Interspeech 2022
18-22 September 2022, Incheon, Korea

Deep Residual Spiking Neural Network for Keyword Spotting in Low-Resource Settings

Qu Yang[1], *Qi Liu*[1,2], *Haizhou Li*[3,1,4]

[1]Department of Electrical and Computer Engineering, National University of Singapore
[2]School of Future Technology, South China University of Technology
[3]The Chinese University of Hong Kong, Shenzhen, China [4]Kriston AI, China

quyang@u.nus.edu.sg, drliuqi@scut.edu.cn, haizhouli@cuhk.edu.cn

Abstract

We propose a practical solution for the implementation of keyword spotting (KWS) system on portable devices, that features *all* three properties required for battery-powered portable scenarios: low power usage, small footprint, and high accuracy. In particular, we study an end-to-end KWS system with deep residual Spiking Neural Network (SNN), perform experiments on Google Speech Commands Dataset, and compare with both state-of-the-art ANN and SNN models. First, the proposed solution outperforms its ANN counterpart and other SNN in terms of energy efficiency. Second, it requires a smaller footprint (86.5K) than other ANN and SNN (210K) models. Third, in terms of classification accuracy, it outperforms the existing power-efficient SNN benchmark by 4% to 17%. The proposed solution is an example of the unparalleled performance of spiking neural network in real-world applications.

Index Terms: Keyword spotting system, Portable devices, Deep spiking neural networks, Power efficiency

1. Introduction

Portable devices, such as smart watches, smart speakers, smart phones, and autonomous cars have become the nexus of AI technologies. In particular, they are often constrained by the portable nature. Under these scenarios, AI applications must satisfy three requirements simultaneously: (1) low energy consumption; (2) small memory footprint; and (3) high accuracy.

In this paper, we propose an end-to-end keyword spotting (KWS) system with deep residual spiking neural network that is particularly suitable for portable scenarios and fulfills all three requirements. KWS is a low-cost solution for automatic speech recognition (ASR) system that only detects a relatively small set of predefined keywords. In contrast to ASR system which works continuously, the KWS system passively recognizes the predefined wakewords, such as "Hey Siri", "Hi, Alexa", and etc., to activate the smart device ready for subsequent tasks.

Recently, artificial neural networks (ANNs) for KWS have seen remarkable performance that represents the state-of-the-art [1, 2, 3, 4]. However, these implementations are often computationally expensive [1, 3]. To tackle this problem, abundant model compression techniques are implemented [5, 6, 7, 8]. The model size is reduced by 10x, but the number of parameters is still considerably large especially when compared to other low-energy solutions. In addition, these KWS systems typically utilize hand-crafted speech feature extractors (i.e., front-end), such as Mel Frequency Cepstral Coefficients (MFCC) and log-mel spectrum, that require Fourier transformations. These front-ends are computationally expensive and may not be feasible on resource-constrained devices. Therefore, we propose an end-to-end KWS system using time-domain 1D convolutions to replace the hand-craft front-end, where the raw audio waveform is directly fed into the model.

Spiking neural networks (SNNs) mimic the biological neurons that process information with action potential (i.e., spike) in massive parallel. Hence, on the basis of this asynchronous feature, SNNs possess higher computational potential in comparison with ANNs [9]. It has been proved that the accumulate (AC) operation is more energy-efficient and compact compared with Multiply-and-Accumulate (MAC) operation, that is, 14x lower energy and 21x fewer spatial area on the Global Foundry 28 nm chip [10]. In addition, because the SNN event-driven feature is particular suited for speech signal processing, there are plentiful success implementation of SNN-based ultra-lower power speech related tasks [11, 12, 13, 14, 15, 16]. Therefore, the benefits of SNN and its successful implementations in other tasks motivates us to build a KWS system based on SNN.

However, the SNN-based KWS systems from [17, 18] are limited on shallow networks (less than 4 layers), which limits their capability to achieve better performance. It remains a challenge to train large-scale SNN due to the discrete and non-differentiable nature of SNN. Recently, a layer-wise ANN-to-SNN conversion method namely, progressive tandem learning (PTL) algorithm [19], has been proposed to enable deeper SNN training with few accuracy loss. Motivated by this work, we develop an enhanced PTL version and train a deep residual SNN for KWS.

In the end-to-end KWS system, a speech encoder is designed using 1D convolutional layers [20] in place of a spectral analyzer for computational efficiency. As the information processing is completely spike-based in the entire model, the system is a fully event-driven solution. In addition, we enhance the layer-wise PTL algorithm with block-wise residual conversion. We seek to achieve power and footprint efficiency as well as accuracy through the model design. First, with the end-to-end design, our model achieves a fully convolutional architecture which is of smaller footprint than existing models with hand-crafted front-end and hidden fully connected layers. Second, the SNN implementation allows for improved power-efficiency over the ANN counterpart. Lastly, we improve the PTL algorithm so that we can scale up to a deeper SNN for higher KWS accuracy. To the best of our knowledge, our solution is the first research to implement deep residual SNN model for end-to-end KWS tasks and satisfies all three requirements for portable devices, and provides the highest accuracy conditional on lowest energy usage when compared to counterparts.

This paper is organized as follows. In Section 2, we formulate the SNN-based KWS system. In Section 3, we report the experiments. Finally Section 4 concludes the study.

10.21437/Interspeech.2022-107

2. KWS with Deep Residual SNN

In Fig.1, we show a system architecture that consists of a multi-layer 1D convolution front-end, which converts raw audio to spike trains, and a deep residual SNN (spiking ResNet8 in this work) for keyword recognition.

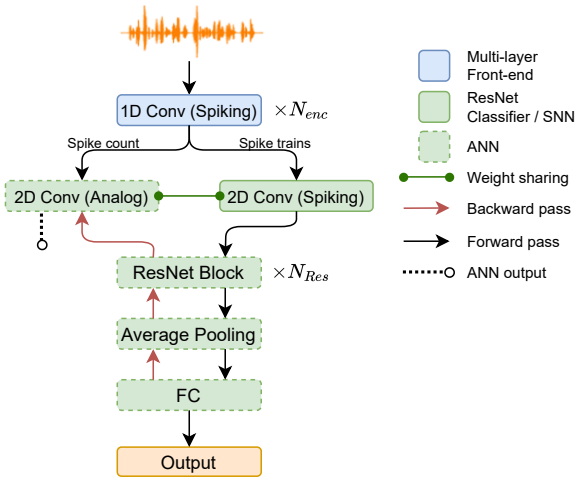


Figure 1: *The proposed end-to-end KWS system. It consists of N_{enc} = 2 1D convolutional layers for front-end and N_{Res} = 3 ResNet blocks in the classifier. The PTL algorithm converts ANN to SNN, both sharing the weights, layer-wisely (block-wisely for ResNet block).*

2.1. Feature Extraction using 1D Convolution

The design of feature extraction front-end, i.e. feature extractor, is motivated by SpEx [21], which applies a 1D convolution to encode the raw audio waveform into spectrum-like feature map. The 1D convolution convolutes over the time series. Its channels are treated as different filter banks in frequency-domain. This time-domain encoding is different from hand-craft methods in that the parameters are not pre-defined as sines or cosines, but fully data-driven. Moreover, to enhance the time-domain encoder with larger receptive field, inspired by wav2vec 2.0 [20], we develop a multi-layer convolutional feature extractor that consists of four blocks containing a time-domain 1D convolution followed by batch normalization and ReLU activation function.

To process information in SNNs, we need to convert the analog-valued raw audio into spike trains. The analog value is treated as the input current fed into the spiking neuron and added to its membrane potential. Then, the spike trains are generated by distributing the membrane potential over consecutive time steps according to the neuronal dynamics, and they start from the output of the first SNN layer which performs neural encoding.

2.2. Network Architecture

We employ a deep network structure, ResNet, and study a learning algorithm for deep SNN. The residual block is the basic component of ResNet. Fig. 2(a) illustrates the l-th block of a typical spiking ResNet (sResNet) [22, 23, 24]. $o^{l-1}[t]$ and $o^{l}[t]$ are the input and output at time step t. In Fig. 2(a), f_n denotes a block that contains a convolutional layer and batch normalized layer. Θ is the step function of spiking neuron as described in Eq. (3). $U_n^l[t]$ denotes the membrane potential of f_n and $o_n^l[t]$ denotes the spike generated by f_n. We have the input-output mapping of this sResNet block as,

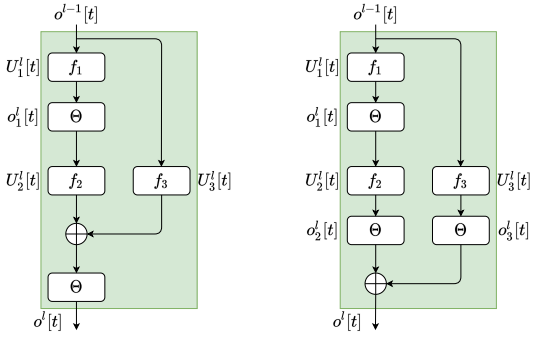
$$o^l[t] = \Theta(U_2[t] + U_3[t] - \vartheta). \tag{1}$$

When both $U_2[t]$ and $U_3[t]$ are below the threshold ϑ, Θ is not supposed to generate an output spike. However, if the sum of $U_2[t]$ and $U_3[t]$ is greater than the threshold, the residual block will generate a spike at time step t, which is undesired. We propose the sResNet block B, as shown in Fig.2(b) to mitigate this issue, where the input-output mapping of this sResNet block can be formulated as,

$$o^l[t] = \Theta(U_2[t] - \vartheta) + \Theta(U_3[t] - \vartheta). \tag{2}$$

Hence, only when both $U_2[t]$ and $U_3[t]$ exceed the threshold, the residual block will then output a spike.

As Eq. (1) accumulates the membrane potential first before generating the spikes while Eq. (2) directly accumulates the spikes. We expect that Eq. (2) is more accurate than Eq. (1) when inducing activation values. We conduct experiments on both A and B blocks, and find that block B has lower conversion accuracy loss than A (1.97% vs. 3.08%). Therefore, we adopt sResNet block B for all the experiments next.



(a) sResNet Block A (b) sResNet Block B

Figure 2: *The residual blocks in a spiking ResNet (sResNet).*

2.3. Bridging between Analog and Spiking Neurons

We use the Non-Leaky Integrate and Fire (NLIF) model with reset scheme [10] which is a widely used spiking neuron model [18, 17, 25]. At each time step t, the state of neuron j can be represented by its membrane potential $U_j[t]$. Fig. 3(a) illustrates the computational process of spiking neuron, which accumulates the incoming spike trains and generates the outgoing spikes once the membrane potential exceeds the firing threshold ϑ, and the resulting outgoing spike trains are denoted by $o_j[t] \in \{0, 1\}$ which is described by the following process:

$$o_j^l[t] = \Theta\left(U_j^l[t] - \vartheta\right) \text{ with } \Theta(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Since the feature representation is encoded and represented by the total spike count, we assume that the spike trains are evenly distributed over time; therefore, the sub-threshold membrane potential of neuron j in layer l across the simulation time window T is:

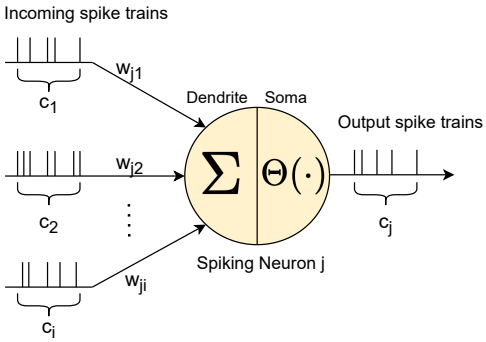
$$U_j^l = \sum_i w_{ji}^{l-1} c_i^{l-1} + b_j^l T, \;\; c_i^{l-1} = \sum_t^T o_i^{l-1}[t]. \tag{4}$$

where w_{ji}^{l-1} represents the synaptic weight between neuron i and j, b_i^l denotes the constant injecting current to neuron j in layer l, and c_i^{l-1} denotes the total count of incoming spike trains from pre-synaptic neuron i at layer $l-1$.

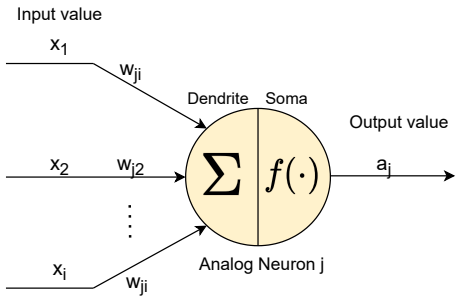
Fig. 3(b) depicts the the computational model of analog neuron, which the process is described by

$$a_j^l = f(\sum_i w_{ji}^{l-1} x_i^{l-1} + b_j^l), \tag{5}$$

where a_j^l and x_i^{l-1} are the output and input of the analog neuron respectively; w_{ji}^{l-1} and b_j^l denotes the weight and the bias. $f(\cdot)$ is the activation function. Note that by treating $b_j^l T$ as the bias term and c_i^{l-1} as input, Eq. (4) is exactly the same as the pre-activation of analog neurons in ANN. With this link, the SNN layers are capable to couple with ANN layers for parameter sharing.



(a) Spiking Neuron



(b) Analog Neuron

Figure 3: *Neuron model of SNN and ANN*

2.4. Progressive Tandem Learning for Deep SNN

Our purpose is to build a deep SNN to spot multiple keywords, which is a multi-class classification task. Training a deep SNN remains a challenge. We pre-train an ANN model that has the same structure as the SNN model, but with different neuronal functions. The ANN model consists of two parts: (1) the feature extractor described in Section 2.1; and (2) the classifier using the famous eight-layer residual network (i.e., ResNet8).

Empirical evidences have shown that an ANN of over 10 layers may converge easily, while an SNN of the same depth finds it difficult to converge [19]. This is because gradient approximation error is prone to accumulate over layers in deep SNN training algorithms, which adversely affects the learning process.

To address this issue, we adopt the progressive tandem learning (PTL) algorithm [19] for an SNN with the NLIF neurons introduced in Section 2.3. The PTL algorithm allows the SNN model to benefit from the ANN model in terms of effective back-propagation and high modeling accuracy. It performs fine-tuning layer by layer, which can effectively overcome the accumulated gradient approximation errors and scale-up freely to deep SNNs. As illustrated in Fig.1, this algorithm consists of both SNN and ANN coupled through weight sharing. The SNN layers are used to propagate the exact representation in spike trains and spike count to subsequent SNN and ANN layers. The synchronized ANN layers work as an auxiliary structure to facilitate the error back-propagation path during training.

In this layer-wise conversion method, it takes L stages to finish the conversion and the fine-tuning process for ANN with L layers. In each stage, the conversion process terminates as soon as the accuracy of hybrid model is not improved after a pre-defined patience period T_p, and enters the next stage for subsequent layer training.

However, such layer-wise conversion method is incompatible with the residual block that involves the shortcut connection. We extend the PTL algorithm with additional block-wise conversion function especially for the residual block, in which the conversion and the fine-tuning processes are performed on the whole sResNet block described in Section 2.2.

3. Experiment

3.1. Dataset

We use the Google Speech Commands (GSC) dataset [26] for experiments. The dataset is split into training, validation, and test sets at a ratio of 8:1:1 following [26]. The GSC dataset includes 30 short commands for version 1 (V1) and 35 for version 2 (V2) by 1,881 and 2,618 speakers respectively. However, for ease of comparison, we conduct experiments on the same task as that in the prior work on SNN, which only recognizes 12 classes, that include 10 commands, namely, “yes”, “no”, “up”, “down”, “left”, “right”, “on”, “off”, “stop” and “go”, and two additional classes, namely silence, and an unknown class. The unknown class covers the remaining of 20 (25) speech commands in the set of 30 (35). The number of training samples varies from 1,300 to 1,800 per command.

We add 10% of the silence samples to each of training, validation, and test sets. The silence class is artificially generated by extracting one-second clips from background noise files. With V1, this results in 56,196 training, 7,477 validation, and 7,518 test utterances, while with V2, it results in 93,327 training, 10,979 validation, and 12,105 test utterances.

During the training, three data prepossessing approaches are applied with probability 0.5 to obtain better performance. The first approach is to randomly scale amplitude of audio within the range of [0.7, 1.1]; the second approach is to randomly change the speed of audio within the range of [0.8, 1.2]; and the third approach is to randomly shift the timing of audio within the range of [-0.2, 0.2] seconds. We mainly focus on V2 but also report results on V1, because many previous published KWS works reported the results only on the first version.

3.2. Experimental Setup

The time-domain feature extractor contains two blocks and the 1D convolutions in each block using channels (128, 32) with strides (20, 2), kernel (40, 4) and dilation size (1, 2).

During the ANN pre-training, following the ResNet formu-

lation [27], we utilize stochastic gradient descent with momentum of 0.9 as the optimizer and cross-entropy loss function. We set the initial learning rate to 1e-2 which reduces by 10x every 40 epochs, the mini-batch size as 32, and the weight decay as 1e-5. During the ANN to SNN conversion, the patience period T_p is set to 8 based on the total training epoch and network depth. The encoding time window is set as 32 to reduce the gap between ANN and SNN representations. All models are implemented with PyTorch and accuracy is reported for the whole test set.

3.3. Accuracy and Model Size

We report the result in terms of test accuracy and parameters (i.e., model size), and compare with the SOTA KWS models in Table 1. It is worth noting that the MFCC and log-Mel coefficient front-end need to compute fast Fourier transformation (FFT), in which the complexity is $\mathcal{O}(N \log N)$ where N is commonly used as 256 or 512 in KWS tasks. Therefore, we approximately estimate the parameters of the MFCC and log-Mel coefficient front-end as 100k based on the Perf calculation, which is a command-line tool for Linux kernel performance monitoring. Since the parameters of SOTA models exclude the computational expensive hand-crafted front-end while the parameters of our model include the 1D convolutions front-end, we add the estimated front-end parameters to SOTA models for a fair comparison. It is evident that our system requires far less parameters.

For accuracy, our model achieves 92.2% and 92.9% on the GSC dataset V1 and V2. Our model outperforms the SNN-based models [17, 18], and is closer to the accuracy of ANN-based models [28, 29, 30]. This result shows that the deep SNN has a higher capacity than a shallow SNN, thus better performance.

Table 1: *A summary of KWS accuracy and model size. All SNN-based models are based on Non-Leaky Integrate-and-Fire (NLIF) neurons. The numbers in parentheses of the last column, if any, represent the estimated number of parameters accounting for feature extraction front-end.*

Model	**Network**	**Acc. (%)**	**Params. (est.) (K)**
Google Speech Commands Dataset Version 1			
Attention RNN [29]	ANN	95.6	202 (302)
Res8 [28]	ANN	94.1	110 (210)
ConvNet on raw WAV [30]	ANN	89.4	700
NLIF full SNN [18]	SNN	87.9	120 (230)
E2E residual SNN (ours)	SNN	**92.2**	**86.5**
Google Speech Commands Dataset Version 2			
Attention RNN [29]	ANN	96.9	220 (320)
Rate-based SNN [17]	SNN	75.2	117 (217)
E2E residual SNN (ours)	SNN	**92.9**	**86.5**

3.4. Energy Efficiency

In terms of energy efficiency, we calculate the total synaptic operations SynOps for our model following the norm of the Neuromorphic Computing community [31, 10, 32]. For ANN model, the total SynOps (i.e., Multiply-and-Accumulate (MAC)) required is defined as below:

$$SynOps(ANN) = \sum_{l}^{L} f_{in}^{l} N_l, \tag{6}$$

where the f_{in}^{l} is the fan-in connections to the neuron in layer l, N_l is the number of neurons in layer l, and L is the total layers of network. For SNN model, the total SynOps (i.e., Accumulate (AC)) required is defined as below:

$$SynOps(SNN) = \sum_{t}^{T} \sum_{l}^{L-1} \sum_{j}^{N_l} f_{out,j}^{l} o_j^l[t], \tag{7}$$

where the $f_{out,j}^{l}$ is the fan-out connections from neuron j in layer l to neurons in the next layer, N_s is the simulation time window, and $o_j^l[t]$ is the occurrence of spike from neuron j.

To compare energy efficiency, we report the ratio of average synaptic operations of SNN to ANN. The ratio is 1.46 when computed using a single mini-batch of data during inference. According to a recent study [10], the AC operations in SNN is 14x cheaper and 21x more compact compared with MAC operation in ANN when implemented on the Global Foundry 28nm chip. Hence, our SNN model achieves more than 9 times energy saving over the ANN counterpart, and the saving can be further boosted on neuromorphic hardware.

4. Conclusions

We present an end-to-end KWS system for low-resource settings based on a deep residual SNN model. By utilizing the 1D convolutions to replace the computationally expensive hand-craft front-end (e.g., MFCC), it enables the end-to-end KWS solution for resource-constrained devices. In addition, this model adopts the enhanced version of PTL algorithm and achieves efficient deep residual SNN training. The experiment results based on the GSC dataset show that our model is small-footprint, outperforms the SNN-based systems in terms of accuracy, and is closer to ANN-based accuracy. The average synaptic operation ratio of SNN to ANN suggests that our SNN implementation is more energy-efficient when compared to ANN counterparts. We also plan to implement our solution on neuromorphic hardware.

5. Acknowledgements

This research project is supported by IAF, A*STAR, SOITEC, NXP and National University of Singapore under FD-fAbrICS: Joint Lab for FD-SOI Always-on Intelligent & Connected Systems (Award I2001E0053). This research is also supported by the internal project of the Guangdong Provincial Key Laboratory of Big Data Computing under the Grant No. B10120210117-KP02, The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen)

6. References

[1] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4087–4091.

[2] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, and S. Vitaladevuni, "Multi-task learning and weighted cross-entropy for dnn-based keyword spotting." in *Interspeech*, vol. 9, 2016, pp. 760–764.

[3] C. Lengerich and A. Hannun, "An end-to-end architecture for keyword spotting and voice activity detection," *arXiv preprint arXiv:1611.09405*, 2016.

[4] S. Sigtia, R. Haynes, H. Richards, E. Marchi, and J. Bridle, "Efficient voice trigger detection for low resource hardware." in *Interspeech*, 2018, pp. 2092–2096.

[5] Y. Zhang, P. Li, Y. Jin, and Y. 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] S. Myer and V. S. Tomar, "Efficient keyword spotting using time delay neural networks," *arXiv preprint arXiv:1807.04353*, 2018.

[7] J. Fernandez-Marques, V. W.-S. Tseng, S. Bhattachara, and N. D. Lane, "On-the-fly deterministic binary filters for memory efficient keyword spotting applications on embedded devices," in *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning*, 2018, pp. 13–18.

[8] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6351–6355.

[9] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[10] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 682, 2017.

[11] A. Tavanaei and A. 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.

[12] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," *arXiv preprint arXiv:1803.09574*, 2018.

[13] J. Wu, Y. Chua, and H. 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.

[14] M. Zhang, J. Wu, Y. Chua, X. Luo, Z. Pan, D. Liu, and H. Li, "Mpd-al: an efficient membrane potential driven aggregate-label learning algorithm for spiking neurons," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 1327–1334.

[15] Z. Pan, M. Zhang, J. Wu, and H. Li, "Multi-tones' phase coding (mtpc) of interaural time difference by spiking neural network," *arXiv preprint arXiv:2007.03274*, 2020.

[16] J. Wu, E. Yılmaz, M. Zhang, H. Li, and K. C. Tan, "Deep spiking neural networks for large vocabulary automatic speech recognition," *Frontiers in neuroscience*, vol. 14, p. 199, 2020.

[17] E. Yilmaz, Ö. B. Gevrek, J. Wu, Y. Chen, X. Meng, and H. Li, "Deep convolutional spiking neural networks for keyword spotting," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2020-Octob, 2020, pp. 2557–2561.

[18] T. Pellegrini, R. Zimmer, and T. Masquelier, "Low-activity supervised convolutional spiking neural networks applied to speech commands recognition," nov 2020.

[19] J. Wu, C. Xu, D. Zhou, H. Li, and K. C. Tan, "Progressive tandem learning for pattern recognition with deep spiking neural networks," *arXiv preprint arXiv:2007.01204*, 2020.

[20] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.

[21] C. Xu, W. Rao, E. S. Chng, and H. Li, "Spex: Multi-scale time domain speaker extraction network," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 28, pp. 1370–1384, 2020.

[22] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going deeper with directly-trained larger spiking neural networks," *arXiv preprint arXiv:2011.05280*, 2020.

[23] Y. Hu, H. Tang, and G. Pan, "Spiking deep residual network," *arXiv preprint arXiv:1805.01352*, 2018.

[24] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in neuroscience*, 2020.

[25] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, "A tandem learning rule for effective training and rapid inference of deep spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[26] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5484–5488.

[29] D. C. de Andrade, S. Leo, M. L. D. S. Viana, and C. Bernkopf, "A neural attention model for speech command recognition," *arXiv preprint arXiv:1808.08929*, 2018.

[30] P. Jansson, "Single-word speech recognition with convolutional neural networks on raw waveforms," 2018.

[31] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[32] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, p. 95, 2019.