# Energy-Efficient Speaker Identification with Low-Precision Networks

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th>Citation</th>
<th>Koppula, Skanda et al. “Energy-Efficient Speaker Identification with Low-Precision Networks.” 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), April 2018, Calgary, AB, Canada, Institute of Electrical and Electronics Engineers (IEEE), September 2018. © 2018 IEEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1109/icassp.2018.8462498">http://dx.doi.org/10.1109/icassp.2018.8462498</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Institute of Electrical and Electronics Engineers (IEEE)</td>
</tr>
<tr>
<td>Version</td>
<td>Author’s final manuscript</td>
</tr>
<tr>
<td>Citable link</td>
<td><a href="https://hdl.handle.net/1721.1/121168">https://hdl.handle.net/1721.1/121168</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Creative Commons Attribution-Noncommercial-Share Alike</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td><a href="http://creativecommons.org/licenses/by-nc-sa/4.0/">http://creativecommons.org/licenses/by-nc-sa/4.0/</a></td>
</tr>
</tbody>
</table>
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 a power-efficient speaker ID 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 accesses to the much smaller on-chip memory (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 of our speaker identification network on a 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 Sections 5 and 6, we describe results from our baseline and bitwidth-reduction experiments on SID networks. Section 7 describes our implementation of one of our low-precision networks in FPGA, where we report estimated 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 research efforts 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
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 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 the network's initialization and operational changes to decompose ternary networks into hardware-amiable binary operations.

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 \( \tilde{W}^l \) by bucketing \( W^l \)'s individual values, \( W^l_{ijk} \), based on a layer-specific threshold \( \Delta^l \):

\[
\tilde{W}^l_{ijk} = \begin{cases} 
K_1^l & \text{if } W^l_{ijk} \geq \Delta^l \\
0 & \text{if } -\Delta^l < W^l_{ijk} < \Delta^l \\
K_2^l & \text{if } W^l_{ijk} \leq -\Delta^l 
\end{cases}
\]

This formulation is the original TTQ construction from [8], which uses two separate scaling constants \( K_1^l \) and \( K_2^l \) for the top and bottom thresholding regimes. In this work, we explored the simplification \( K_2^l = -K_1^l \) (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. The backward pass is also slightly modified: the gradient updates are applied to both the \( K^L \) scaling values, and the original \( W^l \) shadow weights. The threshold \( \Delta^l \) 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-\( l \). At validation and test-time, to avoid a costly max-accumulation in hardware, we use a cached average value for \( \max|h^l| \) of the training set.

To avoid full-precision floating-point operations, we can optionally constrain intermediate activations to 32 or 16-bit fixed point. After the multiply-and-accumulates in every layer, we normalize to [-1,1] (either by way of dynamic re-scaling or batch normalization) and downscale the activations to fit within 16-bits via bit-shifting and bit truncation.

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 hardware design needs to only support additions and subtractions across each row. The 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 \( 12 \times 12 \) 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 \( w \times h \times c \) convolutional kernels, depth-separable kernels use a set of \( c \times w \times h \) intra-channel spatial filters followed by a set of \( c \times 1 \times 1 \) cross-channel filters to reduce multiplications.

[2]. In our work, we introduce two depth-separable convolutional (DSC) layers (Figure 1). Note that since our initial input is single-channel (50 \( \times \) 20 stacked), only the second and third layers benefit from depth separation. Using depth-wise separable convolution attempts, we hoped 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) to establish our text-dependent closed-set speaker identification benchmarks. The reported equal error rates (EERs) are the aver-
We test each of these architectures after training the models to use ternary weights and 32-bit fixed-point activations and input. 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 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 decomposition 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) from \( T^l \times x^l \) [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 by 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 ±3 EER between utterance types.

6. PERFORMANCE OF LOW-BITWIDTH SID

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. All full-precision architectures achieved less than 2% EER on our closed-speaker set, text-dependent task, matching our expectations from prior work. We were able to obtain consistent results (within 1% 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 accesses, contributing to the similar energy costs.

<table>
<thead>
<tr>
<th>Arch</th>
<th>Size</th>
<th>Bin Op</th>
<th>Mult</th>
<th>EER</th>
<th>E, ( \mu J )</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN</td>
<td>14.52MB</td>
<td>452K</td>
<td>0.652</td>
<td>2325.6</td>
<td></td>
</tr>
<tr>
<td>LCN</td>
<td>14.39MB</td>
<td>448K</td>
<td>0.931</td>
<td>2304.6</td>
<td></td>
</tr>
<tr>
<td>CNN</td>
<td>14.58MB</td>
<td>2756K</td>
<td>0.838</td>
<td>2345.8</td>
<td></td>
</tr>
<tr>
<td>Maxout</td>
<td>14.56MB</td>
<td>453K</td>
<td>1.305</td>
<td>2329.3</td>
<td></td>
</tr>
<tr>
<td>DSC</td>
<td>14.51MB</td>
<td>483K</td>
<td>0.393</td>
<td>2323.9</td>
<td></td>
</tr>
</tbody>
</table>

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 (\( \mu J \)).

Without the DRAM accesses refresh the on-chip cache, our energy costs reduce significantly reduce. In all our models, error rates increased compared to their full-precision counterparts. The standard FCN/CNN/LCN models perform the best relative to their full-precision counterparts.
The ternary LCN/DSC architectures were unable to learn (diverging error [DIV], or converging to very high error). Analysis into why these network diverged was unsuccessful, but we noticed extremely fast convergence to unusually small weights and gradients. Various fixed, including adjusting the initializers, learning rates, and optimizers 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. This form of weight initialization was used in the original TTQ manuscript, and our experiments suggest this bootstrapping is not always effective. Instead, we initialized from other ternary networks, which did not exhibit this problem. Six example TTQ kernels are shown in Figure 2. While the sparsity in the bottom two rows of kernels is roughly the same (55% and 54%), the center TTQ CNN 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 pre-loading 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 an output-stationary approach: output accumulation occurs in each PU’s local register, as input and weights are fed in the subsequent cycles. Figure 3 describes the connectivity of these different modules.

The architecture we use is based on the output-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 $256 \times 1000$ 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 the recent depthwise-separable 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. While running counter to prior work, 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.
REFERENCES


