

# Spikemax: Spike-based Loss Methods for Classification

Sumit Bam Shrestha<sup>1\*</sup>, Longwei Zhu<sup>1</sup>, Pengfei Sun<sup>1,2</sup>

<sup>1</sup>*Institute for Infocomm Research, Agency for Science, Technology and Research (A\*STAR), Singapore*

<sup>2</sup>*Ghent University, Belgium*

Email: {sumit\_bam@i2r.a-star.edu.sg, wayne\_zhu@i2r.a-star.edu.sg, pengfei.sun@ugent.be}

**Abstract**—Spiking Neural Networks (SNNs) are a promising research paradigm for low power edge-based computing. Recent works in SNN backpropagation has enabled training of SNNs for practical tasks. However, since spikes are binary events in time, standard loss formulations are not directly compatible with spike output. As a result, current works are limited to using mean-squared loss of spike count. In this paper, we formulate the output probability interpretation from the spike count measure and introduce spike-based negative log-likelihood measure which are more suited for classification tasks especially in terms of the energy efficiency and inference latency. We compare our loss measures with other existing alternatives and evaluate using classification performances on three neuromorphic benchmark datasets: MNIST, DVS Gesture and N-TIDIGITS18. In addition, we demonstrate state of the art performances on these datasets, achieving faster inference speed and less energy consumption.

## I. INTRODUCTION

Error backpropagation is the key technology that propels the current deep learning revolution. The basic strategy for training these Artificial Neural Networks (ANNs) is to compute the gradient of a loss measure of the output of the network and its desired target, and use gradient descent steps to arrive at an optimal/sub-optimal set of parameters. It has been very successful in various applications ranging from image classification, object recognition, object tracking, signal processing, natural language processing etc. and many more.

Spiking Neural Networks (SNNs) have garnered increased interest in recent times as a promising option for extremely low power edge-based computing devices with the development of neuromorphic hardware [1]–[5]. They are a more biologically plausible form of ANNs whose computational unit is a spiking neuron. Therefore, it's input and output are both in the form of spike events in time. Until recently, using backpropagation for SNNs has been quite challenging, especially for deep architectures. ANN to SNN conversion strategies [6]–[11] were the only available options to configure practical SNN systems. These methods, although effective, require a substantially large amount of time steps to produce a reliable output. In addition, they cannot be used to process event-based data directly as the data is already in the form of spike which cannot be handled natively by an ANN.

With recent efforts like [12]–[17], it is now possible to train relatively deep SNNs (by SNN standards) using backpropagation. In these works, the loss measure is usually a mean-square error of the spike count at the output or a loss measure of the internal state, i.e. membrane potential of the output neurons [18], [19]. In this paper, we propose spike-based negative log-likelihood losses which are better suited for classification tasks. We first propose an output probability estimate from the output spike-trains and then subsequently use it in the cross-entropy setting. We further derive the gradient of this spike-based loss formulation. The proposed loss methods are hyperparameter free which is a plus. We call this spike-based loss formulation *spikemax*.

We have released<sup>1</sup> the implementation of *spikemax* loss as an add-on to SLAYER-PyTorch [15] repository which is an SNN backpropagation implementation in PyTorch.

We experimentally evaluate the effectiveness of our loss measure with other existing alternatives on three different neuromorphic benchmark problems: MNIST [20], DVS Gesture [21], and N-TIDIGITS18 [22]. We use these neuromorphic benchmark datasets for evaluation because the SNN can directly process the raw data. As a result, there is no contribution of input to spike conversion method and the performance variation is solely due to the SNN training method. We report state of the art, if not competitive, results on these benchmark problems. In addition we also analyze the inference latency and spike activity of the resulting network which translate to power consumption in the hardware implementation and evaluate the methods on based on their relative power profile.

The rest of the paper is organized as follows. We will briefly introduce the preliminaries of an SNN and its gradient-based training. Then, we will delve into spike-based losses. Here we will formally introduce our *spikemax* loss and its variants *spikemax<sub>g</sub>* and *spikemax<sub>s</sub>*. We follow it with experiments on the aforementioned neuromorphic benchmark datasets and their analysis. Finally, we will present the concluding remarks.

## II. BACKGROUND

In this section, we will briefly introduce the preliminary concepts of an SNN and gradient based training of an SNN in the spiking domain.

<sup>1</sup>The code for *spikemax* loss is publicly available at: <https://github.com/lava-nc/lava-dl/blob/main/src/lava/lib/dl/slayer/loss.py>

\*Corresponding Author

### A. SNN Model

An SNN is a biologically plausible form of an ANN. Succinctly put, an SNN is a special form of an ANN which uses a spiking neuron as its activation/computational unit. A spiking neuron is a mathematical abstraction of a biological neuron. As a result an SNN inherits the intrinsic property of information exchange in the form of events in time known as spikes.

There are many models of a spiking neuron such as Leaky Integrate and Fire (LIF) [23], [24], Adaptive Exponential Integrate and Fire (AdEx) [25], Izhikevich [26], Spike Response Model (SRM) [27], Hodgkin Huxley [28], etc. which represents the behavior of a biological neuron with a varying degree of realism. In this paper, we focus on the simple yet versatile SRM model of a spiking neuron. A SRM model of a spiking neuron is completely defined by a spike response kernel,  $\varepsilon(\cdot)$  which describes the temporal response of the neuron to an input spike; a refractory kernel,  $\nu(\cdot)$ , which describes the post-spike behavior of the neuron; and a neuron threshold,  $\vartheta$ , which describes how easily the neuron spikes. We use  $\varepsilon(t) = t/\tau_s \exp(1 - t/\tau_s)\Theta(t)$  and  $\nu(t) = -2\vartheta t/\tau_r \exp(1 - t/\tau_r)\Theta(t)$  as the spike response kernel and refractory kernels where  $\tau_s$  and  $\tau_r$  are their respective time constants.

Mathematically, for a layer  $l$  with  $N_l$  neurons and inbound spikes,  $\mathbf{s}^{(l-1)}(t)$ , through the synaptic weights  $\mathbf{W}^{(l-1)} \in \mathbb{R}^{N_l \times N_{l-1}}$ , the internal voltage or the membrane potential, of the neuron is described as

$$\mathbf{u}^{(l)}(t) = \mathbf{W}^{(l-1)} \left( \varepsilon * \mathbf{s}^{(l-1)} \right) (t) + \left( \nu * \mathbf{s}^{(l)} \right) (t) \quad (1)$$

and the output of the layer is described as

$$\mathbf{s}^{(l)}(t) = f_s \left( \mathbf{u}^{(l)}(t) \right) \quad (2)$$

where  $f_s(u(t)) = \sum_{t_f} \delta(t - t_f) : \{t_f : u(t_f) \geq \vartheta\}$  is the spike function.

### B. Gradient Based Training of SNN

Error backpropagation has been the workhorse for the current advancement in deep learning. One of the earliest attempts for error backpropagation in SNN was formulated in SpikeProp [29]. The authors used event-based error backpropagation. Further developments in the work have been proposed in [30]–[33]

On the other end of the gradient-based learning for SNN spectrum is dt-based error backpropagation. These approaches [12]–[16], [34] have been successful in recent times for training deep SNNs. There are two main obstacles to be tackled for dt-based error backpropagation in SNN. The first is the derivative of the spike function (it is also an issue in event-based backpropagation) and the second is the temporal dependencies of the signals.

Since, the spike generation mechanism in a spiking neuron is a discontinuous function, its derivative does not exist in a mathematical sense. However, it is essential in the backpropagation chain for gradient propagation. This issue can be circumvented by using a proxy function for the spike

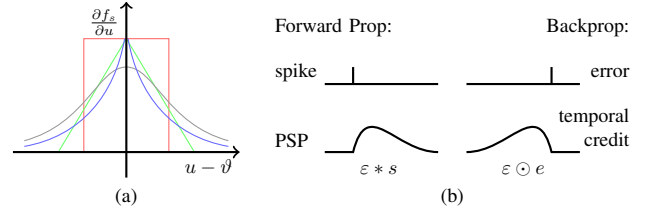


Fig. 1: Error backpropagation in SNN (a) Spike function derivative. (b) Temporal credit assignment of an impulse error during backpropagation.

function derivative. There are different possible representations of this proxy function such as spike escape rate function [15], or simply a linear function with finite voltage support [12], or a half-sigmoid function [34]. These proxy functions are best interpreted as a surrogate gradient function as described in [13]. Various forms of surrogate gradient functions are depicted in Figure 1 (a).

Another important, and often overlooked (often for computational reasons) aspect of dt-based error backpropagation in SNN. However, it is important because of the inherent temporal nature of a spiking neuron. An input spike at a point of time to a spiking neuron induces a post-synaptic response over a range of time. Therefore, it is necessary to compensate for this behavior during error backpropagation. As shown in Slayer [15], this forward temporal effect (represented by the convolution operation in time:  $*$ ) must be compensated by similar temporal distribution of error signal back in time i.e. correlation operation ( $\odot$ ). This is illustrated in Figure 1 (b).

In this work, we use SLAYER-PyTorch [15] framework as the SNN training method as it is publicly available<sup>2</sup> and has proven to be effective for training SNNs.

### III. SPIKE BASED LOSSES

The training cost of the SNN, given a target spike train  $\hat{\mathbf{s}}(t)$  is usually formulated as

$$\mathcal{L} = \int_0^T l(\mathbf{s}^{(n_l)}(t), \hat{\mathbf{s}}(t)) dt \quad (3)$$

where  $n_l$  is the last/output layer and  $t \in [0, T]$  is the simulation time interval. In spike regression problem, the target spike train is usually known. In this case, with  $l(t) = \{\varepsilon * (\mathbf{s}^{(n_l)} - \hat{\mathbf{s}})(t)\}^2$ , the cost can be formulated as a van-Rossum distance [35].

However, the target spike train is not known for classification problems. Typically, in SNN based classification, we want the true output neuron to spike the most number of times. One of the prevalent strategies is to maximize the membrane potential of the true output neuron [14], [36], usually at the end of simulation time. One would have to wait till the end of the simulation to get the classification output. In addition, in a real neuromorphic hardware, the membrane potential is not visible. Although maximizing membrane potential usually

<sup>2</sup>SLAYER-PyTorch is publicly available at: <https://github.com/bamsumit/slayerPytorch>

results in more spike count, however, it does not guarantee it always.

Another common strategy is to directly maximize the spike-count or spike-rate [12], [15]. The *spike-rate* loss can be formulated as

$$\mathcal{L} = \left\{ \frac{1}{T} \int_0^T s^{(n_i)}(t) dt - \hat{r} \right\}^2 \quad (4)$$

where  $\hat{r}$  is the desired spike rate at the output.

Next, we will formulate negative log-likelihood losses based on the probability interpretation of spikes. This is the main theoretical contribution of the paper.

#### A. Probability Interpretation of Spikes

Each spike event at the output is like a vote for the output being that particular class. The count of the spike, thus, represents the totality of the votes, which can be defined as

$$c_i(t) = \int_{t-W}^t s_i(\tau) d\tau \quad (5)$$

where  $W$  is the spike count estimate window and  $i \in \{0, 1, \dots, N_{n_l} - 1\}$  is the neuron index. Then the probability estimate of the output at time  $t$  is

$$p_i(t) = \frac{c_i(t)}{\sum_{i=0}^{N_{n_l}-1} c_i(t)} \quad (6)$$

One can make the length of the sliding window as large as the simulation interval. In that case, the global probability estimate is

$$p_i = \frac{c_i}{\sum_{i=0}^{N_{n_l}-1} c_i}, \quad \text{where } c_i = \int_0^T s_i(\tau) d\tau \quad (7)$$

Note the absence of time dependency disambiguates between running probability estimate and global probability estimate.

#### B. Spikemax Losses

With the probability estimate from the spike output and one-hot target output  $\hat{y}_i(t) = \hat{y}_i$ , we formulate the negative log-likelihood loss as follows:

$$\mathcal{L} = \frac{1}{T} \int_0^T l(t) dt, \quad l(t) = - \sum_i \hat{y}_i(t) \log(p_i(t)) \quad (8)$$

We term this formulation of loss as *spikemax*. The gradients with respect to spikemax loss are

$$\frac{\partial \mathcal{L}}{\partial p_i(t)} = \frac{\partial \mathcal{L}}{\partial l(t)} \frac{\partial l(t)}{\partial p_i(t)} = - \frac{\hat{y}_i(t)}{p_i(t)} \quad (9)$$

$$\frac{\partial p_k(t)}{\partial c_i(t)} = \begin{cases} \frac{1-p_i(t)}{\sum_i c_i(t)} & \text{if } i = k \\ -\frac{p_k(t)}{\sum_i c_i(t)} & \text{otherwise} \end{cases} \quad (10)$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial s_i(t)} &= \left( \frac{\partial \mathcal{L}}{\partial p_i(t)} \frac{\partial p_i(t)}{\partial c_i(t)} + \sum_{k \neq i} \frac{\partial \mathcal{L}}{\partial p_k(t)} \frac{\partial p_k(t)}{\partial c_i(t)} \right) \frac{\partial c_i(t)}{\partial s_i(t)} \\ &= \frac{p_i(t) - \hat{y}_i(t)}{c_i(t)/W}. \end{aligned} \quad (11)$$

Note, for numerical stability, an infinitesimal count can be added. In addition, using the global probability estimate, we define the *spikemax<sub>g</sub>* loss and derive it's gradient as

$$\mathcal{L} = - \sum_i \hat{y}_i \log(p_i), \quad \frac{\partial \mathcal{L}}{\partial s_i(t)} = \frac{p_i - \hat{y}_i}{c_i/T}. \quad (12)$$

It is also possible to use softmax probability estimate from spike count and use negative log-likelihood loss.

$$p_i^s = \frac{\exp(c_i)}{\sum_{i=0}^{N_{n_l}-1} \exp(c_i)} \quad (13)$$

$$\mathcal{L} = - \sum_i \hat{y}_i \log(p_i^s), \quad \frac{\partial \mathcal{L}}{\partial s_i(t)} = p_i^s - \hat{y}_i \quad (14)$$

We will simply call it *spikemax<sub>s</sub>* in the rest of this paper.

With losses and their respective gradient formulation derived, we can incorporate them in the SLAYER-PyTorch computational graph to train an SNN. The advantage of the proposed loss methods in contrast to the common spike-rate loss formulation is that spikemax losses are parameter free i.e. there is no need to tune the output spike rates manually. In addition, spikemax loss optimizes the spiking output for producing continuous classification output using a sliding window, therefore, is more suitable for use in a real-time system.

## IV. EXPERIMENTS AND RESULTS

In this section, we will compare the performance of spikemax losses with spike-rate loss on three different neuromorphic classification tasks. We use SLAYER-PyTorch [15] as our SNN backpropagation framework. Our loss methods are implemented on top of it.

All the results presented in this paper are averaged over 5 different independent trials. Same random seeds were used for each of the loss methods for a fair comparison. We will follow the shorthand notation in [15] to represent the architecture: layers are separated by -, spatial dimensions are separated by x, an  $N \times N$  convolution filter with  $K$  channels is represented by KcN, an  $N \times N$  aggregate pooling filter is represented by Na and a dense layer with  $N$  neurons is represented by the number itself. Note that the convolution and dense layer neurons also have trainable axonal delays [37]. In all our experiments, we use neuron threshold  $\vartheta = 10$  mv, and sampling time of 1 ms. The spike response time constant,  $\tau_s$ , and refractory time constant,  $\tau_r$ , were optimized for each of the datasets. We consider the datasets with inputs as spike events which are then fed directly to the SNN. This also eliminates spike encoding out of the processing pipeline and hence we can focus on SNN training only.

We will compare the results based on the overall accuracy obtained. We will also see how the methods compare in terms of inference latency. In a real system, the network that can classify with reliable accuracy faster does not need to look at the entire duration of the input, therefore, can result in a power efficient inference system. We compare the network's classification accuracy versus the time-length

TABLE I: Benchmark Classification Results

|               | Method                         | Params     | Accuracy                            |
|---------------|--------------------------------|------------|-------------------------------------|
| NMNIST        | Lee et al. [38]                | 1,857,600  | 98.66%                              |
|               | Wu et al. [12]                 | 17,664,256 | 99.53%                              |
|               | Spike-based BP. [39]           | 68,537,760 | <b>99.61%</b>                       |
|               | spike-rate                     | 2,171,728  | 99.33 $\pm$ 0.03%                   |
|               | spikemax                       | 2,171,728  | 99.27 $\pm$ 0.02%                   |
|               | spikemax <sub>g</sub>          | 2,171,728  | 99.33 $\pm$ 0.04%                   |
|               | spikemax <sub>s</sub>          | 2,171,728  | 99.26 $\pm$ 0.06%                   |
| DVS Gesture   | TrueNorth [21]                 | 1,992,476  | 91.77(94.59)%                       |
|               | DECOLLE [36]                   | 1,245,696  | 95.54%                              |
|               | Ghosh et al. [40] <sup>†</sup> | 2,119,080  | 95.94%                              |
|               | Spike-based BP. [39]           | 6,798,292  | <b>97.57%</b>                       |
|               | spike-rate                     | 1,068,368  | 96.21 $\pm$ 0.63%                   |
|               | spikemax                       | 1,068,368  | 95.83 $\pm$ 0.48%                   |
|               | spikemax <sub>g</sub>          | 1,068,368  | 95.53 $\pm$ 0.37%                   |
| N-TDIDIGITS18 | spikemax <sub>s</sub>          | 1,068,368  | 95.15 $\pm$ 0.65%                   |
|               | GRU-RNN [22] <sup>†</sup>      | 109,200    | 90.90%                              |
|               | Phased-LSTM [22] <sup>†</sup>  | 610,500    | 91.25%                              |
|               | ST-RSBP [41]                   | 351,241    | 93.63 $\pm$ 0.27%                   |
|               | spike-rate                     | 84,736     | <b>94.19 <math>\pm</math> 0.18%</b> |
|               | spikemax                       | 84,736     | 93.21 $\pm$ 0.32%                   |
|               | spikemax <sub>g</sub>          | 84,736     | 93.01 $\pm$ 0.38%                   |
|               | spikemax <sub>s</sub>          | 84,736     | 92.43 $\pm$ 0.25%                   |

<sup>†</sup> Non SNN implementation.

of the input sequence to evaluate the network’s latency for classification. Lower classification latency means that we do not need to process the rest of the inputs which saves the energy consumption of the end system. In addition, we will compare the networks in terms of average spike count. Since we are not implementing the system in a hardware, spike count is a good measure of the relative inference energy of the network [14]. More spike activity usually means more energy is consumed by the neuromorphic hardware during inference. Ideally, one would want higher accuracy, earlier inference and low spike count.

#### A. MNIST digit classification

MNIST dataset [20] is the neuromorphic version of standard MNIST images. The images are converted into spikes using a Dynamic Vision Sensor (DVS) moving on a pan-tilt unit in a three-saccadic motion, each lasting 100 ms long. Resulting event-data is  $34 \times 34$  pixels, with “on” and “off” spike-events. The events last for 300 ms per sample. In our experiments, we do not stabilize the MNIST data to compensate the saccadic movement. Raw spike data is used, without any processing. The train and test split is the same as standard MNIST: 60,000 training samples and 10,000 testing samples.

We use a spiking CNN architecture with the following specification:  $34 \times 34 \times 2 - 16c5 - 2a - 32c3 - 2a - 64c3 - 512 - 10$ . The neuron time constants are  $(\tau_s, \tau_r) = (1, 1)$  ms, the target spike rate are  $(\text{True}, \text{False}) = (0.2, 0.04)$  for spike-rate loss, and  $W = 30$  for spikemax loss. The overall classification results for our loss methods are listed in Table I with other reported benchmarks. The overall accuracy for spike-rate, spikemax<sub>s</sub>, spikemax<sub>g</sub> and spikemax loss are very similar at around 99.3%. The result is lower than the best reported

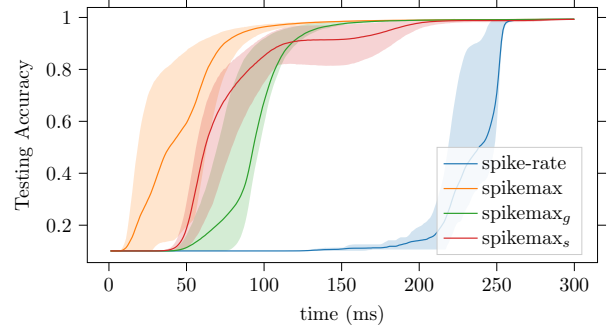


Fig. 2: Testing accuracy over inference runtime for MNIST classification.

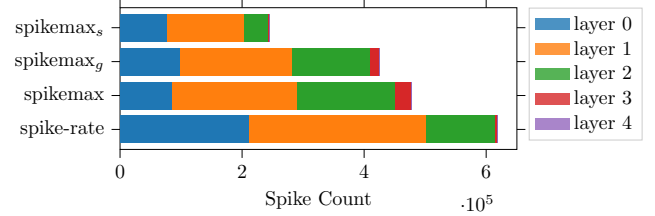


Fig. 3: Spike count distribution per layer for MNIST classification.

accuracy of 99.61% [39] on NMNIST. However, the network we use is significantly smaller ( $31 \times$  less parameter).

Figure 2 shows the plot of accuracy and inference latency. Spikemax networks clearly show the minimum inference latency ( $\approx 70$  ms) among all the methods whereas spike-rate networks demonstrate the slowest inference latency ( $\approx 250$  ms). The networks trained with negative log-likelihood losses (spikemax<sub>s</sub>, spikemax and spikemax<sub>g</sub>) clearly show faster inference times compared to spike-rate networks.

The plot of average spike count per layer is shown in Figure 3. The networks trained with spike-rate spike the most whereas spikemax<sub>s</sub> networks demonstrate minimum spike.

#### B. DVS Gesture classification

The DVS Gesture [21] is a public dataset by IBM Research. It is a neuromorphic dataset that consists of 29 different individuals performing 11 hand gestures (clapping, drumming, hand-wave, etc.) under three different lighting environments captured using a DVS camera. The standard train-test split of first 23 subjects for training and the last 6 subjects for testing is used. The data is  $128 \times 128$  pixels wide with “on” and “off” polarity. We train on randomly sampled 300 ms long sequence and test on first 1.5 s event sequence.

The network architecture is  $128 \times 128 \times 2 - 4a - 16c5 - 2a - 32c3 - 2a - 512 - 11$ . The neuron time constants are  $(\tau_s, \tau_r) = (5, 5)$  ms, the target spike rate are  $(\text{True}, \text{False}) = (0.35, 0.07)$  for spike-rate loss, and  $W = 35$  for spikemax loss. The results are listed in Table I. We report the best accuracy of 96.97% using spike-rate loss. spikemax<sub>s</sub>, spikemax and spikemax<sub>g</sub> results are also very good: better than the results using non-spiking CNN [40] and other SNN

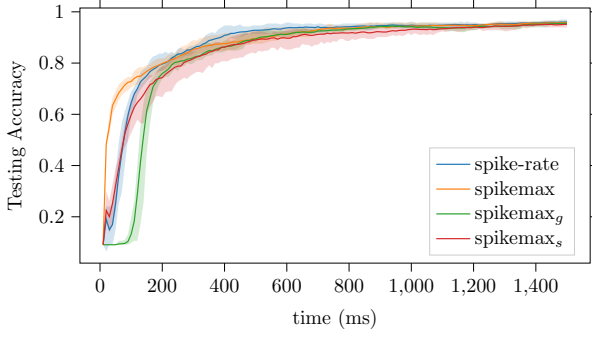


Fig. 4: Testing accuracy over inference runtime for DVS Gesture classification.

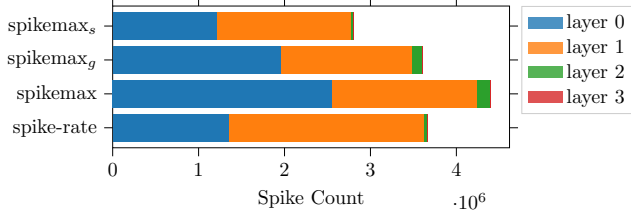


Fig. 5: Spike count distribution per layer for DVS Gesture classification.

based approaches. Compared with the latest best-reported result [39], we only use  $6\times$  less parameter.

Figure 4 shows the plot of accuracy versus inference time for all the loss methods. We can observe that spikemax loss is able to reliably predict the classification output as early as approx. 50 ms. The networks trained using spike-rate and spikemax<sub>s</sub> show inference latency of approx. 100 ms and the networks trained using spikemax show inference latency of approx. 150 ms. Past 200 ms, the inference accuracy of all the networks are similar and saturate after 700 ms.

The average spike count distribution of each layer of the networks is shown in Figure 5 for the final inference time of 1500 ms. Spikemax<sub>g</sub> networks demonstrate the least spike activity among all, therefore, are relatively energy efficient. However, these networks require a longer inference time.

### C. N-TIDIGITS18 audio classification

N-TIDIGITS18 [22] is the neuromorphic version of the TIDIGITS [42] audio classification dataset. The TIDIGITS audio signals were converted into a 64 channel spiking events using a silicon cochlea sensor: CochleaAMS1b [43]. The output classes are digit utterances from “zero” to “nine” and “oh”. We use the standard train-test split of the dataset as used in [22].

We use a fully connected architecture, 64-256-256-11, with neuron time constants,  $(\tau_s, \tau_r) = (5, 5)$  ms,  $(\text{True}, \text{False}) = (0.2, 0.02)$ , and  $W = 40$ , in our experiments. The classification results are tabulated in Table I. We achieve the best performance of 94.45% using spike-rate loss, better than the recurrent SNN architecture [41] with  $4\times$  less parameters. Networks trained with spikemax also show good

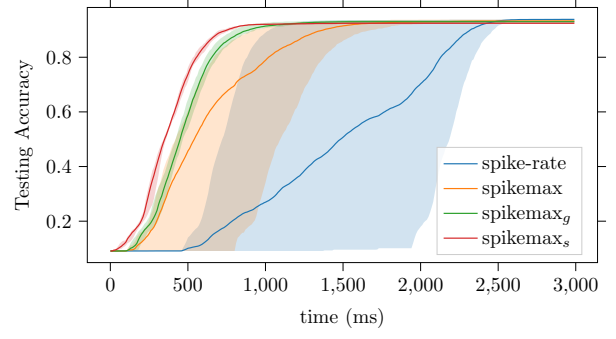


Fig. 6: Testing accuracy over inference runtime for N-TIDIGITS18 classification.

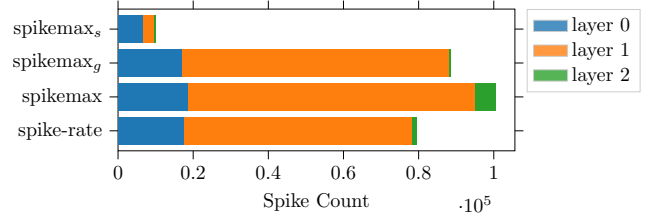


Fig. 7: Spike count distribution per layer for N-TIDIGITS18 classification.

classification performance. Note that we achieve better results than the conventional LSTM network [22] for all the loss methods with  $7\times$  less network parameters.

Accuracy versus inference time is plotted in Figure 6. It is clear that the networks trained with negative log-likelihood based losses (spikemax<sub>s</sub>, spikemax and spikemax<sub>g</sub>) are able to classify much earlier than the networks trained with spike-rate: approx. 600 ms compared to approx. 2000 ms.

In Figure 7, the average spike count per layer for the trained networks is plotted. The spikemax<sub>s</sub> networks demonstrate a remarkably low spike count. They also show earlier inference amongst all other networks.

## V. DISCUSSION

In this paper, we have proposed spike-based negative log-likelihood based losses suitable to train an SNN for classification tasks. We demonstrate state of the art, if not competitive, classification performances on neuromorphic audio and video datasets using these loss methods. We focus on purely neuromorphic datasets so that we exclude the effectiveness of spike-encoding process in the comparison and work directly on the raw spike data.

We also compare the resulting networks in terms of their inference time as well as the spike activity of the network. These measures are a proxy for the energy consumed per inference in a neuromorphic hardware. From the results, we can clearly see that the proposed losses result in networks that start to produce usable inference results earlier. Spikemax<sub>s</sub> loss, in particular, demonstrated considerably low spike activity in the trained networks in two of the three benchmarks.

In addition, we also demonstrate very good classification performances, using networks with fewer learnable parameters. Particularly in N-TIDIGITS18 audio classification tasks, our networks were able to outperform spiking recurrent networks as well as conventional LSTM and GRU networks. This performance is not solely due to the loss method.

In conclusion, we show a way to use negative log-likelihood loss in the spiking domain by estimating the probability confidence from spikes. We see that these losses demonstrate improvements in spike-based classification tasks, especially from the energy perspective. The idea of probability estimate can also be extended for other spike-based learning tasks. We will consider these avenues in the future.

## VI. ACKNOWLEDGMENTS

This research is partially supported by Programmatic grant no. A1687b0033 from the Singapore government's Research, Innovation and Enterprise 2020 plan (Advanced Manufacturing and Engineering domain) and the Flemish Government under the "Onderzoeksprogramma Artificiele Intelligentie (AI) Vlaanderen".

## REFERENCES

- [1] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.
- [2] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.
- [3] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [4] Alexander Neckar, Sam Fok, Ben V Benjamin, Terrence C Stewart, Nick N Oza, Aaron R Voelker, Chris Eliasmith, Rajit Manohar, and Kwabena Boahen, "Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 144–164, 2018.
- [5] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al., "Towards artificial general intelligence with hybrid tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.
- [6] Eric Hunsberger and Chris Eliasmith, "Spiking deep networks with LIF neurons," *CoRR*, vol. abs/1510.08829, 2015.
- [7] Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proceedings of the National Academy of Sciences*, vol. 113, no. 41, pp. 11441–11446, 2016.
- [8] Qian Liu, Yunhua Chen, and Steve B. Furber, "Noisy softplus: an activation function that enables snns to be trained as anns," *CoRR*, vol. abs/1706.03609, 2017.
- [9] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–8.
- [10] Peter U Diehl, Guido Zarrella, Andrew Cassidy, Bruno U Pedroni, and Emre Neftci, "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 2016, pp. 1–8.
- [11] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, pp. 682, 2017.
- [12] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 1311–1318.
- [13] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke, "Surrogate gradient learning in spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, pp. 61–63, 2019.
- [14] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in Neuroscience*, vol. 14, 2020.
- [15] Sumit Bam Shrestha and Garrick Orchard, "SLAYER: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., pp. 1412–1421. Curran Associates, Inc., 2018.
- [16] Yingyezhe Jin, Wenrui Zhang, and Peng Li, "Hybrid macro/micro level backpropagation for training deep spiking neural networks," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., pp. 7005–7015. Curran Associates, Inc., 2018.
- [17] Priyadarshini Panda, Sai Apama Aketi, and Kaushik Roy, "Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization," *Frontiers in Neuroscience*, vol. 14, 2020.
- [18] Pengjie Gu, Rong Xiao, Gang Pan, and Huajin Tang, "Stca: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. 7 2019, pp. 1366–1372. International Joint Conferences on Artificial Intelligence Organization.
- [19] Chenxiang Ma, Junhai Xu, and Qiang Yu 0005, "A deep spike learning through critical time points," in *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*. 2021, pp. 1–8, IEEE.
- [20] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, pp. 437, 2015.
- [21] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha, "A low power, fully event-based gesture recognition system," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [22] Jithendar Anumula, Daniel Neil, Tobi Delbruck, and Shih-Chii Liu, "Feature representations for neuromorphic audio spike streams," *Frontiers in Neuroscience*, vol. 12, pp. 23, 2018.
- [23] Wulfram Gerstner and Werner M Kistler, *Spiking neuron models: Single neurons, populations, plasticity*, Cambridge university press, 2002.
- [24] Hélène Paugam-Moisy and Sander M. Bohte, *Handbook of Natural Computing*, vol. 1, chapter Computing with Spiking Neuron Networks, pp. 335–376, Springer Berlin Heidelberg, 1st edition, 2011.
- [25] Romain Brette and Wulfram Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of Neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 2005.
- [26] E.M. Izhikevich, "Simple model of spiking neurons," *Neural Networks, IEEE Transactions on*, vol. 14, no. 6, pp. 1569–1572, 2003.
- [27] Wulfram Gerstner, "Time structure of the activity in neural network models," *Phys. Rev. E*, vol. 51, pp. 738–758, Jan 1995.
- [28] Alan L Hodgkin and Andrew F Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500, 1952.

- [29] Sander M Bohte, Joost N Kok, and Han La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1, pp. 17–37, 2002.
- [30] Sumit Bam Shrestha and Qing Song, "Robustness to training disturbances in SpikeProp learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 7, pp. 3126–3139, 2018.
- [31] Sumit Bam Shrestha and Qing Song, "Event based weight update for learning infinite spike train," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec 2016, pp. 333–338.
- [32] Yan Xu, Xiaoqin Zeng, Lixin Han, and Jing Yang, "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks," *Neural Networks*, vol. 43, no. 0, pp. 99 – 113, 2013.
- [33] Iulia M. Comsa, Krzysztof Potempa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala, "Temporal coding in spiking neural networks with alpha synaptic function," 2019.
- [34] Friedemann Zenke and Surya Ganguli, "SuperSpike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018.
- [35] Justin Dauwels, François Vialatte, Theophane Weber, and Andrzej Cichocki, "On similarity measures for spike trains," in *Advances in Neuro-Information Processing*, pp. 177–185. Springer, 2008.
- [36] Jacques Kaiser, Hesham Mostafa, and Emre Neftci, "Synaptic plasticity dynamics for deep continuous local learning (decolle)," *Frontiers in Neuroscience*, vol. 14, pp. 424, 2020.
- [37] Pengfei Sun, Longwei Zhu, and Dick Botteldooren, "Axonal delay as a short-term memory for feed forward deep spiking neural networks," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 8932–8936.
- [38] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in Neuroscience*, vol. 10, pp. 508, 2016.
- [39] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2661–2671.
- [40] Rohan Ghosh, Anupam Gupta, Andrei Nakagawa, Alcimar Soares, and Nitish Thakor, "Spatiotemporal filtering for event-based action recognition," *arXiv preprint arXiv:1903.07067*, 2019.
- [41] Wenrui Zhang and Peng Li, "Spike-train level backpropagation for training deep recurrent spiking neural networks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, Eds., pp. 7802–7813. Curran Associates, Inc., 2019.
- [42] R Gary Leonard and George Doddington, "Tidigits speech corpus," *Texas Instruments, Inc*, 1993.
- [43] Shih-Chii Liu, André van Schaik, Bradley A Minch, and Tobi Delbruck, "Asynchronous binaural spatial audition sensor with  $2\times 64\times 4$  channel output," *IEEE transactions on biomedical circuits and systems*, vol. 8, no. 4, pp. 453–464, 2013.