# The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks

Benjamin Cramer[ID], Yannik Stradmann[ID], Johannes Schemmel[ID], *Member, IEEE*, and Friedemann Zenke[ID]

*Abstract*—Spiking neural networks are the basis of versatile and power-efficient information processing in the brain. Although we currently lack a detailed understanding of how these networks compute, recently developed optimization techniques allow us to instantiate increasingly complex functional spiking neural networks in-silico. These methods hold the promise to build more efficient non-von-Neumann computing hardware and will offer new vistas in the quest of unraveling brain circuit function. To accelerate the development of such methods, objective ways to compare their performance are indispensable. Presently, however, there are no widely accepted means for comparing the computational performance of spiking neural networks. To address this issue, we introduce two spike-based classification data sets, broadly applicable to benchmark both software and neuromorphic hardware implementations of spiking neural networks. To accomplish this, we developed a general audio-to-spiking conversion procedure inspired by neurophysiology. Furthermore, we applied this conversion to an existing and a novel speech data set. The latter is the free, high-fidelity, and word-level aligned Heidelberg digit data set that we created specifically for this study. By training a range of conventional and spiking classifiers, we show that leveraging spike timing information within these data sets is essential for good classification accuracy. These results serve as the first reference for future performance comparisons of spiking neural networks.

*Index Terms*—Audio, benchmark, classification, data set, neuromorphic computing, spiking neural networks, spoken digits, surrogate gradients.

## I. INTRODUCTION

SPIKING neural networks (SNNs) are biology's solution for fast and versatile information processing. From a computational point of view, SNNs has several desirable properties: They process information in parallel, are noise-tolerant, and highly energy-efficient [1]. Precisely which computations are carried out in a given biological SNN depends

in large part on its connectivity structure. To instantiate such functional connectivity in-silico, a growing number of SNNs training algorithms have been developed [2]–[8] both for conventional computers and neuromorphic hardware [9]–[14]. However, this diversity of learning algorithms urgently calls for principled means to compare them. Unfortunately, widely accepted benchmark data sets for SNNs that would permit such comparisons are scarce [14], [15]. Hence, in this article, we seek to fill this gap by introducing two new broadly applicable classification data sets for SNNs.

In the following, we provide a brief motivation for why benchmarks are crucial before reviewing existing tasks that have been used to assess SNN performance in the past. By analyzing the strengths and shortcomings of these tasks, we motivate our specific choices for the data sets we introduce in this article. Finally, we establish the first set of baselines by testing a range of conventional and SNN classifiers on these data sets.

### A. Why Benchmarks?

The ultimate goal of a benchmark is to provide a quantitative unbiased way of comparing different approaches and methods to the same problem. While each modeler usually works with a set of *private* benchmarks, tailored to their specific problem of study, it is equally important to have shared benchmarks, which ideally everybody agrees to use, to allow for unbiased comparison and to foster constructive competition between approaches [14], [15].

The last decades of machine learning research would be hard to imagine without the ubiquitous MNIST data set [16], for instance. To process MNIST using a SNNs, it has to be transformed into spikes. This transformation step puts comparability at risk by leaving fundamental design decisions to the modeler. Presently, the SNN network community has a shortage of established benchmarks that avoid the conversion step by directly providing spike trains to the end-user. By impeding the quantitative comparison between methods, the lack of suitable benchmarks has the potential to slow down the progress of the SNN research community as a whole.

Since community benchmarks are essential, then why is there little agreement on which benchmark to use? There are several possible reasons, but the most likely ones are the following: First, an existing benchmark may be *unobtainable*. For instance, it could be unpublished, behind a paywall, or too difficult to use. Second, a published benchmark might be *tailored* to a specific problem and, therefore, not general enough

to be of interest to other researchers. Third, a benchmark may be *saturated*, which means that it is already solved with high precision by an existing method. Naturally, this precludes the characterization of improvements over these approaches. Finally, a benchmark could require extensive preprocessing.

The question, therefore, is: What would an ideal benchmark data set for learning in SNNs be? While this question is difficult, if not impossible, to answer, it is probably fair to say that an ideal benchmark should be at least unsaturated, require minimal preprocessing, be sufficiently general, easy to obtain, and free to use.

### B. Previous Work

There are no unified approaches to measuring performance in SNNs which is partially due to the numerous different learning approaches and architectures. SNN architectures can coarsely be categorized into steady-state rate-coding and temporal coding networks, although also hybrids between the two exist. In steady-state rate-coding, SNNs approximate conventional analog neural networks by using an effective firing rate code in which both input and output firing rates remain constant during the presentation of a single stimulus [3], [17], [18]. Inputs to the network enter as Poisson distributed spike trains with rates proportional to the current input level. Similarly, network outputs are given as a firing rate or spike count of designated output units. Because of these input-output specifications, steady-state rate-coding networks can often be trained using network translation [3] and they can be tested on standard machine learning data sets (e.g. MNIST [16], CIFAR10 [19], or SVHN [20]).

The capabilities of SNNs, however, go beyond such rate-coding networks. In temporally coding networks, input and output activity varies during the processing of a single input example. Within this coding scheme, outputs can be either individual spikes [21]–[24], spike trains with predefined firing times [25], varying firing rates [26]–[28], or continuously varying quantities derived from output spikes. The latter are typically defined as linear combinations of low-pass filtered spike trains [26], [27], [29]–[31].

One of the simplest temporal coding benchmarks is the temporal exclusive-OR (XOR) task, which exists in different variations [22], [31], [32]. A simple SNN without hidden layers cannot solve this problem, similar to the Perceptron's inability to solve the regular XOR task. Hence, the temporal XOR is commonly used to demonstrate that a specific method supports hidden-layer learning. In the temporal XOR task, a neural network has to solve a Boolean XOR problem in which the logical *off* and *on* levels correspond to early and late spike times respectively. While the temporal XOR does require a hidden layer to be solved correctly, its intrinsic low-dimensionality and the low number of input patterns render this benchmark saturated. Therefore, its possibilities for quantitative comparison between training methods are limited.

To assess learning in a more fine-grained way, several studies have focused on SNNs' abilities to generate precisely timed output spike trains in more general scenarios [25], [33]–[37]. To that end, it is customary to use several

Poisson input spike trains to generate a specific target spike train. Apart from regular (see [37]), also random output spike trains with increasing length and Poisson statistics have been considered [25]. Similarly, the Tempotron [38] uses an interesting hybrid approach in which random temporally encoded spike input patterns are classified into binary categories corresponding to spiking versus quiescence of a designated output neuron. In the associated benchmark, task performance is measured as the number of binary patterns that can be classified correctly. Although mapping random input spikes to output spikes allows a fine-grained comparison between methods, the aforementioned tasks lack a nonrandom structure.

Finally, some data sets for n-way classification were born out of practical engineering needs. The majority of these data sets are based on the output of neuromorphic sensors like, for instance, the dynamic vision sensor (DVS) [39] or the silicon cochlea [40]. An early example of such a data set is Neuromorphic MNIST [41], which was generated by a DVS recording MNIST digits that were projected on a screen. The digits were moved at certain intervals to elicit spiking responses in the DVS. The task is to identify the corresponding digits from the elicited spikes. This benchmark has been used widely in the SNN community. However, being based on the MNIST data set it is nearing saturation. The DASDIGITS data set [40] was created by processing the TIDIGITS with a 64 channel silicon cochlea. Unfortunately, the license requirements for the derived data set are not entirely clear as the TIDIGITS are released under a proprietary license. Moreover, because the TIDIGITS contains sequences of spoken digits, the task goes beyond a straightforward n-way classification problem and therefore is beyond the scope for several current SNN implementations. More recently, IBM has released the DVS128 Gesture Data set [42] under a Creative Commons license. The data set consists of numerous DVS recordings of 11 unique hand gestures performed by different persons under varying lighting conditions. The spikes in this data set are provided as a continuous data stream, which makes extensive cutting and preprocessing necessary. Finally, the $128 \times 128$ pixel size renders this data set computationally expensive unless additional preprocessing steps such as downsampling are applied.

In this article, we sought to generate two widely applicable SNN benchmarks with comparatively modest computational requirements. Thus, we focused on audio signals of spoken words due to their natural temporal dimension and lower bandwidth compared to video data and developed a processing framework to convert these audio data into spikes. Using this framework, we generated two spike-based data sets for speech classification and keyword spotting that are not saturated by current methods. Moreover, solving these problems with high accuracy requires taking into account spike timing.

## II. METHODS

To improve the quantitative comparison between SNNs, we have created two large spike-based classification data sets from audio data. Specifically, we recorded the Heidelberg digits (HD) data set for this purpose and used the published
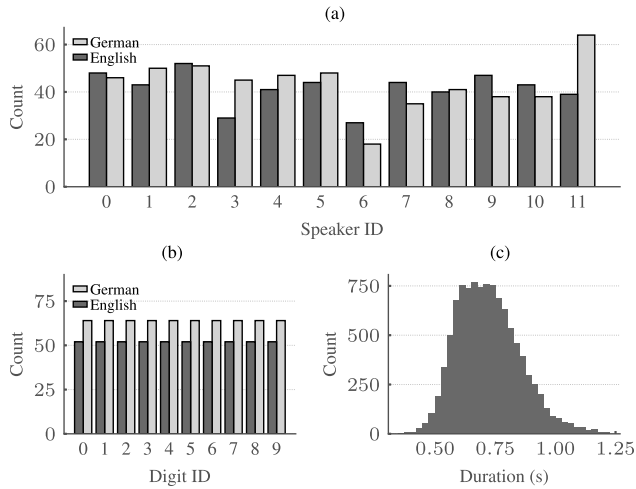
Fig. 1. The Heidelberg Digits HD have a balanced class count and variable temporal duration. The HD consist of 10420 recordings of spoken digits ranging from *zero* to *nine* in English and German language. (a) Histogram of per-speaker digit counts. Variable numbers of digits are available for each speaker and each language. (b) Histogram of per-class digit counts. The data set is balanced in terms of digits within each language. (c) Histogram of audio recording durations. The HD audio recordings were cut for minimal duration to keep computation time at bay.

speech commands (SC) data set by the TensorFlow and AIY teams [43]. In the following, we describe the data sets (Section II-A), the audio-to-spike conversion (Section II-B), and the data format used for publication (Section II-C). We close with a depiction of the SNN model (Section II-D). The nonspiking classifiers are outlined in Section B. All reported error measures in this work correspond to the standard deviation of ten experiments.

### A. Audio Data Sets

In the following, we consider the HD (Section II-A1) and the SC data set (Section II-A2). While the HD were optimized for recording quality and precise audio alignment, the SC are intended to closely mimic real-world conditions for key-word spotting on mobile devices.

*1) Heidelberg Digits:* The HD data set[1] consists of approximately 10 k high-quality recordings of spoken digits ranging from *zero* to *nine* in English and German language. In total 12 speakers were included, six of which were female and six male. The speaker ages ranged from 21 yr to 56 yr with a mean of 29(9)yr. We recorded around 40 digit sequences for each language with a total digit count of 10 420 (see Fig. 1).

The digits were acquired in sequences of ten successive digits. Recordings were performed in a sound-shielded room at the Heidelberg University Hospital with three microphones; two AudioTechnica Pro37 in different positions and a Beyerdynamic M201 TG (Fig. 2). Digitized by a Steinberg MR816 CSX audio interface, recordings were made in WAVE format with a sample rate of 48 kHz and 24 bit precision.

To improve the yield of the following automated processing, a manual pre-selection and cutting of the raw audio tracks were

performed accompanied by conversion to free lossless audio codec (FLAC) format. The cleaned-up tracks were externally mastered [44]. The cutting times of the digit sequences were determined using a gate with speaker-dependent threshold and release time which were optimized by the blackbox-optimizer described in [45]. The loss function was designed to produce ten single files with the lowest possible threshold and shortest gate opening to prevent unnecessary computation during successive analysis and modeling. Additionally, speaker-specific ramp-in and ramp-out times were determined by visual inspection. The final digit files differ in duration due to speaker differences (Fig. 1). 30 ms Hann windows were applied to the start and end of the peak normalized audio signals as further processing stages involve the computation of fast Fourier transformations (FFTs).

To separate the data into training and test sets, we held out two speakers exclusively for the test set. The remainder of the test set was filled with samples (5% of the trials) from speakers also present in the training set. This division allows one to assess a trained network's ability to generalize across speakers.

*2) Speech Commands:* The SC data set[2] is composed of 1 s WAVE-files with 16 kHz sample rate containing a single English word each [43]. It is published under *Creative Commons BY 4.0* license and contains words spoken by 1864 speakers. In this study, we considered version 0.02 with 105 829 audio files, in which a total of 24 single word commands (*Yes*, *No*, *Up*, *Down*, *Left*, *Right*, *On*, *Off*, *Stop*, *Go*, *Backward*, *Forward*, *Follow*, *Learn*, *Zero*, *One*, *Two*, *Three*, *Four*, *Five*, *Six*, *Seven*, *Eight*, *Nine*) were repeated about five times per speaker, whereas ten auxiliary words (*Bed*, *Bird*, *Cat*, *Dog*, *Happy*, *House*, *Marvin*, *Sheila*, *Tree*, and *Wow*) were only repeated approximately once. Partitioning into training, testing, and validation data set was done by a hashing function as described in [43].

For all our purposes, we applied a 30 ms Hann window to the start and end of each waveform. Most importantly, throughout this article, we consider top one classification performance on all 35 different classes which is more difficult than the originally proposed key-word spotting task on only a subset of 12 classes (ten key-words, unknown word, and silence). However, the data can still be used in the originally intended keyword spotting way.

### B. Spike Conversion

The audio files described above served as the basis for our spiking data sets. Audio data were converted into spikes using an artificial model[3] of the inner ear and parts of the ascending auditory pathway (Fig. 2; Section A). This biologically inspired model effectively performs similar signal processing steps as customary spoken language processing applications [46]. First, a hydrodynamic basilar membrane (BM) model (Appendix A-A) causes spatial frequency dispersion, which is comparable to computing a spectrogram with Mel-spaced filter banks. Second, these separated frequencies are

---

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

4                                                                                    IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS
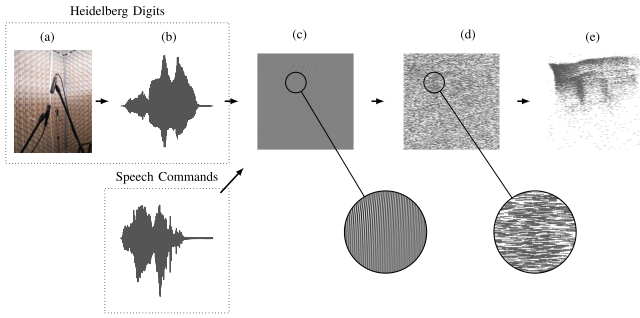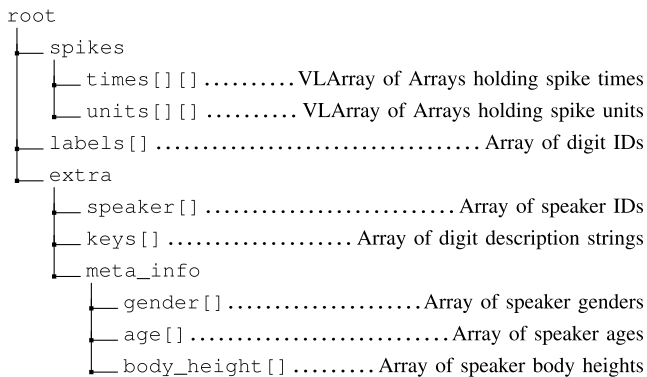


Fig. 2. Processing pipeline for the HD and the SC data set. (a) HD are recorded in a sound-shielded room. (b) Afterward, the resulting audio files are cut and mastered. (c) HD as well as the SC are fed through a hydrodynamic BM model. (d) BM decompositions are converted to phase-coded spikes by use of a transmitter-pool based HC model. (e) Phase-locking is increased by combining multiple spike trains of hair cells at the same position of the basilar membrane in a single bushy cell.

converted to instantaneous firing rates through a biologically motivated transmitter pool based hair cell (HC) model (Appendix A-B), which adds refractory effects, and a layer of bushy cells (BCs) (Appendix A-C) that increase phase locking (see Fig. 2). All model parameters were chosen to mimic biological findings, thereby reducing the amount of free parameters (see Appendix A).

Overall, the inner-ear model approximates the spiking activity observed in the auditory system while retaining a low computational cost. This biologically inspired conversion allowed us to sidestep the issue of user-specific audio-to-spike transformation, which can confound comparability, and served as the basis for our benchmark data sets.

### C. Event-Based Data Format

We used an event-based representation of spikes in the Hierarchical Data Format 5 (HDF5) to facilitate the use of the data sets and to simplify the access to a broader community. This choice was to ensure short download times and ease of access from most common programming environments. For each partition and data set, we provide a single HDF5 file which holds spikes, digit labels, and additional meta information. We made these files publicly available[2] [47], together with supplementary information on the general usage as well as code snippets. A single file is organized as follows:

```
root
├── spikes
│   ├── times[][] ......... VLArray of Arrays holding spike times
│   └── units[][] ......... VLArray of Arrays holding spike units
├── labels[] ................................ Array of digit IDs
└── extra
    ├── speaker[] .......................... Array of speaker IDs
    ├── keys[] .................... Array of digit description strings
    └── meta_info
        ├── gender[] ..................... Array of speaker genders
        ├── age[] ........................... Array of speaker ages
        └── body_height[] ........ Array of speaker body heights
```

In more detail, each element $i$ in *keys* describes the transformation between the digit ID $i$ and the spoken words. Furthermore, the entry $i$ of each array in *meta_info* corresponds to

the information for speaker $i$. The *meta_info* is only available for Spiking Heidelberg digits (SHD).

### D. Spiking Network Models

We trained networks of leaky integrate-and-fire (LIF) neurons with surrogate gradients and backpropagation through time (BPTT) using supervised loss functions to establish a performance reference on the two spiking data sets. In the following, we give a description of the network architectures (Section II-D1), followed by the applied neuron and synapse model (Section II-D2). We close with a depiction of the weight initialization (Section II-D3), the supervised learning algorithm (Section II-D4) and the loss function (Section II-D5) as well as the regularization techniques (Section II-D6).

*1) Network Model:* The spiketrains emitted by the $N_{ch} = 700$ BCs were used to stimulate the actual classification network. In this article, we trained both feed-forward and recurrent networks; each hidden layer contains $N = 128$ LIF neurons. For all network architectures, the last layer was accompanied by a linear readout consisting of leaky integrators which did not spike.

*2) Neuron and Synapse Models:* We considered LIF neurons where the membrane potential $u_i^{(l)}$ of the $i$th neuron in layer $l$ obeys the differential equation

$$\tau_{\text{mem}} \frac{du_i^{(l)}}{dt} = -\left[u_i^{(l)}(t) - u_{\text{leak}}\right] + R I_i^{(l)}(t) \tag{1}$$

with the membrane time constant $\tau_{\text{mem}}$, the input resistance $R$, the leak potential $u_{\text{leak}}$, and the input current $I_i^{(l)}(t)$. Spikes were described by their firing time. The $k$th firing time of neuron $i$ in layer $l$ is denoted by $_k t_i^{(l)}$ and defined by a threshold criterion

$$_k t_i^{(l)} : u_i^{(l)}\left(_k t_i^{(l)}\right) \geq u_{\text{thres}}. \tag{2}$$

Immediately after $_k t_i^{(l)}$, the membrane potential is set to the leak potential $u_i^{(l)}(t) = u_{\text{leak}}$. The synaptic input current onto the $i$th neuron in layer $l$ was generated by the arrival of presynaptic spikes from neuron $j$, $S_j^{(l)}(t) = \sum_k \delta(t - _k t_j^{(l)})$. A common first-order approximation to model the time course of synaptic currents are exponentially decaying currents which sum linearly [48]

$$\frac{dI_i^{(l)}}{dt} = -\frac{I_i^{(l)}(t)}{\tau_{\text{syn}}} + \sum_j W_{ij}^{(l)} S_j^{(l-1)}(t) + \sum_j V_{ij}^{(l)} S_j^{(l)}(t) \tag{3}$$

where the sum runs over all presynaptic partners $j$ and $W_{ij}^{(l)}$ are the corresponding afferent weights from the layer below. The $V_{ij}^{(l)}$ resemble the recurrent connections within each layer.

In this work, the reset was incorporated in (1) through an extra term

$$\frac{du_i^{(l)}}{dt} = -\frac{u_i^{(l)} + u_{\text{leak}} - R I_i^{(l)}}{\tau_{\text{mem}}} + S_i^{(l)}(t)(u_{\text{leak}} - u_{\text{thres}}). \tag{4}$$

To formulate the above equations in discrete time for time step $n$ and stepsize $\delta t$ over a duration $T = n \cdot \delta t$, the output spiketrain $S_i^{(l)}[n]$ of neuron $i$ in layer $l$ at time step $n$ is expressed as a nonlinear function of the membrane potential

$S_i^{(l)}[n] = \Theta(u_i^{(l)} - u_{\text{thres}})$ with the Heaviside function $\Theta$. For small time steps $\delta t$, we can express the synaptic current in discrete time as follows:

$$I_i^{(l)}[n+1] = \kappa I_i^{(l)}[n] + \sum_j W_{ij}^{(l)} S_j^{(l)}[n] + \sum_j V_{ij}^{(l)} S_j^{(l)}[n]. \quad (5)$$

Furthermore, by asserting $u_{\text{leak}} = 0$ and $u_{\text{thres}} = 1$, the membrane potential can be written compactly as

$$u_i^{(l)}[n+1] = \lambda u_i^{(l)}[n]\left(1 - S_i^{(l)}[n]\right) + (1 - \lambda)I_i^{(l)}[n] \quad (6)$$

where we have set $R = (1 - \lambda)$ and introduced the constants $\kappa \equiv \exp(-\delta t/\tau_{\text{syn}})$ and $\lambda \equiv \exp(-\delta t/\tau_{\text{mem}})$.

*3) Weight Initialization:* In all our spiking network simulations we use Kaiming's uniform initialization [49] for the weights $W_{ij}$ and $V_{ij}$. Specifically, the initial weights were drawn independently from a uniform distribution $\mathcal{U}(-\sqrt{k}, \sqrt{k})$ with $k = (\text{\# afferent connections})^{-1}$.

*4) Supervised Learning:* The goal of learning was to minimize a cost function $\mathcal{L}$ over the entire data set. To achieve this, surrogate gradient descent was applied which modifies the network parameters $W_{ij}$

$$W_{ij} \leftarrow W_{ij} - \eta \frac{\partial \mathcal{L}}{\partial W_{ij}} \quad (7)$$

with the learning rate $\eta$. In more detail, we used custom PyTorch [50] code implementing the SNNs. Surrogate gradients were computed using PyTorch's automatic differentiation capabilities by overloading the derivative of the spiking nonlinearity with a differentiable function as described previously [8], [51]. An instructive example of such an implementation in PyTorch can be found online.[4] Specifically, we chose a fast sigmoid for the surrogate gradient

$$\sigma\left(u_i^{(l)}\right) = \frac{u_i^{(l)}}{1 + \beta|u_i^{(l)}|} \quad (8)$$

with the steepness parameter $\beta$.

*5) Loss Functions:* We applied a cross entropy loss to the activity of the readout layer $l = L$. On data with $N_{\text{batch}}$ samples and $N_{\text{class}}$ classes, $\{(\mathbf{x}_s, y_s)|s = 1, \dots, N_{\text{batch}}; y_s \in \{1, \dots, N_{\text{class}}\}\}$ it takes the form

$$\mathcal{L} = -\frac{1}{N_{\text{batch}}} \sum_{s=1}^{N_{\text{batch}}} \mathbb{1}(i = y_s)$$
$$\cdot \log\left\{\frac{\exp\left(u_i^{(L)}[\tilde{n}_i]\right)}{\sum_{i=1}^{N_{\text{class}}} \exp\left(u_i^{(L)}[\tilde{n}_i]\right)}\right\} \quad (9)$$

with the indicator function $\mathbb{1}$. We tested the following two choices for the time step $\tilde{n}$: For the *max-over-time* loss, the time step with maximal membrane potential for each readout unit was considered $\tilde{n}_i = \text{argmax}_n u_i^{(L)}[n]$. In contrast, the last time step $T$ for all samples was chosen for each readout neuron $\tilde{n}_i = T$ in case of the *last-time-step* loss. We minimized the cross entropy in (9) using the Adamax optimizer [52].

[4]https://github.com/fzenke/spytorch

*6) Regularization:* For our experiments, we added synaptic regularization terms to the loss function to avoid pathologically high or low firing rates. In more detail, we used two different regularization terms: As a first term, we used a per neuron lower threshold spike count regularization of the form

$$\mathcal{L}_1 = \frac{s_l}{N_{\text{batch}} + N} \sum_{s=1}^{N_{\text{batch}}} \sum_{i=1}^{N} \left[\max\left\{0, \frac{1}{T}\sum_{n=1}^{T} S_i^{(l)}[n] - \theta_l\right\}\right]^2 \quad (10)$$

with strength $s_l$, and threshold $\theta_l$. Second, we used an upper threshold mean population spike count regularization

$$\mathcal{L}_2 = \frac{s_u}{N_{\text{batch}}} \sum_{s=1}^{N_{\text{batch}}} \left[\max\left\{0, \frac{1}{N}\sum_{i=1}^{N}\sum_{n=1}^{T} S_i^{(l)}[n] - \theta_u\right\}\right]^2 \quad (11)$$

with strength $s_u$, and threshold $\theta_u$.

## III. RESULTS

To analyze the relevance of our newly created spiking data sets we first sought to establish that the data sets were not saturated and that spike timing information is essential to solve the tasks with high accuracy. To test this, we first generated a reduced version of the data sets in which we removed all temporal information. To that end, we computed spike count patterns from both data sets, which, by design, do not contain temporal information about the stimuli. Using these reduced spike count data sets, we then trained different linear and nonlinear support vector machine (SVM) classifiers (Section B-A) and measured their classification performance on the respective test sets. We found that while a linear SVM readily overfitted the data in the case of SHD, its test performance only marginally exceeded the 55% accuracy mark [Fig. 3(a)]. For the spiking speech commands (SSC), overfitting was less pronounced, but also the overall test accuracy dropped to 20% [Fig. 3(b)]. Thus linear classifiers provided a low degree of generalization.

To assess whether this situation was different for nonlinear classifiers, we trained SVMs with polynomial kernels up to a degree of 3. For these kernels, overfitting was less pronounced. Slightly better performance of about 60% on the SHD and 30% on the spiking SC (SSC) was achieved when using a SVM with a radial basis function (RBF) kernel. The performance on the SHD test set, which includes speakers that are not part of the training set, was noticeably lower compared to the accuracy on the validation data. Especially, for polynomial and RBF kernels the generalization across speakers was worse than for the linear kernel [Fig. 3(a)]. In contrast, we found the performance on the SSC test set to be on par with the accuracy on the validation set [Fig. 3(b)], which is most likely an effect of the uniform speaker distribution. These results illustrate that both linear and nonlinear classifiers trained on spike count patterns without temporal information were unable to surpass the 60% accuracy mark for the SHD and the 30% mark for the SSC data set. Therefore, spike counts are not sufficient to achieve high classification accuracy on the studied data sets.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

6                                                                                                              IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS

Next, we wanted to assess whether decoding accuracy could be improved when training classifiers that have explicit access to temporal information of the spike times. Therefore, we trained long short-term memorys (LSTMs) on temporal histograms of spiking activity (Section B-B). In spite of the small size of the SHD data set, LSTMs showed reduced over-fitting and were able to solve the classification problem with an accuracy of 85.7(14)% [Fig. 3(a)] which was substantially higher than the best performing SVM. Similarly, for the SSC data set the LSTM test accuracy 75.0(2)% was more than twice as high as the best-performing classifier on the spike count data. However, the degree of overfitting was slightly higher than on SHD.

Since both, kernel machines and LSTMs, were affected by overfitting, we tested whether the performance could be increased with convolutional neural networkss (CNNs) due to their inductive bias on translation invariance in both frequency and time and their reduced number of parameters. To that end, we binned spikes in spatio-temporal histograms and trained a CNN classifier (Section B-C). CNNs showed the least amount of overfitting among all tested classifiers; the accuracy dropped by only 1.4% on SHD and by 1.5% on SSC (Fig. 3). In particular, the performance on the SHD test data was on par with the one on the validation set, demonstrating a high degree of generalization.

These findings highlight that the temporal information contained in both data sets can be exploited by suitable neural network architectures. Moreover, these results provide a lower bound on the performance ceiling for both data sets. It seems likely that a more careful architecture search and hyperparameter tuning will only improve upon these results. Thus, both the SHD and the SSC will be useful for quantitative comparison between SNNs up to at least these empirical accuracy values.

### A. Training Spiking Neural Networks

Having established that both spiking data sets contain useful temporal information that can be read out by a suitable classifier, we sought to train SNNs of LIF neurons using BPTT to establish the first set of baselines and to assess their generalization properties. One problem with training SNNs with gradient descent arises because the derivative of the neural activation function appears in the evaluation of the gradient. Since spiking is an intrinsically discontinuous process, the resulting gradients are ill-defined. To nevertheless train networks of LIF neurons using supervised loss functions, we used a surrogate gradient approach [8]. Surrogate gradients can be seen as a continuous relaxation of the real gradients of a SNN which can be implemented as an in-place replacement while performing BPTT. Importantly, we did not change the neuron model and the associated forward-pass of the model, but used a fast sigmoid as a surrogate activation function when computing gradients (Methods Section II-D4).

Although not a requirement [8], [53], we only considered SNNs with fixed, finite time constants on the order of ms inspired by biology. Because of this constraint, we investigate two different loss functions for both LSTMs and SNNs [Fig. 4(a)]. The results for LSTMs shown in Fig. 3 were
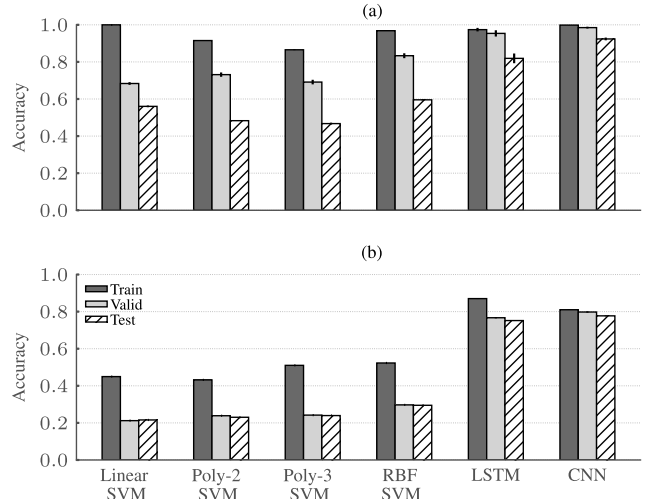


Fig. 3.    Temporal information is essential to classify the SHD and the SSC data sets with high accuracy. (a) Bar graph of classification accuracy for different SVMs trained on spike count vectors and LSTM as well as CNN classifiers trained on the binned spiketrains of the SHD data set. Classification accuracy on SHD is substantially higher for LSTMs and CNNs which also show a lower degree of overfitting. (b) Same as in (a), but showing performance on the SSC. LSTMs and CNNs with access to temporal information outperform the SVM classifiers by a large margin.

obtained by training with a *last-time-step loss*, where the activation of the last time step of each example and readout unit was used to calculate the cross entropy loss at the output. In addition, we also considered a *max-over-time loss*, in which the time step with maximum activation of each readout unit was taken into account [Fig. 4(a)]. This loss function is motivated by the Tempotron [38] in which the network signals its decision about the class membership of the applied input pattern by whether a neuron spiked or not.

We evaluated the performance of LSTMs and SNNs for both aforementioned loss functions on the SHD. Training LSTMs with a cross entropy loss based on the activity of the last time step of every sample was associated with high performance in contrast to SNNs [Fig. 4(b)]. The slightly reduced performance of feed-forward SNNs trained with last-time-step loss compared to recurrently connected spiking neural networks (RSNNs) suggests that time constants were too low to provide all necessary information at the last time step. This was presumably due to active memory implemented through reverberating activity through the recurrent connections. Overall, SNNs performed better in combination with the max-over-time loss function [Fig. 4(c)]. Also LSTMs showed increased performance in combination with a max-over-time loss; the validation accuracy increased from 95.4(17)% for the last-time-step loss to 97.2(09)% for the max-over-time loss. Motivated by these results, we used a max-over-time loss for SNNs as well as LSTMs throughout the remainder of this article.

Surrogate gradient learning introduces a new hyperparameter $\beta$ associated with the steepness of the surrogate derivative (Methods Equation (2)). Because changes in $\beta$ may require a different optimal learning rate $\eta$, we performed a grid search over $\beta$ and $\eta$ based on a single-layer RSNN architecture
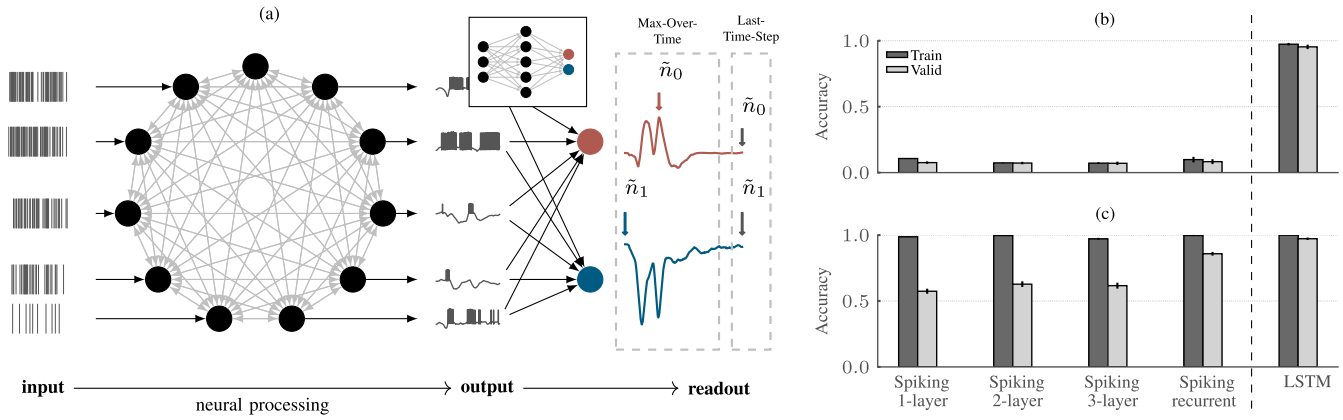
Fig. 4. Setup and multiple choices of loss functions for SNNs. (a) Schematic of a single layer recurrent network with two readout units. We applied two different loss functions for LSTMs and SNNs: First, a max-over-time loss was considered, where the time step with the maximal activity of each readout was used to calculate the cross entropy (marked by colored arrows). Second, a last-time-step loss was utilized where only the last time step of the activation was considered in the calculation of the cross entropy (marked by gray arrow). The inset illustrates the corresponding feed-forward topology. (b) Bar graph of classification accuracy for different SNNs and a LSTM on the SHD. Only the LSTM generalized well when trained with a last-time-step loss. (c) Same as in (b), but showing performance for a max-over-time loss. Overall, SNNs and LSTMs performed better when trained with the max-over-time loss.
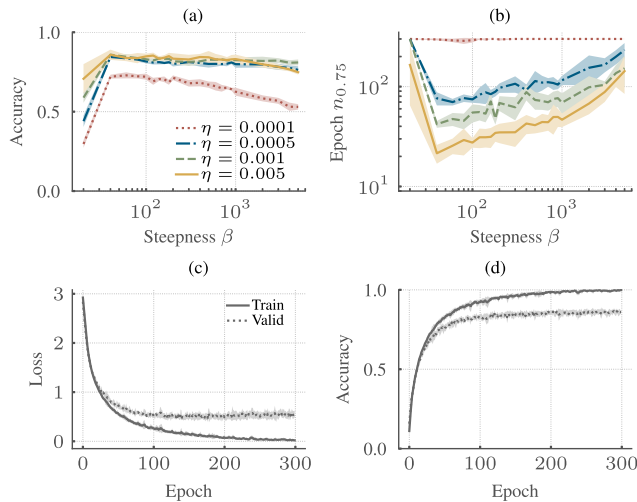


Fig. 5. Accuracy, but not convergence time is only mildly affected by the steepness $\beta$ of the surrogate derivative. (a) Accuracy as a function of $\beta$ on a validation set of the SHD for different learning rates $\eta$. Performance is highest for a wide range of $\beta$ values ($\beta \geq 40$) and depends only slightly on $\eta$. (b) Number of epochs needed to reach an accuracy > 0.75, $n_{0.75}$. In contrast to the performance, $n_{0.75}$ strongly depends on both $\beta$ and $\eta$. (c) Loss curves on the SHD for $\beta = 40$ and $\eta = 10^{-3}$. (d) Same as in (c), but showing the accuracy on the SHD.

trained on the SHD. We found that sensible combinations for both parameters lead to stable performance plateaus over a large range of values [Fig. 5(a)]. Only for small $\beta$ the accuracy dropped dramatically, whereas it decreased only slowly for high values. Interestingly, the learning rate had hardly any effect on peak performance for the tested parameter values. As expected, convergence speed heavily depended on both $\eta$ and $\beta$. These results motivated us to use $\beta = 40$ and $\eta = 1 \times 10^{-3}$ for all SNNs architectures presented in this article unless mentioned otherwise. For this choice, the performance of the RSNN on the validation set reached its peak after about 150 epochs [Fig. 5(d)]. Additional training only increased performance on the training data set [Fig. 5(c)], but did not impact generalization [Fig. 5(d)].

With the parameter choices discussed above, we trained various SNN architectures on the SHD and the SSC. To that end, we considered multilayer feed-forward SNNs with $l$ layers and a single-layer RSNN. Interestingly, increasing the number of hidden layers $l$ did not notably improve performance on the SHD [Fig. 6(a)]. In addition, all choices of $l$ caused high levels of overfitting. Moreover, feed-forward SNNs reached slightly lower accuracy levels than the SVMs on the SHD [Fig. 3(a)]. For the larger SSC data set, the degree of overfitting was much smaller [Fig. 6(b)] and performance was markedly better than the one reached by SVMs [Fig. 3(b)]. Here, increasing the number of layers of feed-forward SNNs led to a monotonic increase of performance on the test set from 32.5(5)% for a single layer to 41.0(5)% in the three-layer case ($l = 3$). However, when testing RSNNs, we found consistently higher performance and improved generalization across speakers. In comparison to the accuracy of LSTMs, RSNNs showed higher overfitting and generalized less well across speakers. The RSNN achieved the highest accuracy of 71.4(19)% on the SHD and 50.9(11)% on the SSC which was still less than the LSTM with 85.7(14)% on the SHD and 75.0(2)% on the SSC.

### B. Generalization Across Speakers and Data Sets

For robust spoken word classification, the generalization across speakers is a key feature. This generalization can be assessed by evaluating the accuracy per speaker on SHD, as the digits spoken by speakers four and five are only present in the test set. We compared the performance on the digits of the held-out speakers to all other speakers and found a clear performance drop across all classification methods for the speakers four and five [Fig. 7(a) and (c)]. For SVMs, the linear kernel led to the smallest accuracy drop of about 18%, whereas we found a decrease of 26% for the RBF kernel. CNNs generalized best with a drop of only 8%, followed by the LSTMs with 10%. Among SNNs, feed-forward architectures were most strongly affected with a drop of about 24% to 27%. RSNNs, however, only underwent a decline of 21% in
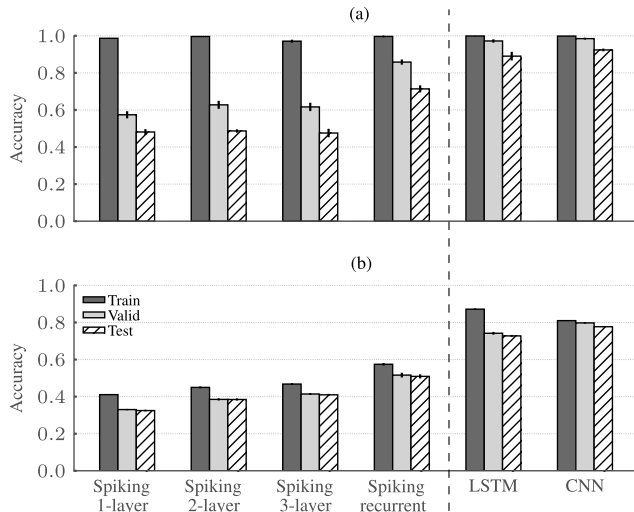
Fig. 6. Recurrent SNNs outperform feed-forward architectures on both data sets. (a) Bar graph of classification accuracy for different SNN architectures on the SHD. The accuracy reached by RSNN is comparable to the performance of LSTMs with a max-over-time loss. Increasing the number of layers in feed-forward architectures hardly affected performance. (b) Same as in (a), but showing performance on the SSC. The performance of SNNs was lower than the one reached by LSTMs. In contrast to (a), an increasing number of layers leads to a monotonic increase in accuracy.
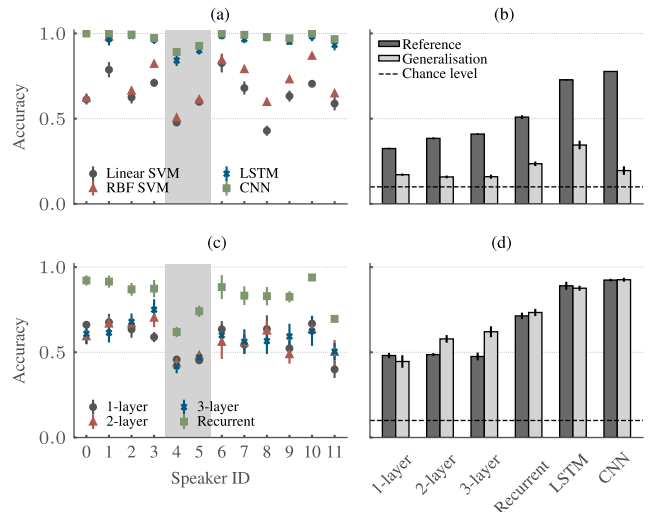


Fig. 7. Networks generalize across speakers and data sets. By reserving two speakers for the test set, the SHD data set allows to assess speaker generalization performance. (a) Per-speaker classification accuracy on the test set of the SHD for different classifiers. A clear decrease in performance is observable for samples spoken by the held-out speakers four and five (highlighted). (b) Bar graph of the performance of SNNs, LSTM and CNN trained on the SHD and tested on the English digits of the SSC. The reference is given by the performance on the SHD test set and the generalization by the performance on the English digits of the SSC test set. Accuracy on the SSC digits is substantially lower than on the digits in the SSC test set. (c) Same as in (a), but showing the per-speaker accuracy of SNN. As for (a), a decrease in performance for the held-out speakers is observed. (d) Same as in (b), but showing the performance of networks when trained on SSC and tested on the English digits of the SHD. As opposed to (b), networks trained on the SSC digits generalize well across data sets.

performance [Fig. 7(b)]. This illustrates that the composition of the test set of SHD can provide meaningful information with regard to generalization across speakers.

Because English digits are part of both data sets, we were able to test the generalization across data sets by training SNNs, LSTM and CNNs on the full SHD data set while testing on a restricted SSC data set and vice versa [Fig. 7(b) and (d)]. For testing, the data sets were restricted to the common English digits *zero* to *nine*. Perhaps not surprisingly, networks generalized better, when trained on the larger SSC data set as a reference and tested on SHD. Nevertheless, all architectures trained on the SHD and tested on the SSC reached performance levels above chance. Again, recurrent architectures reached the highest performance among all tested SNNs.

## C. Improving Generalization Performance Through Data Augmentation and Larger Networks

We first tested the effect of network size on the generalization performance of the RSNN architecture using the smaller SHD data set. As expected, we found that increasing the network size up to 1024 neurons indeed resulted in improved validation and test accuracy up to 76.5(10)% with 1024 neurons [Fig. 8(a)].

Next, we compressed the input in terms of channels, since a similar modification led to good results for the CNNs (see Fig. 6). To that end, we coarse-grained the inputs by condensing the spike trains of neighboring input units. Merging ten neighboring channels did improve validation accuracy with an associated test accuracy of 72.2(16)% [Fig. 8(b)]. This choice resulted in an overall channel count of 70. Despite the only slight increase in test accuracy, the corresponding networks
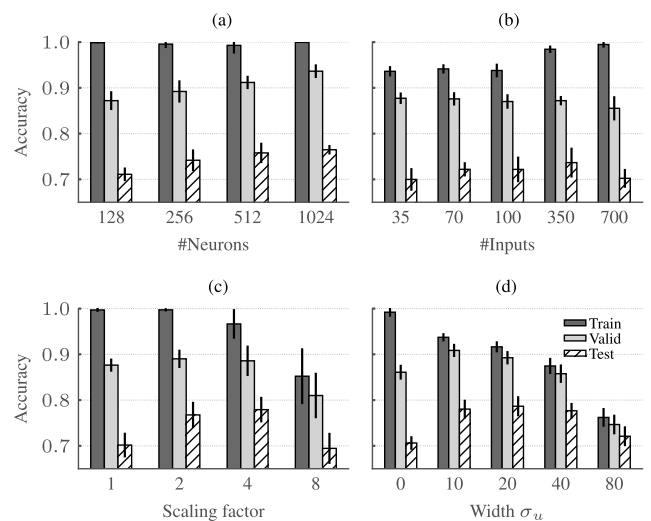


Fig. 8. Generalization performance improves with network size and data augmentation. (a) Bar graph of classification accuracy on the SHD for different network sizes. Larger networks generalize best. (b) Same as before, but showing the accuracy for different numbers of input channels. Networks generalized best when the spikes of the 700 input units were compressed to 70 units. (c) Network performance for temporally scaled synaptic and membrane time constants. The highest performance on the validation set was reached for an expansive scaling factor of 4. (d) Classification accuracy for spatially noisy (channel jitter) input spikes. An optimum occurs for a standard deviation of $\sigma_u = 20$ channels.

with fewer channels benefited from a smaller computational footprint.

We next studied whether compressing the time domain would bear additional benefits. In more detail, we rescaled all neuronal time constants in the network. For a scaling factor

TABLE I
PERFORMANCE COMPARISON

| | Architecture | SHD | SSC |
|---|---|---|---|
| **SVM** | Linear | $56.0 \pm 0.4$ | $21.6 \pm 0.0$ |
| | Poly-2 | $48.3 \pm 0.2$ | $23.0 \pm 0.0$ |
| | Poly-3 | $46.7 \pm 0.5$ | $23.9 \pm 0.0$ |
| | RBF | $60.0 \pm 0.3$ | $29.5 \pm 0.0$ |
| **LSTM**[1] | | $89.0 \pm 0.2$ | $73.0 \pm 0.1$ |
| **CNN** | | $92.4 \pm 0.7$ | $77.7 \pm 0.2$ |
| **Spiking**[1] | 1-layer | $48.1 \pm 1.6$ | $32.5 \pm 0.5$ |
| | 2-layer | $48.6 \pm 0.9$ | $38.5 \pm 0.6$ |
| | 3-layer | $47.5 \pm 2.3$ | $41.0 \pm 0.5$ |
| | Recurrent | $71.4 \pm 1.9$ | $50.9 \pm 1.1$ |
| | Recurrent (best effort)[2] | $83.2 \pm 1.3$ | |
| | SNU[2] | $79.0 \pm 1.6$ | |

[1] Trained with max-over-time loss
[2] 1024 neurons + data augmentation (optimized time constants and channel number combined with noise injection)

of 4 corresponding to $\tau_{\mathrm{mem}} = 80\,\mathrm{ms}$ and $\tau_{\mathrm{syn}} = 40\,\mathrm{ms}$, the manipulation led to a marked reduction of overfitting that also resulted in a higher test accuracy of 79.9(28)% [Fig. 8(c)].

As an alternative means to reduce overfitting, we sought to explored noise injection at the input layer. Specifically, we implemented event-based spike jitter across channels by adding a random number $\mathcal{N}(0, \sigma_u)$ to the unit index $i$ of every input spike, which was then rounded to the nearest integer. We found that such input noise was effective in decreasing overfitting. For instance, a value $\sigma_u = 20$ led to an increased test accuracy of 78.7(22)% [Fig. 8(d)].

When combined, the aforementioned strategies resulted in a further improved *best effort* test accuracy of 83.2(13)% on the SHD test set (Table I). This is not only noticeably higher than the previous result of 71.4(19)% obtained by the RSNN, but also exceeds the effect of every contribution on its own.

Finally, to relate our LIF neurons to other work on RSNNs, we also evaluated the performance of networks composed of spiking neural units (SNUs) on the SHD. Although closely related to our LIF neuron model, spiking neural units (SNUs) feature delta synapses and rely on a different surrogate derivative. When trained under the same conditions as our *best effort* network, the SNU network reached a test accuracy of 79.0(16)% (Table I) which is slightly lower than networks using our LIF neuron model.

## IV. DISCUSSION

In this article, we introduced two new public domain spike-based classification data sets to facilitate the quantitative comparison of SNNs. Furthermore, we provide the first set of baselines for future comparisons by training a range of spiking and nonspiking classifiers.

With these developments, we address a lack of comprehensive benchmark data sets for SNNs. To advance the neuromorphic computing field, we need a set of benchmarks that pose real-world challenges to quantify gains and standardize evaluation across different platforms [15]. We view the data sets in this article as our contribution toward this goal. But, since it is difficult to foresee the pace of future developments, we acknowledge that the present data sets may not prove final. Thus, to facilitate their refinement, extension, and the creation of novel data sets, we released in addition to the spiking data,

our conversion software[3] and raw audio data sets[2], both under permissive public domain licenses.

Both spiking data sets are based on auditory classification tasks but were derived from data that was acquired in different recording settings. We chose audio data sets as the basis for our benchmarks because audio has a temporal dimension which makes it a natural choice for spike-based processing. However, in contrast to movie data, audio requires fewer input channels for a faithful representation, which renders the derived spiking data sets computationally more tractable.

We did not use one of the other existing audio data sets as a basis for the spiking version for different reasons. For instance, a large body of spoken digits is provided by the TIDIGITS data set [54]. However, this data set is only available under a commercial license and we were aiming for fully open data sets. In contrast, the Free Spoken Digit Data set [55] is available under Creative Commons BY 4.0 license. Since this data set only contains 2k recordings with an overall lower recording and alignment quality, we deemed recording the HD as a necessary contribution. Other data sets, such as Mozilla's Common Voice [56], LibriSpeech [57], and TED-LIUM [58] are also publicly available. However, these data sets pose more challenging speech detection problems since they are only aligned at the sentence level. Such more challenging tasks are left for future research on functional SNNs. The Spoken Wikipedia Corpora [59], for instance, also provides alignment at the word level but requires further preprocessing such as the dissection of audio files into separate words. Moreover, the pure size and imbalance in samples per class render the data set more challenging. We, therefore, left its conversion for future work.

The only existing public domain data set with word-level alignment, tractable size, and preprocessing requirements that we were aware of at the time of writing this manuscript was the SC data set. This is the reason why we chose to base one spiking benchmark on the SC while simultaneously providing the separate and the smaller HD data set with the higher recording quality and alignment precision. Finally, the high-fidelity recordings of the HD also make it suitable for quantitative evaluation of the impact of noise on network performance, because well-characterized levels of noise can be added.

The spike conversion step consists of a published physical inner-ear model [60] followed by an established hair-cell model [61]. The processing chain is completed by a single layer of BCs to increase phase-locking and to decrease the overall number of spikes. This approach is similar to the publicly available DASDIGIT data set [40]. DASDIGIT is composed of recordings from the TIDIGIT data set [54] which have been played to a dynamic audio sensor with $2 \times 64$ frequency selective channels. In contrast to the SHD and the SSC, the raw audio files of the TIDIGIT data set are only available under a commercial license. Also, the frequency resolution, measured in frequency selective bands of the BM model, is about a factor of 10 lower. As the software used for processing the SHD and the SSC data sets are publicly available, it is straightforward to extend the present data sets.

This step is more difficult for DASDIGIT, because it requires a dynamic audio sensor.

We standardize the conversion step from raw audio signals to spikes by generating spikes from the HD and the SC audio data sets. In doing so, we both improve the usability settings and reduce a common source of performance variability due to differences in the preprocessing pipelines of the end-user.

To establish the first set of baselines (Table I), we trained a range of nonspiking and spiking classifiers on both the SHD as well as the SSC. In comparing the performance on the full data sets with performance obtained on reduced spike count data sets, we found that the temporal information available in the spike times can be leveraged for better classification by suitable classifiers. Moreover, architectures with explicit recurrence, like LSTMs and RSNNs, were the best performing models among all architectures we tested. Most likely, the reverberating activity through recurrent connections implements the required memory, thereby bridging the gap between neural time constants and audio features. Therefore, the inclusion of additional state variables evolving on a slower time scale as in [5] will be an interesting extension to improve the performance of SNNs.

In the present manuscript, we trained SNNs using surrogate gradients in combination with BPTT [5]–[8], [62], [63]. However, it is essential to realize that there exists a plethora of alternative gradient-based approaches based on network translation [3], [64]–[66], single spike timing [23], [67], mean firing rate [68], [69], and stochastic approximations [37], [70]–[72]. Furthermore, there are biologically inspired online approximations of surrogate gradients [2], [73], [74] and, finally, a body of work has used biologically motivated spike-timing dependent plasticity (STDP)-like learning rules [75], [76] (see [4] for a comprehensive review). An in-depth comparison of this plethora of approaches was beyond the scope of this manuscript. However, the present data sets might prove useful in facilitating a more detailed comparison of the work mentioned above. As such, it is left as interesting future work to study how STDP interacts with the present data sets.

Our analysis of the SHD and the SSC using LSTMs and SNNs showed that the choice of loss functions can have a marked effect on classification performance. While LSTMs perform best with a last-time-step loss, in which only the last time step is used to calculate the cross entropy loss, SNNs achieved their highest accuracy for a max-over-time loss, in which the maximum membrane potential of each readout unit is considered. A detailed analysis of suitable cost functions for training SNNs is an interesting direction for future research.

In summary, we have introduced two versatile and open spiking data sets and conducted a first set of performance measurements using SNN classifiers. This constitutes an important step forward toward the more quantitative comparison of functional SNNs in-silico both on conventional computers and neuromorphic hardware.

## APPENDIX A
## INNER EAR MODEL

Audio data were converted into spikes using a model of the inner ear and the ascending auditory pathway (see Fig. 2)
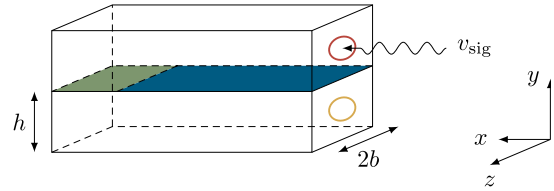


Fig. 9.    Schematic view of the BM model. The BM (blue) separates the *scala tympani* (lower chamber) from the *scala vestibuli* (upper chamber). At the *helicotrema* (green), the two scalae are connected. The *scala tympani* ends in the *round window* (yellow). A sound wave $v_{\text{sig}}$ is penetrating the eardrum, applying pressure at the *oval window* (red) by moving the *ossicles*, leading to compression and slower traveling wave. We have neglected the *scala media* [60] and consider a stretched form.

which combines a basilar membrane (bm) model with a population of hair cells (HCs) followed by a population of bushy cells (BCs) for spike generation. We now describe the model components individually.

### A. Basilar Membrane Model

As a complete consideration of hydrodynamic BM models is beyond the scope of this manuscript, we closely follow the steps of [60] and highlight the key steps of their derivation. A fundamental aspect of a cochlea model is the interaction between a fluid and a membrane causing spatial frequency dispersion [60], [77], [78]. Key mechanical features of the cochlea are covered by the simplified geometry of the BM in Fig. 9. Here, we assumed the fluid to be inviscid and incompressible. Furthermore, we expect the oscillations to be small that the fluid can be described as linear. The BM was expressed in terms of its mechanical impedance $\xi(x, \omega)$ which depends on the position in the $x$-direction and the angular frequency $\omega = 2\pi \nu$

$$\xi(x, \omega) = \frac{1}{i\omega}\left[S(x) - \omega^2 m + i\omega R(x)\right] \tag{12}$$

with a transversal stiffness $S(x) = C_0 e^{-\alpha x} - a$, a resistance $R(x) = R_0 e^{-\alpha x/2}$ and an effective mass $m$ [77]. The damping of the BM was described by $\gamma = R_0/(C_0 m)^{1/2}$. Variations of the stiffness over several orders of magnitude allowed to encompass the entire range of audible frequencies.

Let $p(\boldsymbol{x}, \omega)$ be the difference between the pressure in the upper and lower chambers. The following expression fulfills the boundary conditions $v_y = 0$ for $y = h$, and $v_z = 0$ at $z = \pm b$ [77]

$$
\begin{aligned}
p(\boldsymbol{x}, \omega) \\
= \sum_n \int_0^\infty \frac{dk}{2\pi} e^{-ikx} p_0(k) \left[ \frac{\cosh(m_0(h - y))}{\cosh(m_0 h)} \right. \\
\left. + \frac{m_0 \tanh(m_0 h) \cosh(m_1(h - y))}{m_1 \tanh(m_1 h) \cosh(m_1 h)} \cdot \cos\left(\frac{\pi z n}{b}\right) \right].
\end{aligned}
\tag{13}
$$

The Laplace equation yields expressions for $m_0 = k$ and $m_1 = (k^2 + \pi^2/b^2)^{1/2}$. Only the principal mode of excitation in the $z$-direction was considered by setting $n = 1$. With the assumptions made above, the Euler equation reads for the $y$-component of the velocity in the middle of the BM

$$\partial_y p(x, \omega) = -i\omega\rho v_y(x, \omega) = \frac{2i\omega\rho}{\xi(x, \omega)} p(x, \omega) \tag{14}$$
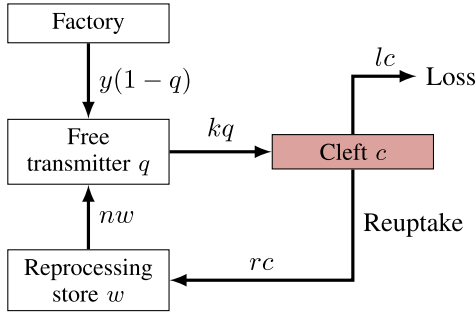
Fig. 10. Schematic view of the transmitter flow within the HC model. Figure adapted from [79]. The model comprises four transmitter pools which allow to describe the transmitter concentration in the synaptic cleft.

where we dropped the $y$ and $z$ argument for readability. In the following, we consider the limiting case of long waves with $kh \ll 1$. By combining (13) and (14), one gets

$$\partial_x^2 p(x, \omega) = \frac{i\omega\rho}{h\xi(x, \omega)} p(x, \omega) \quad (15)$$

where the replacement $\hat{p}(k) \to p(x)$, $k \to i\partial_x$ and $k^2 \to \partial_x^2$ has been applied. Here, $\hat{p}(k)$ denotes the Fourier transform of $p(x, 0, 0)$. The solution of this equation was approximated by

$$p(x, \omega) = \sqrt{\frac{G(x, \omega)}{g(x, \omega)}} H_0^{(2)}(G(x, \omega)) \quad (16)$$

where $H_0^{(2)}$ is the second Hankel function and $g(x, \omega)$ and $G(x, \omega)$ are given by

$$g(x, \omega) = \omega \sqrt{\frac{\rho}{h\xi(x, \omega)}}, \quad (17)$$

$$G(x, \omega) = \int_0^x dx' g(x', \omega) + \frac{2}{\alpha} g(0, \omega). \quad (18)$$

An analytical expression for $G(x, \omega)$ can be found in [60]. The model was applied to a given stimulus by

$$v_y(x, t) = \int \frac{d\omega}{2\pi} i Z_{\text{in}} \frac{v_y(x, \omega)}{p(0, \omega)} e^{-i\omega t} v_{\text{sig}}(\omega) \quad (19)$$

where $v_{\text{sig}}(\omega)$ is the Fourier transformation of the stimulus. The input impedance of the cochlea was modeled by

$$Z_{\text{in}}(\omega) = \frac{p(x = 0)}{v_x(x = 0)} \approx \sqrt{\frac{2C_0}{h}} \frac{i J_0(\zeta) + Y_0(\zeta)}{J_1(\zeta) - i Y_1(\zeta)} \quad (20)$$

with the Bessel functions of first $J_\beta$ and second $Y_\beta$ kind of order $\beta$ and $\zeta = 2\omega/\alpha(2/(hC_0))^{1/2}$.

To process the audio data, we evaluated $v_y(x, t)$ in the range $(0, 3.5\,\text{cm}]$ in $N_{ch}$ steps of equal size. Specifically, we chose $N_{ch} = 700$ as a compromise between a faithful representation of the underlying audio signal and manageable computational cost when using the data set. Before applying the BM model, each recording $v_{\text{sig}}(t)$ was normalized to a root mean square (rms) value of $0.3\,\text{cm s}^{-1}$. For all model parameters see Table II.

### B. Hair Cell Model

The transformation of the movement of the BM to spikes was realized by the HC. The following description illustrates the key steps of [79], to which we refer for further details.

TABLE II
MODEL PARAMETERS

| Parameter | Symbol | Value | |
|---|---|---|---|
| Damping const. | $\gamma$ | $0.15\,\text{s}^{-1}$ | |
| Greenwoods const. | $a$ | $35\,\text{kg s}^{-2}\,\text{cm}^{-2}$ | |
| Stiffness const. | $C_0$ | $10^9\,\text{g s}^{-2}\,\text{cm}^{-2}$ | |
| Fluid density | $\rho$ | $1.0\,\text{g cm}^{-3}$ | |
| Attenuation factor | $\alpha$ | $3.0\,\text{cm}^{-1}$ | |
| Height of scala | $h$ | $0.1\,\text{cm}$ | |
| Effective mass | $m$ | $0.05\,\text{g cm}^{-2}$ | |
| Number of channels | $N_{\text{ch}}$ | 700 | |
| Input scaling factor | $c$ | $1.0\,\text{s cm}^{-1}$ | |
| Permeability offset | $A$ | 5 | |
| Permeability rate | $B$ | 300 | |
| Maximum permeability | $g$ | 1000 | |
| Replenishing rate | $y$ | 11.11 | |
| Loss rate | $l$ | 1250 | |
| Reuptake rate | $r$ | 16 667 | |
| Reprocessing rate | $n$ | 250 | |
| Propability scaling | $h$ | 50 000 | |
| Number of HCs at same position | $N_{\text{HC}}$ | 40 | |
| Synaptic time const. | $\tau_{\text{syn}}$ | $0.5\,\text{ms}^1$ | $10\,\text{ms}^2$ |
| Membrane time const. | $\tau_{\text{mem}}$ | $1\,\text{ms}^1$ | $20\,\text{ms}^2$ |
| Refractory time const. | $\tau_{\text{ref}}$ | $1\,\text{ms}^1$ | $0\,\text{ms}^2$ |
| Leak potential | $u_{\text{leak}}$ | 0 | |
| Reset potential | $u_{\text{reset}}$ | 0 | |
| Threshold potential | $u_{\text{thres}}$ | 1 | |
| Number of neurons per layer | $N$ | 128 | |
| Simulation step size | $\delta t$ | $0.5\,\text{ms}$ | |
| Simulation duration | $T$ | $1.0\,\text{s}$ | |
| Batch size | $N_{\text{batch}}$ | 256 | |
| Learning rate | $\eta$ | 0.001 | |
| Steepness of gradient | $\beta$ | 100 | |
| Regularization lower threshold | $\theta_l$ | 0.01 | |
| Regularization lower strength | $s_l$ | 1.0 | |
| Regularization upper threshold | $\theta_u$ | 100.0 | |
| Regularization upper strength | $s_u$ | 0.06 | |
| First moment estimates decay rate | $\beta_1$ | 0.9 | |
| Second moment estimates decay rate | $\beta_2$ | 0.999 | |

[1] BC parameter
[2] SNN parameter

In the HC model, one assumes that the cell contains a specific amount of free transmitter molecules $q(x, t)$ which could be released by use of a permeable membrane to the synaptic cleft (Fig. 10). The permeability is a function of the velocity of the BM, $v_y(x, t)$

$$k(x, t) = \begin{cases} \dfrac{g \cdot [c \cdot v_y(x, t) + A]}{c \cdot v_y(x, t) + A + B} & \text{for } v_y(x, t) + A > 0 \\ 0 & \text{else.} \end{cases} \quad (21)$$

The amount $c(x, t)$ of transmitter in the cleft is subject to chemical destruction or loss through diffusion $l \cdot c(x, t)$ as well as reuptake into the cell $r \cdot c(x, t)$

$$\frac{dc}{dt} = k(x, t)q(x, t) - l \cdot c(x, t) - r \cdot c(x, t). \quad (22)$$

A fraction $n \cdot w(x, t)$ of the reuptaken transmitter $w(x, t)$ is continuously transferred to the free transmitter pool

$$\frac{dw}{dt} = r \cdot c(x, t) - n \cdot w(x, t). \quad (23)$$

The transmitter originates in a manufacturing base that replenishes the free transmitter pool at a rate $y[1 - q(x, t)]$

$$\frac{dq}{dt} = y[1 - q(x, t)] + n \cdot w(x, t) - k(x, t)q(x, t). \quad (24)$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

12

IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS

While in the cleft, transmitter quanta have a finite probability $\mathcal{P}_{\text{spike}} = h \cdot c(x, t)\, dt$ of influencing the postsynaptic excitatory potential. A refractory period was imposed by denying any event which occurs within 1 ms of a previous event. At each position $x$ of the BM, we simulated $N_{\text{HC}} = 40$ independent HCs.

### C. Bushy Cell Model

The phase-locking of HC outputs was increased by feeding their spike output to a population of $N_{ch}$ BCs. In contrast to [80], we implemented the BCs as standard LIF neurons. In more detail, we considered a single layer ($l = 1$) of BCs without recurrent connections ($V_{ij}^{(l)} = 0 \; \forall i, j$). The feed-forward weights were set to $W_{ij}^{(l)} = 0.54/N_{\text{HC}} \, \forall i, j$. A single BC was used to integrate the spiketrains of the $N_{\text{HC}} = 40$ HCs for each channel of the BM.

## APPENDIX B
## NONSPIKING CLASSIFIERS

For validation purposes, we applied three standard non-spiking methods for time-series classification to the data sets, namely SVMs, LSTMs, and CNNs. We give details on each of them in the following.

### A. Support Vector Machines

We trained linear and nonlinear SVMs using *scikit-learn* [81]. Specifically, we trained SVMs with polynomial (up to third degree) and RBF kernels. The vectors in the input space were constructed such that for each sample a $N_{ch}$-dimensional vector $\mathbf{x}_i$ is generated by counting the number of spikes emitted by each BC in each sample. Furthermore, features were standardized by removing the mean and scaling to the unit variance.

### B. Long Short Term Memories

We used LSTMs for validation purposes of the temporal data [82]. The inputs to the LSTM consist of the $N_{ch}$ spike trains emitted by the BCs, but binned in time bins of size 10 ms. We trained LSTM networks using *Tensor-Flow 1.14.0* with the *Keras 2.3.0* application programming interface (API) [83], [84]. For all used layers, we stuck to the default parameters and initialization unless mentioned otherwise. Specifically, we considered a single LSTM layer with 128 cells with a dropout probability of 0.2 for the linear transformation of the input as well as for the linear transformation of the recurrent states. Last, a readout with softmax activation was applied. The model was trained with the Adamax optimizer [52] and a categorical cross entropy loss defined on the activation of the last time step and the time step with maximal activation.

### C. Convolutional Neural Networks

We applied CNNs to further test for separability of the data sets. To that end, the spike trains were not only binned in time, but also space. The temporal binwidth was set to 10 ms. Along the spatial dimension, the data were binned to result in 64 distinct input units. As for LSTMs, CNN networks were trained using *Tensorflow* with the *Keras* APIs with default parameters and initialization unless mentioned otherwise. First, a 2-D convolution layer with 32 filters of size $11 \times 11$ and rectified linear unit (ReLU) activation function was applied. Next, the output was processed by three successive blocks, each composed of two 2-D convolutional layers, each of them accompanied by batch normalization and ReLU activation. Both convolutional layers contain 32 filters of size $3 \times 3$. We finalized the blocks by a 2-D max-pooling layer with pool size $2 \times 2$ and a dropout layer with rate 0.2. The output of the last of the three blocks was processed by a dense layer with 128 nodes and ReLU activation function followed by a readout with softmax activation. The whole model is trained with the Adamax optimizer [52] and a categorical cross entropy loss for optimization.

## REFERENCES

[1] K. Boahen, "A neuromorph's prospectus," *Comput. Sci. Eng.*, vol. 19, no. 2, pp. 14–28, 2017.

[2] F. Zenke and S. Ganguli, "SuperSpike: Supervised learning in multilayer spiking neural networks," *Neural Comput.*, vol. 30, no. 6, pp. 1514–1541, Jun. 2018.

[3] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers Neurosci.*, vol. 12, p. 774, Oct. 2018.

[4] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Netw.*, vol. 111, pp. 47–63, Mar. 2019.

[5] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 787–797.

[6] S. B. Shrestha and G. Orchard, "SLAYER: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018, pp. 1419–1428.

[7] S. Woániak, A. Pantazi, T. Bohnstingl, and E. Eleftheriou, "Deep learning incorporating biologically inspired neural dynamics and in-memory computing," *Nature Mach. Intell.*, vol. 2, no. 6, pp. 325–336, Jun. 2020.

[8] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks," 2019, *arXiv:1901.09948*. [Online]. Available: http://arxiv.org/abs/1901.09948

[9] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 1947–1950.

[10] S. Friedmann, J. Schemmel, A. Grubl, A. Hartel, M. Hock, and K. Meier, "Demonstrating hybrid learning in a flexible neuromorphic hardware system," *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 1, pp. 128–142, Feb. 2017.

[11] S. B. Furber *et al.*, "Overview of the SpiNNaker system architecture," *IEEE Trans. Comput.*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013.

[12] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[13] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 106–122, Feb. 2018.

[14] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, Nov. 2019. [Online]. Available: https://www.nature.com/articles/s41586-019-1677-2

[15] M. Davies, "Benchmarks for progress in neuromorphic computing," *Nature Mach. Intell.*, vol. 1, no. 9, pp. 386–388, Sep. 2019.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] J. Zylberberg, J. T. Murphy, and M. R. DeWeese, "A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of v1 simple cell receptive fields," *PLoS Comput. Biol.*, vol. 7, no. 10, Oct. 2011, Art. no. e1002250.

[18] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Frontiers Neurosci.*, vol. 11, p. 324, Jun. 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2017.00324

[19] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Princeton, NJ, USA, Tech. Rep. 7, 2009.

[20] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. (2011). *Reading Digits in Natural Images with Unsupervised Feature Learning*. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[21] R. Gátig, "To spike, or when to spike?" *Current Opinion Neurobiol.*, vol. 25, pp. 134–139, Apr. 2014.

[22] S. M. Bohte, J. N. Kok, and H. La Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, nos. 1–4, pp. 17–37, Oct. 2002.

[23] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3227–3235, Jul. 2017.

[24] I. M. Comsa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala, "Temporal coding in spiking neural networks with alpha synaptic function: Learning with backpropagation," 2019, *arXiv:1907.13223*. [Online]. Available: http://arxiv.org/abs/1907.13223

[25] R.-M. Memmesheimer, R. Rubin, B. P. Ölveczky, and H. Sompolinsky, "Learning precisely timed spikes," *Neuron*, vol. 82, no. 4, pp. 925–938, May 2014.

[26] A. Gilra and W. Gerstner, "Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network," *eLife*, vol. 6, Nov. 2017, Art. no. e28295.

[27] W. Nicola and C. Clopath, "Supervised learning in spiking neural networks with FORCE training," *Nature Commun.*, vol. 8, no. 1, p. 2208, Dec. 2017.

[28] D. Thalmeier, M. Uhlmann, H. J. Kappen, and R.-M. Memmesheimer, "Learning universal computations with spikes," *PLOS Comput. Biol.*, vol. 12, no. 6, Jun. 2016, Art. no. e1004895.

[29] C. Eliasmith and C. H. Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. Cambridge, MA, USA: A Bradford Book, 2004.

[30] S. Denève and C. K. Machens, "Efficient codes and balanced networks," *Nature Neurosci.*, vol. 19, no. 3, pp. 375–382, Mar. 2016.

[31] L. F. Abbott, B. DePasquale, and R.-M. Memmesheimer, "Building functional networks of spiking model neurons," *Nature Neurosci.*, vol. 19, no. 3, pp. 350–355, Mar. 2016.

[32] D. Huh and T. J. Sejnowski, "Gradient descent for spiking neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1433–1443.

[33] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting," *Neural Comput.*, vol. 22, no. 2, pp. 467–510, Feb. 2010.

[34] J.-P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner, "Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning," *Neural Comput.*, vol. 18, no. 6, pp. 1318–1348, Jun. 2006.

[35] R. V. Florian, "The chronotron: A neuron that learns to fire temporally precise spike patterns," *PLoS ONE*, vol. 7, no. 8, Aug. 2012, Art. no. e40233.

[36] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, "Span: Spike pattern association neuron for learning spatio-temporal spike patterns," *Int. J. Neural Syst.*, vol. 22, no. 04, Aug. 2012, Art. no. 1250012.

[37] B. Gardner and A. Gráning, "Supervised learning in spiking neural networks for precise temporal encoding," *PLoS ONE*, vol. 11, no. 8, Aug. 2016, Art. no. e0161335.

[38] R. Gütig and H. Sompolinsky, "The tempotron: A neuron that learns spike timing-based decisions," *Nature Neurosci.*, vol. 9, no. 3, pp. 420–428, 2006.

[39] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15 $\mu$s latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[40] J. Anumula, D. Neil, T. Delbruck, and S.-C. Liu, "Feature representations for neuromorphic audio spike streams," *Frontiers Neurosci.*, vol. 12, p. 23, Feb. 2018.

[41] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers Neurosci.*, vol. 9, p. 437, Nov. 2015.

[42] A. Amir *et al.*, "A low power, fully event-based gesture recognition system," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7243–7252.

[43] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018, *arXiv:1804.03209*. [Online]. Available: http://arxiv.org/abs/1804.03209

[44] D. Schumann. *Dev-Core*. Accessed: Aug. 23, 2019. [Online]. Available: https://dev-core.org/

[45] P. Knysh and Y. Korkolis, "Blackbox: A procedure for parallel optimization of expensive black-box functions," 2016, *arXiv:1605.00998*. [Online]. Available: http://arxiv.org/abs/1605.00998

[46] X. Huang, A. Acero, H.-W. Hon, and R. Reddy, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[47] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke. (2019). *Heidelberg Spiking Datasets*. [Online]. Available: http://dx.doi.org/10.21227/51gn-m114

[48] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[50] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. NIPS*, 2017, p. 5.

[51] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *bioRxiv*, p. 2020.06.29.176925, Jun. 2020. [Online]. Available: https://www.biorxiv.org/content/10.1101/2020.06.29.176925v1

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[53] Y. Li, R. Kim, and T. J. Sejnowski, "Learning the synaptic and intrinsic membrane dynamics underlying working memory in spiking neural network models," *BioRxiv*, Jun. 2020, doi: 10.1101/2020.06.11.147405.

[54] R. G. Leonard and G. R. Doddington, "A speaker-independent connected-digit database," Tech. Rep., 1991, doi: 10.35111/72xz-6x59.

[55] J. Zohar, S. Cãar, F. Jason, P. Yuxin, N. Hereman, and T. Adhish. (Aug. 2018). *Jakobovski/Free-Spoken-Digit-Dataset: V1.0.8*. [Online]. Available: https://doi.org/10.5281/zenodo.1342401

[56] Mozilla. (Aug. 2019). *Mozilla Common Voice*. [Online]. Available: https://voice.mozilla.org/en

[57] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 5206–5210.

[58] A. Rousseau, P. Deléglise, and Y. Esteve, "Ted-LIUM: An automatic speech recognition dedicated corpus," in *Proc. LREC*, 2012, pp. 125–129.

[59] A. Köhn, F. Stegen, and T. Baumann, "Mining the spoken wikipedia for speech data and beyond," in *Proc. 10th Int. Conf. Lang. Resour. Eval.*, N. C. C. Chair, K. Choukri, T. Declerck, M. Grobelnik, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, Eds. Paris, France: European Language Resources Association, May 2016.

[60] N. Sieroka, H. G. Dosch, and A. Rupp, "Semirealistic models of the cochlea," *J. Acoust. Soc. Amer.*, vol. 120, no. 1, pp. 297–304, Jul. 2006.

[61] R. Meddis, "Simulation of auditory–neural transduction: Further studies," *J. Acoust. Soc. Amer.*, vol. 83, no. 3, pp. 1056–1063, Mar. 1988.

[62] S. M. Bohte, "Error-backpropagation in networks of fractionally predictive spiking neurons," in *Proc. ICANN* (Lecture Notes in Computer Science). Berlin, Germany: Springer, Jun. 2011, pp. 60–68. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-21735-7_8

[63] B. Yin, F. Corradi, and S. M. Bohté, "Effective and efficient computation with multiple-timescale spiking recurrent neural networks," 2020, *arXiv:2005.11633*. [Online]. Available: http://arxiv.org/abs/2005.11633

[64] D. Zambrano, R. Nusselder, H. Steven Scholte, and S. Bohte, "Efficient computation in adaptive artificial spiking neural networks," 2017, *arXiv:1710.04838*. [Online]. Available: http://arxiv.org/abs/1710.04838

[65] B. Ráckauer, N. Känzig, S.-C. Liu, T. Delbruck, and Y. Sandamirskaya, "Closing the accuracy gap in an event-based visual recognition task," 2019, *arXiv:1906.08859*. [Online]. Available: http://arxiv.org/abs/1906.08859

[66] C. Stöckl and W. Maass, "Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes," 2020, *arXiv:2002.00860*. [Online]. Available: http://arxiv.org/abs/2002.00860

[67] J. Göltz *et al.*, "Fast and deep: Energy-efficient neuromorphic learning with first-spike times," 2019, *arXiv:1912.11443*. [Online]. Available: http://arxiv.org/abs/1912.11443

[68] E. Hunsberger and C. Eliasmith, "Spiking deep networks with LIF neurons," 2015, *arXiv:1510.08829*. [Online]. Available: http://arxiv.org/abs/1510.08829

[69] C. Lee, S. Shakib Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," 2019, *arXiv:1903.06379*. [Online]. Available: http://arxiv.org/abs/1903.06379

[70] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*. [Online]. Available: http://arxiv.org/abs/1308.3432

[71] D. Jimenez Rezende and W. Gerstner, "Stochastic variational learning in recurrent spiking networks," *Frontiers Comput. Neurosci.*, vol. 8, p. 38, Apr. 2014.

[72] H. Jang, O. Simeone, B. Gardner, and A. Gruning, "An introduction to probabilistic spiking neural networks: Probabilistic models, learning rules, and applications," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 64–77, Nov. 2019.

[73] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (DECOLLE)," 2018, *arXiv:1811.10766*. [Online]. Available: http://arxiv.org/abs/1811.10766

[74] G. Bellec *et al.*, "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature Commun.*, vol. 11, no. 1, p. 3625, Jul. 2020. [Online]. Available: https://www.nature.com/articles/s41467-020-17236-y

[75] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Netw.*, vol. 99, pp. 56–67, Mar. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608017302903

[76] T. Zhang, Y. Zeng, D. Zhao, and M. Shi, "A plasticity-centric approach to train the non-differential spiking neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2018, p. 25.

[77] E. de Boer, "Auditory physics. Physical principles in hearing theory. I," *Phys. Rep.*, vol. 62, no. 2, pp. 87–174, Jun. 1980.

[78] E. de Boer, "Auditory physics. Physical principles in hearing theory. II," *Phys. Rep.*, vol. 105, no. 3, pp. 141–226, Mar. 1984.

[79] R. Meddis, "Simulation of mechanical to neural transduction in the auditory receptor," *J. Acoust. Soc. Amer.*, vol. 79, no. 3, pp. 702–711, Mar. 1986.

[80] J. S. Rothman, E. D. Young, and P. B. Manis, "Convergence of auditory nerve fibers onto bushy cells in the ventral cochlear nucleus: Implications of a computational model," *J. Neurophysiol.*, vol. 70, no. 6, pp. 2562–2583, Dec. 1993.

[81] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[82] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[83] M. Abadi. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/

[84] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

**Benjamin Cramer** received the M.Sc. degree in physics from Heidelberg University, Heidelberg, Germany, in 2016, where he is currently pursuing the Ph.D. degree with the Electronic Vision(s) Group.

His research focus is on learning in neuromorphic hardware.

**Yannik Stradmann** received the M.Sc. degree in physics from Heidelberg University, Heidelberg, Germany, in 2019.

His research within the Electronic Vision(s) Group focuses on the development and characterization of mixed-signal VLSI circuits for neuromorphic hardware.

**Johannes Schemmel** (Member, IEEE) received the Ph.D. degree in physics from Heidelberg University, Heidelberg, Germany, in 1999.

He is currently an Akademischer Oberrat with the Kirchhoff-Institute of Physics, Heidelberg, where he is the Head of the ASIC Laboratory and the Electronic Vision(s) Group. His research interests are mixed-mode VLSI systems for information processing, especially the analog implementation of biologically realistic neural network models. He is also the Architect of the Spikey and BrainScaleS accelerated Neuromorphic hardware systems.

**Friedemann Zenke** received the Ph.D. degree from the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, on the interaction of synaptic and homeostatic plasticity in spiking neural network models, in 2014.

He is currently a Junior Group Leader with the Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland. His research focuses on learning in biologically inspired neural network models with a focus on credit assignment and unsupervised learning.