ENERGY-EFFICIENT SPEAKER IDENTIFICATION WITH LOW-PRECISION NETWORKS

Skanda Koppula, James Glass, Anantha P. Chandrakasan

Massachusetts Institute of Technology, Cambridge, MA 02139
{*skoppula, glass, anantha*}*@mit.edu*

ABSTRACT

Power-consumption in small devices is dominated by off-chip memory accesses, necessitating small models that can fit in on-chip memory. In the task of text-dependent speaker identification, we demonstrate a 16x byte-size reduction for state-of-art small-footprint LCN/CNN/DNN speaker identification models. We achieve this by using ternary quantization that constrains the weights to {-1,0,1}. Our model comfortably fits in the 1 MB on-chip BRAM of most off-the-shelf FPGAs, allowing for power-efficient implementation with 100x fewer floating point multiplications, and a 1000x decrease in estimated energy cost. Additionally, we explore the use of depth-wise separable convolutions for speaker identification, and show while significantly reducing multiplications in full-precision networks, they perform poorly when ternarized. We simulate hardware designs for inference on our model, the first hardware design targeted for efficient evaluation of ternary networks and end-to-end neural network-based speaker identification.

Index Terms— speaker identification, low-precision, end-to-end, neural network, on-chip memory, FPGA

1. INTRODUCTION

Speaker identification (SID) is the task of identifying that a speech sample belongs to a registered identity among a closed set of speakers. Recently, speaker identification has enjoyed a flurry of attention by researchers as more consumer devices have started using SID as part of their authentication sequence. State-of-art work has demonstrated success applying various end-to-end neural network architectures for both text-dependent and text-independent speaker recognition [1]. These methods exhibit performance on par traditional i-vector/GMM speaker models, for short and medium-length utterances [2].

In practical applications of speaker identification, efficient evaluation of a model is as important as accuracy. Small devices, such as cell phones, have limited energy and memory budgets, placing a different set of demands on the choice of model. Recent research on deep learning application-specific integrated circuits (ASICs) has demonstrated that energy consumption during network inference is dominated by off-chip memory accesses (e.g. DRAM), which are an order of magnitude more power-expensive than the much smaller on-chip memory access (e.g. SRAM) [3]. To reduce power and memory consumption for efficient inference, speaker verification models would require small network sizes and fewer multiplications along the chip's critical path.

In this paper, we demonstrate applying numeric quantization algorithms as an effective technique to reduce the model size and complexity in the task of text-dependent speaker identification. Our quantization to {1,0,-1} also reduces inference-time multiplications by two orders of magnitude, further reducing the energy consumption. We use our network improvements in our hardware design simulations (targeted at eventual evaluation our speaker identification network on the Xilinx Zync-7000 FPGA). To the best of our knowledge, this is the first FPGA-design for neural network-based speaker identification.

The next section describes the prior work in speaker identification and hardware translation of models. Sections 3 and 4 details our bitwidth-reduction training procedure and depth-separable CNN architecture. In Section 6, we describe results from our bitwidth-reduction experiments in SID networks. Then, in Section 5, we compare our previous networks with our proposed depth-wise separable convolutional network. Section 7 describes our implementation of one of our low-precision networks in FPGA, where we report latency and power consumption results. We conclude with a brief discussion of our results in Section 8.

All work can be found at `https://github.com/skoppula/sid-ternary-network`. The repository includes training scripts and links to pre-trained models and training logs.

2. PRIOR WORK

Efficient hardware evaluation of neural networks has been the focus of various manuscripts in the past two years. One avenue of research in the field has focused on building application-specific integrated circuits to efficiently evaluate large pre-trained neural networks ([4], [3], [5]). Another avenue of research has focused on algorithmic-based model optimization: reducing model complexity by introducing sparsity and Huffman-encoding parameters [6], student-

teacher models to distill knowledge to smaller models [7], bit-width reduction [8, 9, 10], and kernel-shaping techniques [11, 12]. These works have primarily been used for only computer vision tasks, with the exception of [13], which is a limited demonstration of FPGA-based phoneme recognition.

Less work has focused on adapting speaker recognition networks for efficient inference. [14] and [15] propose architectures for small end-to-end SV networks, the smallest of which is above 1 MB. Prior attempts at speaker verification in FPGA has focused on implementing speaker recognition using GMM and SVM-based systems [16, 17]. Our key contribution in this work is demonstrating drastically smaller networks for speaker identification, that are optimized for energy-efficient inference on custom hardware.

3. TRAINING LOW-BITWIDTH PARAMETERS

To reduce the size of networks, we constrain the network weights to a small set of quantized values instead of the standard 32-bit floating point. In particular, we explore *trained ternary quantization* (TTQ), a method to learn weights in the set {-1,0,1} [8]. In this work, we demonstrate that TTQ can be used successfully in text-dependent speaker identification, with modification to network's regularization function. In Section 6, we compare using TTQ in speaker ID with using another adjustable-bitwidth quantization technique [10].

In regular fully-connected and convolutional networks, every network layer l has a full-precision kernel W^l. Under TTQ, W^l is maintained, but not used during the forward pass. Rather, W^l is converted to its ternary approximation $\widetilde{W}^l$ by bucketing W^l's individual values, W^l_{ijk}, based on a layer-specific threshold Δ^l:

$$\widetilde{W}^l_{ijk} = \begin{cases} K^l_1 & \text{if } W^l_{ijk} \geq \Delta^l \\ 0 & \text{if } -\Delta^l < W^l_{ijk} < \Delta^l \\ K^l_2 & \text{if } W^l_{ijk} \leq -\Delta^l \end{cases}$$

This formulation is the original TTQ construction from [8], which uses two separate scaling constants K^l_1 and K^l_2 for the top and bottom thresholding regimes. In this work, we explored the simplification $K^l_2 = -K^l_1$ (briefly discussed in Section 6), so that the full-precision scalar factors out of the weight matrix, reducing kernel multiplications to additions and subtractions, substantially decreasing circuit area and latency. We retain the same back-propagation derivation, where the gradient updates are applied to both the K^L scaling values, and the original W^l shadow weights. The threshold Δ^l can either be learned, but as is used in [8] and in our experiments, we approximate $\Delta^l = 0.05 \times \max|h^l|$, where h^l are the incoming activations to layer-l.

To avoid full-precision floating-point operations, we can optionally constrain intermediate activations to 16-bit fixed point. After the multiply-and-accumulates in every layer, we normalize to fixed-point [-1,1] and downscale the activation bitwidth to fit within 16-bits. This is the same approach taken by [10].

Of particular note, in this form of constraining weights, no multiplications are required during layer evaluation. With weights in the set {-1,0,1}, the parallelized units in each layer evaluation module needs to only support additions and subtractions across each row. Fixed point multiplication with K^l occurs lazily after these activations have been computed.

4. SMALL-FOOTPRINT ARCHITECTURES

Various architectures have been proposed for the design of small-footprint networks to extract salient speaker characteristics (for identification or verification). For the purpose of obtaining streaming results in a latency-constrained real-world setting, most prior works use as input a sliding fixed-length context of approximately 50 Mel-Frequency Cepstral Coefficient (MFCC) frames. Heigold et al. propose a simple 4-layer, 500-wide fully-connected network (FCN) [18]. A similar linear FCN is used in [19] as well. Chen et al. builds on this, proposing a 256-wide four layer network, with the first FC layer replaced with a convolution of size 12×12 with stride 12 and 16 output channels [15]. [15] also considered locally-connected layers, splitting the first layer into four locally-connected sub-networks. Variani et al. propose using four 256-wide max-out layers as an alternative to fully-connected layers [14]. All architectures use ReLU non-linearities and tuned levels of last-layer dropout.

As an candidate alternative, we draw from the *Mobilenet* network used by Howard et al. for small-model real-time mobile vision applications [11]. The key optimization is the use of depth-separable convolution: instead of using a set of $w \times h \times c_i$ convolutional kernels, depth-separable kernels use a set of c_i $w \times h$ intra-channel spatial filters followed by a set of c_o 1×1 cross-channel filters to reduce multiplications. Using deeper CNNs for SID was an idea explored by Torfi et al. [2]. In our work, we introduce two depth-separable convolutional (DSC) layers (Figure 1). Note that since our initial input is single-channel (50×20 stacked), only the second and third layers benefit from depth separation. Using depth-wise separable convolution attempts, we hope to exploit the demonstrated modeling power of CNNs for SID (as demonstrated by [2] in the text-independent setting), while reducing the number of inference-time arithmetic operations.

5. COMPARING SID ARCHITECTURES

We conduct experiments on each of these proposed architectures to establish a baseline. We examine the size, the number of multiply-and-accumulates (MACs), and the equal error rate of each network. We use the RSR2015 corpus (300 speakers/73 unique phrases/196844 utterances) in our text-dependent speaker identification benchmarks. The reported equal error rates (EERs) are the average across 10 randomly

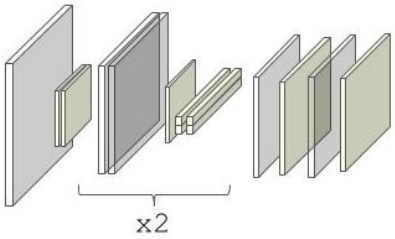

Table 1: Candidate network with depthwise-separable convolution (DSC) layers preceding two fully-connected layers.

selected phrases that we used to conduct experiments on. As input to our networks, we used 50 stacked frames of 20 MFCCs, corresponding to 20ms speech segments after energy-based voice activity detection (VAD) filtering. Matching the procedure in prior work, EER was computed by comparing the cosine distance of last layer activations ('d-vector'). All EER values are reported as the percent error rate.

Each of the full-precision models were sized to match in byte-size, targeting roughly 450K parameters, the approximate original size for in the reference networks given in [14], [15], and [19]. We use a fixed 200 epochs for training with a step-wise decreasing learning rate schedule. Reported EERs are from the 15% testing partition, using the model checkpoint with best validation identification accuracy. From our literature search, this is one of the first comparative text-dependent SID benchmarks of these architectures on a non-proprietary corpus, with code available for extension.

We compute energy estimates of each network during inference of one example based on the energy/operation model from Han et al. [3]. Our estimated energy consumption is a lower bound on the sum of multiplication/additions costs, SRAM costs (for storing/loading intermediate activations and weights), and DRAM costs (for loading weights that do not fit in our model's 2MB on-chip cache). Energy does not include pre-loading the weights into SRAM/DRAM (a one time initial cost).

Arch	Size	Mult	EER	Energy, μJ
FCN	14.52MB	452K	0.652	2325.6
LCN	14.39MB	448K	0.931	2304.6
CNN	14.58MB	2756K	0.838	2345.8
Maxout	14.56MB	453K	1.305	2329.3
DSC	14.51MB	483K	0.393	2323.9

Table 2: Performance of full-precision networks from Sec. 4. We compare average EER, floating point multiplies, estimated energy consumption, and model size for different architectures.

A summary of our results is listed in Table 2. We were able to obtain consistent results (within 1.5% EER) for each of the 10 unique utterances used in our text-dependent speaker ID task. Notably, despite improving on normal convolutions by decreasing multiplies by 5x, our depth-separable convolution network performed better than a standard CNN. Energy costs were dominated by DRAM weight accesses, contributing to the similar energy costs.

6. PERFORMANCE OF LOW-BITWIDTH SID

We test each of these architectures after training the models to use ternary weights and full-precision activations. In our accounting of operations, we apply the same transformations used in our hardware implementation. In particular, we factorize each ternary dot product into the addition of two dot-products with binary kernels:

$$T^l \times x^l = (B_1^l \oplus x^l) \times K_1^l - (B_2^l \oplus x^l) \times K_2^l$$

where T^l is a ternary kernel with weights $\in \{K_1^l, 0, K_2^l\}$, x^l is the layer-l intermediate full-precision activation, and B_1^l and B_2^l are binary matrices demarcating the positions of K_1^l's and K_2^l's in T^l, respectively. $\oplus$ is the binary matrix multiplication operator (bit-wise AND and addition). This procedure reduces energy consumption: the amortized cost of a 32-bit AND (<0.1 pJ) and the extra 32-bit addition (0.9 pJ) exceeds the cost of a 32-bit multiply (3.7 pJ) [3].

The alternative optimization to using separate K_1^l and K_2^l is to fix $K_2^l = -K_1^l$, which would remove the need for this decomposition trick. We found that with this change our networks were unable to learn ($> 50\%$ EER after 300 epochs), even with sweeping learning rate schedules and various architectural changes. We also found significantly greater deviation between each of the 10 unique utterances than with our full-precision networks. Within each model class, EERs fluctuated within a range of $\pm$ 3 EER.

Arch	Size	Bin Op	Mult	EER	E, μJ
FCN	907.6KB	905K	1.18K	1.337	1.235
LCN	899.4KB	896K	1.64K	39.25	1.228
CNN	911.4KB	5.51M	17.0K	2.064	1.838
Maxout	909.0KB	906K	1.41K	6.803	1.239
DSC	906.9KB	930K	5.05K	DIV	1.274

Fig. 1: Performance of networks from Sec. 4 after ternarization. We compare model size, binary operation count, floating point multiplies, average EER, and estimated energy consumption (μJ).

Without the DRAM accesses to re-load weights in the on-chip cache, our energy costs reduce significantly reduce. In all our models, error rates increase. In particular, the standard FCN/CNN/LCN models perform the best relative to their full-precision counterparts. For the LCN/DSC architectures, we were unable get the ternary networks to learn (diverging error [DIV], or converging to very high error). Analysis into why

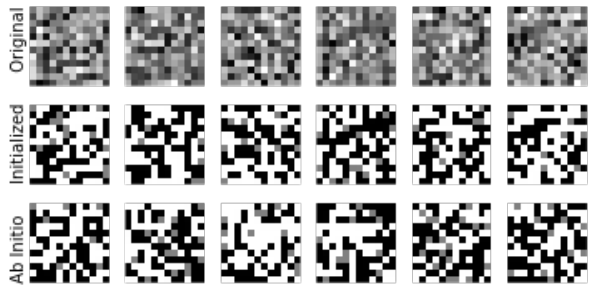

Fig. 2: Visualization of the first six filters in three SID CNNs. *First row*: Filters from a full-precision SID CNN. *Second row*: Filters from the corresponding TTQ network, initialized from the first-row's CNN's. *Third row*: TTQ CNN kernels trained from scratch on the same dataset.

these network diverged yielded fast convergence to unusually small weights and gradients, despite random initialization. Various fixed, including adjusting the learning rates, optimizers, and initializers, could not correct the training problem.

We also obtained diverging networks when we tried to initialize the ternary network's shadow weights with the pre-trained floating point weights, across all architectures. This form of weight initialization was used in the original TTQ manuscript, and our experiments suggest this bootstrapping is not always effective. Instead, initialization of ternary networks from other ternary networks did not exhibit such divergence. Six example TTQ kernels are illustrated in Figure 2. An interesting note is that sparsity in both TTQ kernels is roughly the same (55% and 54%), but the middle TTQ network encountered problems in training (>90% EER), whereas the *ab initio* TTQ network achieved an EER of 1.9.

7. HARDWARE DESIGN AND SIMULATION

We designed custom hardware for evaluation of our hardware-amiable ternary networks. In particular, we targeted inference on our FCN network, which achieved the lowest ternary EER and multiplication count among all our tested architectures. For implementation, we used Bluespec Verilog, and used the in-built Bluesim simulator to test our design. Our design comprises of the following modules:

- **Processing Units (PUs)** Performs the base arithmetic logic. Each PU computes the multiply-and-accumulates of a single row of ternary weights with input activations. After cycles finish to compute the MAC, scaling the cached output by either K_1 or K_2, and computes the ReLU of the row's output.

- **PU Controller** Coordinates feeding input and preloading weights of a particular layer into the PUs. Essentially, a large finite state machine.

- **UART I/O Module** Communicates input MFCCs/output speaker ID to off-chip ARM processor that performs input pre-processing.

- **On-chip BRAM** Stores intermediate activations, and network weights. This memory has single-cycle latency. The Bluespec library interface to BRAM is used.

In the taxonomy of neural network accelerators, our design falls under a weight-stationary approach: a subset of network weights are stored in the local registers (in the PU) as input is fed in the subsequent cycles. Figure 3 describes the connectivity of these different modules.

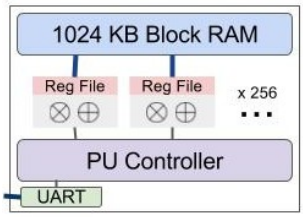

Fig. 3: Block diagram of the SID inference evaluator. Replicated in-parallel PUs perform the core combinational logic.

The architecture we use is based on the weight-stationary reference architectures provided by Sze, et al. [4]. The key modification is trimming down the combinational logic in the replicated processing units to exploit our ternary weights, allowing an order of magnitude fewer 32-bit multiplications along the critical path.

We were able to successfully test our designs in simulation. Excluding cycles used to read in UART frames into BRAM, a complete example takes 1820 cycles to evaluate. More than half the time is used to process the first FC layer as four sequential chunks, a 256×1000 by 1000 multiply. Currently, we are synthesizing the Verilog for placement on a Xilinx Zync-7000 FPGA, so that we can perform end-to-end tests.

8. DISCUSSION

We demonstrate a substantially smaller, hardware-amiable text-dependent speaker identification model, using weight ternarization. We compare various network architectures used in prior research, and demonstrate the poor results ternarizing a recent, highly-efficient convolutional architecture. We design and test in simulation inference on our ternary networks, demonstrating its suitability for hardware evaluation.

In future work, we intend on removing the discrete cosine transform (DCT) at the end of MFCC generation, as performed in recent work [2]. This would preserve spatial locality of the inputs, and possibly boost performance of our convolution-based architectures. Furthermore, we plan on power-gating the PU's so as to exploit the inherent sparsity in ternary networks to further reduce power consumption.

9. REFERENCES

[1] Pavel Matějka, Ondřej Glembek, Ondřej Novotnỳ, Oldřich Plchot, František Grézl, Lukáš Burget, and Jan Honza Cernockỳ, "Analysis of DNN approaches to speaker identification," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5100–5104.

[2] Amirsina Torfi, Nasser M Nasrabadi, and Jeremy Dawson, "Text-Independent Speaker Verification Using 3D Convolutional Neural Networks," *arXiv preprint arXiv:1705.09422*, 2017.

[3] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally, "EIE: efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*. IEEE Press, 2016, pp. 243–254.

[4] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[5] Michael Price, James Glass, and Anantha P Chandrakasan, "14.4 A scalable speech recognizer with deep-neural-network acoustic models and voice-activated power gating," in *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*. IEEE, 2017, pp. 244–245.

[6] Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[7] Liang Lu, Michelle Guo, and Steve Renals, "Knowledge distillation for small-footprint highway networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*.

[8] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[9] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[10] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[12] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.

[13] Jinhwan Park and Wonyong Sung, "FPGA based implementation of deep neural networks using on-chip memory only," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1011–1015.

[14] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4052–4056.

[15] Yu-hsin Chen, Ignacio Lopez-Moreno, Tara N Sainath, Mirkó Visontai, Raziel Alvarez, and Carolina Parada, "Locally-connected and convolutional neural networks for small footprint speaker recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[16] Phak Kan, Tim Allen, and Steven Quigley, "A GMM-based speaker identification system on FPGA," *Reconfigurable Computing: Architectures, Tools and Applications*, pp. 358–363, 2010.

[17] Jingjiao Li, Dong An, Lili Lang, and Dan Yang, "Embedded speaker recognition system design and implementation based on FPGA," *Procedia Engineering*, vol. 29, pp. 2633–2637, 2012.

[18] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer, "End-to-end text-dependent speaker verification," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5115–5119.

[19] Gautam Bhattacharya, Jahangir Alam, Themos Stafylakis, and Patrick Kenny, "Deep Neural Network based Text-Dependent Speaker Recognition: Preliminary Results," .