A surrogate gradient spiking baseline for speech command recognition

Alexandre Bittar 1,* , Philip N. Garner 1

1 Idiap Research Institute, Speech and Audio Processing, Martigny, Switzerland

Correspondence*:
Alexandre Bittar
abittar@idiap.ch

ABSTRACT

Artificial neural networks (ANNs) are the basis of recent advances in artificial intelligence (AI); they typically use real valued neuron responses. By contrast, biological neurons are known to operate using spike trains. In principle, spiking neural networks (SNNs) have a greater representational capability than ANNs; however their adoption has been held back by both a lack of stable training algorithms and a lack of compatible baselines. We begin with a fairly thorough review of literature around the conjunction of ANNs and SNNs. Focusing on surrogate gradient approaches, we proceed to define a simple but relevant evaluation based on recent speech command tasks. After evaluating a representative selection of architectures, we show that a combination of adaptation, recurrence and surrogate gradients can yield spiking architectures that not only compete with ANN solutions, but even exceed their performance in many cases. We conclude tangibly that SNNs are appropriate for future research in AI, and more speculatively that they may also assist in inference about biological function.

Keywords: spiking neurons, physiologically plausible models, deep learning, signal processing, speech recognition

1 INTRODUCTION

Recent years have seen the success of artificial neural networks (ANNs) in speech processing technologies. The neurons traditionally used in these modern networks take real numbers as inputs and produce real-valued outputs. In biological neurons, the information is transmitted in the form of binary sequences of events, called spike trains. The neurons in ANNs can be seen as an instantaneous firing rate approximation of biological spiking neurons, so that the information about the individual timings of the spikes is neglected. Several neuroscience studies suggest that precise spike timings are important in transmitting information, especially in the visual cortex and in auditory neurons (Mainen and Sejnowski, 1995; Van Rullen and Thorpe, 2001; Butts et al., 2007; Gollisch and Meister, 2008). With the idea of simulating brain-like networks to process information, this firing rate interpretation of spikes can be improved to spiking neuron models. Physiologically plausible mathematical models have been developed to describe the neuronal dynamics (Gerstner and Kistler, 2002; Izhikevich, 2007). The resulting spiking neurons constitute the building blocks of spiking neural networks (SNNs), and have been called the third generation of neural network models (Maass, 1997). The temporal dimension of spike trains make them naturally adapted to sequential input data, such as speech, for which, SNNs in principle have a higher representation capability compared to traditional ANNs (Kasabov, 2019).
Another motivation towards SNNs is that the sparsity of spikes over time can allow energy-efficient hardware implementations (Davies et al., 2018; Roy et al., 2019; Panda et al., 2020; Dellaferrea et al., 2020), stimulated by event-based sensors, resulting in portable, low-powered devices. As pointed out by Pfeiffer and Pfeil (2018), this constitutes an advantage over conventional ANNs that rely on energy consuming high-end GPUs.

ANNs are most commonly trained using stochastic gradient descent (SGD), which relies on the chain rule of derivatives. During the forward pass, a batch of input examples is passed through the network, and a loss function is applied to the final outputs. During the subsequent backward pass, the network trainable parameters are updated to minimize the loss. The network gradually adapts to the task at hand by repeating this operation over a data set of prepared examples. SNNs are not directly compatible with gradient descent owing to their non-differentiable threshold behaviour. Different methods have been developed to alleviate the problem. We will review a representative set of them in section 2.4. In particular, the surrogate gradient approach allows SNNs to be trained like recurrent ANNs using the Back-Propagation Through Time (BPTT) algorithm, which is a generalization of gradient descent to process sequential data. Recurrent neural networks (RNNs) have proven to be efficient on speech recognition tasks. In general, the reported performance of SNNs is inferior to that of the best ANNs (Wu et al., 2018, 2020; Cramer et al., 2020; Yin et al., 2020, 2021; Yao et al., 2021; Shaban et al., 2021), even if the gap is gradually closing. By concentrating on the task of speech command recognition, the main general aims of this paper are the following,

1. Assess the general capability of SNNs, and how they might represent an attractive alternative to standard ANNs.
2. Identify which SNN training techniques are compatible with successful modern ANN frameworks, and establish a method that is able to compete with the ANN performance, while retaining the advantages of energy efficiency.
3. Given that ANNs have proven to be good at problem solving, use a physiologically plausible approach to provide some insights on how the corresponding biological mechanisms in humans might be functioning.

Spiking versions of the Heidelberg Digits (SHD) and Google Speech Command (SSC) datasets have recently been released using physiological models of the cochlea (Cramer et al., 2020). We use these as well as their respective non-spiking, traditional versions (HD and SC) to conduct experiments with both SNNs and ANNs. Using physiologically plausible neuron models with surrogate gradients, we achieve new state-of-the-art results with SNNs, even outperforming ANNs in some cases.

We will start by introducing the mathematical models used to describe spiking neurons in section 2.1. We will then show how ANNs are typically implemented (section 2.2), to then build an equivalent forward pass for SNNs (section 2.3). We will then focus on the training methods for SNNs in section 2.4 and define our selected approach using surrogate gradients. To complete the description of our spiking networks, the loss function and readout layer will be defined in section 2.5. We will then explain the different speech perception tasks on which we conduct the experiments in section 2.6 and finally present and discuss our results in sections 3 and 4.
2 MATERIALS AND METHODS

2.1 Single spiking neuron models

The neuron models presented below will be used as building blocks of potentially deep spiking neural networks. In order to achieve a satisfying degree of compatibility with the modern frameworks developed for machine learning, our analysis will focus on single neuron models that rely on a limited number of parameters and are not excessively expensive in terms of computations.

2.1.1 Leaky integrate and fire

The simplest and most widely used single neuron model is the leaky integrate and fire (LIF), the origin of which dates back to the beginning of the twentieth century with the work of Lapicque (1907). The dynamics of a single neuron are described by the membrane potential \( u(t) \), which evolves in time as a function of some input current \( I(t) \). In the absence of stimuli, i.e., when \( I(t) = 0 \), the membrane potential \( u(t) \) decays exponentially to some resting value \( u_{\text{rest}} \) with a time constant \( \tau_u \approx 10\text{ms} \). When \( I(t) \neq 0 \), the membrane potential \( u(t) \) integrates the incoming stimuli and increases or decreases accordingly. As presented by Gerstner and Kistler (2002), the dynamics in continuous time follow the differential equation,

\[
\tau_u \frac{du}{dt}(t) = -u(t) - u_{\text{rest}} + RI(t), \tag{1}
\]

where \( R \) is the membrane resistance. In order to have spikes, a threshold value \( \vartheta \) must be added to the model, so that when the potential reaches the critical value, a spike is emitted and the potential is reset to a new value \( u_r < \vartheta \).

\[
\text{if } u(t = t^f) \geq \vartheta \text{ then } s(t^f) = 1 \text{ and } \lim_{\delta \to 0; \delta > 0} u(t^f + \delta) = u_r. \tag{2}
\]

2.1.2 Adding an adaptation variable

Although widely used, the LIF model is not sufficient to reproduce many of the various firing patterns observed in biological neurons, such as adaptive, bursting, transient and delayed (Gerstner and Kistler, 2002). The idea of a second equation to describe an adaptation (or accommodation) variable between threshold and subthreshold voltage can be traced back to Hill (1936). The work of Treves (1993), Izhikevich (2001) and Brunel et al. (2003) have notably lead to its modern formulation, in which a recovery variable \( w(t) \) is linearly coupled to the membrane potential \( u(t) \) in the subthreshold regime, and a mechanism is used for spike-triggered adaptation. The resulting more complex neuronal dynamics of an adaptive, linear LIF model (adLIF) follow the differential equations,

\[
\tau_u \frac{du}{dt}(t) = -(u(t) - u_{\text{rest}}) - Rw(t) + RI(t) \tag{3}
\]

\[
\tau_w \frac{dw}{dt}(t) = -w(t) + a(u(t) - u_{\text{rest}}), \tag{4}
\]

where the adaptation current typically evolves more slowly than the potential, i.e., with a longer time constant \( \tau_w \approx 100\text{ms} \) compared to \( \tau_u \approx 10\text{ms} \). When the potential reaches a peak value \( u(t) = \vartheta \), its displacement is considered large enough to represent a spike, which defines the firing
time $t^f = t$. The potential is then reset, $u(t^f) = u_r$ and the recovery variable pushed by an amount $b$, $w(t^f) = w(t^f) + b$.

2.1.3 Adding a nonlinearity

Using a nonlinearity instead of the linear relation to the membrane potential in equation (3) transforms the strict voltage threshold into a more biologically plausible smooth spike initiation zone (Brette and Gerstner, 2005). For such adaptive, nonlinear LIF models, the complete dynamics can be represented by a trajectory $(u(t), w(t))$ on a 2D-plane. Before receiving any stimuli, the neuron is at equilibrium on a stable fixed point. Depending on the model’s parameters ($a$, $b$, $u_r$, $u_{\text{rest}}$, $R$, $\tau_u$ and $\tau_w$) a particular set of incoming spike trains can cause the trajectory to go through a bifurcation and form a limit cycle, resulting in repetitive firing. The neuron model of Izhikevich (2003) uses a quadratic function,

$$f(u) = -(u(t) - u_{\text{rest}})(\theta - u(t)), \quad (5)$$

and the adaptive exponential integrate and fire model (AdEx) of Brette and Gerstner (2005) uses a combination of linear and exponential functions,

$$f(u) = -(u(t) - u_{\text{rest}}) + \Delta \exp \left[ -\frac{\theta - u(t)}{\Delta} \right]. \quad (6)$$

This last neuron model seems to be the most physiologically plausible in terms of fitting with naturalistic pyramidal-neuron voltage traces (Badel et al., 2008).

2.2 Artificial neural networks

The vast majority of neural networks used in modern machine learning tasks are organized in layers of artificial neurons from the second generation as defined by Maass (1997). Here we are interested in networks that can process sequential inputs, in particular speech. Starting from some discrete signals $y^0_j \in \mathbb{R}^T$ of length $T$ and time step $\Delta t$, $j = 1, \ldots, N^0$, a standard ANN processes the information layer by layer, as follows. In the $l$-th layer, a single non-spiking artificial neuron $i$ receives inputs from neurons $j = 1, \ldots, N^{l-1}$ of the previous layer. If recurrent connections are enabled, neuron $i$ also receives inputs from all neurons $k = 1, \ldots, N^{l}$ in the same $l$-th layer. The overall stimulus of neuron $i$ in layer $l$ at time step $t$ is then computed as,

$$I_i^l[t] = \sum_{j=1}^{N^{l-1}} W^l_{ji} y^l_{j-1}[t] + \sum_{k=1}^{N^{l}} V^l_{ki} y^l_k[t-1] + b^l_i, \quad (7)$$

where $W^l$ and $V^l$ are the trainable feedforward and recurrent weight matrices respectively, and $b^l_i$ is a trainable bias vector. The network is called a multilayer perceptron (MLP) if only feedforward connections are implemented, i.e., $V = 0$, and a recurrent neural network (RNN) when additional recurrent connections are present, i.e., $V \neq 0$. In both cases, the sequential output of the neuron $y^l_i \in \mathbb{R}^T$ is simply computed using a nonlinear activation function $g(\cdot)$,

$$y^l_i[t] = g(I^l_i[t]), \quad (8)$$
which produces real-valued signals. Using a sigmoid activation function \( g(x) = \frac{1}{1 + e^{-x}} \) for instance, the analog neuron output \( y \in [0, 1] \) can be interpreted as the firing rate of a spiking neuron (over some arbitrary period of time).

The most successful recurrent architectures are based on the long short-term memory (LSTM), defined by Hochreiter and Schmidhuber (1997), in which gates are used to filter out irrelevant information and tackle the vanishing/exploding gradient problem. Each gate uses its own distinctive feedforward and recurrent weights, which increases the total number of trainable parameters. The gated recurrent unit (GRU) of Cho et al. (2014) and the light GRU (liGRU) of Ravanelli et al. (2018) constitute gradual simplifications of the LSTM with fewer gates in an effort to reduce the size of recurrent units. Very recently, Bittar and Garner (2021) have derived a probabilistically interpretable version of the liGRU called light Bayesian recurrent unit (liBRU) that showed slight improvements over the liGRU on speech recognition tasks. We will implement MLPs, RNNs, liBRUs and GRUs, which will serve as an ANN-baseline to compare with our SNNs.

### 2.3 Spiking neural networks

In a spiking neural network, a single neuron \( i \) in the \( l \)-th layer receives pre-synaptic inputs from neurons \( j = 1, \ldots, N^{l-1} \) of the previous layer in the form of spike trains \( s^{l-1}_j \in \{0, 1\}^T \). If recurrent connections are enabled, it also receives spike trains \( s^l_k \) from all other neurons \( k = 1, \ldots, N^l, k \neq i \) in the same \( l \)-th layer. The overall stimulus of neuron \( i \) in layer \( l \) can then be written as

\[
I^l_i[t] = \sum_{j=1}^{N^{l-1}} W^l_{ji} s^{l-1}_j[t] + \sum_{k=1; k \neq i}^{N^l} V^l_{ki} s^l_k[t-1] + b^l_i,
\]

where the weight matrices \( W^l \) and \( V^l \) correspond to the strength of the synaptic connections, and the bias \( b^l_i \) to heterogeneous resting values of the membrane potential among neurons. In equations (1) and (3), the electric current \( I \) was multiplied by the membrane resistance \( R \) to have units of electric potential. The value of \( R \) can actually be integrated in the trainable parameters \( W, V \) and \( b \), so that in the rest of the paper, \( I \) is used instead of \( R I \) to describe the stimuli in units of electric potential. The excitatory and inhibitory connections between physiological neurons are here represented by positive and negative weights respectively. After firing, a biological neuron enters a period of refractoriness so that it cannot immediately fire a second spike. In order to maintain this physiological feature of self inhibition, the diagonal elements of \( V \) can be set to 0 in equation (9), as positive diagonal weights would act against refractoriness. The latter can be then modelled via the reset of the membrane potential to a value \( u_r \) after spiking. Alternatively, negative diagonal elements could be used to directly decrease the value of the membrane potential. With equations (7) and (9), we see that the stimulus of a spiking neuron can be computed in the exact same way as for a standard artificial neuron. The main difference with ANNs is therefore that the dynamics of the membrane potential combined with the threshold behaviour replace the simple activation function of equation (8) and produce binary signals \( s^l_1 \in \{0, 1\}^T \times N_i \) instead of analog ones \( y^l_1 \in \mathbb{R}^T \times N_i \). As pointed out by Neftci et al. (2019), such dynamics can be viewed as a nonlinear activation function which makes SNNs a special case of RNNs. In this paper, however, the term RNN describes a purely non-spiking recurrent network, as defined in section 2.2. In terms of number of trainable parameters, SNNs that only have feedforward connections are thus analogous to MLPs, and SNNs with recurrent connections are comparable to RNNs.
For an adLIF model, by making the following changes of variables \( u \rightarrow u - u_{\text{rest}} \) and \( w \rightarrow Rw \), equations (3) and (4) can be rewritten in discrete time and we get the following forward pass,

\[
\begin{align*}
    u[t] &= \alpha u[t-1] + (1 - \alpha)(I[t] - w[t-1]) - s[t-1] u[t-1] \\
    w[t] &= \beta w[t-1] + (1 - \beta) \tilde{a} u[t-1] + s[t-1] b \\
    s[t] &= (u[t] \geq \vartheta)
\end{align*}
\]

where \( \tilde{a} := Ra / [1, 1] \), \( b \in [0, 2] \), \( \alpha := \exp(-\Delta t/\tau_u) \in [0.60, 0.96] \), \( \beta := \exp(-\Delta t/\tau_w) \in [0.96, 0.99] \) and \( u_r = 0 \) based on physiologically plausible ranges of values. In equations (10) and (11), the first term describes the leak, the second the excitation, and the third the effect of having spiked at the previous time step. Equation (12) produces a 0 or a 1 when the membrane potential is below or above threshold. The threshold value \( \vartheta \) can be set to 1, which corresponds to additional changes of variables \( u \rightarrow u/\vartheta \) and \( w \rightarrow w/\vartheta \). Based on ad-hoc experiments, the model parameters \( \alpha \), \( \beta \), \( \tilde{a} \) and \( b \) are made trainable, leading to heterogeneous values among neurons and thus to different types of firing. The LIF model can then be viewed as a simplification of the adLIF with \( a = b = 0 \), so that there is no recovery current \( w[t] = 0 = \) constant, and the only model parameter is \( \alpha \).

The above equations can be vectorized and looped over time to describe the dynamics of a whole layer, which defines the forward pass through an SNN. Equation (12) is however not differentiable and therefore incompatible with gradient descent.

### 2.4 Training methods for SNNs

Biological neurons exhibit activity-dependent synaptic plasticity characterized by long term potentiation (LTP) and depression (LTD). This behaviour can be modelled using a form of spike timing dependent plasticity (STDP), as defined by Dan and Poo (2006). The synaptic efficacy between a pre- and a post-synaptic neuron can be represented by a trainable scalar weight, which is adjusted as a function of the relative spike timings of the two neurons. When the pre-synaptic neuron fires shortly (\( \approx 10 \text{ms} \)) before the post-synaptic neuron, a causal relation can be assumed and the connection is strengthened (\( \Delta w > 0 \)), which corresponds to LTP. In contrast there cannot be causality if the pre-synaptic neuron fires briefly after the post-synaptic neuron. In this case the connection is weakened (\( \Delta w < 0 \)), which corresponds to LTD. Without the need of labeled examples, this form of unsupervised Hebbian learning is sufficient to detect correlations in the input stimuli and learn encodings of real-world data. However, in order to perform motor tasks, STDP must be combined with a form of global reward-based learning (Gerstner and Kistler, 2002). In the brain, neuromodulators such as dopamine play the role of globally broadcasting the success signal of an ongoing task to different areas of the brain (Schultz et al., 1997; Schultz, 2007, 2010). On top of the pre- and post-synaptic activity, the update mechanism can additionally depend on a neuromodulator, which leads to three-factor learning rules described by Frémaux and Gerstner (2016).

Artificial neurons, on the other hand, are most commonly trained using stochastic gradient descent (SGD). This technique compares the model predictions to desired outputs on a batch of examples via a loss function. The error of the whole batch is then back-propagated through the network using the chain-rule of derivatives, and the trainable parameters of the entire network are updated accordingly. This form of supervised, global and offline learning is however highly...
biologically implausible (O’Reilly and Munakata, 2000). In comparison, the weight adjustment with STDP happens online, i.e., each time a spike is emitted, and has only a local dependence on the pre- and post-synaptic neurons. Nevertheless, SGD represents the most successful training algorithm used in ANNs. With the aim of evaluating the compatibility of SNNs within ANN frameworks, we will focus on training SNNs with gradient descent.

Using SGD with SNNs is challenging because the derivative of the spike function in equation (12) with respect to the membrane potential is zero in the subthreshold regime (when no spikes are emitted, i.e., almost everywhere) and undefined when threshold is reached and a spike is produced. Moreover, small perturbations of the synaptic weights can either lead to considerably different output spike trains, or produce no change at all. The numerous discontinuities caused by the threshold mechanism make the search of a global optimum particularly difficult. This problem is especially visible when training multi-layered architectures with SGD. Nevertheless, training SNNs with SGD can still be achieved through a variety of approaches that can be grouped into the following three general categories:

1. Using a hybrid ANN-SNN model
2. Using a differentiable model
3. Using surrogate gradients

which we describe in detail in turn.

2.4.1 Hybrid ANN-SNN models

The first category, which has been reviewed by Abbott et al. (2016), circumvents the problem of training SNNs by using conventional rate-based ANNs instead. The resulting architectures do exhibit a spiking behaviour during the forward pass, but the spike timings are ignored in the learning rule. Hunsberger and Eliasmith (2015) have used an ANN to closely approximate the firing rate of individual spiking neurons. More recently, Wu et al. (2021) have managed to ensure an efficient gradient based back-propagation by coupling an SNN with an ANN through layer-wise weight sharing. During the forward pass, the SNN computes the exact spiking neural representations, and the ANN the corresponding approximate spike counts (or firing rates). During the backward pass, the error is backpropagated through the ANN via SGD and the weight updates are transferred to the SNN. This tandem learning technique allows fast and efficient learning with multi-layered architectures and has notably proven to be successful on speech recognition tasks (Wu et al., 2020). One could argue that the information about the timings of the individual spikes is still reduced to a rate-based approximation during the backward pass.

More generally, this category of hybrid methods could prove to be convenient for low-powered hardware implementations of neural networks. The general procedure would then be to train a potentially deep ANN, translate it into an equivalent SNN, and implement the trained SNN on hardware.

2.4.2 Differentiable neuron models

As presented by Neftci et al. (2019), a second category consists of smoothed SNNs and can be further split into the following subcategories,
• **Soft-threshold models**
  Some conductance-based neuron models (Hodgkin and Huxley, 1952; FitzHugh, 1961; Morris and Lecar, 1981) can exhibit a soft threshold behaviour that is differentiable. Huh and Sejnowski (2018) were able to train such a differentiable SNN using gradient descent. Nonetheless, spikes in this approach cannot be considered as binary events anymore. Instead of binary spike trains, continuous real-valued variables (that can be seen as gates) are propagated through the model, which means that the energy efficiency advantage of SNNs over ANNs is lost. This type of model is therefore not considered in our experiments.

• **Probabilistic models**
  *In vivo* experiments observe a high degree of irregularity in neuronal activity. The origin of these unpredictable dynamics (whether it is a highly efficient way of coding information or plain noise) remains rather poorly understood in neuroscience (Gerstner and Kistler, 2002; Uddin, 2020). Nevertheless, the variability of spike trains can be modelled by adding noise to spiking neurons, which leads to probabilistic models. The Generalized Linear Model (GLM) (McCullagh and Nelder, 2019), which is a probabilistic generalization of the Spike Response Model (SRM) (Jolivet et al., 2003), introduces stochasticity in the spike firing mechanism. In GLMs, the negative log-likelihood of a spike-train is a differentiable quantity and can then be minimized with SGD. This specific class of models is beyond the scope of this paper and hence a matter of future work.

• **Spike train convolution models**
  The discrete spike trains can be transformed into a continuous representation with the use of kernel functions and allow the application of common mathematical functions. Based on the Widrow-Hoff learning rule (Widrow and Lehr, 1990), the SPAN (Mohemmed et al., 2012), and PSD (Yu et al., 2013) algorithms have been developed but are constrained to single-layered architectures. More recent learning rules based on the inner product of convolved spike trains have managed to extend this class of methods to multi-layered (Lin et al., 2017) and recurrent networks (Lin and Shi, 2018). Some also allow the time delays of the neurons to be trainable (Wang et al., 2019). Nevertheless, convolution-based models are generally designed for learning precise spike patterns, i.e., for tasks where the desired outputs take the form of spike trains, which is usually not the case in conventional ANN frameworks, and in particular for speech recognition. In order to make them compatible with a classification task, one must assign a fixed, arbitrary spike pattern to each class. The error function then computes a similarity measure between these desired chosen patterns and the actual output spike trains of the network. In order to be differentiable, the update rules of the trainable parameters typically only depend on the input, output and target spike trains and not on the specific dynamics of the neuron model. This class of methods is therefore so far limited to rather shallow networks, which means that the desired compatibility with modern ANN frameworks remains modest.

• **Single-spike timing-based models**
  Certain simplified neuron models use the timing of the first spike instead of spike trains as the information carrier. Using the Lambert W function described by Corless et al. (1996), the timing of a post-synaptic spike can be analytically derived from the timings of pre-synaptic spikes. Using this technique, Bodyanskiy and Dolotov (2013) as well as Comsa et al. (2020) managed to train differentiable SNNs with SGD. Mostafa (2017) also defined a timing-based differentiable SNN, with the persisting limitation of having a single-spike per neuron.
2.4.3 Surrogate gradient methods

Also presented by Neftci et al. (2019), the problem of the non-differentiable threshold behaviour can be solved using surrogate gradients. During the backward pass, the Heaviside step function of the spike generation is smoothed into a suitable differentiable function. With this approach, the threshold operation is only approximated during the backward pass, and remains a step function inside the forward computations. The derivative has notably been approximated using a rectifying linear unit (Bohte et al., 2002), a sigmoid derivative (Schrauwen and Van Campenhout, 2006; Bellec et al., 2018; Zenke and Ganguli, 2018), an exponential function (Shrestha and Orchard, 2018), a Gaussian (Yin et al., 2020), a multi-Gaussian (Yin et al., 2021) and a boxcar function (Kaiser et al., 2020). An SNN can then be considered as a special case of a recurrent neural network (RNN) and the error Back-Propagation Through Time (BPTT) algorithm becomes applicable. Nevertheless, the sparsity in time of non-zero gradients, combined with the problems of exploding/disappearing gradients remain. This third and last category of SGD-based training methods is rather versatile compared to the first two (2.4.1 and 2.4.2), as it is not limited to a specific neuron model and allows the use of the different spiking neuron models described in 2.1.

We will therefore concentrate our analysis on the surrogate gradient approach, but still include a comparison with the tandem method of Wu et al. (2021) presented in 2.4.1.

In an SNN as defined in section 2.3, by exploiting auto-differentiation inside the deep learning framework PyTorch (Paszke et al., 2017), one can manually replace the undefined gradient of the step function in equation (12) with a surrogate, and make the backward pass and therefore gradient descent possible for the whole network. Different choices of surrogate gradients illustrated in Figure 1 can be considered. The boxcar method, previously used by Kaiser et al. (2020), is chosen by default in this study based on ad-hoc experiments. It is defined as,

$$\frac{\partial s[t]}{\partial u[t]} = \begin{cases} 0.5 & \text{if } |u[t] - \vartheta| \leq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

and is quite inexpensive in terms of computations.

2.5 Loss function and readout layer

In order to use an SNN as a classifier that can be trained inside a typical ANN framework, we choose a cross-entropy loss function to be applied to the outputs of the final layer \(L\) of the architecture. Instead of a sequence of spikes, this readout layer must output one value \(o_i\) per neuron \(i = 1, \ldots, N^L\) that indicates its level of activity over time. During inference, the neuron with the highest activity will be chosen. We considered four different methods for the readout layer,

1. a spiking layer using the spike count, \(o_i = \sum_{t=1}^{T} s_i^L[t]\)
2. a non-spiking layer using the last potential value over time, \(o_i = u_i^L[T]\)
3. a non-spiking layer using the maximal potential value over time, \(o_i = \max_{t=1, \ldots, T} u_i^L[t]\)
4. a non-spiking layer using a cumulative sum of the potential over time, \(o_i = \sum_{t=1}^{T} \text{softmax}(u_i^L[t])\).
Figure 1. Different surrogate gradient functions to approximate the derivative of the step-function responsible for spike generation.

In ad-hoc experiments, the last technique gave the best performance and is what is used in all presented results. Non-spiking LIF neurons with no recurrent connections are used inside the readout layers of all presented models.

All models take inputs of size \((N, T, F)\) and return outputs of size \((N, C)\), where \(N\) is the batch size (i.e., the number of examples in one batch), \(T\) the number of time steps, \(F\) the number of input features/channels and \(C\) the number of classes (labels). The ground truths are given as a vector \(y\) of size \((N)\) containing the label indexes. The cross-entropy loss is then computed as,

\[
\mathcal{L} = -\frac{1}{N} \sum_{n=1}^{N} \log \frac{\exp(o[n, y[n]])}{\sum_{c=1}^{C} \exp(o[n, c])}.
\] (14)

The Adam optimizer (Kingma and Ba, 2015) is used for all experiments with an initial learning rate of 0.001. A scheduler is defined to reduce the learning rate by a factor 0.7 if there is no improvement on the test set accuracy during 10 epochs in a row.

The general network architecture, used for all SNNs in this work, is presented in Figure 2. We will now explain the tasks on which they will be evaluated.
2.6 Speech perception tasks for SNNs

This research focuses on the bio-inspired processing of auditory information, leading to the formation of appropriate representations and the extraction of relevant features that can then be used for different tasks. The long-term objective is to perform automatic speech recognition (ASR) using a physiologically plausible approach that includes waveform to spike conversion followed by processing of the information via spiking neural networks.

However, ASR is a complex task; modern approaches involve end-to-end deep networks, whereas previous techniques needed to solve a series of subtasks, typically feature extraction, phoneme recognition and decoding. In the field of SNNs, it appears that one could benefit from first focusing on the simpler task of speech command recognition to better understand spiking networks. While retaining the processing of auditory information, this more elementary task neither involves too many components in the pipeline, nor requires very deep networks and therefore constitutes a first necessary step in the direction of efficient ASR with SNNs.

We will start by giving a short summary of the biological processes involved in speech perception. We then review LAUSCHER, a bio-inspired model to convert audio waveforms into spike trains, and some resulting, newly available spiking datasets.
2.6.1 From waveform to spikes

A speech utterance arrives at the ear in the form of air vibrations. From the eardrum it travels via the ossicles to the cochlea, hence the basilar membrane and the organ of Corti, ultimately stimulating hair cells that convert the physical movement into electrical signals. The signals take the form of spike trains on the auditory nerve. Many conventional ASR “filterbank” front-ends are rough analogues of this process, notably modelling the logarithmic response to frequency and to amplitude.

Cramer et al. (2020) have developed LAUSCHER, a biologically plausible cochlear model to convert audio waveforms into spike trains. A cochlea model, based on the models developed by Sieroka et al. (2006) is used to calculate the hydrodynamic shallow water basilar membrane response to the input waveform. The output of the cochlea then goes into a transmitter pool-based hair cell model, derived from the work of Meddis (1986, 1988). Finally, a layer of auditory neurons called bushy cells convert the signal to spike trains using LIF dynamics.

Such a framework allows a direct conversion from audio waveforms into spike trains, while solely relying on physiological processes. In order to train SNNs on speech data, the general and most commonly used alternative is to extract acoustic features from the waveform and interpret them as firing rates to produce spike trains via Poisson processes. Even though the latter approach still shows some physiological plausibility, a single firing rate value is used to produce spikes during the length of a frame (typically 25ms). This concession comes from the need of using datasets that were originally designed for ANNs, i.e., rate-based approximations of SNNs.

2.6.2 Spiking datasets

In order to rectify the absence of free spike-based benchmark datasets, Cramer et al. (2020) recently released two spiking datasets using LAUSCHER:

- The Spiking Heidelberg Digits (SHD) dataset contains spoken digits from 0 to 9 in both English and German (20 classes). The recordings are from 12 different speakers, 2 of which are only present in the test set. The train set contains 8332 examples and the test set 2088 (there is no validation set).
- The Spiking Speech Commands (SSC) dataset is based on the Google Speech Commands v0.2 dataset and contains 35 classes from a larger number of speakers. The number of examples in the train, validation and test splits are 75466, 9981 and 20382 respectively.

In both datasets, the original waveforms have been converted to spike trains over 700 input channels. These spiking datasets form an adapted benchmark and allow the investigation of SNNs as well as the comparison of different techniques.

The current state-of-the-art on the SHD and SSC datasets is summarized in Tables 1 and 2. The SNN methods are presented in the upper section of the tables, and in the lower sections, the non-spiking CNN and LSTM serve as a point of comparison with the ANN performance.

2.6.3 Non-spiking datasets

The original, non-spiking versions of the SHD and SSC datasets are available and will also be considered in this work. For the Heidelberg Digits (HD) and Google Speech Commands (SC) datasets, acoustic features are extracted from the waveforms and fed into neural networks. An
Figure 3. Standard representation via filterbank features (A) and spike train representation via LAUSCHER (B) of the same spoken digit (seven in English) from the SHD dataset.

Table 1. State-of-the-art on SHD dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attention (Yao et al., 2021)</td>
<td>91.1%</td>
</tr>
<tr>
<td>Recurrent + adaptation (Yin et al., 2021)</td>
<td>90.4%</td>
</tr>
<tr>
<td>Recurrent + adaptation (Yin et al., 2020)</td>
<td>84.4%</td>
</tr>
<tr>
<td>Recurrent + data augm. (Cramer et al., 2020)</td>
<td>83.2%</td>
</tr>
<tr>
<td>Recurrent + heter. time const. (Perez-Nieves et al., 2021)</td>
<td>82.7%</td>
</tr>
<tr>
<td>Recurrent (Cramer et al., 2020)</td>
<td>71.4%</td>
</tr>
<tr>
<td>Non-recurrent (Cramer et al., 2020)</td>
<td>47.5%</td>
</tr>
<tr>
<td>CNN (Cramer et al., 2020)</td>
<td>92.4%</td>
</tr>
<tr>
<td>LSTM (Cramer et al., 2020)</td>
<td>89%</td>
</tr>
</tbody>
</table>

Table 2. State-of-the-art on SSC dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recurrent + adaptation (Yin et al., 2021)</td>
<td>74.2%</td>
</tr>
<tr>
<td>Recurrent + heter. time const. (Perez-Nieves et al., 2021)</td>
<td>57.3%</td>
</tr>
<tr>
<td>Recurrent (Cramer et al., 2020)</td>
<td>50.9%</td>
</tr>
<tr>
<td>Non-recurrent (Cramer et al., 2020)</td>
<td>41.0%</td>
</tr>
<tr>
<td>CNN (Cramer et al., 2020)</td>
<td>77.7%</td>
</tr>
<tr>
<td>LSTM (Cramer et al., 2020)</td>
<td>73%</td>
</tr>
</tbody>
</table>

input example is illustrated in Figure 3, where the filterbank and spiking approaches are compared. The second version of the original Speech Commands (SC) dataset introduced by Warden (2018) has the same number of examples as its spiking version (SSC), but different training, validation and testing splits of 84843, 9981 and 11005 examples respectively. The SSC has a 70% / 10% / 20% partition instead of 80% / 10% / 10% for the SC. This makes a direct comparison impossible between the accuracies on the two tasks, as the SC has considerably more training data. For the
HD and SHD datasets however, the splits are the same. We were not able to find state-of-the-art results on the non-spiking HD dataset, however, for the SC dataset, the state-of-the-art is presented in Table 3. In all state-of-the-art tables, the best test accuracies by SNNs and ANNs are written in bold.

Table 3. State-of-the-art on SC dataset (version 2 with 35 labels).

<table>
<thead>
<tr>
<th>Method</th>
<th>Test acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recurrent + adaptation (Shaban et al., 2021)</td>
<td>91%</td>
</tr>
<tr>
<td>Transformers (Sahu et al., 2021)</td>
<td>99.74%</td>
</tr>
<tr>
<td>Attention RNN (De Andrade et al., 2018)</td>
<td>93.9%</td>
</tr>
</tbody>
</table>

3 RESULTS

We present results for the LIF and adLIF neuron models defined in 2.1.1 and 2.1.2. The more complex, nonlinear Izhikevich and Adex models defined in 2.1.3 did not bring improvements over the linear adaptive LIF and are left out of this analysis. We also distinguish between models with and without recurrent connections, in the form of a weight matrix \( V \) applied to a layer-wise feedback as defined in 2.3 so that SNNs without recurrent connections are considerably lighter in terms of number of trainable parameters. The hidden size of a network corresponds to the number of neurons in each of the hidden layers. Although different hidden layers can have different sizes, we focused on hidden layers of the same size in this study. The number of layers is the number of hidden layers plus one (the readout layer). Networks of increasing size and depth were investigated by varying the hidden size from 128 to 1024 neurons per layer, and the number of layers from 2 to 5. Overall, 3-layered architectures appeared as the best compromise between size and performance and are used in all presented results (with both SNNs and ANNs). Nevertheless, increasing the number of layers showed that the training of the networks was robust to considerable depth. The chosen approach was therefore able to discard scalability limitations of SNNs on these 4 tasks, which is very encouraging for the compatibility of SNNs with modern deep learning frameworks. In the following sections dedicated to each of the 4 tasks, the results with SNNs will be presented in the upper region of the tables. We distinguish between the following types of network:

1. tandem: non-recurrent network of non-adaptive IF neurons (no leak) trained with tandem learning rule.
2. LIF: non-recurrent network of non-adaptive LIF neurons trained with a surrogate gradient.
3. adLIF: non-recurrent network of adaptive linear LIF neurons trained with a surrogate gradient.
4. RLIF: recurrent network of non-adaptive LIF neurons trained with a surrogate gradient.
5. RadLIF: recurrent network of adaptive linear LIF neurons trained with a surrogate gradient.

Based on ad-hoc experiments, all presented surrogate gradient SNNs use (i) trainable neuron parameters within fixed ranges of values (\( \alpha \) for LIF and \( \alpha, \beta, \tilde{a} \) and \( b \) for adLIF neurons as defined in 2.3), (ii) a surrogate gradient with the boxcar function as defined in 2.4.3, and (iii) a non-spiking readout layer with a cumulative sum over time as defined in 2.5.

This is a provisional file, not the final typeset article
for the ANN baseline, the following types of network will be presented in the lower section of the tables of results:

1. MLP: a simple feed-forward network without recurrence
2. RNN: a standard recurrent network
3. liBRU: a network of light Bayesian recurrent units
4. GRU: a network of gated recurrent units

The liBRU is a probabilistic version of the liGRU with a Softplus activation function instead of a rectified linear unit. We show results with the liBRU instead of the liGRU, as they were slightly better on all 4 tasks. In terms of number of trainable parameters, on the one hand, tandem, LIF and adLIF networks are roughly equivalent to MLPs, and on the other hand, RLIF and RadLIF networks are comparable to standard RNNs. By contrast, gated non-spiking networks are considerably larger than all SNNs, since each gate includes weight matrices of its own. With respectively one and two gates, liBRUs and GRUs contain approximately two and three times as many parameters as RNNs of the same size. All presented ANNs and SNNs use dropout with $p = 0.1$ as well as batch normalization (Ioffe and Szegedy, 2015).

3.1 Spoken digit recognition on SHD

Owing to its small size, the SHD data set allows a thorough investigation of the best choice of architecture. On this specific task, we show for the first time that SNNs can surpass ANNs. Our results are illustrated in Table 4. First notice that our best SNN results are better than the attention based SNN state of the art of 91.1% by Yao et al. (2021) (see Table 1). More importantly, our approach also improves upon the best reported ANN-performance of 92.4% by Cramer et al. (2020), which used a convolutional neural network (CNN). Our own attempts with recurrent ANNs only reached 90.40% with the gated recurrent unit (GRU) of Cho et al. (2014). Even using non-recurrent connections and a relatively small network (3x128), we obtained an accuracy of 93.06% with adaptive spiking neurons. This shows a remarkable ability of SNNs to compete with much larger standard networks. With recurrence and a higher number of neurons, our best performing SNN even reached a test accuracy of 94.62%, which is extremely promising for the future of spiking networks with surrogate gradients.

We also tested the tandem approach of Wu et al. (2021) presented in 2.4.1, which is an alternative to surrogate gradients. This method so far does not allow recurrent connections. Even if the results are slightly higher (62.64%) than those with a MLP (61.63%), they are significantly lower than what we get with the surrogate gradient approach for a network of the same size (87.04% and 93.06% for LIF and adLIF neurons respectively). This can be seen as evidence of the importance of using the precise spike timings inside the training mechanism.

3.2 Spoken digit recognition on HD

In order to compare with standard methods for speech recognition, some experiments were made on the original, non-spiking Heidelberg digits (HD) dataset. Filterbank features were extracted from the waveforms, and directly fed into various networks. As illustrated in Figure 3 compared to a spiking input generated with LAUSCHER, which is a 700 neurons × 100 timesteps sparse binary tensor, here a non-spiking input typically takes the form of a 40 features × 250 frames real-valued tensor. Even though the first hidden layer receives real-valued sequences instead of spike trains,
Table 4. Results on SHD dataset. Larger ANNs were also tested but only obtained slight improvements and remained under the performance of SNNs of the same size. Assuming a beta-distribution, the 95% confidence intervals are between ±2.1% and ±0.9% for test set accuracies between 61.63% and 94.62%.

<table>
<thead>
<tr>
<th>Network type</th>
<th>Recurrent connections</th>
<th>Number of layers</th>
<th>Hidden size</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>tandem</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>62.64%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1024</td>
<td>68.01%</td>
</tr>
<tr>
<td>LIF</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>87.04%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1024</td>
<td>89.29%</td>
</tr>
<tr>
<td>adLIF</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>93.06%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1024</td>
<td>93.57%</td>
</tr>
<tr>
<td>RLIF</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>89.75%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1024</td>
<td>92.51%</td>
</tr>
<tr>
<td>RadLIF</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>92.88%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1024</td>
<td>94.62%</td>
</tr>
<tr>
<td>MLP</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>61.63%</td>
</tr>
<tr>
<td>RNN</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>73.48%</td>
</tr>
<tr>
<td>liBRU</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>89.61%</td>
</tr>
<tr>
<td>GRU</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>90.40%</td>
</tr>
</tbody>
</table>

spiking networks remain compatible with this approach. They even outperform their non-spiking equivalents, as presented in Table 5, where the LIF and RLIF networks surpass the MLP and RNN respectively. The light Bayesian recurrent unit (liBRU) of Bittar and Garner (2021) was also tested and gave the best overall performance, although it requires roughly twice as many trainable parameters as a RNN or RLIF network. The accuracies reached with this filterbank approach are considerably higher than the ones on the spiking dataset. Most investigated models were able to reach a test accuracy close to 100%, which is why we only show a few relevant results. This seems to indicate that some information is lost when performing the conversion from waveform to spikes with LAUSCHER, compared to the extraction of acoustic features. Here the conversion from filterbank features to spike trains happens in a trainable fashion inside the neuronal dynamics of the first hidden layer. Moreover, the initial (non-trainable) transformation of the audio waveforms into filterbank features is fast enough to be performed during training, so that our approach with the non-spiking data sets does not require any preliminary processing of the audio, and could be suitable for low-powered hardware implementations.

3.3 Speech command recognition on SSC

The SSC dataset is roughly 10 times bigger than the SHD and has 35 labels instead of 20. It already represents a more complicated classification task to solve for a neural network. Our results are presented in Table 6. Here, we managed to close the gap between the SNN and ANN performances by reaching a test accuracy of 77.4% with a SNN. Even though this already represents considerable improvements to the best previously reported SNN result of 74.2% by Yin et al. (2021), our results remain slightly lower than the (non-spiking) CNN performance of 77.7%.
Table 5. Results on HD dataset. Assuming a beta-distribution, the 95% confidence intervals are between ±0.5% and ±0.1% for test set accuracies between 96.99% and 99.96%.

<table>
<thead>
<tr>
<th>Network type</th>
<th>Recurrent connections</th>
<th>Number of layers</th>
<th>Hidden size</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIF</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>98.40%</td>
</tr>
<tr>
<td>RLIF</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td><strong>99.35%</strong></td>
</tr>
<tr>
<td>MLP</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>96.99%</td>
</tr>
<tr>
<td>RNN</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>99.13%</td>
</tr>
<tr>
<td>liBRU</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td><strong>99.96%</strong></td>
</tr>
<tr>
<td>GRU</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>99.91%</td>
</tr>
</tbody>
</table>

reported by [Cramer et al. (2020)](), and also lower than our best ANN-performance of 79.05% with a GRU. Nevertheless, in terms of number of trainable parameters, if we compare SNNs to ANNs of the same size, the LIF and adLIF networks score substantially better (66.67% and 71.66%) than the MLP (only 29.27%), and the RLIF and RadLIF outperform (73.87% and 73.25%) the RNN (70.01%).

Table 6. Results on SSC dataset. Assuming a beta-distribution, the 95% confidence intervals on the accuracies are all about ±0.6% due to the large size of the test set.

<table>
<thead>
<tr>
<th>Network type</th>
<th>Recurrent connections</th>
<th>Number of layers</th>
<th>Hidden size</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIF</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>66.67%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td>68.14%</td>
</tr>
<tr>
<td>adLIF</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>71.66%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td>73.58%</td>
</tr>
<tr>
<td>RLIF</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>73.87%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td>75.91%</td>
</tr>
<tr>
<td>RadLIF</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>73.25%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td><strong>77.40%</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>MLP</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>29.27%</td>
</tr>
<tr>
<td>RNN</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>70.01%</td>
</tr>
<tr>
<td>liBRU</td>
<td>yes</td>
<td>3</td>
<td>512</td>
<td>78.70%</td>
</tr>
<tr>
<td>GRU</td>
<td>yes</td>
<td>3</td>
<td>512</td>
<td><strong>79.05%</strong></td>
</tr>
</tbody>
</table>

3.4 Speech command recognition on SC

Our results on the non-spiking SC dataset are presented in Table [7]. While using a similar neuron model that includes recurrence and adaptation, we find that our approach is able to reach even better accuracies than the current SNN state-of-the-art of 91% by [Shaban et al. (2021)](). We also
find that the chosen SNN approach surpasses the performance of almost all considered ANNs. With a similar number of trainable parameters, the non-recurrent LIF and adLIF networks gave much better results (82.12% and 90.46%) than the MLP which only scored 48.80% on this task. Similarly, the recurrent RLIF and RadLIF networks achieved accuracies of 90.71% and 92.48% respectively, compared to only 84.65% for a non-spiking equivalent RNN. We even observe that a non-recurrent, adaptive SNN (adLIF), which is significantly lighter in terms of trainable parameters, is able to outperform a conventional RNN, which illustrates the advantage of physiologically plausible spiking neuron models. By adding recurrence and a larger number of hidden units, we find that our best performing SNN (94.51%) even surpassed the Attention RNN approach of De Andrade et al. (2018) (93.9%), which remained as the ANN state-of-the-art on this task for a long time. With roughly twice as many trainable parameters, the liBRU was the only ANN in our baseline that was able to modestly exceed the RadLIF SNN performance. Generally, the SNN approach appears able to compete with state-of-the-art gated recurrent networks, while retaining a definitive advantage of energy efficiency; this is extremely encouraging for further work in this direction.

Table 7. Results on SC dataset. Assuming a beta-distribution, the 95% confidence intervals are between ±0.9% and ±0.4% for test set accuracies between 48.80% and 95.06%.

<table>
<thead>
<tr>
<th>Network type</th>
<th>Recurrent connections</th>
<th>Number of layers</th>
<th>Hidden size</th>
<th>Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIF</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>82.12%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td>83.03%</td>
</tr>
<tr>
<td>adLIF</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>90.46%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td>93.12%</td>
</tr>
<tr>
<td>RLIF</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>90.71%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td>93.58%</td>
</tr>
<tr>
<td>RadLIF</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>92.48%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td><strong>94.51%</strong></td>
</tr>
<tr>
<td>MLP</td>
<td>no</td>
<td>3</td>
<td>128</td>
<td>48.80%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td>53.16%</td>
</tr>
<tr>
<td>RNN</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>84.65%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td>92.09%</td>
</tr>
<tr>
<td>liBRU</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>94.55%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td><strong>95.06%</strong></td>
</tr>
<tr>
<td>GRU</td>
<td>yes</td>
<td>3</td>
<td>128</td>
<td>93.65%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
<td>94.32%</td>
</tr>
</tbody>
</table>

4 DISCUSSION

4.1 Towards physiological plausibility

Different neuron models were investigated in our study. In order of increasing complexity and physiological plausibility: the leaky integrate-and-fire (LIF), adaptive linear LIF (adLIF), adaptive quadratic LIF (Izhikevich) and adaptive exponential LIF (AdEx), described in 2.1. As
presented in the results section adding adaptation with the adLIF model consistently improved the performance compared to the LIF. Nevertheless, the more physiological nonlinear AdEx and Izhikevich models (that were left out of the presented results) did not achieve better performances than the simpler linear adLIF model. We believe this is an effect of the weaker compatibility of these inherently more complex models with gradient descent.

On the SC dataset, adding adaptation had the same impact as adding recurrent connections (see Table 7), even though the former requires remarkably less trainable parameters than the latter. On the SHD dataset, the effect of adding adaptation is even more pronounced as the considerably lighter adLIF networks scored better than the RLIFs (see Table 4). This shows the importance of the neuron model and, more generally, of a physiologically plausible approach. As pointed out by Perez-Nieves et al. (2021), the heterogeneity of the spiking neurons is a metabolically and computationally efficient strategy. In ANNs, as defined in equation (8), all neurons have the same activation function. The resulting homogeneity in the behaviour of standard artificial neurons implies that the only source of heterogeneity lies in the synaptic connections, that can be different for each neuron. However, adding neurons to the network increases the computational cost by an order of $O(N^2)$. With spiking neurons, the more complex neuronal dynamics allow heterogeneous behaviours among neurons by depending on trainable parameters that only scale with $O(N)$, hence the more efficient computational strategy.

In terms of the learning rule, the compatibility with deep learning methods was favoured over the biological plausibility of the approach. Nevertheless, the surrogate gradient technique can actually be a good candidate towards more physiologically plausible learning algorithms. Kaiser et al. (2020) have recently defined a deep continuous local learning rule (DECOLLE) using random readouts at each layer. Their method still uses surrogate gradients to allow SGD, but is closer to a form of bio-inspired plasticity. This seems to indicate that the evaluated compatibility of training SNNs within ANN frameworks could lead to further improvements of the training methods, and allow more physiologically plausible learning rules by retaining the advantages of well developed ANN techniques.

Compared to the SNN-ANN tandem method of Wu et al. (2021), the chosen surrogate gradient approach does not ignore the spike timings during the backward pass. In addition to being more flexible to easily include recurrence and different neuron models, the latter produced better results than the former, which suggests that the precise timing of the spikes is of importance in processing temporal information.

### 4.2 Towards energy-efficient hardware

In other papers that used the spiking datasets (see Tables 1 and 2), non-recurrent SNNs were always reported to perform substantially less well compared to their recurrent counterparts. In this study, we managed to raise the performance of lighter, non-recurrent SNNs. Our results with non-recurrent adLIF models on the SHD and SC data sets were even able to surpass those of the best previously reported recurrent SNNs on the same tasks. This allows competitive networks with much fewer trainable parameters, and could lead to hardware implementations that require less space, power and memory.

The average firing rate $\bar{\nu}$ of the implemented spiking networks (over all neurons and all time steps) was observed to consistently converge around $\bar{\nu} \approx 0.1$. To compare the energy consumption of SNNs with ANNs, similarly to Panda et al. (2020), one can count the number of accumulate
(AC) and multiply-and-accumulate (MAC) operations that are required at each time step. Here, we consider ANNs that process sequential inputs and focus on the case with recurrent connections, i.e., RNNs. In contrast to equation (7) where the matrix multiplications involve non-zero real numbers and results in $N_l(N_l-1 + N_l + 1)$ MACs for RNNs, equation (9) only requires $\bar{\nu}N_l(N_l-1 + N_l + 1)$ ACs for SNNs. The first energy gain therefore comes from the sparsity of the spike trains in equation (9), which gets rid of $1 - \bar{\nu} \approx 90\%$ of the required operations as most neurons are not activated. Even if the internal neuronal dynamics of SNNs described by equations (10–12) require additional operations compared to the ANN activation of equation (8), these only scale with the number of hidden units $N_l$ in the current layer $l$, whereas the benefits of sparsity scale with $(N_l)^2$.

Moreover, SNNs replace the MACs by ACs in the dot-product computations as a consequence of the binary nature of spike trains, which constitutes the second gain of energy. As presented by Han et al. (2015) with a 45nm CMOS process, a single 32-bit integer AC operation only requires 0.1pJ compared to 3.2pJ for a MAC. This reduces the energy consumption by another factor of 32. Even if they usually lead to slightly worse accuracies, regularizers can additionally be used to obtain even sparser spike trains and reduce the value of $\bar{\nu}$. By cummulating the advantages of the sparse and binary nature of the information, a recurrent SNN without regularizers already requires roughly 320 times less energy than a non-spiking RNN of the same size.

In addition to being more energy efficient, we have seen that SNNs outperformed RNNs. On all 4 tasks, even the much lighter non-recurrent SNNs (with adaptive neurons and $V = 0$) managed to surpass the performance of standard RNNs. Only gated RNNs were able to compete with SNNs and surpass them in some cases. However, layers of light BRUs and GRUs require 2 and 3 times more operations respectively compared to standard RNN layers, thus expanding the energy gap even more drastically. The sparse event-driven processing of the information in SNNs therefore makes them extremely attractive for reaching lower powered hardware implementations dedicated to real-world applications.

### 4.3 Conclusion

In the introduction we set out three broad goals for the work. In concluding, we have established a SNN method that, on top of being compatible with standard deep learning frameworks, is capable of competing with ANNs on the same tasks, while conserving the advantage of energy efficiency. This represents the main contribution of this paper, which in fact fulfils the second goal. The chosen surrogate gradient approach allows SNNs to be trained with gradient descent like conventional ANNs. The resulting compatibility with modern ANN frameworks combined with the observed scalability of our spiking networks to deep architectures point towards further applications of this method to more advanced tasks. In terms of energy consumption, the implemented SNNs are drastically more efficient compared to ANNs of the same size, showing promising pathways for low-powered hardware implementations of neural networks.

At the same time, we have also achieved the first goal of assessing the more general capability of SNNs in comparison to conventional ANNs. We have shown that the particular combination of spiking neurons, surrogate gradients and automatic differentiation can not only equal but actually outperform strong ANN baselines on speech recognition tasks. In particular, even light non-recurrent SNNs were able to compete with much larger, standard gated recurrent units. This in turn confirms that SNNs do indeed have greater representational capability than their ANN counterparts when applied to temporal data. While the neurons inside a standard ANN all share the
same activation function, the firing behaviours among spiking neurons can become heterogeneous by making the neuron parameters trainable. This allows more complex representations of the information with less neurons inside the network.

The success of this physiologically plausible approach to modelling neural networks indicates that our more general third and last goal is still valid. The experiments do not attempt to say anything about biological function. However, they show that a representational capability, that is available to biological entities, is capable of solving the same problems as (artificial) networks that are known to be capable of exceeding human performance on many tasks. This provides a strong hypothesis for future understanding of the biological mechanisms of the brain.

CONFLICT OF INTEREST STATEMENT

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

AUTHOR CONTRIBUTIONS

AB performed the experiments, did much of the literature search and wrote the bulk of the manuscript. PNG secured funding, supervised the experimental work and assisted with literature and writing.

FUNDING

This project received funding under NAST: Neural Architectures for Speech Technology, Swiss National Science Foundation grant 200021_185010.

DATA AVAILABILITY STATEMENT

The SHD, SSC and HD datasets used for this study can be found at [https://compneuro.net/datasets/](https://compneuro.net/datasets/). The SC dataset is available at [https://www.tensorflow.org/datasets/catalog/speech_commands](https://www.tensorflow.org/datasets/catalog/speech_commands).

SOFTWARE

In order to further encourage the development of spiking neural networks, we are in the process of making our code available open source; a URL will be provided in place of this text.

REFERENCES


Alexandre Bittar et al.  A surrogate gradient spiking baseline for speech command recognition


