Convolutional Neural Network with Spikes for Voice Keyword Classification at the Edge

Abstract

Convolutional neural networks (CNNs) have shown to be useful for image and even audio classification. However, deep CNNs can be computationally heavy and unsuitable for edge intelligence where embedded devices are constrained by memory and energy requirements. Spiking neural networks (SNNs) offer potential as energy-efficient networks but typically underperform typical deep neural networks. This paper proposes the combination of CNN and SNN to form a spiking convolutional neural network (SCNN) which exhibited excellent accuracy on a multi-class audio classification task. We further investigate the computational cost of the SCNN on general-purpose hardware and search for shallower networks for improved computational cost. This paper further illustrates the potential of threshold voltage in SCNN as a tuning hyperparameter for sparsity and accuracy optimization.

Introduction

By mimicking the sparse and event-driven computations occurring in the biological brain through spiking neural networks (SNNs), researchers have been trying to achieve neural networks which are as efficient (i.e., energy and memory efficiency) and as effective (i.e., accuracy, ability to learn, and adaptability for different tasks) as the biological brain [1]. While the sparse information encoding of SNNs helps with energy efficiency for edge implementations, the information loss results in SNNs underperforming as compared to conventional neural networks generally [1], [2].

In the space of audio classification tasks, SNN solutions have been explored in recent years in various forms – pure SNNs in the form of fully-connected layers [3], SNNs with recurrent connections [4], SNN with convolutional layers [5], SNNs with self-organizing maps [6] and even using single SNN layer for feature extraction [7]. While convolutional neural networks (CNNs) excel in audio classification using both the temporal and spectral features of audio [8]–[10], deep networks are computationally expansive in terms of both energy and memory. Thus, it is of interest to explore the combination of SNN and CNN to achieve a balance of performance and energy efficiency.

The work in [5] indicated the potential of the use of a spiking convolutional neural network (SCNN) for real-time speech command detection. As the task involved in [5] is a binary classification ("Yes" and "No" speech commands) requiring a specialized Neuromorphic Auditory Sensor (NAS), more research is required to evaluate the performance of SCNN for multiclass classification for a more readily available sensors. On the other hand, [11] explores SCNN with deep neural networks (DNNs) with a more mature feature extraction technique – Mel-cepstral Coefficients (MFCCs), which allow for integration with more readily-available sensors, especially for edge detection [12]. While tandem learning has shown great promise in achieving great performance using SNN solutions [13], [14], the overall training method is non-trivial, and not all layers can exploit the sparsity property of SNN. This paper aims to achieve the following:

1. Proposes an SCNN architecture for multiclass classification for voice detection at the edge using backpropagation techniques.
2. Examine the impact of encoding from the first spiking layer on optimization for sparsity and accuracy.

Methods

A. Audio-processing

Going beyond the binary classification problem, the free-spoken digit dataset (FSDD) was used where an audio recording of spoken digits ‘0’ to ‘9’ was sampled at 8 kHz [15]. A study of the overall dataset indicated that 90 % of the audio falls within 0.8 s of the audio file. However, trimming based on a fixed length was not done at this stage of the processing to preserve as much information and reduce computation. For each audio recording, the period(s) where there is silence was trimmed based on a cut-off threshold of 10 dB. 16 MFCCs were then extracted from each audio at a hop length of 512. MFCCs were chosen as it has been highly effective for speech classification tasks as it provides frequency-domain information at bands like the response of biological auditory systems [16], [17]. As the computation of the MFCC is dependent on the parameters involved as seen in Figure 1, the conservative set of configurations was chosen to enable sufficient features to be extracted without increasing computation excessively.

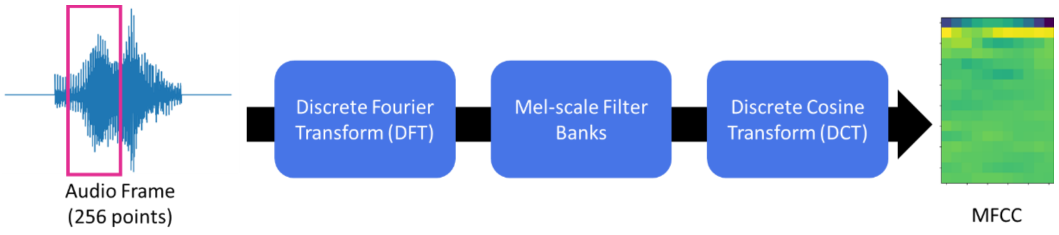
Audio Frame
(256 points)
Discrete Fourier Transform (DFT)
Mel-scale Filter Banks
Discrete Cosine Transform (DCT)
MFCC

Figure 1 outlines the computation steps for MFCC extraction.

B. Network Design

i. CNN

The design space for neural networks like CNN is large, with proposed deep networks like LeNet [18], AlexNet [19], and VGGNet [20] performing exceptionally well for images and [8]–[10] performing well for audio. However deep networks result in a high number of parameters and computations which may not be easily implemented on hardware. As such, the CNN proposed by one of the contributors to the FSDD was used as a reference [21] due to its effectiveness in the dataset and relatively shallow architecture as compared to the architectures listed above. The network was modified further to obtain the architecture shown in Figure 2A. To ensure that the MFCCs obtained from the audio-processing fit the input size of the neural network, zero-padding was used to pad in both the horizontal and vertical dimensions as padding has shown to have minimal to no impact on the performance of CNNs [22]. It should be noted that this size requirement exceeds most of the MFCCs generated from FSDD and should a dataset with larger MFCCs be used, trimming will be required unless the input size of the network is adjusted.

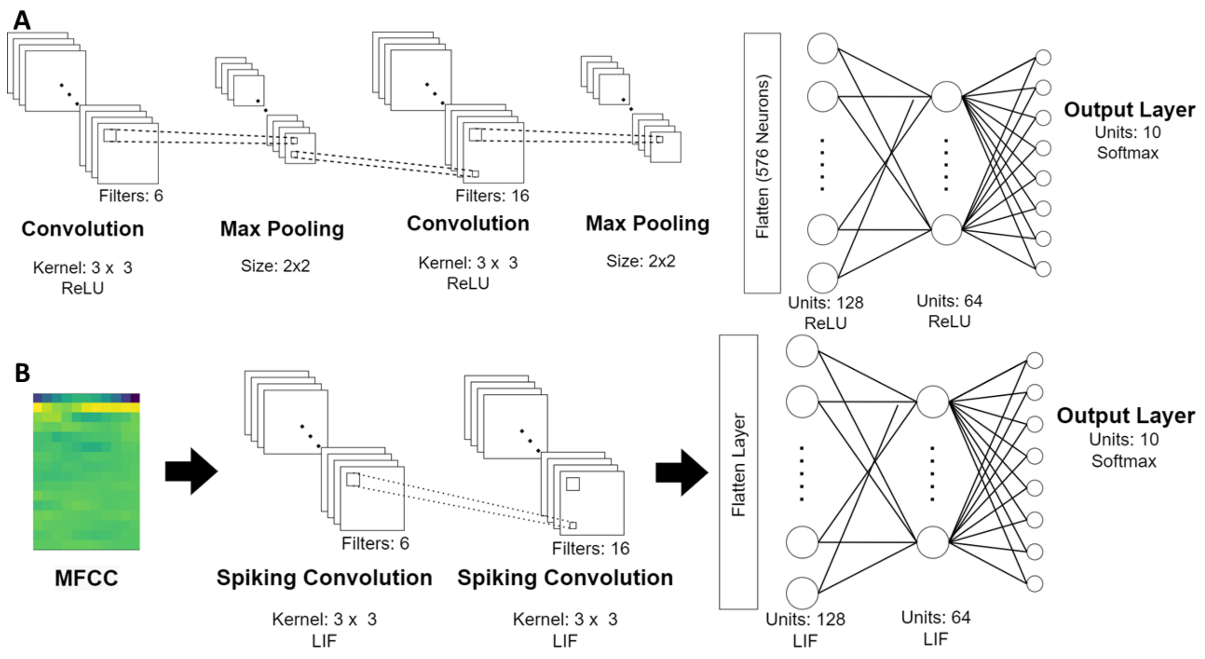

Figure 2 A (top) shows the CNN architecture as a base reference while B (bottom) shows the SCNN architecture proposed.

ii. SCNN

Using the open-source PyTorch package snnTorch [23], the CNN was converted to an SCNN through the use of spiking activations, with the architecture shown in Figure 2B. The use of snnTorch allowed for powerful optimizations and configurations available in PyTorch [24]. Of the different neurons available, the leaky integrate-and-fire (LIF) neuron was chosen in consideration of the effectiveness in many SNN solutions [2], [25] and the simplicity in computation at the expense of its closeness to the biological neuron, especially as compared to other neuron models like the Hodgkin-Huxley neuron [26], and the Izhikevich neuron [27]. Each LIF neuron consists of several parameters to adjust the spiking dynamics and mechanisms, including the resting membrane voltage, decay constant, threshold voltage, and time steps.

Traditionally, SNNs face challenges in optimization due to the non-continuous and non-differentiability characteristics of the spiking neurons, meaning that traditionally effective gradient descent and backpropagation methods cannot be used trivially to train the model. While alternative such as spike-time-dependent plasticity (STDP) has been proposed, the implementation is challenging and the performance of the network in most cases fail to compare to their DNN-counterpart [1], [2], [28]. SnnTorch enables the use of surrogate gradient to be implemented in the neuron, making backpropagation possible for training the model [29], [30]. The default neuron using the sigmoid function as the surrogate gradient function was implemented with a slope of 25 for a total of 10 timesteps at a threshold of 1.0.

C. Experiment

In this study, it was hypothesized that a spiking neural network version of the CNN can be implemented for voice detection with minimal accuracy degradation while having a more efficient implementation. Instead of using spiking datasets, the first convolutional spiking layer serves as a form of an encoder to the original data input where the output from the layer is in the form of spikes. The first spiking convolutional layer was hypothesized to be the most important in the overall architecture, where further analysis can be done to optimize the overall network while focusing on the first layer.

The different SCNN architectures were first used to investigate the performance of the networks consisting of accuracy, and computational resources. While different hardware will provide varying performance, this study uses the number of parameters in the network and the number of floating-point operations per second (FLOPS) as approximates for computational resources, both in terms of memory (parameters) and energy (FLOPS). While there are many options to profile the number of FLOPS, this study uses fvcore implementation from the Meta Artificial Intelligence Research Team [31].

The SCNN was first trained using a train-test split of 80-20 for 20 epochs before testing the networks to obtain accuracy. The optimizer used is the Adam optimizer [32] with the loss function being the cross entropy loss. A comparison was done between the SCNN and the CNN trained using the same approach. With accuracy being the optimizing metric and the computational resources as the satisficing metric, the SCNN was modified further. Through the removal of layers and studying the impact of the modifications, the importance and impact of the layers on the accuracy and computation resources can be observed.

Results and Discussion

i. Comparison between SCNN and CNN

The CNN used performed well as expected and remains resilient to architectural changes, such as the removal of one of the convolutional layers and one of the fully connected layers. With random data partitioning implemented in the train-test split, average accuracy was obtained from running three runs as a representation of the performance. On the other hand, the number of parameters and number of FLOPS remain the same for each architecture and are used without the need for averaging.

Table 1 compares the accuracy of CNN and SCNN.

Metrics	CNN	SCNN
Architecture	6C3-P2-16C3-128FC-64FC-10FC	6C3-16C3-128FC-64FC-10FC
Test Accuracy	97.3 %	98.6 %
Average Loss	0.006655	0.048219
No. of Parameters	83.702K	1.616M
No. of FLOPS	3.326M	0.281G

Note: In architecture naming, nCx refers to an n-channel convolutional neural network with kernel size x (x can be m x n for non-square kernels), Px refers to max pooling with kernel size x, and yFC refers to fully connected layers with y number of neurons.

The proposed SCNN performed better than the CNN in terms of accuracy as seen in Table 1. However, a closer examination of the loss values obtained indicated that the loss incurred in SCNN is generally higher than in CNN by a factor of nearly 10x. In this case, the sparsity of using spiking activations affected the network such that when the prediction is wrong, the prediction is far from the actual values. Spiking activations led to difficulties in differentiating the classes.

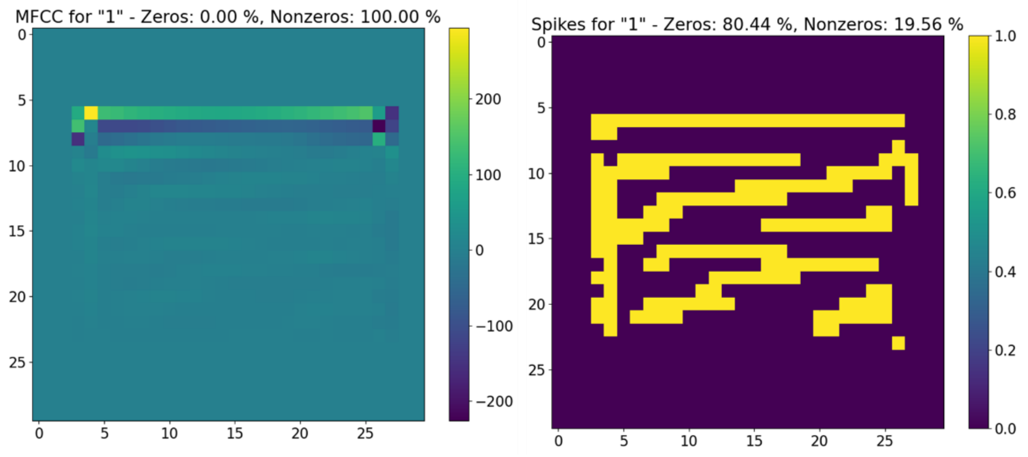

Figure 3 shows the output of one filter in the first layer after convolution (left) and after the spiking activation (right) with an increase in the number of zeros from 0 % to 80.44 % within the data representation.

Table 2 shows the performance of different SCNN architectures tested using FSDD [15].

Model	SCNN Architecture	Test Accuracy	Quantized Test Accuracy	No. of parameters	FLOPS
A	6C3-16C3-128FC-64FC-10FC	98.6 %	99.4 %	1.616 M	281.000 M
A	6C3-16C3-128FC-64FC-10FC (only 1 timestep)	95.8 %	97.3 %	1.616 M	28.0850 M
B	6C3-128FC-10FC	98.6 %	98.9 %	0.702 M	88.930 M
B	6C3-128FC-10FC (only 1 timestep)	96.9 %	93.1 %	0.702 M	8.8930 M
C	6C3-P2-128FC-10FC	93.9 %	96.9 %	0.183 M	26.722 M

In Table 1, while the number of parameters and FLOPS in the SCNN profiled was higher than that in CNN, it does not account for hardware implementation techniques which are incredibly key to utilizing the full potential of SNN. As seen in Figure 3, the sparsity derived from using spiking layers is significant, and the binary output can be highly efficient when it comes to both memory and computation at the hardware level. At the same time, it displays a challenge in the use of SNN, especially with general-computing devices. With a reduction of the timestep from 10 to 1, the computational expenditure was reduced significantly at the expense of the accuracy, which saw a degradation of around 2-3 % in Table 2.

As hypothesized, the first spiking layer remains to be highly important to maintain the accuracy of the network. Just by using the max-pooling operation on the output of the first layer from Model B to Model C, it was observed that the accuracy can drop as far as 4 %. On the other hand, removing the max-pooling operation on the second layer and so on did not affect the accuracy as much. At the encoding layer, any error or information loss was propagated throughout the network, leading a poor performance. To observe suitability for a different dataset, the Google Speech Commands (GSC) dataset [33] was used based on similar pre-processing (MFCC calculated and padding before input to network), and the performance of the SCNN degraded significantly. As shown in Table 3, shallower networks (Model B and C) suffered degradation more severely than deeper networks (Model) A. With more classes than FSDD, the GSC dataset challenged the performance of the SCNN not just in the accuracy but increased the number of parameters and number of FLOPS as well. The SCNN implemented is therefore not ideal for multiple datasets.

Table 3 shows the performance of different SCNN architectures tested using the GSC Dataset [33].

Model	SCNN Architecture	Test Accuracy	No. of parameters	FLOPS
A	6C3-16C3-128FC-64FC-10FC	50.9 %	1.617 M	304.000 M
B	6C3-128FC-10FC	1.50 %	0.705 M	96.756 M
C	6C3-P2-128FC-10FC	1.50 %	0.187 M	29.364 M

ii. Quantization results

To better simulate the performance of the SCNN proposed, the model was quantized to int8 precision from FP32 and re-evaluated through quantization-aware training (QAT), which snnTorch currently supports. The QAT method follows the cosine annealing method proposed by [34]. The quantization was implemented using *Brevitas* [35] and snnTorch. The former provides the QAT capabilities within the convolution and linear layers, and the latter provides the quantization of the activation at the membrane state levels. Table 2 shows the quantization test accuracy without quantized states. Based on the different architecture, it was observed that using the QAT, the model remains resilient to quantization, which was expected as the data within the model is already binarized by the spiking layers. Similarly, to the unquantized models, timesteps play a significant role in providing high performance to the SCNN quantized through QAT.

With state quantization, the network in Model B with 10 timesteps experienced a sharp decrease in test accuracy, reaching 70.0 % in one of the runs. On the other hand, the network in Model C performed better, reaching an average test accuracy of 94 % across 3 runs, like the unquantized performance. The quantization of the snnTorch weights increased the training time tremendously and degraded the performance of the SNN more than simple weights quantization.

iii. Comparison with other SCNN counterparts

Table 4 compares this work and the related SCNN or CSNN counterparts. Similarly, SNN-related solutions including self-organizing maps (SOM-SNN) and spiking recurrent neural networks (SRNN) are reported in the table for completeness. In the context of SNN-based solutions, the SCNN network proposed in this paper achieved high accuracy while remaining shallow and easy to train comparable to conventional CNN approaches. Instead of using a teacher-student network or weight transfer, backpropagation using surrogate gradient allowed the SCNN to be trained via the PyTorch framework with optimizers like Adam made available without the need to train additional networks. However, the use of MFCC as data input requires feature extraction on the hardware as compared to time-domain implementations such as [4], and [7]. Hardware acceleration for MFCC extraction such as the one proposed by [12] could help reduce feature extraction overhead and enable a modular approach in hardware implementation.

iv. Hyperparameters, Sparsity, and Accuracy

While the performance above was obtained using a manual search through the hyperparameters, it is desirable to study the impact of hyperparameters on the accuracy and the sparsity of data, which could be exploited in hardware implementation for energy and memory optimization. Since the first spiking layer was determined to be of great importance due to the encoding capability, only the sparsity of the first layer output was examined.

Beta was defined to be the decay factor of the membrane factor and can also be viewed as a factor to vary the current potential based on the potential at the previous timestep. As seen in Figure 4, the accuracy value follows a relatively similar trend across all architectures at different beta values where the peak occurs at around 0.50. However, the sparsity of the first layer for different architectures varies differently for different beta values without a clear correlation between sparsity and accuracy due to beta values.

Table 4 compares the performance of this work with similarly published works using spiking neural networks for audio speech recognition datasets which the author is aware of.

Year	References	Type of Neural Network	Data Pre-processing	Architecture	Dataset	Training Method	Test Accuracy
2018	[6]	SOM-SNN (Self-organising map-SNN)	STFT (Short-time Fourier Transform)	SOM-SNN	Speech Digit	Maximum-Margin Tempotron	97.4 %
2018	[5]	SCNN	Neuromorphic Auditory Sensor with Sonogram	6C5-P3-12C3-P2-2F	Speech Commands (Left, Right)	Weight transfer	89.9 %
2019	[7]	CNN	Energy with 1-layer SNN	8C12-P2-16C4-10F	Speech Digit	Backpropagation	96 %
2019	[37]	CSNN	Mel-spectrogram	64C4x3-64C4x3-64C4x3-10F	Google Speech Commands (GSC)	Backpropagation with Surrogate Gradient	94 %
2021	[4]	SRNN (Spiking recurrent neural network)	Minimal (Time-domain classification)	Input-2 Spiking recurrent layers – Output layer	Spiking Heidelberg Digits (SHD) / GSC /TIMIT	Backpropagation with multi-Gaussian surrogate gradient	90.4 % /92.1 % /66.1 %
2022	[36]	CSNN	Minimal (Spiking dataset)	3 Hidden layers (full architecture unreported)	SHD	Backpropagation with Surrogate Gradient and Kaiming initialization	83.1 % (95.9 % Validation accuracy)
2022	This work	SCNN	MFCC	6C3-128F-10F	Free Spoken Digit/GSC	Backpropagation with Surrogate Gradient	<u>98.6 % /50.9%</u>

Note: Test accuracies obtained from [4] correspond to the respective datasets. [36] conducted comprehensive studies on different architectures and the best result was used for comparison.

On the other hand, increasing the number of timesteps as seen previously increases the computation time and resource, with the sparsity of the data at the first layer oscillating for Model A and Model C in Figure 4. While accuracies generally increased for all architectures as timesteps increase, the sparsity oscillates and computation increases. This meant that to optimize hyperparameters for sparsity, computation, and accuracy, functions with oscillatory properties such as sinusoidal functions may be used. Nonetheless, without a clear oscillating period and patterns, this optimization will remain difficult.

Lastly, the impact of threshold value was the most consistent across the different architectures. As the threshold value increases, it is expected for sparsity to increase. This increase in sparsity led to an obvious decrease in accuracies across the different architectures. For shallower networks present in Model B and Model C, the accuracies remain consistent for a larger range of threshold values as compared to the deeper network in Model A. A shallower network reduced the impact of the error propagation from the threshold voltage as compared to a deeper network. Thus, for sparsity-aware neural architecture search or hyperparameter-tunning for SCNNs, the threshold voltage can be possibly used, while the impact of timesteps and beta remains unclear for optimization, especially with different architectures.

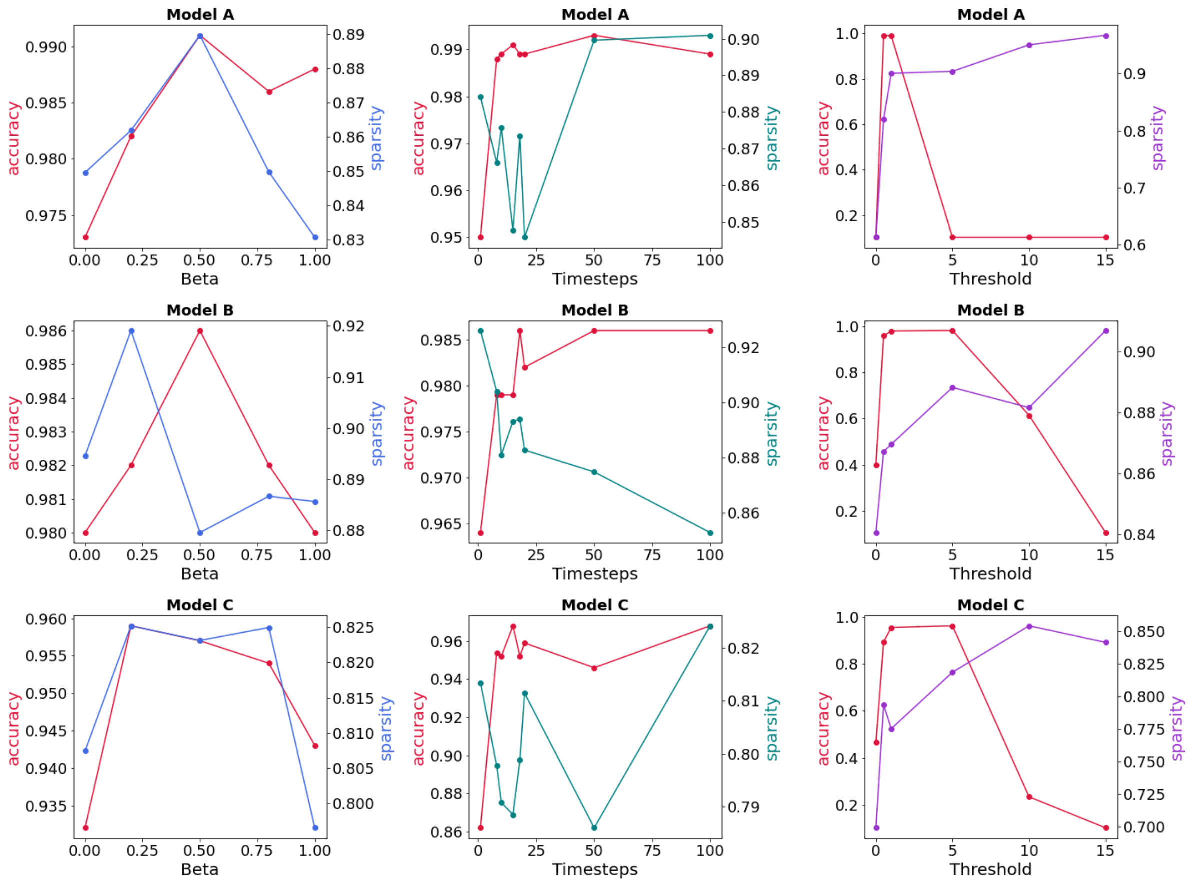

Figure 4 shows accuracy (left axis) and sparsity (right axis) for the different SCNN architecture and the different hyperparameters. Rows 1, 2, and 3 (left to right) show the results for models A, B and C respectively. Columns 1, 2, and 3 (top to bottom) show the results when beta, timesteps, and threshold were varied respectively.

Conclusion

This paper proposed a 5-layer SCNN for voice keyword classification which achieved 98.6 % test accuracy. Modifications to the network architecture were examined with quantization to simulate the hardware implementation. One of the modified architectures achieved minimal accuracy degradation with reduced computational cost. Of the three hyperparameters studied (beta, timesteps, and threshold), the threshold value was determined to be suitable for optimization for accuracy and sparsity. Further analysis of the relationship between the hyperparameters and the implementation of the networks on actual hardware will offer greater insights into actual performance, computational cost, and training cost.

References

[1] M. Pfeiffer and T. Pfeil, ‘Deep Learning With Spiking Neurons: Opportunities and Challenges’, *Front. Neurosci.*, vol. 12, 2018, Accessed: Sep. 27, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2018.00774

[2] J. D. Nunes, M. Carvalho, D. Carneiro, and J. S. Cardoso, ‘Spiking Neural Networks: A Survey’, *IEEE Access*, vol. 10, pp. 60738–60764, 2022, doi: 10.1109/ACCESS.2022.3179968.

[3] J. Wu, E. Yılmaz, M. Zhang, H. Li, and K. C. Tan, ‘Deep Spiking Neural Networks for Large Vocabulary Automatic Speech Recognition’, *Front. Neurosci.*, vol. 14, 2020, Accessed: Sep. 09, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2020.00199

[4] B. Yin, F. Corradi, and S. M. Bohté, ‘Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks’, *Nat. Mach. Intell.*, vol. 3, no. 10, Art. no. 10, Oct. 2021, doi: 10.1038/s42256-021-00397-w.

[5] J. P. Dominguez-Morales *et al.*, ‘Deep Spiking Neural Network model for time-variant signals classification: a real-time speech recognition approach’, in *2018 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2018, pp. 1–8. doi: 10.1109/IJCNN.2018.8489381.

[6] ‘Frontiers | A Spiking Neural Network Framework for Robust Sound Classification’. https://www.frontiersin.org/articles/10.3389/fnins.2018.00836/full (accessed Sep. 11, 2022).

[7] J. Hu, W. L. Goh, Z. Zhang, and Y. Gao, ‘Voice Keyword Recognition Based on Spiking Convolutional Neural Network for Human-Machine Interface’, in *2020 3rd International Conference on Intelligent Autonomous Systems (ICoIAS)*, Feb. 2020, pp. 77–82. doi: 10.1109/ICoIAS49312.2020.9081859.

[8] S. Hershey *et al.*, ‘CNN architectures for large-scale audio classification’, in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 131–135. doi: 10.1109/ICASSP.2017.7952132.

[9] M. Espi, M. Fujimoto, K. Kinoshita, and T. Nakatani, ‘Exploiting spectro-temporal locality in deep learning based acoustic event detection’, *EURASIP J. Audio Speech Music Process.*, vol. 2015, no. 1, p. 26, Sep. 2015, doi: 10.1186/s13636-015-0069-2.

[10] H. Zhang, I. McLoughlin, and Y. Song, ‘Robust sound event recognition using convolutional neural networks’, in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 559–563. doi: 10.1109/ICASSP.2015.7178031.

[11] E. Yılmaz, Ö. B. Gevrek, J. Wu, Y. Chen, X. Meng, and H. Li, ‘Deep Convolutional Spiking Neural Networks for Keyword Spotting’, in *Interspeech 2020*, Oct. 2020, pp. 2557–2561. doi: 10.21437/Interspeech.2020-1230.

[12] K. Yang, L. Zhu, and W. Shan, ‘Design of an ultra-low Power MFCC Feature Extraction Circuit with Embedded Speech Activity Detector’, in *2021 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA)*, Nov. 2021, pp. 82–83. doi: 10.1109/ICTA53157.2021.9661980.

[13] 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’. arXiv, Jun. 30, 2020. Accessed: Sep. 11, 2022. [Online]. Available: http://arxiv.org/abs/1907.01167

[14] J. Wu, C. Xu, D. Zhou, H. Li, and K. C. Tan, ‘Progressive Tandem Learning for Pattern Recognition with Deep Spiking Neural Networks’. arXiv, Jul. 02, 2020. Accessed: Sep. 11, 2022. [Online]. Available: http://arxiv.org/abs/2007.01204

[15] Z. Jackson, C. Souza, J. Flaks, Y. Pan, H. Nicolas, and A. Thite, ‘Jakobovski/free-spoken-digit-dataset: v1.0.8’. Zenodo, Aug. 09, 2018. doi: 10.5281/zenodo.1342401.

[16] P. Rao, ‘Audio Signal Processing’, *Speech Audio Image Biomed. Signal Process. Using Neural Netw.*, pp. 169–189, 2008, doi: 10.1007/978-3-540-75398-8_8.

[17] T. Fukadat, K. Tokudatt, T. Kobayashzjtt, and S. Imaittt, ‘An Adaptive Algorithm for Mel-Cepstral Analysis of Speech’.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha, ‘Gradient-Based Learning Applied to Document Recognition’, p. 46, 1998.

[19] 'ImageNet classification with deep convolutional neural networks | Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1'. https://dl.acm.org/doi/10.5555/2999134.2999257 (accessed Sep. 13, 2022).

[20] K. Simonyan and A. Zisserman, 'Very Deep Convolutional Networks for Large-Scale Image Recognition'. arXiv, Apr. 10, 2015. doi: 10.48550/arXiv.1409.1556.

[21] 'adhishthite/sound-mnist: A Convolutional Neural Network to identify spoken digits.' https://github.com/adhishthite/sound-mnist (accessed Sep. 13, 2022).

[22] M. Hashemi, 'Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation', *J. Big Data*, vol. 6, no. 1, p. 98, Nov. 2019, doi: 10.1186/s40537-019-0263-7.

[23] J. K. Eshraghian *et al.*, 'Training Spiking Neural Networks Using Lessons From Deep Learning'. arXiv, Jan. 14, 2022. Accessed: Sep. 13, 2022. [Online]. Available: http://arxiv.org/abs/2109.12894

[24] A. Paszke *et al.*, 'Automatic differentiation in PyTorch', p. 4.

[25] K. Roy, A. Jaiswal, and P. Panda, 'Towards spike-based machine intelligence with neuromorphic computing', *Nature*, vol. 575, no. 7784, Art. no. 7784, Nov. 2019, doi: 10.1038/s41586-019-1677-2.

[26] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014.

[27] E. M. Izhikevich, 'Simple model of spiking neurons', *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003, doi: 10.1109/TNN.2003.820440.

[28] C. Lee, P. Panda, G. Srinivasan, and K. Roy, 'Training Deep Spiking Convolutional Neural Networks With STDP-Based Unsupervised Pre-training Followed by Supervised Fine-Tuning', *Front. Neurosci.*, vol. 12, 2018, Accessed: Oct. 05, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2018.00435

[29] E. O. Neftci, H. Mostafa, and F. Zenke, 'Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks', *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 51–63, Nov. 2019, doi: 10.1109/MSP.2019.2931595.

[30] B. Cramer *et al.*, 'Surrogate gradients for analog neuromorphic computing', *Proc. Natl. Acad. Sci.*, vol. 119, no. 4, p. e2109194119, Jan. 2022, doi: 10.1073/pnas.2109194119.

[31] 'fvcore'. Meta Research, Sep. 13, 2022. Accessed: Sep. 14, 2022. [Online]. Available: https://github.com/facebookresearch/fvcore

[32] D. P. Kingma and J. Ba, 'Adam: A Method for Stochastic Optimization'. arXiv, Jan. 29, 2017. Accessed: Sep. 14, 2022. [Online]. Available: http://arxiv.org/abs/1412.6980

[33] P. Warden, 'Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition'. arXiv, Apr. 09, 2018. doi: 10.48550/arXiv.1804.03209.

[34] J. K. Eshraghian, C. Lammie, M. R. Azghadi, and W. D. Lu, 'Navigating Local Minima in Quantized Spiking Neural Networks'. arXiv, Feb. 15, 2022. Accessed: Sep. 11, 2022. [Online]. Available: http://arxiv.org/abs/2202.07221

[35] Alessandro *et al.*, 'Xilinx/brevitas: Release version 0.7.1'. Zenodo, Dec. 14, 2021. doi: 10.5281/zenodo.5779154.

[36] J. Rossbroich, J. Gygax, and F. Zenke, 'Fluctuation-driven initialization for spiking neural network training'. arXiv, Jun. 21, 2022. doi: 10.48550/arXiv.2206.10226.

[37] R. Zimmer, T. Pellegrini, S. F. Singh, and T. Masquelier, 'Technical report: supervised training of convolutional spiking neural networks with PyTorch'. arXiv, Nov. 22, 2019. Accessed: Sep. 19, 2022. [Online]. Available: http://arxiv.org/abs/1911.10124