Low-activity supervised convolutional spiking neural networks applied to speech commands recognition

Thomas Pellegrini, Romain Zimmer, Timothée Masquelier

LOW-ACTIVITY SUPERVISED CONVOLUTIONAL SPIKING NEURAL NETWORKS APPLIED TO SPEECH COMMANDS RECOGNITION

Thomas Pellegrini[1,*], *Romain Zimmer*[1,2], *Timothée Masquelier*[2]

[1]IRIT, Université de Toulouse, Toulouse, France
[2]CERCO UMR 5549, CNRS – Université Toulouse 3, Toulouse, France

ABSTRACT

Deep Neural Networks (DNNs) are the current state-of-the-art models in many speech related tasks. There is a growing interest, though, for more biologically realistic, hardware friendly and energy efficient models, named Spiking Neural Networks (SNNs). Recently, it has been shown that SNNs can be trained efficiently, in a supervised manner, using backpropagation with a surrogate gradient trick. In this work, we report speech command (SC) recognition experiments using supervised SNNs. We explored the Leaky-Integrate-Fire (LIF) neuron model for this task, and show that a model comprised of stacked dilated convolution spiking layers can reach an error rate very close to standard DNNs on the Google SC v1 dataset: 5.5%, while keeping a very sparse spiking activity, below 5%, thank to a new regularization term. We also show that modeling the leakage of the neuron membrane potential is useful, since the LIF model outperformed its non-leaky model counterpart significantly.

Index Terms— Spiking neural networks, surrogate gradient, speech command recognition

1. INTRODUCTION

Deep Neural Networks (DNNs) are the current state-of-the-art models in many speech related tasks. From a computational neuroscience perspective, DNNs can be seen as rate coding based models [1], in the sense that if a neuron is responsive to a given stimulus, then if we augment the stimulus intensity, the neuron output intensity will also increase. Temporal coding based models [2] try to also take into account information carried by the temporal structure of the stimulus. In the case of Spiking Neural Networks (SNNs), spike timing and delays between spikes is important in order to retrieve patterns in the spike sequences given as input to a model.

There is a growing interest for SNNs applied to speech recognition tasks, from isolated word and phone recognition [3, 4, 5, 6],to large-vocabulary automatic speech recognition (ASR) very recently [7]. Reasons are that the audio speech signal is particularly suited to event-driven models such as SNNs, SNNs are also more biologically realistic than DNNs, hardware friendly and energy efficient models, if implemented on dedicated energy-efficient neuromorphic chips. Furthermore, it has been shown recently that SNNs can be trained efficiently, in a supervised manner, using backpropagation with a surrogate gradient trick [8]. This new approach allows to train SNNs as one would do for DNNs.

In this work, we propose to use supervised SNNs for speech command (SC) recognition. We explore the Leaky Integrate-and-Fire (LIF) neuron model for this task, and show that convolutional SNNs can reach an accuracy very close to the one obtained with state-of-the-art DNNs, for this task. Our main contributions are the following: i) we propose to use dilated convolution spiking layers, ii) we define a new regularization term to penalize the averaged number of spikes to keep the spiking neuron activity as sparse as possible, iii) we show that the leaky variant of the neuron model outperforms the non-leaky one (NLIF), used in [7].

In order to facilitate reproducibility, our code using PyTorch is available online[1].

2. OVERVIEW OF THE LEAKY INTEGRATE-AND-FIRE MODEL (LIF)

The Leaky Integrate-and-Fire (LIF) model [9] is a phenomenological model of a biological spiking neuron, that describes how the neuron membrane potential U behaves through time, given an input current I comprised of spikes incoming from afferent neurons. Despite its simplicity, this model is very popular in com-

*This work was partially supported by the French ANR agency within the LUDAU project (ANR-18-CE23-0005-01) and the French "Investing for the Future — PIA3" AI Interdisciplinary Institute ANITI (Grant agreement ANR-19-PI3A-0004). We used HPC resources from CALMIP (Grant 2020-p20022).

[1]https://github.com/romainzimmer/s2net

putational neuroscience, for studies of neural coding, memory, and network dynamics [10]. From a machine learning point of view, it was shown to outperform more complex models at predicting real neuron activity [11, 12], and it is the basic brick of many recent Spiking Neural Networks (SNNs).

In Integrate-and-Fire (IF) models, a spike (output pulse) $\delta(t - t^f)$ — δ being the Dirac δ-function — is generated at the firing time t^f when the potential U crosses from below a threshold value b. The potential is then instantly reset to a new value U_{rest}. In the standard leaky variant, the sub-threshold dynamics of the membrane potential of the i^{th} neuron are described by the differential equation Eq. 1 [8]:

$$\tau_{\text{mem}} \frac{dU_i}{dt} = -(U_i(t) - U_{\text{rest}}) + RI_i(t) \tag{1}$$

where $U_i(t)$ is the membrane potential at time t, U_{rest} is the resting membrane potential, τ_{mem} is the membrane time constant, I_i is the current injected into the neuron and R is the membrane resistance. When U_i exceeds a threshold B_i, the neuron fires and U_i is decreased. The $-(U_i - U_{\text{rest}})$ term is the leak term that drives the potential towards U_{rest}.

If there is no leak, the model is called Non-Leaky Integrate and Fire (NLIF) and the corresponding differential equation is:

$$\tau_{\text{mem}} \frac{dU_i}{dt} = RI_i \tag{2}$$

Without loss of generality, we take $R = 1$ hereafter.

The input current can be defined as the projection of the input spikes along the preferred direction of neuron i given by W_i, the i^{th} row of W, the synaptic weight matrix:

$$I_i = \sum_j W_{ij} S_j^{\text{in}} \tag{3}$$

where $S_j(t) = \sum_k \delta(t - t_k^j)$ if neuron j fires at time $t = t_1^j, t_2^j, ...$ With this formulation, the potential will rise instantaneously when input spikes are received. Alternatively, the current can be governed by a leaky integration of these projections, similar to the membrane potential. For the sake of simplicity, we chose to use the instantaneous formulation for the input current.

The differential equations of LIF models can be approximated by linear recurrent equations in discrete time. Introducing a reset term $U_i^R[n]$ for the potential, the neuron dynamics can now be fully described by the following equations [13].

$$\begin{aligned} U_i^R[n\text{ - }1] &= b_i||W_i||^2 S_i^l[n\text{ - }1] \\ I_i[n] &= \sum_j W_{ij} S_j^{l-1}[n] \\ U_i[n] &= \beta(U_i[n\text{ - }1] - U_i^R[n\text{ - }1]) + I_i[n] \\ S_i^l[n] &= \Theta(U_i[n] - b_i||W_i||^2) \end{aligned} \tag{4}$$

where $\beta = \exp(-\frac{\Delta t}{\tau_{\text{mem}}})$, with Δt is a time step, Θ is the Heaviside step function and b_i is the threshold of neuron i.

With these equations, LIF neurons can be modeled as Recurrent Neural Network (RNN) cells whose state and output at time step n are given by $(U[n], I[n])$ and $S[n]$, respectively [8].

In practice, we used a trainable threshold parameter b_i, such that:

$$S_i[n] = \Theta(\frac{U_i[n]}{\|W_i\|^2 + \epsilon} - b_i) \tag{5}$$

with b_i initialized to 1 and $\epsilon = 10^{-8}$. We normalize the potential with $\|W_i\|^2$ to avoid squashing the value of the gradients during training. Note that we may choose to also optimize the leak parameters β, during training. Impact of learning those and the thresholds will be discussed in Section 10.

In [7], the authors based their SNNs on a non-leaky variant of IF models. In this case, solving Eq. 2 and with our notations, Eq. 4 boils down to:

$$U_i[n] = U_i[n\text{ - }1] - U_i^R[n] + I_i[n] \tag{6}$$

We will report results using this NLIF formulation in Section 10.

3. SURROGATE GRADIENT

In order to train SNNs in a supervised fashion, just as we do in standard deep learning using the back-propagation algorithm and stochastic gradient descent, we need to address the issue regarding the calculation of the gradient of the threshold function. Indeed, the gradient of the Heaviside function is zero everywhere and is not defined at zero. No gradient would be propagated unless we approximate it by a smooth function. That has been proposed in [8], in the name of a *surrogate gradient* function. In our work, we approximate the gradient of the Heaviside step function by the gradient of a sigmoid function, with a scale parameter $a \geq 0$ to control the quality of the approximation. The sigmoid function we use is:

$$\text{sig}_a(x) = 1/(1 + \exp(-ax)) \tag{7}$$

where a is the scale parameter, x a real-valued input. The larger a, the steeper the curve. This hyperparameter can be set empirically.

Hence, when using our SNN, on the forward pass the Heaviside function is used, whereas on the backward pass, the gradients are computed using the sigmoid gradient:

$$\Theta'(x) \approx \text{sig}'_a(x) = a \, \text{sig}_a(x) \, \text{sig}_a(-x) \tag{8}$$

4. SPIKING LAYERS

The proposed SNN is a feed-forward model with multiple spiking layers, and a decision output layer, sometimes called a readout layer. The input of a spiking layer is a spike sequence, also called a spike train, except for the first layer, for which the input is a multidimensional real-valued signal (log-FBANK coefficients). Each layer outputs a spike train except for the readout layer that outputs real values that can be seen as a linear combination of spikes, and on which predictions are made.

For fully-connected spiking layers, the input current at each time step would be a weighted sum of the input spikes emitted by the previous layer at the given moment (or a weighted sum of the input signal if it is the first layer). The state and the output of the cells are updated following the equations given in Section 2.

Just as in standard deep learning, we expect convolution filters to be pertinent for our task, given that our input are spectrograms. We thus defined and implemented convolutional spiking layers, as described in Algorithm 1. In the 2-d case (time$\times$frequency), the input current at each time step is given by a 2-d convolution between a kernel and the input spike train. The output spike train is computed through a for loop on the time steps, using the equations of the LIF model. As a side note, when processing spike trains as input, convolution in time can be seen as propagation delays of the input spikes. It is well known that such delays enrich the network's dynamics [14] and expressivity [15].

Algorithm 1 Spiking convolution layer algorithm

Inputs: input spike train S^{in}
Outputs: output spike train S^{out}
Initialize the potential U and S^{out} to zero tensors
Convolve S^{in} with kernels w: cS $\leftarrow$ conv2d(S^{in}, w)
for n=1,. . .,T **do**
    Compute $R[n]$
    Get input current at time step n: $I[n] \leftarrow$ cS$[n]$
    Compute $U[n]$
    Apply threshold function to compute $S^{\text{out}}[n]$
end for

We used the readout layer proposed in [8], with non-firing neurons (no reset). For classification tasks, the dimension of the output is equal to the number of labels. The label probabilities are given by a Softmax function applied to the maximum value over time of the membrane potential of each neuron.

In practice, we have found that using time-distributed fully connected layer and taking the mean activation of this layer over time as output makes training more stable. Thus, the output is the mean over time of a linear combination of input spikes.

5. REGULARIZING THE SPIKING ACTIVITY

A desirable property of SNNs would be energy efficiency, meaning that their neuron spiking activity should be as sparse as possible. We would like the number of emitted spikes by each spiking layer as small as possible, while still performing the requested task with high accuracy. This property is also desirable from a biological point of view, since biological neurons are very energy-efficient and emit limited amounts of spikes in a given amounts of time. If sparse, patterns of activity might also be more explainable.

In order to enforce sparse spiking activity, one could apply a ℓ_1 or ℓ_2 regularization on the total number of spikes emitted by each layer. However, due to the surrogate gradient, some neurons will be penalized even if they have not emitted any spike. As a regularization term, we propose to use the squared number of spikes at each layer l:

$$L_r(l) = \frac{1}{2KN} \sum_n \sum_k S_k^2[n]$$

where K is the number of neurons and N is the number of time steps. Using $S_k[n]^2$ instead of $S_k[n]$ is a simple way to ensure that the regularization will not be applied to neurons that have not emitted any spikes, i.e. for which $S_k[n] = 0$.

Indeed,

$$\frac{dS_k^2[n]}{dU_k[n]} = 2S_k[n]\text{sig}'_a(U_k[n])$$

which is zero when $S_k[n] = 0$. We will report the regularization impact in the experiment section.

6. GOOGLE SPEECH COMMANDS DATASET AND PREPROCESSING

The Google Speech Commands dataset [16] is a dataset of short audio recordings (at most 1 second, sampled at

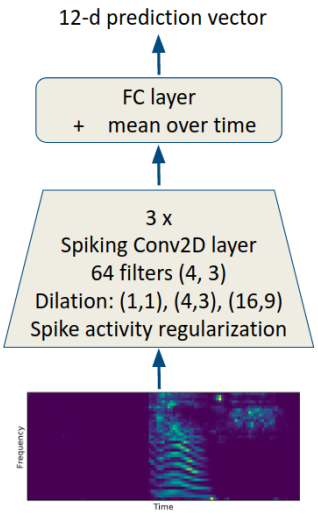


Fig. 1. Proposed SNN architecture.

16 kHz) of 30 different commands pronounced by different speakers for its first version and 35 for the second. All experiments were conducted on the first version of the dataset. The task considered is to discriminate among 12 classes: silence, unknown, "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go".

Commands that are not in the ten target classes are labeled as unknown and silence training data are extracted from the background noise files provided with the dataset. We used the standard validation and testing subsets provided with the corpus to evaluate the accuracy of our approach.

Forty log-Mel coefficients were extracted from raw signals with LibROSA [17], using a Mel scale defined between 20 Hz and 4000 Hz. We used a window size of 30 ms and a hop length of 10 ms, which also corresponds to the time step of the simulation Δt. The resulting input feature maps are of dimension 100×40 in channels, time and frequency. Finally, the spectrograms are re-scaled to ensure that the signal in each frequency band has a variance of 1 across time.

7. MODEL ARCHITECTURE

Fig. 1 shows the architecture of our proposed SNN for the speech commands recognition task.

We used three stacked convolution spiking layers and an output/readout layer, which is a time-distributed fully connected layer with twelve output neurons for the twelve classes to be predicted. Each convolution layer has one β learnable parameter controlling the time constant of the layer, C channels and one learnable threshold parameter b per channel.

Table 1 shows the details of our best model. Kernels are of size $H = 4$ along the time axis and $W = 3$ along the frequency axis. All convolutional layers have a stride of 1 and dilation factors of d_H and d_W along time and frequency axes respectively. The dilation coefficients were chosen so that the receptive fields r grow exponentially according the layer index: 4×3, 16×9 and 64×27 for the three convolution layers, respectively. With these settings, the total number of trainable parameters is 0.13M parameters.

The scale a of the surrogate gradient was empirically set to ten. Values below gave worse results and higher values did not bring improvements.

Conv. layer	C	H	W	d_H	d_W	r
1	64	4	3	1	1	4×3
2	64	4	3	4	3	16×9
3	64	4	3	16	9	64×27

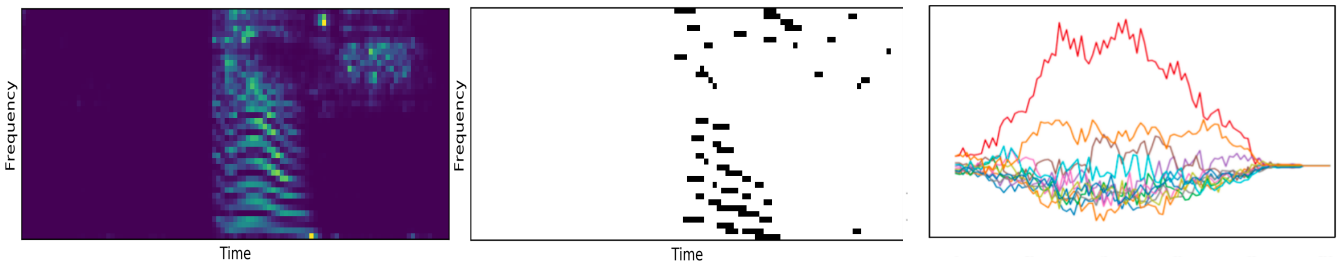
Table 1. Convolutional spiking layer characteristics. C: number of channels, H, W: kernel size in time and frequency, d_H, d_W: dilation coefficients in time and frequency, r: receptive fields.

8. TRAINING DETAILS

The model was trained using 128-samples minibatches, with the Rectified-Adam optimizer [18], with a learning rate of 10^{-3}, for 20 epochs with one epoch of warm-up, an exponential learning rate scheduler with a 0.85 coefficient, and a weight decay of 10^{-5}. We used a weighted random sampler to address the unbalanced number of samples of each class. The thresholds b (one per channel) and the β (one per layer) parameters were randomly initialized with normal distributions of respective means 1 and 0.7, and 0.01 standard deviations. Gradient values were clipped to $[-5, 5]$, β to $[0, 1]$ and b to $[0, +\infty[$. For each convolution layer $l = 1, ..., L$, the spiking activity regularization loss $L_r(l)$ was added with a 0.1 weighting coefficient. One training epoch took 15 minutes on a 16-GB Tesla V100.

9. RESULTS

Fig. 2 shows an example of a log-Mel spectrogram for the word "off" (a), the corresponding output after the first convolutional spiking layer (b), for one of its channels, and the network output predictions (c), after training. As can be seen, the first layer binarizes the input spectrogram ; each channel with a different threshold. In Fig. 2 (c), the twelve curves correspond to the twelve non-normalized score values at the different time steps (X-axis). As can be seen, one class gets high scores fast,



(a) Log-Mel spectrogram (b) Spike train, for one channel (first layer) (c) Model output

Fig. 2. Example for the spoken word "off". (a) Input features given to the SNN, (b) the corresponding spike train emitted by one channel of the first layer, (c) the model output, X-axis: time, Y-axis: predictions for the twelve classes.

at the start of the recording, indicating that the convolution layers integrated the relevant information.

Globally speaking, our SNN achieved 5.5% error rate (ER) on the standard test subset. In comparison, the best standard deep learning models reported in the literature achieved error rates from 3% to 6% [19, 16]. In [19], for instance, a ResNet-8 (110k trainable parameters, close to our model size of 131k params), and a ResNet-16 (248k parameters) models reached 5.9% and 4.2% ER on the same dataset. Regarding the *unknown* class, precision is lower for this class than for the other classes: 80% compared to 94% in average for all the classes. Recall is similar to other classes: about 94%. The *silence* class is perfectly recognized.

Our implementation keeps track of the number of spikes emitted by each of the convolution layers of the network. On the test subset, the averaged spiking rates are 2.6%, 4.9% and 6.1%, for the first, second and third convolution layers, respectively. Thus, the layer activation is around 5%, which is sparse as desired.

After training, we observe that the β values decrease in average with the convolution layer index: 0.60, 0.35, 0.23, for the three layers. The smaller the beta, the faster the leak and the more the neurons forget the past and reacts to the incoming spikes. The values of the thresholds b also follow this pattern in average: 0.9541, 0.9434, 0.9284. This is consistent with the previous observation that the deeper the layer is in the network, the lower the thresholds are and the more spikes are emitted.

10. ABLATION STUDIES

In order to estimate the impact of different components of our model, we conducted four complementary experiments: 1) when we use non-dilated convolution layers instead of dilated ones, 2) when we remove the spiking activity regularization, 3) when we simplify the LIF model to be the non-leaky Integrate-and-Fire model (NLIF) from [7], 4) when we freeze the leak and threshold parameters: β and b.

All the model variants used in this section are SNNs. We did another experiment where we removed the spiking components of the convolution and readout layers, thus turning the SNN into a standard convolutional neural network (CNN), comprised of dilated convolution layers and a time-distributed readout layer. This CNN performed much worse, with 52.8% ER obtained on the test set, showing that our proposed architecture has been optimized as an SNN, and other design choices should be made to build an efficient CNN.

Table 2 shows the results of these experiments, in terms of accuracy values obtained on the valid and test subsets, and the average number of spikes for the three convolution layers (# spikes 1, 2, 3). The first line of the table reports the numbers for our proposed SNN ("full model"), already discussed in the previous section.

10.1. Dilated convolutions: 1-a,b

As can be seen in Table 2, line 1-a, there is a significant increase in ER when using standard instead of dilated convolutions, from 5.5% to 9.1% ER on the test subset. When removing dilation, the receptive fields of the three layers are notably reduced, compared to the ones listed in 1. This explains the accuracy reduction, as confirmed by our second experiment, reported in line 1-b, in which we used the model with no dilation but with convolution kernels with the same receptive fields as in our proposed SNN. This larger model recovers the points lost by the 1-a model, with 5.2% and 6.4% ER on the validation and test sets, respectively. This model is, nevertheless, much larger than our SNN, with 4.22 Million parameters. Dilated convolutions are a way to achieve similar results but with much less parameters. Finally, as a side comment, it is interesting to see that the large model emit much less spikes: below 2% for the first two layers and 4.7% for the third one.

Table 2. Results and ablation studies. ER: error rate (%), # spikes i: averaged percentage of spikes emitted by the i^{th} convolution layer, during inference on the test subset. NLIF: Non-Leaky Integrate-and-Fire variant [7].

	Model size (Million params)	valid ER (%)	test ER (%)	#spikes 1 (%)	#spikes 2 (%)	#spikes 3 (%)
ResNet-8 [19]	0.11	_	5.9	_	_	_
Full SNN model (ours)	0.13	6.6	5.5	2.6	4.9	6.1
1-a) w/o dilation	0.13	15.8	9.1	4.1	3.7	6.0
1-b) w/o dilation, large kernels	4.22	5.2	6.4	1.7	1.3	4.7
2) w/o spik. regul.	0.13	5.8	5.7	12.5	12.5	6.4
3-a) NLIF, w/ spik. regul.	0.13	16.3	12.8	9.0	6.0	7.2
3-b) NLIF, w/o spik. regul.	0.13	18.6	12.1	11.9	7.2	8.3

10.2. Spiking activity regularization: 2

When we remove the penalization on the spiking activity, accuracy remains very close to the full model one (line 2). As expected, the averaged numbers of spikes are larger than for the full model: 12.5% for the first two layers, and 6.4% for the third layer. For this last layer, the difference is small (full model: 6.1%), indicating that the regularization weight, that we chose to be constant to 0.1 for the three layers, was too small for that layer. It could be useful, in future experiments, to use layer-dependent weights, although we do not expect accuracy improvements from tweaking those.

10.3. Non-leaky neuron model: 3-a,b

As explained in Section 2, the LIF model can be simplified by removing the potential leak term from the model. The resulting NLFI model has been used in [7] for acoustic modeling. In this case, the potential is governed by Eq. 6. We replaced Eq. 4 by Eq. 6 in the convolution layer definition of our model, without any other change. Lines 3-a and 3-b lead to the same conclusion, either with or without spike regularization. ER increased significantly compared to using the LIF model (full model): about 13-12% on the test set. It indicates that the β parameters play an important role.

10.4. Frozen leak coefficients β and thresholds b

We did three more experiments, freezing either the β values or the threshold b values, or both. We did not report the results in Table 2 since no significant difference has been found compared to the full model ones. For instance, keeping the initial random values for both β and b led to 5.7% ER. Similar figures were also found for the spiking activity, with a number of spikes that increase with the depth index of the convolution layers. This finding indicates that learning these model parameters is not needed, at least with the initial values we used for them.

11. CONCLUSIONS

In this work, we explored the LIF neuron model to define dilated convolution spiking layers for a spoken command recognition application. Contrarily to most works using SNNs applied to speech tasks, in which special mechanisms, usually non-trainable, are needed to first encode the speech input features into some type of neural encoding (spike trains) as a first step to then use SNNs (threshold encoding in [20], a specific layer in [7], etc.), our approach is unified in the sense that the first convolution layer applied to real-valued speech features is trainable and shares the same definition and implementation than the ones processing spike trains as input. Our proposed SNN, trained with back-propagation through time with surrogate gradient, achieved results competitive with standard deep convolutional neural networks.

We defined a regularization term to penalize the averaged number of spikes to keep the spiking neuron activity as sparse as possible, which is a desirable property both from a biological point of view and for a future potential implementation on low-energy dedicated chips.

Finally, we conducted ablation studies in order to estimate the impact of different components of our approach. In particular, an interesting result is that the LIF neuron model outperformed the simpler non-leaky one (NLFI), used in [7] for ASR.

In future work, we will try to confirm these results in acoustic modeling for speech recognition.

12. REFERENCES

[1] Edgar D Adrian and Yngve Zotterman, "The impulses produced by sensory nerve-endings: Part ii. the response of a single end-organ," *The Journal of physiology*, vol. 61, no. 2, pp. 151, 1926.

[2] Peter Dayan, Laurence F Abbott, et al., "Theoretical neuroscience: computational and mathematical modeling of neural systems," *Journal of Cognitive Neuroscience*, vol. 15, no. 1, pp. 154–155, 2003.

[3] Amirhossein Tavanaei and Anthony Maida, "Bio-inspired multi-layer spiking neural network extracts discriminative features from speech signals," in *International conference on neural information processing*. Springer, 2017, pp. 899–908.

[4] Jibin Wu, Yansong Chua, and Haizhou Li, "A biologically plausible speech recognition framework based on spiking neural networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[5] Yong Zhang, Peng Li, Yingyezhe Jin, and Yoonsuck Choe, "A digital liquid state machine with biologically inspired learning and its application to speech recognition," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 11, pp. 2635–2649, 2015.

[6] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Advances in Neural Information Processing Systems*, 2018, pp. 787–797.

[7] Jibin Wu, Emre Yilmaz, Malu Zhang, Haizhou Li, and Kay Chen Tan, "Deep spiking neural networks for large vocabulary automatic speech recognition," *Frontiers in Neuroscience*, vol. 14:199, March 2020.

[8] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[9] Richard B Stein, "Some models of neuronal variability," *Biophysical journal*, vol. 7, no. 1, pp. 37–68, 1967.

[10] Wulfram Gerstner and Werner M Kistler, *Spiking neuron models: Single neurons, populations, plasticity*, Cambridge university press, 2002.

[11] Ryota Kobayashi, Yasuhiro Tsubo, and Shigeru Shinomoto, "Made-to-order spiking neuron model equipped with a multi-timescale adaptive threshold.," *Frontiers in Computational Neuroscience*, vol. 3:9, July 2009.

[12] Wulfram Gerstner and Richard Naud, "How good are neuron models?," *Science*, vol. 326, pp. 379–380, 2009.

[13] Romain Zimmer, Thomas Pellegrini, Srisht Fateh Singh, and Timothée Masquelier, "Technical report: supervised training of convolutional spiking neural networks with PyTorch," *arXiv preprint arXiv:1911.10124*, 2019.

[14] Eugene M. Izhikevich, "Polychronization: computation with spikes," *Neural Computation*, vol. 18, no. 2, pp. 245–282, feb 2006.

[15] Wolfgang Maass and Michael Schmitt, "On the Complexity of Learning for Spiking Neurons with Temporal Coding," *Information and Computation*, vol. 153, no. 1, pp. 26–46, 1999.

[16] Pete Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *CoRR*, vol. abs/1804.03209, 2018.

[17] Brian McFee, Matt McVicar, Colin Raffel, Dawen Liang, Oriol Nieto, Josh Moore, Dan Ellis, Douglas Repetto, Petr Viktorin, João Felipe Santos, and Adrian Holovaty, "librosa: v0.4.0," June 2015.

[18] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han, "On the variance of the adaptive learning rate and beyond," *arXiv preprint arXiv:1908.03265*, 2019.

[19] Douglas Coimbra de Andrade, Sabato Leo, Martin Loesener Da Silva Viana, and Christoph Bernkopf, "A neural attention model for speech command recognition," *CoRR*, vol. abs/1808.08929, 2018.

[20] Malu Zhang, Jibin Wu, Yansong Chua, Xiaoling Luo, Zihan Pan, Dan Liu, and Haizhou Li, "MPD-AL: an efficient membrane potential driven aggregate-label learning algorithm for spiking neurons," in *Proc. AAAI*, 2019, vol. 33, pp. 1327–1334.