in \Gamma(\boldsymbol{x}))$ for ensemble-based SpikeCP—and normalized latency $\mathbb{E}[T_s(\boldsymbol{x})]/T$ as a function of the target accuracy p_{targ}. Ensemble-based DC-SNN increases the decision latency as the target probability p_{targ} increases in order to meet the reliability condition. However, a reliable decision is only attained by DC-SNN when p_{targ} is small since DC-SNN guarantees target accuracy only when the model is well calibrated. In contrast, ensemble-based SpikeCP is always reliable, irrespective of the target accuracy, as proven. Furthermore, ensemble-based SpikeCP using VI and PM requires smaller latency to achieve the target accuracy.

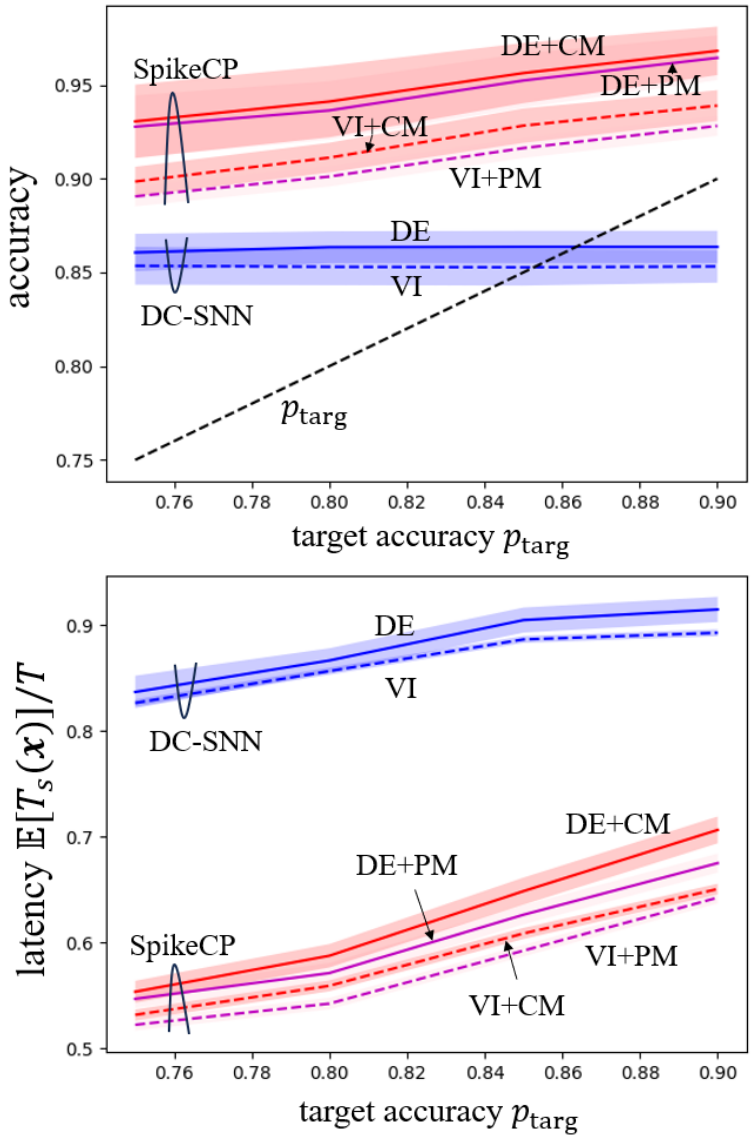

Figure 2. Accuracy ($\Pr(c \in \Gamma^d(\boldsymbol{x}))$ for ensemble-based DC-SNN and $\Pr(c \in \Gamma(\boldsymbol{x}))$ for ensemble-based SpikeCP) and normalized latency $\mathbb{E}[T_s(\boldsymbol{x})]/T$ as a function of the target accuracy p_{targ} for MNIST-DVS dataset.

In Figure 3, we show the accuracy and normalized latency as a function of the ensemble size. Note that even with $K = 1$, DE and VI perform differently, since while DE directly trains a conventional SNN, VI generates a model by sampling from an optimized distribution. With a larger ensemble size, both ensemble-based DC-SNN and SpikeCP exhibit reduced latency to reach a final decision. However, in practice, an excessively large ensemble size K for DE may increase complexity, necessitating the training of K SNN models. Furthermore, while SpikeCP maintains its reliability guarantee, DC-SNN falls short of achieving the target accuracy.

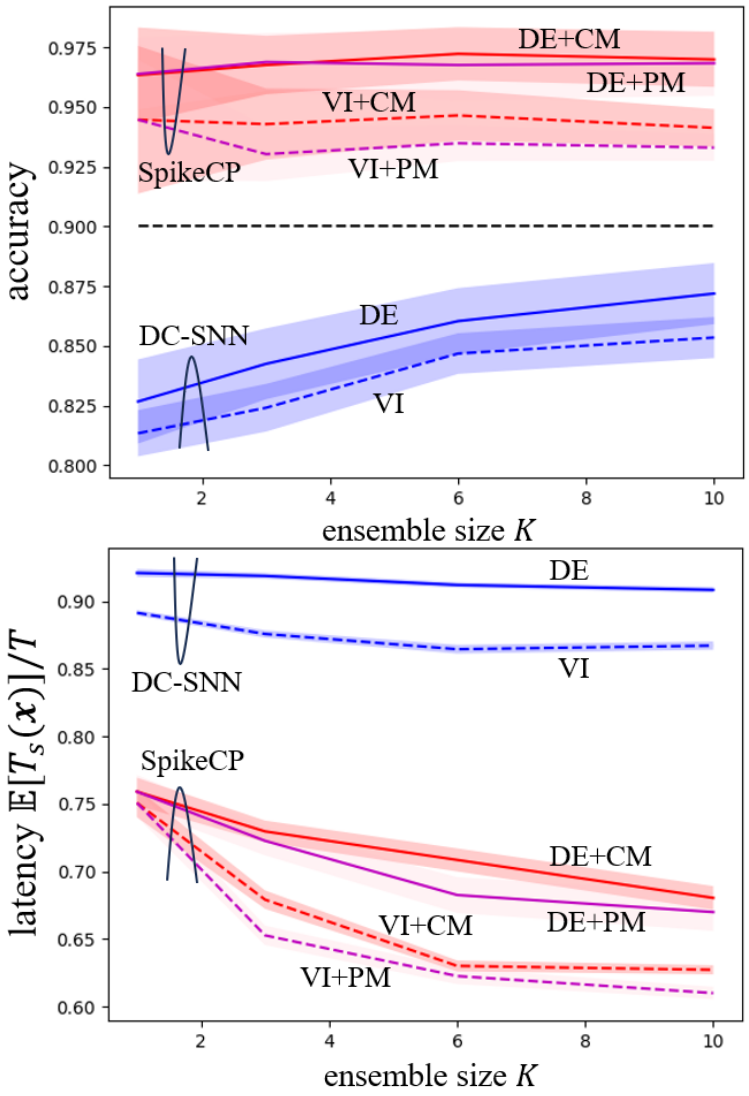

Figure 3. Accuracy ($\Pr(c \in \Gamma^d(\boldsymbol{x}))$ for ensemble-based DC-SNN and $\Pr(c \in \Gamma(\boldsymbol{x}))$ for ensemble-based SpikeCP) and normalized latency $\mathbb{E}[T_s(\boldsymbol{x})]/T$ as a function of the ensemble size K for MNIST-DVS dataset.

To explore the impact of the hyperparameter r in (15) and (19) for ensemble-based SpikeCP, we show in Figure 4 the accuracy and normalized latency as a function of r. To ensure that the p-merging function in (19) produces a valid p-value, we adopt different p-merging function $F(p^1, \ldots, p^K)$ for different values of r as in ([33], Table 1). CM pooling methods exhibit the lowest latency when r is approximately around 1, which aligns with standard Bayesian ensembling, while PM demonstrates a smaller latency with larger values of r.

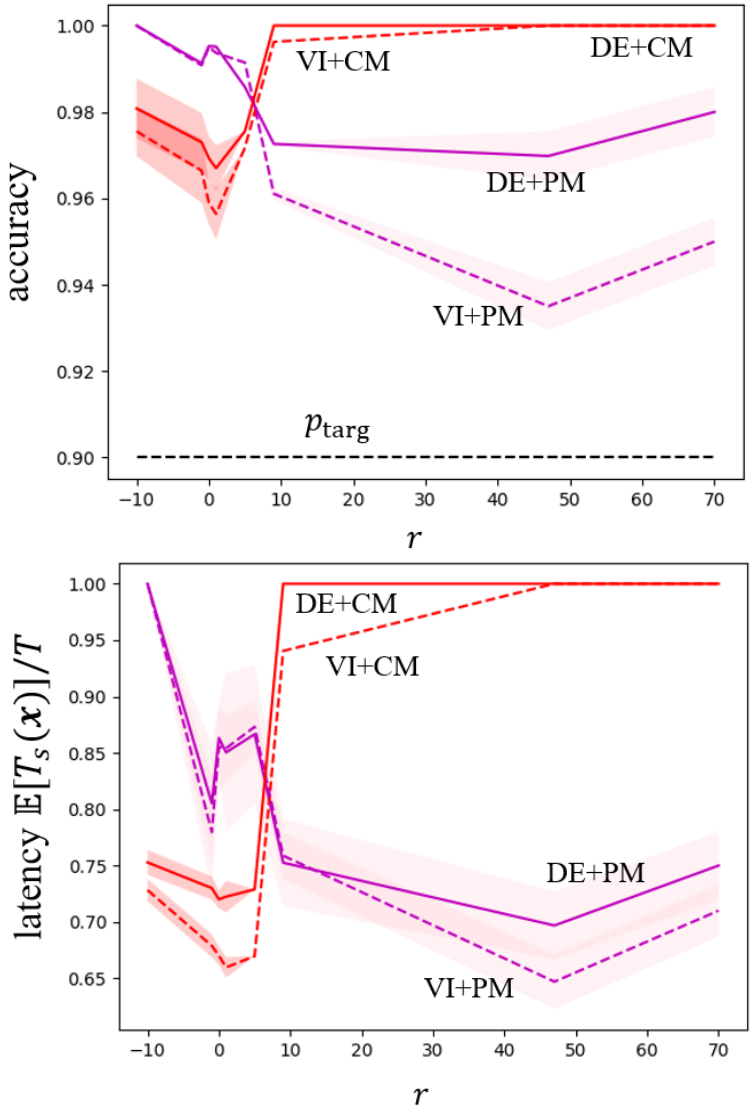

Figure 4. Accuracy $\Pr(c \in \Gamma(\boldsymbol{x}))$ and normalized latency $\mathbb{E}[T_s(\boldsymbol{x})]/T$ as a function of the hyperparameter r (in (15) for SpikeCP with CM and in (19) for SpikeCP with PM) for MNIST-DVS dataset.

5.2. DVS128 Gesture Dataset

The DVS128 Gesture dataset is a collection of videos from a DVS camera that show an actor performing one of 11 different gestures under three different illumination conditions. We divide each time series into $T = 80$ time intervals and integrate the discrete samples within each interval to obtain a (continuous-valued) time sample [36]. The dataset contains 1176 training data and 288 test data, from which 50 examples are chosen to serve as calibration data. The SNN architecture is constructed using a convolutional layer, encompassing batch normalization and a max-pooling layer, as well as a fully-connected layer as described in [36].

In Figure 5, we show the accuracy, given by the probability $\Pr(c \in \Gamma(\boldsymbol{x}))$ in (4), and the average decision latency as a function of the ensemble size K for the DVS128 Gesture dataset. The performance of ensemble-based DC-SNN is similar to that on the MNIST-DVS dataset and fails to meet the target accuracy. To highlight the performance of ensemble-based SpikeCP, we omit the performance of DC-SNN here. Confirming their theoretical properties, all ensemble-based SpikeCP schemes meet the target accuracy $p_{\text{targ}} = 0.9$. Furthermore, the average latency decreases with the ensemble size K, providing substantial improvements as compared to the original SpikeCP scheme with $K = 1$ [8].

VI methods tend to have better performance in terms of latency, showcasing the benefits of VI as a more principled approach for Bayesian learning. Finally, PM generally yields smaller latency values as compared to CM, indicating that merging p-variables offers a more efficient information pooling strategy.

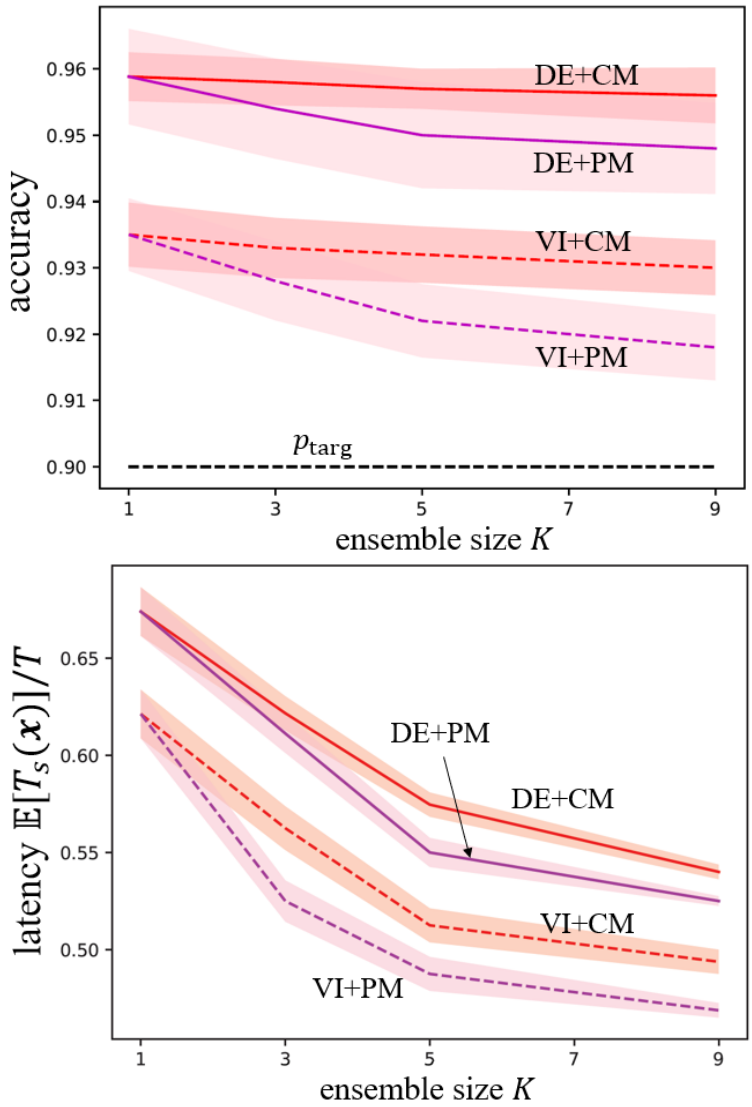

Figure 5. Accuracy $\Pr(c \in \Gamma(\boldsymbol{x}))$ and normalized latency $\mathbb{E}[T_s(\boldsymbol{x})]/T$ as a function of the ensemble size K for DVS128 Gesture dataset.

5.3. CIFAR-10 Dataset

The CIFAR-10 dataset consists of 60,000 32×32 color images that are divided into 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images. We use $|\mathcal{D}^{\text{cal}}| = 50$ calibration samples, which are obtained by randomly selecting 50 data points from the test set. We adopt a ResNet-18 architecture in which conventional neurons are replaced with SRM neurons [36]. Each example is repeatedly presented to the SNN for $T = 80$ times. The CIFAR-10 images are fed directly into the SNN, and the conversion from images to spikes is executed by the first spiking neural layer as in [36].

In Figure 6, we show the accuracy $\Pr(c \in \Gamma(\boldsymbol{x}))$ and normalized latency $\mathbb{E}[T_s(\boldsymbol{x})]/T$ as a function of the ensemble size K on the CIFAR-10 dataset for ensemble-based SpikeCP. As per our theory, SpikeCP can guarantee the reliability condition with all information pooling schemes. Furthermore, VI with PM produces the best performance in terms of latency.

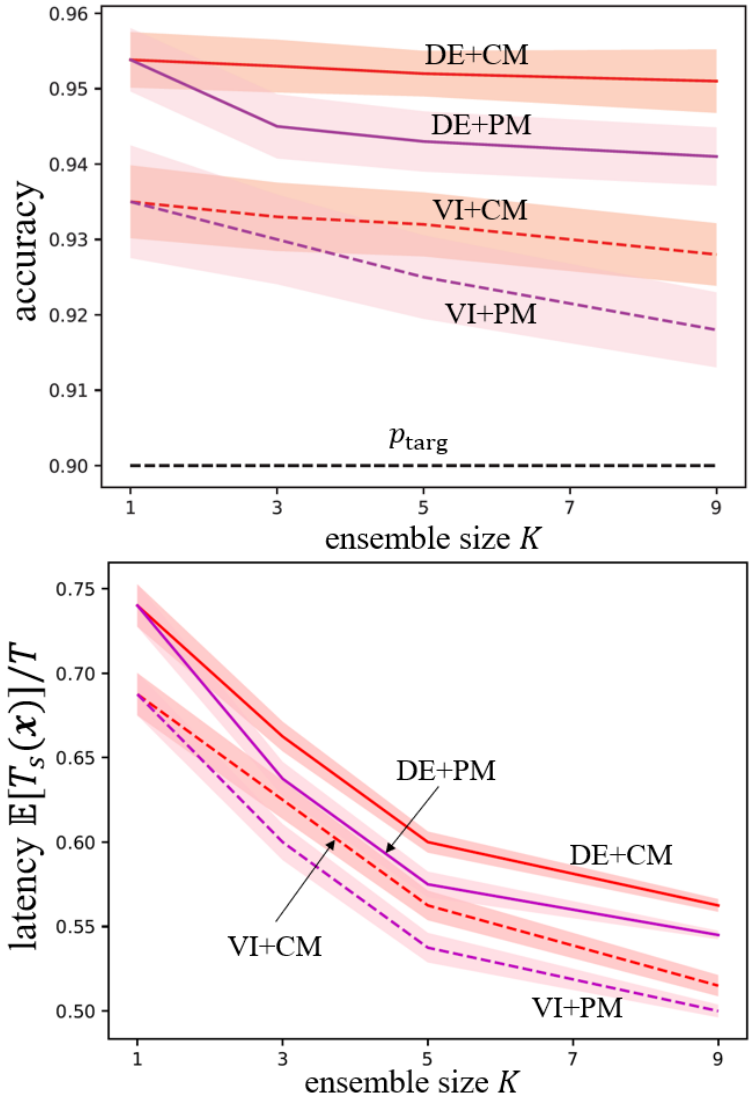

Figure 6. Accuracy $\Pr(c \in \Gamma(\boldsymbol{x}))$ and normalized latency $\mathbb{E}[T_s(\boldsymbol{x})]/T$ as a function of the ensemble size K for CIFAR-10 dataset.

6. Conclusions

In this work, we have introduced ensemble-based SpikeCP, a novel latency-adaptive SNN set predictor with provable reliability guarantees. Ensemble-based SpikeCP leverages the improved epistemic uncertainty quantification capacity of ensemble methods to enhance the reliability of stopping decisions for adaptive-latency classification. Intuitively, combing the predictions of multiple models supports the determination of a more reliable stopping time by focusing on time instants at which most models agree that the current accuracy level is sufficient. Our proposed approach relies on information pooling from ensemble models, and it provides a theoretical guarantee of reliability.

A limitation of our work is the use of the Bonferroni correction, which, while ensuring the reliability condition, may result in higher inference latency for challenging inputs. A potential future direction is to explore the derivation of tighter bounds on the reliability condition, which may lead to a solution with lower average latency. Another research topic involves extending SpikeCP to time decoding for further latency reduction. Finally, further work may address application of the proposed method to domains like wireless communications, where latency and reliability are crucial performance metrics [22].

Author Contributions: Conceptualization, software, formal analysis, and writing: J.C. and S.P.; conceptualization, supervision, writing, project administration, and funding acquisition: O.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the European Union's Horizon Europe project CENTRIC (101096379), by an Open Fellowship of the EPSRC (EP/W024101/1), and by the EPSRC project (EP/X011852/1).

Data Availability Statement: For the experiments in this paper, we used publicly available datasets, including the MNIST-DVS dataset (http://www2.imse-cnm.csic.es/caviar/MNISTDVS.html), the DVS128 Gesture dataset (https://ibm.ent.box.com/s/3hiq58ww1pbbjrinh367ykfdf60xsfm8/folder/50167556794), and the CIFAR-10 dataset (https://www.cs.toronto.edu/~kriz/cifar.html), accessed on 30 November 2023.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jang, H.; Simeone, O.; Gardner, B.; Gruning, A. An Introduction to Probabilistic Spiking Neural Networks: Probabilistic Models, Learning Rules, and Applications. *IEEE Signal Process. Mag.* **2019**, *36*, 64–77. [CrossRef]
2. Ghosh-Dastidar, S.; Adeli, H. Spiking neural networks. *Int. J. Neural Syst.* **2009**, *19*, 295–308. [CrossRef] [PubMed]
3. Tavanaei, A.; Ghodrati, M.; Kheradpisheh, S.R.; Masquelier, T.; Maida, A. Deep learning in spiking neural networks. *Neural Netw.* **2019**, *111*, 47–63. [CrossRef] [PubMed]
4. Mehonic, A.; Sebastian, A.; Rajendran, B.; Simeone, O.; Vasilaki, E.; Kenyon, A.J. Memristors—From in-memory computing, deep learning acceleration, and spiking neural networks to the future of neuromorphic and bio-inspired computing. *Adv. Intell. Syst.* **2020**, *2*, 2000085. [CrossRef]
5. Li, C.; Jones, E.; Furber, S. Unleashing the Potential of Spiking Neural Networks by Dynamic Confidence. *arXiv* **2023**, arXiv:2303.10276.
6. Li, Y.; Geller, T.; Kim, Y.; Panda, P. SEENN: Towards Temporal Spiking Early-Exit Neural Networks. *arXiv* **2023**, arXiv:2304.01230.
7. Guo, C.; Pleiss, G.; Sun, Y.; Weinberger, K.Q. On calibration of modern neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, NSW, Australia, 6–11 August 2017; pp. 1321–1330.
8. Chen, J.; Park, S.; Simeone, O. SpikeCP: Delay-Adaptive Reliable Spiking Neural Networks via Conformal Prediction. *arXiv* **2023**, arXiv:2305.11322.
9. Angelopoulos, A.N.; Bates, S. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv* **2021**, arXiv:2107.07511.
10. Shafer, G.; Vovk, V. A Tutorial on Conformal Prediction. *J. Mach. Learn. Res.* **2008**, *9*, 371–421.
11. Balasubramanian, V.; Ho, S.S.; Vovk, V. *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications*; Morgan Kaufmann: Waltham, MA, USA, 2014.
12. Vovk, V.; Gammerman, A.; Shafer, G. *Algorithmic Learning in a Random World*; Springer: New York, NY, USA, 2022.
13. Hüllermeier, E.; Waegeman, W. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Mach. Learn.* **2021**, *110*, 457–506. [CrossRef]
14. Simeone, O. *Machine Learning for Engineers*; Cambridge University Press: Cambridge, UK, 2022.
15. Skatchkovsky, N.; Jang, H.; Simeone, O. Bayesian continual learning via spiking neural networks. *Front. Comput. Neurosci.* **2022**, *16*, 1037976. [CrossRef]
16. Katti, P.; Skatchkovsky, N.; Simeone, O.; Rajendran, B.; Al-Hashimi, B.M. Bayesian Inference on Binary Spiking Networks Leveraging Nanoscale Device Stochasticity. *arXiv* **2023**, arXiv:2302.01302.
17. Cai, R.; Ren, A.; Liu, N.; Ding, C.; Wang, L.; Qian, X.; Pedram, M.; Wang, Y. VIBNN: Hardware acceleration of Bayesian neural networks. *ACM SIGPLAN Not.* **2018**, *53*, 476–488. [CrossRef]
18. Lakshminarayanan, B.; Pritzel, A.; Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6405–6416.
19. Ganaie, M.A.; Hu, M.; Malik, A.; Tanveer, M.; Suganthan, P. Ensemble deep learning: A review. *Eng. Appl. Artif. Intell.* **2022**, *115*, 105151. [CrossRef]
20. Shrestha, S.B.; Timcheck, J.; Frady, P.; Campos-Macias, L.; Davies, M. Efficient Video and Audio processing with Loihi 2. *arXiv* **2023**, arXiv:2310.03251.
21. Gerstner, W. Spike-response model. *Scholarpedia* **2008**, *3*, 1343. [CrossRef]
22. Chen, J.; Skatchkovsky, N.; Simeone, O. Neuromorphic Wireless Cognition: Event-Driven Semantic Communications for Remote Inference. *IEEE Trans. Cogn. Commun. Netw.* **2023**, *9*, 252–265. [CrossRef]
23. Doya, K. *Bayesian Brain: Probabilistic Approaches to Neural Coding*; MIT Press: Cambridge, MA, USA, 2007.
24. Neftci, E.O.; Mostafa, H.; Zenke, F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Mag.* **2019**, *36*, 51–63. [CrossRef]

25. Papadopoulos, H. Inductive conformal prediction: Theory and application to neural networks. In *Tools in Artificial Intelligence*; InTech: London, UK, 2008.
26. Vovk, V. Conditional validity of inductive conformal predictors. In Proceedings of the Asian Conference on Machine Learning, PMLR, Singapore, 4–6 November 2012; pp. 475–490.
27. Hochberg, Y.; Tamhane, A.C. *Multiple Comparison Procedures*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1987.
28. Koliander, G.; El-Laham, Y.; Djurić, P.M.; Hlawatsch, F. Fusion of probability density functions. *Proc. IEEE* **2022**, *110*, 404–453. [CrossRef]
29. Oh, J.; Kwak, N. Generalized mean for robust principal component analysis. *Pattern Recognit.* **2016**, *54*, 116–127. [CrossRef]
30. Gou, J.; Ma, H.; Ou, W.; Zeng, S.; Rao, Y.; Yang, H. A generalized mean distance-based k-nearest neighbor classifier. *Expert Syst. Appl.* **2019**, *115*, 356–372. [CrossRef]
31. Meng, X.L. Posterior predictive *p*-values. *Ann. Stat.* **1994**, *22*, 1142–1160. [CrossRef]
32. Vovk, V.; Wang, B.; Wang, R. Admissible ways of merging p-values under arbitrary dependence. *Ann. Stat.* **2022**, *50*, 351–375. [CrossRef]
33. Vovk, V.; Wang, R. Combining p-values via averaging. *Biometrika* **2020**, *107*, 791–808. [CrossRef]
34. Amir, A.; Taba, B.; Berg, D.; Melano, T.; McKinstry, J.; Di Nolfo, C.; Nayak, T.; Andreopoulos, A.; Garreau, G.; Mendoza, M.; et al. A low power, fully event-based gesture recognition system. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7243–7252.
35. Serrano-Gotarredona, T.; Linares-Barranco, B. Poker-DVS and MNIST-DVS. Their history, how they were made, and other details. *Front. Neurosci.* **2015**, *9*, 481. [CrossRef] [PubMed]
36. Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; Tian, Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 2661–2671.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.