Interspeech 2022
18-22 September 2022, Incheon, Korea

Knowledge Distillation for In-Memory Keyword Spotting Model

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

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

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

Abstract

We study a light-weight implementation of keyword spotting (KWS) for voice command and control, that can be implemented on an in-memory computing (IMC) unit with same accuracy at a lower computational cost than the state-of-the-art methods. KWS is expected to be always-on for mobile devices with limited resources. IMC represents one of the solutions. However, it only supports multiplication-accumulation and Boolean operations. We note that common feature extraction methods, such as MFCC and SincConv, are not supported by IMC as they depend on expensive logarithm computing. On the other hand, some neural network solutions to KWS involve a large number of parameters that are not feasible for mobile devices. In this work, we propose a knowledge distillation technique to replace the complex speech frontend like MFCC or SincConv with a light-weight encoder without performance loss. Experiments show that the proposed model outperforms the KWS model with MFCC and SincConv front-end in terms of accuracy and computational cost.

Index Terms: Keyword spotting, knowledge distillation, in-memory computing, speech encoder, MFCC, SincConv

1. Introduction

Keyword spotting (KWS) is increasingly used in mobile devices for always-on voice command and control. An effective KWS system is expected to be accurate, memory and computation efficient.

The traditional von Neumann computing architecture faces two common challenges: 1) the disparity between the processing speech of the memory processing units. 2) the high energy cost of moving data between the memory and processing units, which is aggravated by big data processing. In-memory computing (IMC) [1, 2] has been studied to overcome the challenges, where some computations can be performed and organized by the memory itself. The computational complexity can be further reduced when the memory devices are coupled physically [3]. With a higher computational speed of the memory and less data movement, we can conduct the KWS task in IMC more efficiently.

However, IMC research is mainly focused on performing multiplication-accumulation operations inside the memory-macro in a highly parallel and efficient manner. The commonly used audio frontends such as Mel-frequency cepstrum coefficient (MFCC) and SincConv cannot be implemented directly via IMC to achieve an adequate performance because of the logarithm function. Also, the look-up table technique for logarithm is energy consuming in IMC implementation compared to addition or multiplication operations. To address this, we propose an IMC-friendly encoder to replace the logarithm-based feature extraction front-end. In addition, we develop a knowledge distillation framework to transfer a high performance KWS model to an IMC-friendly model in a learning process.

2. Related work

The neural solutions to KWS [4, 5] can be grouped into the pipeline approach and end-to-end approach.

In the pipeline approach, a frontend first extracts features, such as MFCC, from input audio. Then the extracted features are fed into the various downstream classifier [4, 5, 6, 7, 8, 9]. The variants of MFCC include log-Mel filter bank energies (LFBE) [10], the Howl toolkit [11]. On the hardware platform, such as FPGA, MFCC feature extraction can be implemented [12, 13]. However, such implementation usually involves a large amount of computation. As shown in [13], the MFCC frontend consumes much more power (2,137.4μJ) than the neural network classifier (117μJ) for a two-class KWS task because Fast Fourier Transform (FFT) is computationally expensive. As an example, MFCC involves 97.8% and 90.1% of the execution time on CPU and ARM respectively [14].

The end-to-end approach typically employs a convolutional layer, in place of FFT implementation, that performs frequency analysis equivalence. For example, Wav2Vec 2.0 [15] can achieve state-of-the-art performance in ASR and provide a powerful encoder for several downstream tasks [16]. However, its vast parameters (about 97.2 million) are not feasible on mobile devices.

SincNet [17] used a parametrized Sinc-convolutions (SincConv) that seeks to benefit from the best of both pipeline and end-to-end approaches. SincConv is a trainable rectangular bandpass filter bank used for extracting auditory features from the frequency domain. In the frequency domain, the rectangular bandpass filter could be written as the difference between two low-pass filters, which could be converted to the difference between two sinc functions ($sinc(x) = sin(x)/x$) for filters in the time domain (shown in Eq.1).

$$g[n, f_1, f_2] = 2f_2 sinc(2\pi f_2 n) - 2f_1 sinc(2\pi f_1 n) \quad (1)$$

where f_1, f_2 separately are learned lower and upper cutoff frequencies for each filter, all sets of f_1, f_2 are initialized with the cutoff frequencies of the mel-scale filter bank and then further trained with two trainable parameters that control the scale and bandwidth of the filter bank. Then the sinc filters $g[n, f_1, f_2]$ are used as the convolutional kernels in the time domain like other time-domain filter banks.

While SincConv strikes a tradeoff between the number of parameters and accuracy, mel-scale conversion in SincConv involves a logarithm computing that is unfriendly to IMC implementation. The look-up table for logarithm can be considered as a solution to logarithm in hardware implementation, but it is also expensive on the IMC platform. Each random access to

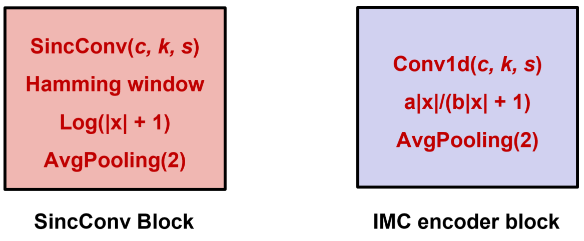

Figure 1: *The architecture of a SincConv block (teacher) and a IMC encoder (student). Parameters c, k, s in SincConv and Conv1d represent the number of output channels, kernel length, and stride.*

RAMs costs a few orders of magnitude more in terms of energy and latency than compute operations. DRAM read, for example, takes around 640 pJ, while computation operations like addition and multiplication take about 0.1 pJ and 3.1 pJ respectively [18].

This study is motivated to replace the complex speech frontend like MFCC or SincConv with an alternative without performance loss.

3. The Proposed KWS Framework

We study an IMC encoder to serve as the feature extraction frontend as part of the end-to-end model. We will also formulate a knowledge distillation framework to transfer knowledge from MFCC or SincConv to the IMC encoder to benefit from the high performance teacher model.

3.1. In-memory computing (IMC) encoder

The SincConv encoder [17] is a trainable frontend with only two parameters, which has been successfully used in energy efficient neural solutions to KWS [19]. To simplify the logarithm function in SincConv operation [17] and the logarithm compression function ($y = log(|x| + 1)$) [20, 21] in the SincConv block (Fig.1 left), we propose the IMC encoder block (Fig.1 right), which is composed of 1d convolution (viz., Conv1d), activation function and average pooling.

The proposed IMC encoder is similar to the SincConv block, except that SincConv and Hamming window are replaced by the Conv1d layer. We approximate the logarithm function with a simple activation function:

$$y = a * |x|/(1 + b * |x|),$$

where a and b are two parameters to approximate the logarithm function.

There are two ways for estimating the parameters a and b: 1) we choose a and b with proper values determined by mean square error between our activation function and log-compression function to approximate the log-compression function and then fix them in training. 2) we train a and b together with a downstream classifier. We consider that the former can better approximate the log-compression function, while the latter takes advantage of better fitting input data samples.

3.2. Knowledge distillation

Knowledge distillation [22] is a method used to transfer knowledge from a large teacher model to a small student model that is suitable for mobile devices and embedded systems. We propose a knowledge distillation framework, as shown in Fig.2, to

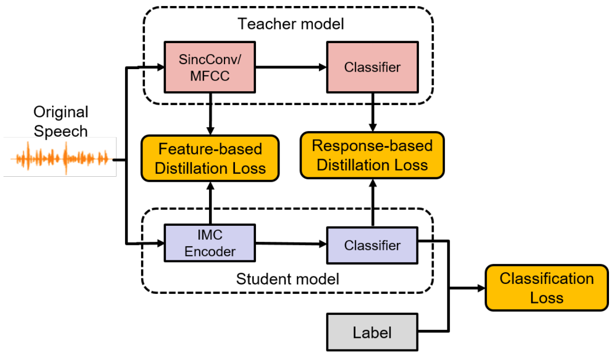

Figure 2: *The IMC encoder as part of the student model in a teacher-student knowledge distillation framework.*

transfer the knowledge from SincConv/MFCC frontend, which is commonly used and represents good performance, to a computation efficient IMC encoder.

Both the teacher model and student model are composed of an encoder and a classifier. The teacher model consists of a MFCC or SincConv frontend, while the student model employs an IMC encoder as the frontend. MFCC is believed to reflect the human auditory responses, whose sensitivity varies on different frequency levels. SincConv is also a parametrized Sinc-convolutions that can capture audio information in frequency domain. Both MFCC and SincConv perform well in KWS and speech recognition in general. The knowledge distillation framework benefits from its high performance speech feature representation. For the classifier, the teacher and student models have the same classifier architecture.

We propose to transfer the knowledge from both the encoder and classifier of teacher model via feature-based ($Loss_f$) and response-based ($Loss_r$) distillation loss. The loss functions are as follows:

$$\begin{aligned}
Loss_f &= MSE_{loss}(S_{enc}, T_{enc}) \\
Loss_r &= D_{KL}(S_{out}||T_{out}) \\
Loss_{cls} &= CrossEntropy(S_{out}, label) \\
Loss &= \alpha_1 Loss_f + \alpha_2 Loss_r + \alpha_3 Loss_{cls}
\end{aligned}$$

where $S_{enc}, S_{out}, T_{enc}, T_{out}$ are separate output of student and teacher model from their encoder and classifier, $MSE_{loss}(\cdot)$ represents the mean square error loss, and $D_{KL}(\cdot)$ represents Kullback Leibler (KL) Divergence. α_1, α_2, α_3 are the weighting factors for the total loss.

4. Experiments

4.1. Dataset and Settings

Datasets: In our experiments, we use the Speech Commands version 2 (v2) dataset from Google [23] with data augmentation and preprocessing methods in [16] to train and evaluate our model. There are 105,829 one-second utterances classified into 35 different keywords spoken by 2,618 different speakers. For our task, this dataset has 12 classes for classification: “yes”, “no”, “up”, “down”, “left”, “right”, “on”, “off”, “stop”, “go”, unknown, or silence. The remaining 25 keywords out of the mentioned ten words are labeled as unknown, and other background noise samples are labeled as silence. In our experiments, the dataset is divided in a ratio of 8:1:1 for training, validation, and testing.

Table 1: *Performance comparison for different encoders in terms of Multiply-Accumulate Operations(MACs) and accuracy (%).*

Encoder	MACs	Accuracy
Conv1d	4.96M	94.75%
MFCC	8.07M	95.85%
SincConv	5.39M	96.32%
Our IMC	**4.96M**	**96.57%**

Table 2: *KWS performance in terms of accuracy (%). '$Encoder_t$' indicates the encoder of the teacher model; and 'a, b' the parameter estimation techniques of activation function in the IMC encoder. We refer "Baseline" to the student model that is trained independently, and refer "After KD" to the student model after knowledge distillation from the "Teacher" model.*

$Encoder_t$	a, b	Baseline	Teacher	After KD
MFCC	Fixed	94.24%	95.85%	94.64%
MFCC	Trainable	94.75%	95.85%	95.03%
SincConv	Fixed	94.24%	96.32%	96.06%
SincConv	Trainable	94.75%	96.32%	96.57%

Experiment settings: In our following experiments, ResNet-8 [24] is used as the default classifier in both the teacher models and student models. For encoder blocks, number of filters of SincConv and IMC encoder are selected to be 128. And MFCC use 128 MFCC features. Parameters c, k, s in SincConv block and the IMC encoder block are 128, 150 and 62. Both teacher model and student model are trained by AdamW optimizer with initial learning rate of 1e-3 and batch size of 32.

Knowledge distillation framework settings: In knowledge distillation, we first train the teacher model with MFCC and SincConv from scratch, then transfer the knowledge of encoder and classifier to the student model. When transferring the knowledge to the student model, the balancing parameters $\alpha_1, \alpha_2, \alpha_3$ are set 0.3, 0.1 and 0.6.

4.2. Results

In Table 1, we compare our proposed IMC encoder with other encoders in terms of computational cost and accuracy. Note that all the models in Table 1 use encoders with same number of filters and classifiers with same architecture (ResNet-8 [24]) for a fair comparison. In most cases, the energy cost of each operation is fixed on a certain device, thus the number of MACs can represent the computation cost of a model. From table 1, our proposed IMC encoder can not only get rid of the complex logarithm operation that is not implementable on the IMC platform but also outperforms both MFCC and SincConv in terms of accuracy with fewer computational costs. Also, compared with the Conv1d encoder that is usually used in in-memory computing, the IMC encoder can achieve better performance.

In Table 2, we compare the accuracy on Speech Commands v2 dataset over different architectures, where "Fixed" means a and b are chosen to minimize MSE between the activation function and log-compression function and are fixed in training, "Trainable" means that a and b are trainable via back-propagation.

From Table 2, we have the following observations: (i) The student models (baseline), if trained independently, has performance lagging behind the teacher models with SincConv and MFCC. However, they are on par with or and even outperform the teacher models through knowledge distillation. (ii) Using trainable a, b can make the IMC encoder perform better both before and after knowledge distillation. With additional parameter to for activation, the activation value from our IMC encoder could be consistent with the expected data distribution of following classifier and easy for optimization. (iii) SincConv outperforms MFCC as the encoder of teacher model. The teacher model with SincConv frontend not only achieves higher accuracy, but also brings about a higher accuracy improvement over the student model through knowledge distillation than MFCC.

Table 3: *KWS performance with different knowledge distillation frameworks. "Encoder + classifier" refers to the teacher-student knowledge distillation framework in Fig. 2. "Encoder" refers to the framework that only transfers knowledge from the encoder of the teacher model to the student model.*

$Encoder_t$	KD framework	Teacher	After KD
MFCC	Encoder	95.85%	94.56%
MFCC	Encoder + classifier	95.85%	95.03%
SincConv	Encoder	96.32%	95.04%
SincConv	Encoder + classifier	96.32%	96.57%

Table 4: *KWS performance with weighting factor of KD loss. Default settings are marked with gray.*

weighting factors for the KD loss			Accuracy (%)
α_1	α_2	α_3	
0.3	0	0.7	96.45
0.1	0.1	0.8	96.34
0.2	0.1	0.7	96.46
0.3	**0.1**	**0.6**	**96.57**
0.4	0.1	0.5	96.38
0.5	0.1	0.4	96.02
0.1	0.3	0.6	96.29
0.2	0.3	0.5	96.02
0.3	0.3	0.4	95.75
0.4	0.3	0.3	94.97
0.5	0.3	0.2	95.26

In our knowledge distillation framework in Fig.2, we include the downstream classifier with the same architecture into both teacher and student models. In Table 3, we compare its performance with another knowledge distillation framework. In the "Encoder" framework, we only transfer the knowledge from the encoder of the teacher model to the student model by setting the balancing parameter of KD loss $[\alpha_1, \alpha_2, \alpha_3]$ to be [1, 0, 0]. Then train the student encoder with the classifier, where the learning rate of the encoder is smaller than the classifier. As we can see from Table 3, the IMC encoder can achieve better performance when training the encoder and classifier together. Because the encoder is shallow, if we train the encoder and classifier separately, the encoder may forget the knowledge transferred from the teacher model when finetuning with the classifier.

There are three parameters α_1, α_2, α_3 used to weight cost functions mentioned in Sec 3.2. We compare the KWS performance of student models under different weighting factors in Table 4. Results in Table 4 demonstrate that: (1) Student models can benefit from the knowledge of teacher encoder since we

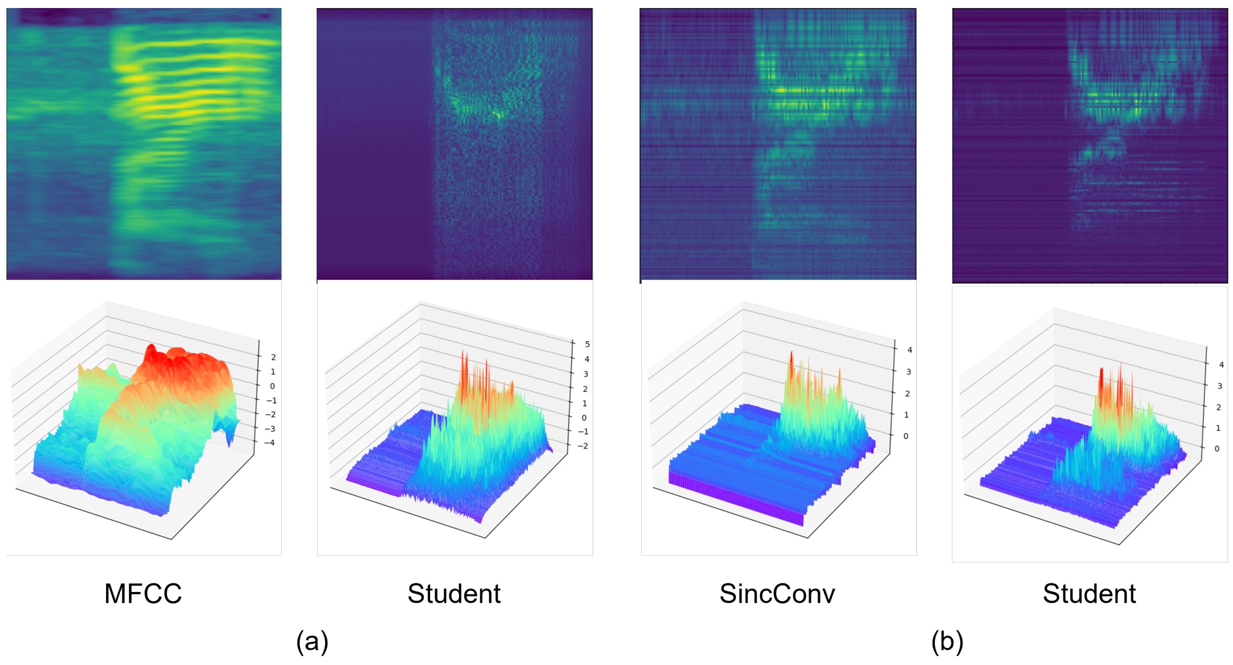

Figure 3: *The feature maps from encoders in our knowledge distillation framework. The upper panel and lower panel show the feature maps and their value distribution. (a) MFCC's and its student's encoding feature maps. (b) SincConv's and its student's encoding feature maps.*

can see that accuracy increase with α_1 when $\alpha_1 < 0.3$. (2) Putting less importance on the classification task (for example $\alpha_3 < 0.5$) can weaken the student model by keeping the student encoder focused on imitating the representation from the teacher model but ignoring the KWS task itself. (3) We found that classification loss ($Loss_{cls}$) can play a better role in optimization than the response-based loss($Loss_r$).

In Table 2, we found that SincConv could outperform MFCC as a teacher model. We also found that in knowledge distillation the feature-based ($Loss_f$) between the IMC encoder and MFCC always larger than that between the IMC encoder and SincConv. So we visualize their feature maps from encoder of those two knowledge distillation frameworks in Fig.3. It is evident from the feature maps (upper panel) in Fig.3 that the IMC encoder taught by teacher model with SincConv (Fig.3 (b) right) has a more distinct feature map than its MFCC counterpart (Fig.3 (a) right). And from the feature maps' value distribution (lower panel) in Fig.3, the IMC encoder model can only learn some basic features from MFCC. In contrast, the feature map of the SincConv frontend, which has the shape similar to the IMC encoder, can be easily imitated. This might be because the filter banks of SincConv are implemented in time domain , while MFCC is an encoding method in the frequency domain.

To further test the encoding performance of our IMC encoder, we test the performance of different types of encoders with two additional classifiers: Fully Convolutional Neural Networks(FCNN) [25] and GRU-based RNN model[26]. When training the IMC encoder in this experiment, we use the SincConv teacher and fixed a, b. Hyperparameters of these models above (including the number and size of each layer, learning rate, batch size, etc.) are tuned for this task. Table 5 shows that, IMC encoder models with different classifiers can achieve comparable performance to MFCC and SincConv, which implies that our IMC encoder can be used to replace MFCC or SincConv in other models for the keyword spotting task.

Table 5: *KWS performance in terms of accuracy(%) with different classifier.*

Encoder \ Classifier	ResNet-8	FCNN	RNN
Conv1d	94.75%	93.47%	93.76%
SincConv	96.32%	95.03%	94.67%
MFCC	95.85%	95.19%	94.98%
Our IMC	96.57%	95.24%	94.85%

5. conclusion

We propose an IMC-friendly KWS encoder and a knowledge distillation framework. The experiments show that our proposed IMC encoder outperforms KWS model with SincConv with less computational cost through knowledge distillation. And when transferring knowledge to the IMC encoder, SincConv could be a better encoder of teacher model compared to MFCC for its nature of time-domain convolution.

6. Acknowledgment

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)

7. References

[1] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, no. 6, pp. 333–343, 2018.

[2] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020.

[3] M. Di Ventra and Y. V. Pershin, "The parallel approach," *Nature Physics*, vol. 9, no. 4, pp. 200–202, 2013.

[4] 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.

[5] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha, "Temporal convolution for real-time keyword spotting on mobile devices," *arXiv preprint arXiv:1904.03814*, 2019.

[6] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword transformer: A self-attention model for keyword spotting," *arXiv preprint arXiv:2104.00769*, 2021.

[7] K. S. Rao and K. Manjunath, *Speech recognition using articulatory and excitation source features*. Springer, 2017.

[8] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[9] T. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," 2015.

[10] Y. Gao, N. D. Stein, C.-C. Kao, Y. Cai, M. Sun, T. Zhang, and S. Vitaladevuni, "On front-end gain invariant modeling for wake word spotting," *arXiv preprint arXiv:2010.06676*, 2020.

[11] R. Tang, J. Lee, A. Razi, J. Cambre, I. Bicking, J. Kaye, and J. Lin, "Howl: A deployed, open-source wake word detection system," *arXiv preprint arXiv:2008.09606*, 2020.

[12] P. Ehkan, F. Zakaria, M. Warip, Z. Sauli, and M. Elshaikh, "Hardware implementation of mfcc-based feature extraction for speaker recognition," in *Advanced Computer and Communication Engineering Technology*. Springer, 2015, pp. 471–480.

[13] J. Lei, T. Rahman, R. Shafik, A. Wheeldon, A. Yakovlev, O.-C. Granmo, F. Kawsar, and A. Mathur, "Low-power audio keyword spotting using tsetlin machines," *Journal of Low Power Electronics and Applications*, vol. 11, no. 2, p. 18, 2021.

[14] C. Choo, Y.-U. Chang, and I.-Y. Moon, "Fpga-based hardware accelerator for feature extraction in automatic speech recognition," *Journal of information and communication convergence engineering*, vol. 13, no. 3, pp. 145–151, 2015.

[15] 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.

[16] D. Seo, H.-S. Oh, and Y. Jung, "Wav2kws: Transfer learning from speech representations for keyword spotting," *IEEE Access*, pp. 1–1, 2021.

[17] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 1021–1028.

[18] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, 2014.

[19] S. Mittermaier, L. Kürzinger, B. Waschneck, and G. Rigoll, "Small-footprint keyword spotting on raw audio data with sinc-convolutions," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7454–7458.

[20] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux, "Learning filterbanks from raw speech for phone recognition," in *2018 IEEE international conference on acoustics, speech and signal Processing (ICASSP)*. IEEE, 2018, pp. 5509–5513.

[21] N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, and E. Dupoux, "End-to-end speech recognition from the raw waveform," *arXiv preprint arXiv:1806.07098*, 2018.

[22] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[23] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *ArXiv e-prints*, Apr. 2018. [Online]. Available: https://arxiv.org/abs/1804.03209

[24] 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.

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[26] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.