Efficient Uncertainty Estimation in Spiking Neural Networks via MC-dropout

Tao Sun[1], Bojian Yin[1], and Sander Bohté[1,2,3](✉)

[1] CWI, Machine Learning Group, Amsterdam, The Netherlands
{tao.sun,byin,sbohte}@cwi.nl
[2] Rijksuniversiteit Groningen, Groningen, The Netherlands
[3] University of Amsterdam, Amsterdam, The Netherlands

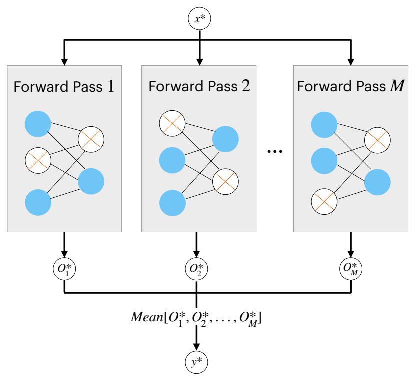
Abstract. Spiking neural networks (SNNs) have gained attention as models of sparse and event-driven communication of biological neurons, and as such have shown increasing promise for energy-efficient applications in neuromorphic hardware. As with classical artificial neural networks (ANNs), predictive uncertainties are important for decision making in high-stakes applications, such as autonomous vehicles, medical diagnosis, and high frequency trading. Yet, discussion of uncertainty estimation in SNNs is limited, and approaches for uncertainty estimation in ANNs are not directly applicable to SNNs. Here, we propose an efficient Monte Carlo(MC)-dropout based approach for uncertainty estimation in SNNs. Our approach exploits the time-step mechanism of SNNs to enable MC-dropout in a computationally efficient manner, without introducing significant overheads during training and inference while demonstrating high accuracy and uncertainty quality.

Keywords: Spiking Neural Network · Uncertainty Estimation · MC-dropout.

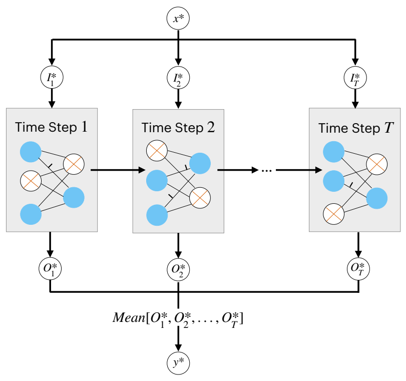
1 Introduction

Inspired by the brain's event-driven and sparse communication, spiking neural networks (SNNs) are enabling applications with high energy-efficiency in the form of neuromorphic computing [21]. Analogous to biological neurons, spiking neurons in SNNs communicate using discrete spikes, and time stepping is typically used to account for the evolution of these neurons' internal state as a response to impinging and emitted spikes. With recent advances in architectures and training methods, SNNs now achieve performance comparable to their artificial neural network (ANN) counterparts in many tasks [25, 26, 3].

To employ SNNs in the real-world however, accurate predictions have to be paired with high-quality uncertainty estimation to enable decision-making in high-stakes applications such as autonomous vehicles, medical diagnosis, and high frequency trading [4]: uncertain predictions in these applications may need to be reviewed by human experts for final decisions. In ANNs, predictive uncertainties in classification models are commonly represented by predictive distributions [13]. While evidence suggests that the brain performs a form of Bayesian



(a) Inference with ANN MC-dropout



(b) Inference in AOT-SNN

Fig. 1: (a) In ANNs, MC-dropout is performed by averaging results for a predefined number (M) of forward passes through a dropout-enabled network. (b) In AOT-SNNs, inference at each time step is taken as functionally equivalent to a forward pass in the MC-dropout method. As the SNN network evaluation requires T time-steps already, only one effective forward pass is needed.

inference based on uncertainty representations [18], the literature on uncertainty in SNNs is relatively limited and primarily concentrates on the sampling of probabilistic distributions, typically from a neuroscience perspective [20, 12].

Approaches for uncertainty estimation in classical deep learning models can be divided into two groups: deterministic methods and Bayesian methods [6]. With a deterministic method, a model learned from training data is essentially a point estimate of the model's parameters. In a deterministic deep network, each predictive distribution is estimated by a single forward propagation followed by the softmax function. Yet, although it is feasible to infer uncertainty with deterministic methods, these methods are known to be prone to output overconfident estimation [13, 6]. In contrast, a Bayesian network learns the posterior distribution of parameters in the network rather than depending on a single setting of parameters. The probability outputs of a Bayesian method can be analytically obtained by marginalizing the likelihood of the input with the estimated posterior distribution; this however is generally an intractable problem. To tackle this issue, many approximation methods and non-Bayesian methods have been introduced [6]. Example of these methods like Monte-Carlo-dropout (MC-dropout) [5] and deep ensembles [13] achieve excellent performance in terms of uncertainty estimation quality, either by repeatedly carrying out inference for each sample in perturbed versions of the network (Figure 1a), or by training a collection of networks and then carrying out inference in each network.

Here, we propose an efficient uncertainty estimation approach for SNNs by exploiting their time-step mechanism. Specifically, we apply continual MC-dropout in SNNs by taking their outputs averaged over time steps as predictive distributions, where we train SNNs with a loss function that also involves their time steps: **A**verage-**O**ver-**T**ime-SNNs (AOT-SNNs, Figure 1b). In AOT-SNNs, we take inference of each time step as functionally equivalent to a forward pass in the classical MC-dropout method. Since only one forward pass is needed in inference, the computational overhead for AOT-SNNs is significantly reduced relative to the MC-dropout method while still allowing effective uncertainty estimation. We compare the performance of AOT-SNNs with more standard SNNs, as well as with SNNs using the classical MC-dropout approach and SNN ensembles, across multiple classification tasks. We demonstrate that for identical network architectures, AOT-SNNs substantially outperform more standard SNNs and achieve comparable accuracy as ensembles and classical MC-dropout SNNs at little cost to uncertainty estimation quality while being much more computationally efficient.

2 Background

2.1 Problem Setup

We assume a training dataset $\mathcal{D}$ that consists of $\mathcal{N}$ i.i.d data points $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, where $\mathbf{x}_n \in \mathbb{R}^d$ and the true label $y_n \in \mathbf{y} = \{1, \ldots, K\}$. Given a sample $\mathbf{x}_n$, a neural network outputs the probabilistic predictive distribution $p_\omega(y_n|\mathbf{x}_n)$, where ω is the parameters of the network.

A number of non-Bayesian methods achieving excellent performance in term of uncertainty estimation have been proposed, among which are deep ensembles [13] and post-hoc calibration methods [10]. Deep ensembles are considered a "gold standard" for uncertainty estimation [24], while a set of models are trained with a proper scoring rule as the loss function. At inference time, the output of all models are then combined to obtain a predictive distribution. Post-hoc calibration methods, such as temperature scaling [10], involve the re-calibration of probabilities using a validation dataset and achieve excellent calibration performance in the i.i.d test dataset.

2.2 Bayesian Neural Networks and MC-Dropout Approximation

In a Bayesian neural network, the predictive distribution for a sample $\mathbf{x}$ is given by:

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \int p(\mathbf{y}|\mathbf{x}, \omega)p(\omega|\mathcal{D})d\omega. \tag{1}$$

The posterior distribution, $p(\omega|\mathcal{D})$ or $p(\omega|\mathbf{X}, \mathbf{Y})$, of the parameters ω can be computed by applying Bayes' theorem

$$p(\omega|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \omega)p(\omega)}{p(\mathbf{Y}|\mathbf{X})}. \tag{2}$$

Due to the intractability of the normalizer in (2), the posterior distribution $p(\omega|\mathcal{D})$ and the predictive distribution $p(\mathbf{y}|\mathbf{x}, \mathcal{D}))$ usually cannot be evaluated analytically. A variety of approximation methods have been introduced to tackle this issue [14, 9]. One such approximation is the MC-dropout method, which is often taken as a baseline model in uncertainty estimation [13, 17] due to its feasibility and relatively good performance.

Dropout [22] is a simple but effective technique used in deep learning models to prevent overfitting. In the MC-dropout method, dropout is applied before each weight layer of a neural network in both **training** and **testing**. The predictive distribution calculation with the MC-dropout method is performed by averaging results over a predefined number of forward passes through a dropout-enabled network. Gal & Gharamani [5] showed that neural networks with such configuration can be viewed as an approximation to a Bayesian method in the form of *deep Gaussian processes* [2].

Either MC-dropout models or deep ensembles involves multiple forward propagation passes in inference. As a result, when naively applied to SNNs, the computational and energy costs becomes relatively high due to the necessity of repeatedly running SNNs for multiple times during inference.

2.3 Source and Quality of Predictive Uncertainty

The only source of predictive uncertainty of deterministic methods is from the noisy data. Uncertainty in a Bayesian method comes from both data and defects of the model itself [6]: uncertainty caused by data is referred to as *data uncertainty*, while uncertainty caused by defects of the model itself is referred to as *model uncertainty.*

The quality of predictive uncertainties can be measured from two aspects [13]. The first concerns uncertainty quality on in-distribution data, where test data and training data share the same distribution. The second aspect evaluates generalization of uncertainty on domain-shifted data. While certain post-hoc calibration methods may generate accurate predictive probabilities for i.i.d data, their effectiveness in predicting uncertainty for domain-shifted data is not ensured [17]. For both aspects, model calibration is examined as the indication of uncertainty quality [17]. For classification tasks, accuracy and calibration are two evaluation measures that are mutually orthogonal [13]. Accuracy, defined as the ratio of corrected classified examples to total number of examples, measures how often a model correctly classifies; calibration measures the quality of predictive probability distributions [13] and indicates the extent to which the probability of a predicted class label reflects the real correct likelihood. A class of metrics to measure calibration is referred to as *proper scoring rules* [8], which include the Brier score (BS) and negative log-likelihood (NLL); another calibration metrics is the *Expected Calibration Error* (ECE) [10], which is a scalar summary statistic of calibration that approximates miscalibration. Although the definition ECE is intuitive and thus widely used, it is not a perfect metric for calibration because optimal ECE values can be generated by trivial solutions [17]; see the Appendix for details on proper scoring rules and ECE.

2.4 SNN

SNNs typically work with the same types of network topologies as ANNs, but computation in SNNs is distinct. SNNs use stateful and binary-valued spiking neurons, rather than the stateless and analog valued neurons of ANNs. As a result, unlike synchronous computation in ANNs, inference in SNNs is in a iterative form through multiple time steps $t = 0, 1, ..., T$: in each time step t, the membrane potential of a spiking neuron $U(t)$ is affected by the impinging spikes from connecting neurons emitted at time step $t-1$, and the past potential $U(t-1)$. Once the membrane potential $U(t)$ reaches a threshold θ, the neuron itself emits a spike. Such sparse and asynchronous communications between connected neurons is key to enabling SNNs to achieve high energy-efficiency.

LIF Neurons Various spiking neuron models exist, ranging in complexity from the detailed Hodgkin-Huxley model to the simplified Leaky-Integrated-and-Fire (LIF) neuron model [7]. The latter is widely used in SNNs, as it is interpretable and computationally efficient. Resembling an RC circuit, the LIF neural model is represented as:

$$\tau \frac{dU}{dt} = -U + RI. \tag{3}$$

where I and R are the current and input resistance, and τ is the time constant of the circuit. The discrete approximation of (3) can be written as:

$$u_i^t = \lambda u_i^{t-1} + \sum_j w_{ij} s_j^t - s_i^{t-1}\theta, \tag{4}$$

$$s_i^t = \begin{cases} 1, & \text{if } u_i^t > \theta \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where u_i is the membrane potential of a neuron i, λ denotes the leaky constant (< 1) for the membrane potential, w_{ij} represents the weight connecting the neuron i and its pre-synaptic neuron j, and s_i indicates whether a neuron spikes.

With the introduction of surrogate gradient methods [16, 25] and learnable LIF neurons [25, 3], both trainability and performance of SNNs have been improved dramatically.

3 Methods

Here, we present our proposed AOT-SNNs. We first explain how we efficiently apply MC-dropout to SNNs, and then introduce the loss function used in AOT-SNNs, which is based on the mean output values over time steps. Lastly, we explain the network architecture we use to demonstrate AOT-SNNs in practice.

3.1 Efficient MC-dropout in SNNs

As noted, the classical MC-dropout method runs a test sample a specified number (M) times in a model with dropout enabled, and takes the output of these forward passes as the final predictive distribution (Figure 1a). Thus applied in ANNs, MC-dropout results in satisfactory predictive uncertainty estimation.

In principle, such MC-dropout can be applied directly to SNNs, as *MC-dropout SNN*. This, however, results in computationally expensive inference as an SNN typically has to be run for multiple time steps to perform inference. Naively performing inference of a single sample in an *MC-dropout SNN* would mean running M forward passes of a sample through a network where each individual pass entails the evaluation of T time steps, incurring $M \times T$ time steps in total.

As an alternative, we propose to leverage the SNN time-step mechanism by enabling MC-dropout in AOT-SNNs during a single evaluation. Specifically, we compute predictive distributions in a dropout-enabled AOT-SNN by averaging outputs at multiple time steps. For a sample $\mathbf{x}$, the AOT-SNN computes at each time step t a probability distribution $p_t(\mathbf{y}|\mathbf{x})$. Thus, the probability distribution for the sample $\mathbf{x}$ is calculated as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} p_t(\mathbf{y}|\mathbf{x}).$$

In this view, each time step in an AOT-SNN is weakly equivalent to a single forward pass in the classical MC-dropout method. As such, only one forward pass is required during inference, which requires just T time steps compared to $M \times T$ for the MC-dropout SNN.

3.2 Loss Function

Loss functions in many current high-performing SNN learning algorithms [25, 3, 19, 27] are computed based on the output values of last time step, and we will refer such loss functions as *last-time-step* loss, resulting in Last-Time-Step-SNNs (*LTS-SNNs*). The last-time-step loss can be written as:

$$L = l(T), \tag{6}$$

where $l(T)$ is the loss function computed from the output values of the final time step T.

Since the last-time-step loss is not compatible with the proposed uncertainty estimation approach in AOT-SNNs, we introduce the *average-over-time* loss, which calculates its output by averaging over multiple time steps:

$$L = \frac{1}{T}\sum_{t=1}^{T} l(t). \tag{7}$$

By combining the average-over-time loss with dropout, we expect that the quality of uncertainty estimation for our approach will be improved, as the AOT loss pushes SNNs to correctly classify as much as possible at every time step. This is in contrast to LTS-SNNs, where dropout is not enabled during inference[4] and the predictive distributions output of only the last time step are used.

For $l(t)$, either negative log-likelihood (NLL) loss or the mean squared error (MSE) loss [3] can be used. Here, we use the MSE loss, as we find that in practice the NLL loss causes a disconnect between NLL and accuracy, which is an indication of miscalibration [10].

3.3 Network Architecture

We use AOT-SNNs with a network architecture very similar to the high-performing PLIF networks in [3]. These networks are composed of a *spiking encoder network* and a *classifier network*. The spiking encoder network consists of multiple downsampling modules. Each downsampling module has a certain number of convolution blocks and a pooling layer ($kernel\,size = 2, stride = 2$). The convolution block is composed of a convolution layer ($kernel\,size = 3, stride = 1, padding = 1$), a batch normalization layer, and a spiking neuron layer.

Our classifier network is slightly modified from [3] and includes a fully-connected layer, a spiking neuron layer, another fully-connected layer, which is then followed by a readout integrator layer. Unlike the original PLIF networks that classify using relatively coarse summed rate-coding collected from a population of output neurons, probabilities of AOT-SNNs are computed based on the membrane potentials of readout integrator neurons as in [25]. This modification enables AOT-SNNs to achieve better uncertainty estimation performance compared to corresponding standard PLIF networks while obtaining similar accuracy. In the spiking neuron layers, PLIF neurons [3] are used, where the time constants τ are learned and shared by neurons within the same layer. Note that dropout is applied to the neurons' output spikes, and input data is directly injected into the network as current into the input neurons.

4 Experiments

We performed a series of experiments to compare AOT-SNNs to LTS-SNNs, as well as MC-dropout SNNs and also with the 'gold standard' of SNN ensembles, across multiple classification tasks. As a proof of concept, we first applied this approach to the MNIST dataset. Second, we experiment on the CIFAR-10 dataset to compare our models with corresponding LTS-SNNs. Additionally, we reported and analyzed results on the CIFAR-100 dataset. Furthermore, we carried out an ablation study where we characterized the uncertainty properties of AOT-SNNs with regard to dropout rates and dropout types.

[4] For LTS-SNNs, dropout is not enabled at inference time as this leads to notably weak performance for LTS-SNNs, similar to that of ANNs.

Table 1: Performance comparisons between the AOT-SNN and its corresponding LTS-SNN on the MNIST dataset (mean±std across 10 models). The numbers after the model names represent time steps.

Model	Accuracy (%) ↑	BS ↓	NLL ↓	ECE ↓
AOT-SNN (8)	99.54±0.030	7.0e-4±4.3e-5	0.0144±7.6e-4	1.2e-3±3.4e-4
LTS-SNN (8)	99.37±0.080	9.8e-4±9.9e-5	0.021±2.5e-3	4e-3±1.1e-3
MC-dropout SNN (8, 10)	99.57±0.033	6.5e-4±5.0e-05	0.0125±9.6e-4	1.1e-3±2.9e-4
SNN Ensembles (8, 10)	99.56	7.5e-4	0.0180	6.7e-3

4.1 Experimental Setup

In our experiments, LTS-SNNs used the same layer structure as their corresponding AOT-SNNs. All the MC-dropout SNNs and SNN ensembles are based on their corresponding LTS-SNNs.

The Adam optimizer was used, with a cosine annealing learning rate scheduler, whose initial learning rate is 0.001 and T_{max} is 64. The default dropout rate used is 0.5. For the MINIST dataset, we used a batch size of 150, while the batch sizes were 60 for CIFAR-10 and 15 for CIFAR-100. The number of epochs used for each dataset were 200 (MNIST), 300 (CIFAR-10), and 300 (CIFAR-100).

4.2 MNIST

The spiking encoder network for the MNIST dataset has two downsampling modules, each of which includes only one convolution block. In Table 1, we compared the AOT-SNNs, its corresponding LTS-SNNs, MC-dropout SNNs, and SNN ensembles, all using best performing models that have eight time steps to evaluate samples. The results demonstrate that the AOT-SNNs outperform the LTS-SNNs in both accuracy and the predictive uncertainty metrics, including Brier score, NLL, and ECE. Furthermore, AOT-SNNs exhibit similar accuracy and uncertainty estimation as both MC-dropout SNNs and SNN ensembles.

4.3 CIFAR-10 and CIFAR-100

The architectures of AOT-SNNs for the CIFAR-10 and CIFAR-100 dataset are similar. They apply the same spiking encoder network, which has two downsampling modules, each with three convolution blocks. Their classifier networks differ only in the last fully-connected layer due to their different number of ground truth classes.

CIFAR-10 held-out test dataset. Table 2 presents a comparison of AOT-SNNs to LTS-SNNs, MC-dropout SNNs, and SNN ensembles. While each MC-dropout SNN ran five forward passes, each SNN ensemble consisted of five models. We show results for 4 and 8 time steps, corresponding to respective best performing duration (see also Table 3). AOT-SNNs exhibit superior performance compared to LTS-SNNs and achieve comparable accuracy to SNN ensembles while yielding slightly lower results on BS and NLL, only underperforming on ECE. In comparison to the MC-dropout SNNs, AOT-SNNs do deliver superior accuracy and performed almost as well as BS and NLL, with only a slight loss in ECE.

Table 3 presents the results of AOT-SNNs and LTS-SNNs with time steps smaller or equal to 10. With each model trained five times, the table lists the mean and standard deviation for all the metrics. In this exhaustive comparison, we see that that AOT-SNNs significantly outperform LTS-SNNs, with all models with more than 3 time steps achieving significantly better accuracy and Brier score, with best results for 8 time steps. Moreover, almost all AOT-SNNs achieve better NLL and ECE, except for the model with a single time step (which however has considerably lower accuracy).

CIFAR-100. Comparing the AOT-SNN with time step eight with its corresponding LTS-SNN for CIFAR-100 (Table 4), we similarly find that AOT-SNNs achieve significantly better results than the LTS-SNN, in both accuracy and predictive uncertainty quality.

CIFAR-10-C: domain-shifted test dataset. As mentioned earlier, the quality of predictive uncertainties needs to be measured on both in-distribution held-out data and domain-shifted data. We evaluated AOT-SNNs on the CIFAR-10-C dataset [11], a domain-shifted test dataset of CIFAR-10. The CIFAR-10-C

Table 2: Comparison on the CIFAR-10 dataset between AOT-SNNs, LTS-SNNs, MC-dropout models, and deep ensembles (mean± std across 5 models). The digits enclosed in brackets following the model names indicate the number of SNN time steps and the number of forward passes or models used in inference.

Model	Accuracy (%) ↑	BS ↓	NLL ↓	ECE ↓
AOT-SNN (4, 1)	90.2±0.26	0.0153±3.0e-4	0.38±1.2e-2	0.040±3.1e-3
AOT-SNN (8, 1)	90.8±0.23	0.0144±4.0e-4	0.37±2.2e-2	0.043±4.1e-3
LTS-SNN (4, 1)	88.9±0.71	0.017±1.1e-3	0.43±2.8e-2	0.058±4.4e-3
LTS-SNN (8, 1)	88.5±0.60	0.0181±8.1e-4	0.47±1.3e-2	0.067±3.4e-3
MC-dropout SNN (4, 5)	90.53±0.37	0.0140±4.1e-4	0.32±1.0e-2	0.026±3.0e-3
MC-dropout SNN (8, 5)	90.43±0.37	0.0145±5.3e-4	0.35±1.3e-2	0.037 ±1.4e-3
SNN Ensembles (4, 5)	90.9	0.0134	0.2919	0.012
SNN Ensembles (8, 5)	90.8	0.0135	0.2967	0.016

Table 3: Performance comparisons between AOT-SNNs and LTS-SNNs on CIFAR10 (mean±std across 5 trials).

Model	Time steps	Accuracy (%) ↑	BS ↓	NLL ↓	ECE ↓
AOT-SNN	2	89.4±0.18	0.0168±1.4e-4	0.417±6.1e-3	0.047±2.3e-3
AOT-SNN	3	89.7±0.26	0.0160±2.7e-4	0.40±2.1e-2	0.044±4.2e-3
AOT-SNN	4	90.2±0.26	0.0153±3.0e-4	0.38±1.2e-2	0.040±3.1e-3
AOT-SNN	5	90.4±0.07	0.0150±2.4e-4	0.39±2.6e-2	0.043±3.6e-3
AOT-SNN	6	90.5±0.16	0.0149±2.8e-4	0.38±1.7e-2	0.043±3.0e-3
AOT-SNN	7	90.2±0.34	0.0151±4.3e-4	0.37±1.2e-2	0.043±1.9e-3
AOT-SNN	8	90.8±0.23	0.0144±4.0e-4	0.37±2.2e-2	0.043±4.1e-3
AOT-SNN	9	90.5±0.55	0.0147±7.3e-4	0.37±2.4e-2	0.044±4.1e-3
AOT-SNN	10	90.7±0.41	0.0146±6.2e-4	0.37±2.4e-2	0.044±5.2e-3
LTS-SNN	1	88.2±0.47	0.0168±6.8e-4	0.36±1.3e-2	0.014±3.4e-3
LTS-SNN	2	88.6±0.40	0.0180±3.1e-4	0.46±1.2e-2	0.067±5.5e-3
LTS-SNN	3	88.0±0.56	0.0184±7.6e-4	0.44±2.3e-2	0.060±3.0e-3
LTS-SNN	4	88.9±0.71	0.017±1.1e-3	0.43±2.8e-2	0.058±4.4e-3
LTS-SNN	5	88.4±0.27	0.0181±4.7e-4	0.46±1.6e-2	0.063±3.1e-3
LTS-SNN	7	88.3±1.12	0.018±1.4e-3	0.48±2.6e-2	0.068±6.2e-3
LTS-SNN	8	88.5±0.60	0.0181±8.1e-4	0.47±1.3e-2	0.067±3.4e-3
LTS-SNN	9	88.0±0.52	0.0189±8.2e-4	0.49±2.5e-2	0.069±3.6e-3
LTS-SNN	10	88.0±0.91	0.019±1.5e-3	0.49±4.6e-2	0.069±6.2e-3

Table 4: Performance comparisons between the AOT-SNN and the corresponding LTS-SNN on the CIFAR-100 dataset.

Model	Time steps	Accuracy (%) ↑	BS ↓	NLL ↓	ECE ↓
AOT-SNN	8	65.15	5.028e-3	1.6749	0.1352
LTS-SNN	8	62.32	5.333e-3	1.7325	0.1665

dataset is designed to evaluate the robustness of image classification models against common corruptions. It contains 19 corruption types that are created by applying a combination of 5 severity levels to the original CIFAR-10 test set. The CIFAR-10-C dataset is commonly used as a benchmark to evaluate the uncertainty estimation in domain-shifted settings [17]. We compared the performance of the AOT-SNN with eight time steps and its corresponding LTS-SNN on all the severity levels of CIFAR-10-C (Figure 2). With the AOT-SNN outperforming the LTS-SNN in all severity levels, we conclude that AOT-SNNs also improve uncertainty estimation over LTS-SNNs in domain-shifted settings.

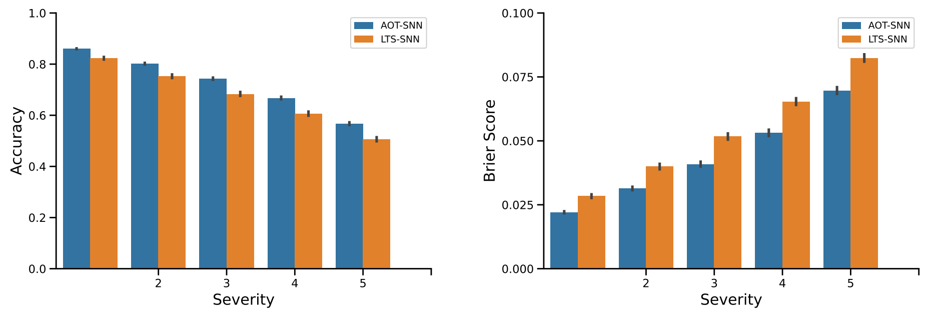


Fig. 2: Comparisons of the AOT-SNN model and its corresponding LTS-SNN on each severity level of CIFAR-10-C.

Ablation study. We further considered the impact of dropout rates and dropout types on the quality of uncertainty estimates of AOT-SNNs.

Dropout type. We replaced the dropout in the LTS-SNN and our best-performing model, both of which have eight time steps, with DropConnect [23]. Instead of dropping the spikes like the regular dropout, DropConnect randomly drops the weights in each layer before the PLIF neuron layer. As shown in Table 5, despite the slightly better performance of the LTS-SNN-DC compared to the corresponding dropout-based models (LTS-SNN), the AOT-SNN-DC outperform LTS-SNN-DC in terms of both accuracy and uncertainty quality (both models in the table have a dropout rate of 0.5). The observation suggests that DropConnect may fulfill the same function as regular dropout in AOT-SNNs, and in some cases even could be preferable.

Table 5: Performance comparisons between the AOT-SNN with DropConnect and its corresponding LTS-SNN on the CIFAR-10 dataset. The numbers after the model names represent time steps.

Model	Accuracy (%) ↑	BS ↓	NLL ↓	ECE ↓
AOT-SNN (8)	90.8±0.23	0.0144±4.0E-4	0.37±0.022	0.043±4.1E-3
AOT-SNN-DC (8)	90.5±0.37	0.0140±4.1E-4	0.32±0.010	0.026±3.0E-3
LTS-SNN (8)	88.5±0.60	0.0181±8.1E-4	0.47±0.013	0.067±3.4E-3
LTS-SNN-DC (8)	90.2±0.25	0.0161±3.6E-4	0.47±0.035	0.065±4.1E-3

Dropout rate. To investigate the impact of dropout rate on performance, we tested AOT-SNNs with dropout rates ranging from 0.1 to 0.9 in increments

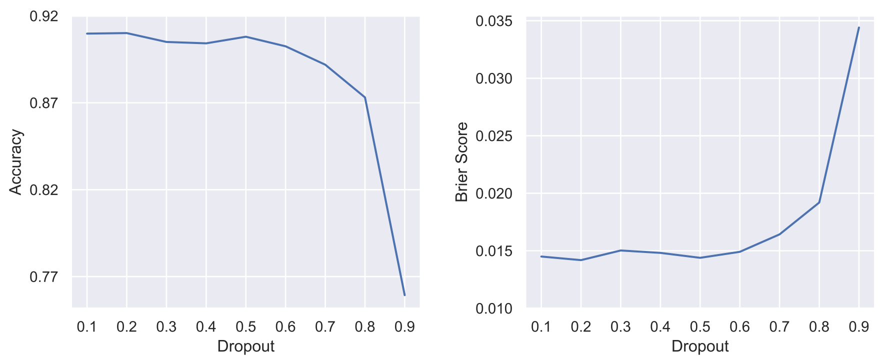


Fig. 3: The impact of dropout rate on performance of AOT-SNNs on the CIFAR-10 dataset. Dropout rates are ranging from 0.1 to 0.9 in increments of 0.1.

of 0.1. These experiments were based on our best-performing model of eight time steps and trained on the CIFAR-10 dataset separately for each amount of dropout. The accuracy and Brier score were plotted in Figure 3. The trends in accuracy, Brier score are consistent, with models having dropout rates lower than 0.5 producing flat results, followed by a decline in performance.

5 Conclusion

We proposed a novel and efficient approach for uncertainty estimation in spiking neural networks SNNs based on the MC-dropout method combined with an appropriate choice of loss-function. Our approach exploits the time-step mechanism of SNNs to enable MC-dropout in a computationally efficient manner, without introducing significant overheads during training and inference. We demonstrated that our proposed approach can be computationally efficient and performant in uncertainty quality at the same time. Future work could investigate the potential of our approach in more applications, such as speech processing and medical imaging.

Acknowledgments TS is supported by NWO-NWA grant NWA.1292.19.298. SB is supported by the European Union (grant agreement 7202070 "HBP").

Appendix

Proper Scoring Rules. A *scoring rule* $S(\mathbf{p}, y)$ assigns a value for a predictive distribution $\mathbf{p}$ and one of the labels y. A *scoring function* $s(\mathbf{p}, \mathbf{q})$ is defined as the expected score of $S(\mathbf{p}, y)$ under the distribution $\mathbf{q}$

$$s(\mathbf{p}, \mathbf{q}) = \sum_{y=1}^{K} q_y S(\mathbf{p}, y). \tag{8}$$

If a scoring rule satisfies $s(\mathbf{p}, \mathbf{q}) <= s(\mathbf{q}, \mathbf{q})$, it is called a *proper scoring rule*. If $s(\mathbf{p}, \mathbf{q}) = s(\mathbf{q}, \mathbf{q})$ implies $\mathbf{q} = \mathbf{p}$, this scoring rule is a *strictly proper scoring rule*. When evaluating quality of probabilities, an optimal score output by a proper scoring rule indicates a perfect prediction [17]. In contrast, trivial solutions could generate optimal values for an improper scoring rule [17, 8].

The two most commonly used proper scoring rules are Brier score [1] and NLL. Brier score is the squared L_2 norm of the difference between $\mathbf{p}$ and one-hot encoding of the true label y. NLL is defined as $S(\mathbf{p}, y) = -\log p(y|\mathbf{x})$ with y being the true label of the sample $\mathbf{x}$. Among these two rules, the Brier score is more recommendable because NLL can unacceptably over-emphasize small differences between small probabilities [17]. Note that proper scoring rules are often used as loss functions to train neural networks. [13, 8].

ECE. The ECE is a scalar summary statistic of calibration that approximates miscalibration [15, 10]. To calculate ECE, the predicted probabilities, $\hat{y}_n = \text{argmax}_y \mathbf{p}(y|\mathbf{x_n})$, of test instances are grouped into M equal-interval bins. The ECE is defined as

$$ECE = \sum_{m=1}^{M} f_m |o_m - e_m|, \tag{9}$$

where o_m is the fraction of corrected classified instances in the m^{th} bin, e_m the average of all the predicted probabilities in the m^{th} bin, and f_m the fraction of all the test instances falling into the m^{th} bin. The ECE is not a proper scoring rule and thus optimum ECEs could come from trivial solutions.

References

1. Brier, G.W., et al.: Verification of forecasts expressed in terms of probability. Monthly weather review **78**(1), 1–3 (1950)
2. Damianou, A., Lawrence, N.D.: Deep gaussian processes. In: Artificial intelligence and statistics. pp. 207–215. PMLR (2013)
3. 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: CVPR. pp. 2661–2671 (2021)
4. Gal, Y.: Uncertainty in Deep Learning. Ph.D. thesis, Department of Engineering, University of Cambridge (2016)
5. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: ICML. pp. 1050–1059. PMLR (2016)
6. Gawlikowski, J., Tassi, C.R.N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., et al.: A survey of uncertainty in deep neural networks. arXiv preprint arXiv:2107.03342 (2021)
7. Gerstner, W., Kistler, W.M.: Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press (Aug 2002)
8. Gneiting, T., Raftery, A.E.: Strictly proper scoring rules, prediction, and estimation. Journal of the American statistical Association **102**(477), 359–378 (2007)
9. Graves, A.: Practical variational inference for neural networks. NIPS **24** (2011)

10. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: ICML. pp. 1321–1330. PMLR (2017)
11. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: ICLR (2019), https://openreview.net/forum?id=HJz6tiCqYm
12. Jang, H., Simeone, O.: Multisample online learning for probabilistic spiking neural networks. IEEE Trans Neural Netw Learn Syst **33**(5), 2034–2044 (May 2022)
13. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. NIPS **30** (2017)
14. Mackay, D.J.C.: Bayesian methods for adaptive models. Ph.D. thesis, California Institute of Technology (1992)
15. Naeini, M.P., Cooper, G., Hauskrecht, M.: Obtaining well calibrated probabilities using bayesian binning. In: AAAI (2015)
16. 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 Processing Magazine **36**(6), 51–63 (2019)
17. Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., Snoek, J.: Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. NIPS **32** (2019)
18. Pouget, A., Beck, J.M., Ma, W.J., Latham, P.E.: Probabilistic brains: knowns and unknowns. Nature neuroscience **16**(9), 1170–1178 (2013)
19. Rathi, N., Srinivasan, G., Panda, P., Roy, K.: Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In: ICML (2020), https://openreview.net/forum?id=B1xSperKvH
20. Savin, C., Deneve, S.: Spatio-temporal representations of uncertainty in spiking neural networks. Adv. Neural Inf. Process. Syst. (2014)
21. Schuman, C.D., Kulkarni, S.R., Parsa, M., Mitchell, J.P., Date, P., Kay, B.: Opportunities for neuromorphic computing algorithms and applications. Nature Computational Science **2**(1), 10–19 (2022)
22. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)
23. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: ICML. pp. 1058–1066. PMLR (2013)
24. Wilson, A.G., Izmailov, P.: Bayesian deep learning and a probabilistic perspective of generalization. NIPS **33**, 4697–4708 (2020)
25. Yin, B., Corradi, F., Bohté, S.M.: Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. Nature Machine Intelligence **3**(10), 905–913 (2021)
26. Yin, B., Corradi, F., Bohté, S.M.: Accurate online training of dynamical spiking neural networks through forward propagation through time. Nature Machine Intelligence (2023, to appear)
27. Yue, Y., Baltes, M., Abujahar, N., Sun, T., Smith, C.D., Bihl, T., Liu, J.: Hybrid spiking neural network fine-tuning for hippocampus segmentation. arXiv preprint arXiv:2302.07328 (2023)