J Sign Process Syst
DOI 10.1007/s11265-016-1202-x

A Fixed-Point Neural Network Architecture for Speech Applications on Resource Constrained Hardware

Mohit Shah[1] · Sairam Arunachalam[3] · Jingcheng Wang[2] · David Blaauw[2] · Dennis Sylvester[2] · Hun-Seok Kim[2] · Jae-sun Seo[1] · Chaitali Chakrabarti[1]

Received: 23 February 2016 / Revised: 15 September 2016 / Accepted: 8 November 2016

Abstract Speech recognition and keyword detection are becoming increasingly popular applications for mobile systems. These applications have large memory and compute resource requirements, making their implementation on a mobile device quite challenging. In this paper, we design low cost neural network architectures for keyword detection and speech recognition. Wepresent techniques to reduce memory requirement by scaling down the precision of weight and biases without compromising on the detection/recognition performance. Experiments conducted on the Resource Management (RM) database show that for the keyword detection neural network, representing the weights by 5 bits results in a 6 fold reduction in memory compared to a floating point implementation with very little loss in performance. Similarly, for the speech recognition neural network, representing the weights by 6 bits results in a 5 fold reduction in memory while maintaining an error rate similar to a floating point implementation. Preliminary results in 40nm TSMC technology show that the networks have fairly small power consumption: 11.12mW for the keyword detection network and 51.96mW for the speech recognition network, making these designs suitable for mobile devices.

Keywords Speech recognition · Keyword detection · Deep neural networks · Memory compression · Fixed-point architecture

✉ Sairam Arunachalam
sarunac4@asu.edu

Mohit Shah
mohit.shah@asu.edu

Jingcheng Wang
jiwang@umich.edu

David Blaauw
blaauw@umich.edu

Dennis Sylvester
dennis@umich.edu

Hun-Seok Kim
hunseok@umich.edu

Jae-sun Seo
jaesun.seo@asu.edu

Chaitali Chakrabarti
chaitali@asu.edu

[1] School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287, USA

[2] Department of Electrical Engineeering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA

[3] School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287, USA

1 Introduction

Automatic Speech Recognition (ASR) refers to the task of converting speech/audio input to text. Such systems are currently used in personal assistant systems like Siri, Google Now, etc. Keyword detection refers to the task of detecting specific keywords embedded in speech. The keyword detection system can be used as a front-end recognition system to trigger an ASR in case a specific keyword is detected. In this case, keyword detection engine runs as a background service in mobile environment. Since it is always 'on', its power consumption has to be very small. Compared to the keyword detection engine, the automatic speech recognition system has high memory and compute resource requirements. Implementing such a system in a mobile or resource

Published online: 25 November 2016

constrained environment is very challenging. Since memory power consumption is dominant, there is a strong need to develop an architectural framework for such applications with the goal of reducing the memory size and thereby lowering power consumption.

There is a vast amount of literature for both speech recognition and keyword detection systems. For speech recognition, the usual process involves using a Hidden Markov Model (HMM) for modeling the sequence of words /phonemes and using a Gaussian Mixture Model (GMM) for acoustic modeling [1, 2]. The most likely sequence can be determined from the HMMs by employing the Viterbi algorithm. The GMMs can be implemented in a parallel fashion, but the Viterbi algorithm is inherently sequential.

Keyword detection falls under speech recognition and is relatively less complex. There are different approaches for detecting a keyword. For instance, the speech recognition system can be used to perform keyword detection. In this approach, first speech recognition is performed and then keywords are detected from the decoded transcription [3–5]. The drawback for such a method is that it requires the entire phrase to be decoded before the keywords can be detected. The second method involves training separate models for keywords and out-of-vocabulary (OOV) words and detecting keywords based on the likelihood over each model. The OOV words are modeled using a garbage or a filler model, while a separate GMM-HMM is trained for each keyword [6–10]. Such a system is suitable in environments where the set of keywords is known beforehand.

Recently, neural network (NN) based methods have shown tremendous success in speech recognition tasks. This success has come after advances made in the field of deep learning [11, 12]. These networks are well-suited to capture the complex, non-linear patterns from the acoustic properties of speech. Detection is again straightforward; a matrix-vector multiplication step followed by a non-linear operation at each layer, making it highly suitable for parallel implementations. One such implementation was presented in [12] for keyword detection. While the detection performance was very good, the network was quite large, requiring upto a few million multiplications every few milliseconds as well as large memory banks for storing the weights. The network for a typical speech recognition system is a lot larger and a straightforward implementation requires huge memory for storing the weights and large number of compute resources. An application specific integrated circuit (ASIC) for ASR that consumes 144 mW was presented in [13]. This ASIC was specifically designed for Gaussian Mixture Model (GMM) computation; it did not include pre- and post-processing modules such as feature extraction and Viterbi search. Another 5000 word ASR architecture that utilizes GMM and WFST models was proposed in [14]. Its power consumption is very low at 6mW, though this number does not include the power consumed to access data from off-chip memory.

In this paper we present neural network architectures for keyword detection and speech recognition that have low hardware cost. This work is an extension[1] of the keyword detection work that was presented in [15]. Specifically, here we present techniques to reduce the memory requirement and compute cost through systematic scaling down the precision of the weights so that the performance quality is minimally affected. Additional reduction in memory can be obtained by utilizing the sensitivity of the weights in determining the precision. These techniques were evaluated by conducting experiments on the Resource Management (RM) database [16] and also by hardware implementations in TSMC 40nm technology. This paper makes the following contributions.

- Design of fixed point neural network architecture with 2 hidden layers and 400 neurons per layer for detecting 10 keywords. The weights and biases are represented by 5 bits resulting in memory requirement reducing from 1.25MB to 200KB, while maintaining a detection performance of 0.928 AUC (which is the area under the receiver operator characteristics).
- Design of fixed point neural network architecture with 4 hidden layers and 1024 neurons per layer for speech recognition. The weights and biases are represented by 6 bits resulting in memory requirement reducing from 19.53MB to 3.66MB, while maintaining a word error rate of 1.77 %.
- Hardware implementations of the keyword detection and speech recognition networks. Preliminary results in TSMC 40nm technology show that the power consumption of the two networks are only 11.12mW and 51.96mW, respectively, making them suitable for mobile devices.

The remainder of the paper is organized as follows. An overview of the proposed approach and neural network design are presented in Section 2. The fixed point implementations of both networks are described in Section 3. The performance of the two networks are presented in Section 4. The hardware implementation details of the

[1]The extensions to the SiPS'15 paper include (i) development of a fixed-point neural network architecture for speech recognition that has significantly reduced memory requirements compared to a floating point implementation, (ii) evaluation of the speech recognition network on RM database, (iii) hardware implementation of the speech recognition network and (iv) a new technique to reduce the precision of the weights through sensitivity analysis.

neural network architectures are described in Section 5, and finally, conclusions are presented in Section 6.

2 Neural Network Design

In this section, we describe the various steps involved in building the multi-layer neural networks for keyword detection and speech recognition systems. First, we pre-process the raw speech data (Section 2.1) and extract meaningful features that are used by the neural network for classification (Section 2.2). We train the network based on a heuristic algorithm to find the optimum weights and biases (Section 2.3). The network outputs are further processed before the final detection or recognition steps (Section 2.4).

2.1 Pre-Processing

In order to derive the networks for keyword detection and speech recognition, we use the Resource Management (RM) database [16] that consists of phrases recorded for scenarios pertaining to the naval forces. Ten keywords are chosen for this work: *ships*, *list*, *chart*, *display*, *fuel*, *show*, *track*, *submarine*, *latitude* and *longitude*. Speech is processed at a frame rate of 100 frames/second, a window size of 25ms and a step size of 10ms.

For the keyword detection network, the first 13 Mel frequency coefficients (MFCC) are extracted for each frame. These features are augmented with MFCCs of the 15 previous frames and 15 future frames to form a 403-dimension feature vector per frame. These 31 frames with 13 MFCCs/frame correspond to 310ms of speech. Since the average word duration for this database is 300ms, this choice is deemed to be appropriate for modeling words or sub-word units. Forced alignment is done using the Kaldi-toolkit [17] to obtain the word boundaries. Then each frame is labeled as one of the three categories: a particular keyword, Out Of Vocabulary (OOV) in case the keyword was not in the list above, or silence. The speaker-independent training and testing data are already specified with the database. Also, there are 109 and 59 speakers in the training and testing set, respectively. The speech features are z-normalized to zero mean and unit variance for each speaker.

For the speech recognition system, we use fMLLR features (feature-space Maximum Likelihood Linear Regression) extracted from the MFCC features of 7 frames (3 past, 1 current and 3 future frames) following the method given in [18]. Specifically, the 13x7 = 91 MFCC features are reduced to 40 features by de-correlation and dimensionality reduction using Linear Discriminant Analysis (LDA). These features are further de-correlated using the Maximum Likelihood Linear Transform (MLLT). The 40 fMLLR features over 11 frames (5 past, 1 current and 5 future frames) is used to construct the final feature vector consisting of 440 fMLLR features per frame. The initial speech recognition system is trained using the traditional GMM-HMM method, such that the GMM method is used to estimate the probability of various HMM states of the triphone model. This training is done using the scripts provided in the Kaldi-toolkit [17] for the RM database [16]. This GMM-HMM model is used as a baseline to train the neural network.

2.2 Neural Network

The feed-forward neural network designed for keyword detection is shown in Fig. 1. For the keyword detection system, the network consists of two hidden layers with 400 neurons per layer [15]. The input nodes are 403 MFCC features, which corresponds to 31 frames with 13 MFCCs

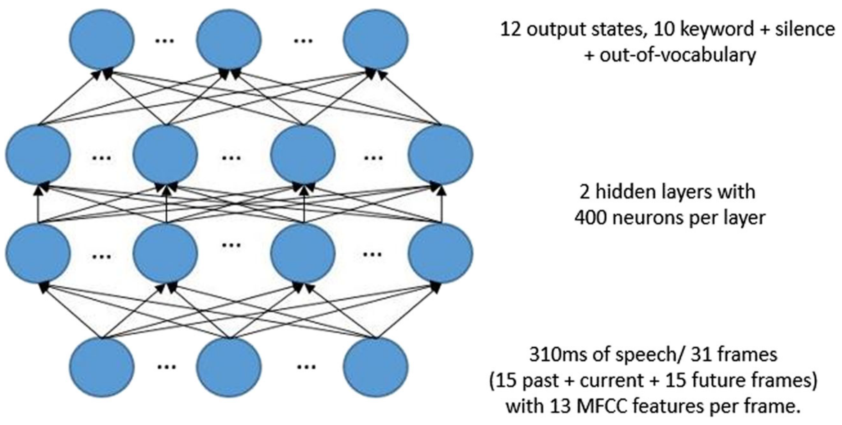

12 output states, 10 keyword + silence + out-of-vocabulary
2 hidden layers with 400 neurons per layer
310ms of speech/ 31 frames (15 past + current + 15 future frames) with 13 MFCC features per frame.

Figure 1 Neural network architecture for keyword detection consists of 2 hidden layers with 400 neurons per layer. The input feature dimension is 403 and the output dimension is 12 (10 keywords, 1 OOV and 1 silence).

per frame. The output layer consists of 12 nodes, where 10 nodes for the 10 keywords, 1 node is for OOV words and 1 node is for silence.

The neural network designed for speech recognition system is shown in Fig. 2, which consists of 4 hidden layers with 1024 neurons per layer. 440 fMLLR features are used as input nodes, and the output layer consists of 1483 HMM states for the phonemes.

The feed-forward computations in the two networks are as follows. Let the input layer be denoted as x_i, where $i = 1, 2 \ldots, N$, is the number of input features. The computation in the first hidden layer h^1 is given in Eq. 1.

$$z_j^1 = \sum_{i=1}^{N} W_{ij}^1 x_i + b_j^1 \tag{1}$$

Here W_{ij}^1 is the weight of the connection between ith node in the input layer and jth node in the current layer and b_j^1 the bias term at the jth node. The computations in all other hidden layers are identical.

To produce the neuron output at each hidden layer, we employ the ReLU (Rectified Linear Unit) activation as described in Eq. 2. The ReLU activation is a simple straightforward function and is easy to implement as it requires only a comparison operation.

$$h_j^1 = \max(0, z_j^1) \tag{2}$$

Finally, the output of the last hidden layer is modeled as a softmax layer that predicts the posterior probabilities of each of the output states. The softmax function described in Eq. 3 is used to estimate the probability of each output state, where N is the number of output states.

$$o_i = \frac{e^{z_i}}{\sum_{n=1}^{N} e^{z_i}} \tag{3}$$

2.3 Training the Neural Networks

Training the neural network is performed by minimizing the cross-entropy error, as described in Eq. 4 [19].

$$E = -\sum_{i=1}^{N} t_i * ln(y_i) \tag{4}$$

Here t_i is the ith target value, y_i is the ith output and N is the size of the output layer.

The keyword detection network is a smaller network with 2 hidden layers and 400 neurons per layer. Here the network weights are randomly initialized and training is done using the back-propagation algorithm. The stochastic gradient descent is used with 500 samples with a learning rate of 0.001 and a momentum of 0.8 for 6 epochs.

For the larger speech recognition neural network system with 4 hidden layers and 1024 neurons per layer, the weights are randomly initialized. This network is then fine-tuned using the back propagation algorithm with a learning rate of 0.008, a momentum of 0.5 with sample size of 256. Here the training is performed using the ’new bob’ approach. In this technique the training is is scaled by a factor of 0.5 for the remaining epochs once the validation error between two epochs is less than 0.2 % and is terminated when the validation error difference between two epochs drops below

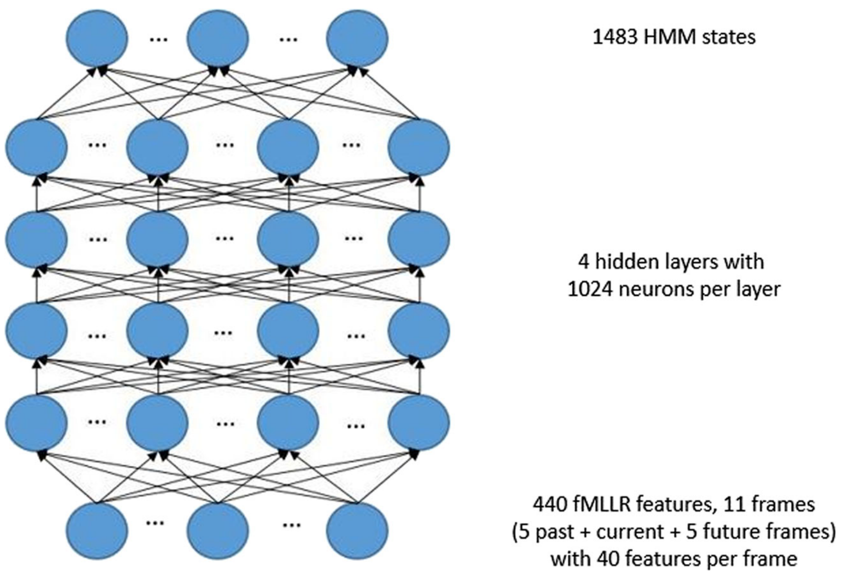

1483 HMM states
4 hidden layers with 1024 neurons per layer
440 fMLLR features, 11 frames (5 past + current + 5 future frames) with 40 features per frame

Figure 2 Neural network architecture for the speech recognition system consists of four hidden layers with 1024 neurons per layer. The input feature dimension is 440 and the output layer represents the posterior probability of the 1483 HMM states.

0.2 %. The neural network training is done using the PDNN toolkit [20].

2.4 Post-Processing

2.4.1 Keyword Detection

The output layer returns a posterior probability estimate for each frame every 10ms. To reduce the inherent noise in estimation, the estimates are smoothed using a symmetrical moving average window of W frames centered around the current frame. This helps eliminate noisy bursts and reduce the false alarm rate. The smoother estimate $\hat{y}_j$ is given by

$$\hat{y}_j = \frac{1}{W} \sum_{i=j-(W-1)/2}^{j+(W-1)/2} y_i \tag{5}$$

where, y_i is the probability estimate obtained from the final softmax layer.

The window size W is varied from 10 to 50 frames (100ms to 500ms) in our experiments. An example highlighting the effect of smoothing is shown in Fig. 3. It can be seen that the frame-wise probabilities returned at the output are quite noisy and smoothing suppresses the noise by combining estimates from the past and the future. For our keyword detection neural network, the optimal W is found to be 50.

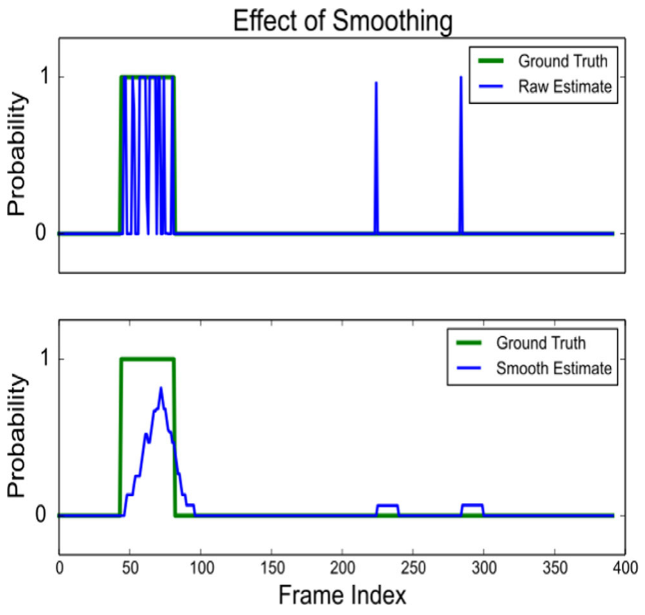


Figure 3 An example of the smoothing process for detection of *ships* in the phrase *do any ships that are in pacific fleet have SPS-48*. (*Top*) Frame-wise, raw posterior probability estimates, and (*Bottom*) Frame-wise, smoothed probability estimates. The window size is $W = 15$.

While this method helps reduce the noisy bursts/spikes and the false alarm rate, to obtain phrase-level decision, an additional post-processing step is applied over the smoothed estimates. Using a sliding window of size C frames, if the average probability estimate within this window exceeds a certain threshold, then a keyword is said to be present in the phrase. The window size C is dependent on the expected duration of a keyword and is varied from 10 to 30 (100ms to 300ms) in our experiments. The optimal size of C is found to be 25.

2.4.2 Speech Recognition

The posterior probabilities of the frames are obtained from the output layer of the neural network. These probability estimates are divided by the prior probabilities that were obtained for each state from the baseline GMM-HMM model. The scaled estimates are then fed to the Viterbi algorithm to determine the best sequence of phonemes. These phonemes are then used to transcribe the words and sentences for the particular input sequence.

3 Fixed Point Neural Network Architecture

The neural networks for both keyword detection and speech recognition are trained using floating point numbers. While this ensures faster and better convergence of the weights, a floating point implementation is expensive both in terms of memory and compute resources. In order to obtain a fixed-point implementation for the weights, we first compute a histogram of the weights for each layer and then use a simple linear scheme to quantize the values. Next, we use the sensitivity of the weights to further lower the precision. Throughout the paper, we denote the fixed point representation using a Q$A.B$ format, where A denotes the integer precision and B denotes the fractional precision. Unless otherwise mentioned, an additional sign bit is always assumed.

3.1 Determining Fixed Point Weights

As mentioned earlier, the histogram of the weights in each layer is used to determine the precision of the weights. The weight histograms of both neural networks are normally distributed, and thus we can use either linear or non-linear quantization schemes. In this work, we use a simple linear quantization scheme owing to its simplicity and generalizability. For keyword detection, such a scheme results in weights being represented by Q2.2; the corresponding memory footprint is 6 times lower than a 32 bit floating point representation as will be shown in Section 4.1.

In order to further reduce the required memory, we classify the weights as either sensitive or insensitive. This enables us to represent weights that have a larger affect on the outcome of the network, namely the sensitive ones, with more bits. This method is derived from the work done in [21], where each node in a particular layer was approximated based on energy considerations. Instead of approximating each node as done in [21], we propose to approximate the weights that aids reducing the memory footprint.

To derive such an architecture, we need to first identify which connections are sensitive. We achieve weight approximation by piggy-backing on the well known back propagation algorithm which calculates the gradients at different levels of the network by back tracking from the output. We train the network by minimizing the cross-entropy error as described in Section 2.3. Once the trained network is obtained, we once again use the back propagation algorithm on a sample of M inputs to reduce bias and compensate for bad predictions during classification. In our experiments, we choose the value of M to be 100,000. The errors are used to determine the sensitivity of the connection. In a specific layer, if the error for a particular weight/bias is greater than the threshold, we classify the weight as sensitive and use more bits to represent the value; if the error is below a threshold we deem the weight to be insensitive and we use fewer bits to represent its value. We sweep the threshold values and determine the value that results in minimal loss in quality, the results of which are shown in Section 4. No retraining is necessary here as the error produced by incremental retraining will not be high enough to update the already approximated weights.

3.2 Fixed Point Computations

The input and intermediate hidden outputs are also stored in fixed-point format to reduce the accumulator size during the multiplication process. To determine the precision of the output of the nodes where the ReLU activation is used, we keep track of the maximum and minimum values produced in a particular layer over the entire test data set. This helps to determine the integer portion of the precision. To determine the fractional precision, we evaluate AUC, the area under the receiver operator characteristic curve [22] for different values of fractional precision and choose the precision for which the reduction in AUC (compared to the floating point implementation) is minimal.

3.3 Node Pruning

In the keyword detection network, there may be certain nodes in the hidden layers that are rarely active (i.e. the output is zero). Such nodes can be pruned thereby reducing memory and computations. To achieve this, we evaluate the network during the training phase and calculate the number of neurons that give an output of 0. This count is used to generate a probability estimate for each node. Using a threshold value of $t \in (0, 1)$, we remove nodes that have probability greater than t. For the keyword detection network, we found that in the first hidden layer all nodes are equally informative and node pruning is not helpful. On the other hand, for the second hidden layer, for $0.7 < t < 1$, we can prune nodes with only a marginal loss in performance [15].

4 Experimental Results

In this section, we describe the experimental results for the fixed point neural network architectures for keyword detection and speech recognition algorithms. In each case, the baseline corresponds to a floating point implementation of the identical neural networks, where all weights, biases and neuron outputs are represented in 32 bit floating point.

4.1 Keyword Detection

The receiver operator characteristics curve (ROC) of a keyword represents its true positive rate as a function of its false alarm rate. The area under the ROC curve, AUC, is the metric that is used to represent the detection quality [22]. Thus a larger value of AUC implies better detection quality (the maximum value is 1). In this paper, we take the average of the AUC values for each keyword and use the average AUC as a measure of the detection quality.

The ROC curve for floating point implementations with different number of neurons per hidden layer (256, 400 and 512 neurons) are shown in Fig. 4. We see that a hidden layer size of 400 neurons has better performance compared to the one with 256 neurons. Since increasing the number of nodes to 512 per layer has only a small gain in performance, we select the number of neurons to be 400 per hidden layer. This reduces not only the number of computations but also the memory requirements as fewer weights need to be stored.

In order to determine the fixed-point precision of the weights and biases, we first derive their histograms. Figure 5 shows the histograms of the weights and biases in different layers. It can be seen that all the weights and biases lie in the range $w \in (-4, 4)$. From this we infer that we need 2

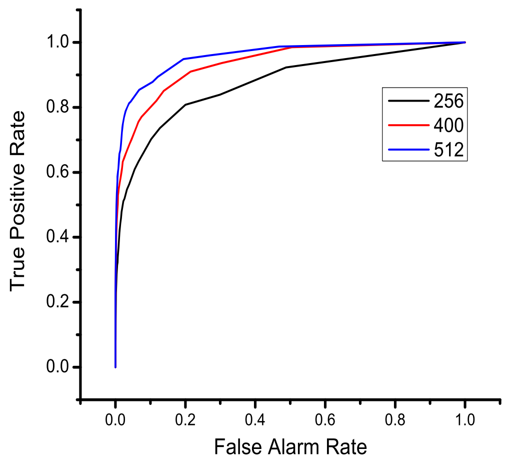


Figure 4 A comparison of the average ROC across all keywords between networks with different number of node per hidden layer.

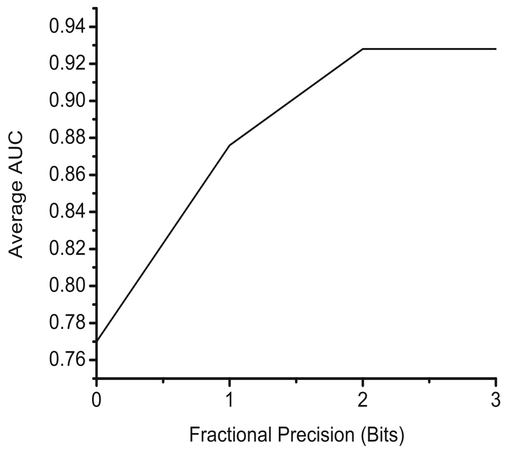


Figure 6 Effect of fractional precision of weights and biases on the average AUC for keyword detection network.

integer bits to represent the weights and biases. We also have a sign bit to distinguish between negative and positive values. To find the fractional precision of weights and biases, we sweep the average AUC for various precision levels in the range $B \in \{0, 1, 2, 3\}$. The results of this experiment are shown in Fig. 6. We see that a fractional precision of 2 bits provides a quality of detection comparable to the floating point implementation. Increase in fractional precision above 2 bits provides negligible gain in performance and decrease in the precision to 1 bit has a drastic impact on the AUC metric. Therefore we choose 2 fractional bits and Q2.2 format for representing weights and biases.

To further reduce the memory footprint, we approximate weights based on their sensitivity as mentioned in Section 3.1. An important parameter is the threshold that enables us to approximate the weights, without impacting the quality of the system. The results of this experiment are shown in Fig. 7. For a threshold of 0.4, the corresponding AUC is 0.895 compared to 0.934 in the floating point implementation. At this threshold, we can categorize ∼30 % of

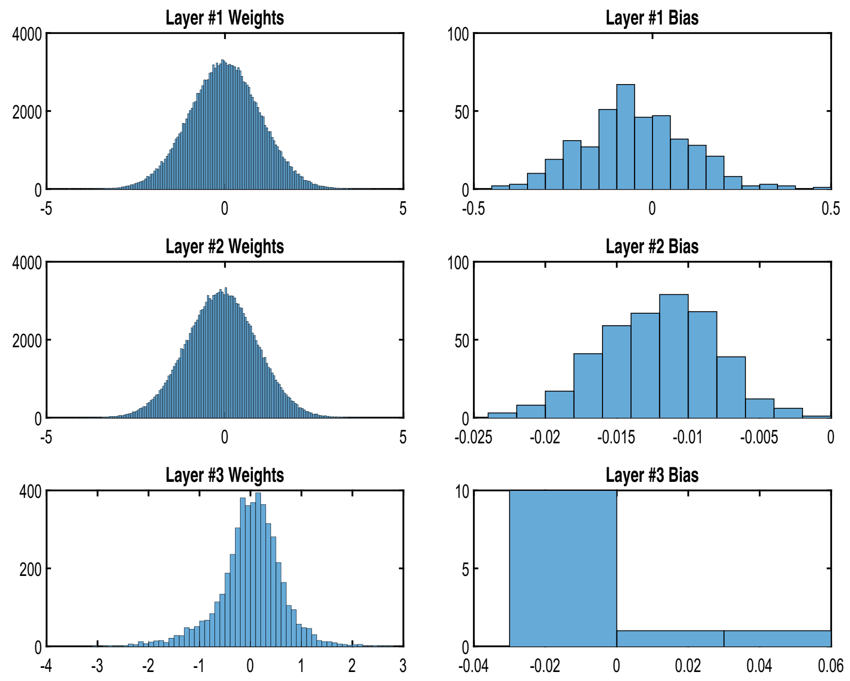


Figure 5 Histogram of weights and biases for keyword detection neural network.

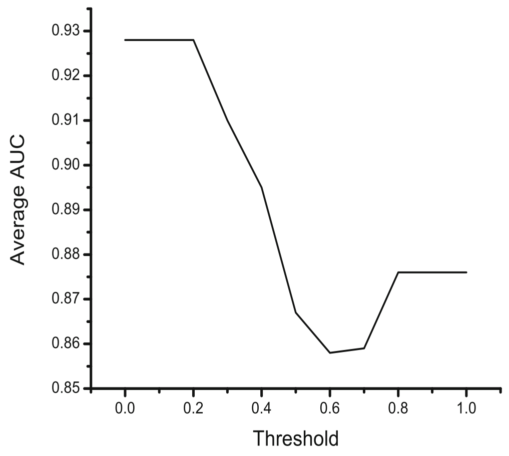


Figure 7 Effect of threshold on AUC for keyword detection neural network.

the weights as insensitive weights that can have a lower precision level. In particular, sensitive weights are represented using higher precision at Q2.2 and insensitive weights are represented using Q2.1. The results of the various architectures are shown in Table 1. We see that if all weights are represented by Q2.2, the memory requirement reduces from 1.25MB to 200KB (a 6 fold reduction). On further approximation by using the weight pruning algorithm, the memory is reduced to 171KB, which is another 15 % reduction from the Q2.2 implementation.

The inputs (MFCC coefficients) are represented by Q2.13 (16 bits) and the hidden node outputs are represented by Q12.10 (22 bits). The dynamic range of the outputs of the hidden nodes is used to determine the number of integer bits, which is 12. Since the activation function is ReLU, the output is always positive and there is no sign bit. To obtain the number of fractional bits, we vary the fractional precision to determine the variation in the AUC of the system. Then we choose the fractional precision that results in an AUC comparable to that of the floating point system. Based on these experiments, the fractional precision is chosen as 10 bits.

4.2 Speech Recognition

To evaluate different speech recognition implementations, we use the Word Error Rate (WER)[23]. WER is derived from Levenshtien distance, which works at the word level instead of the phoneme level. It is measured as

$$WER = \frac{S + D + I}{N} \tag{6}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions and N is the number of reference words. In the RM database, the test portion contains 1460 sentences. While the error rates are for the whole system (neural network + HMM), the analysis here focuses only on the neural network part.

In order to determine the precision of the weights and biases, we first derive the distribution of weights and biases of each layer of the speech recognition neural network. Figure 8 shows the distributions in each of the 5 layers. The weights and biases form a normal distribution centered at 0. The values of the weights and biases are in the range $w \in (-1, 1)$. So no integer bits are required to represent these values.

The histograms of the weights and biases in Fig. 8 show that most of the values are near 0. So the fractional part of the weights is what determines the recognition performance. Figure 9 shows the effect of the precision. Both SER and WER drop substantially when changing the precision from 7 bits to 4 bits. This confirms that a 5 bit fractional part is required for the system to perform as well as the baseline floating point implementation. Thus we choose to represent the weights by Q0.5 (6 bits). This results in the weight memory to be of size 3.66MB compared to 19.53MB when the weights were represented in 32 bit floating point. Table 2 presents the WER and memory requirements of these architectures.

In order to further reduce the memory footprint, a threshold is used to decide whether a weight is sensitive or insensitive, as described in Section 3.1. Figure 10 shows the effect of the threshold on the WER of the speech recognition system. The error rates increase substantially if the weight pruning threshold is greater than 0.4 and thus we choose the threshold to be 0.4. For this, the insensitive weights are fixed to a precision of Q0.5 and sensitive ones are fixed to a precision of Q0.4. The corresponding memory requirement is 3.63MB, which is 0.82 % lower than that of the Q0.5 implementation.

The inputs (fMLLR features) are represented by Q4.11 (16 bits) and the hidden nodes are represented by Q10.5 (15

Table 1 AUC and memory requirements of floating and fixed point implementations for keyword detection network.

Architecture	AUC	Memory
Floating Point	0.934	1.25MB
Fixed Point Q.2.2	0.928	200KB
Weight Pruning	0.895	171KB

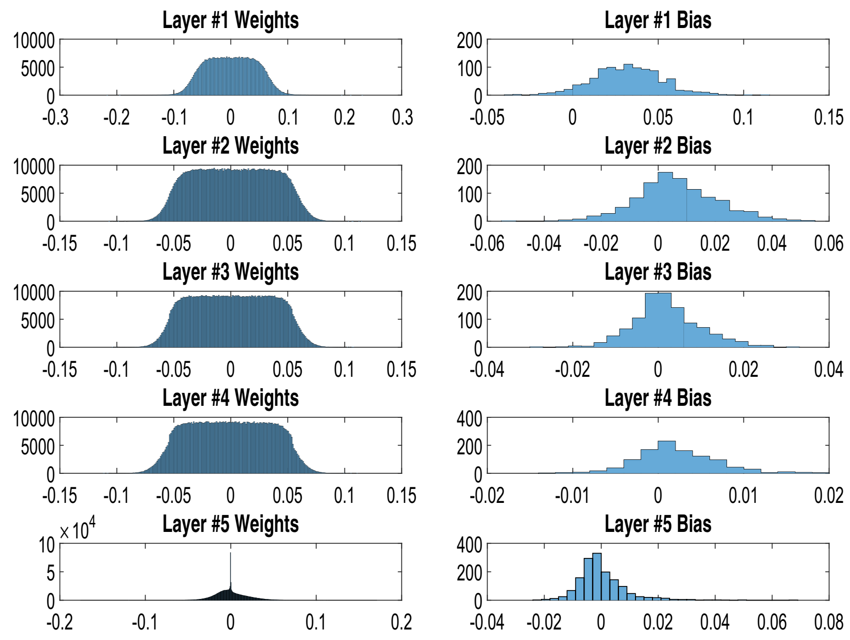


Figure 8 Histogram of weights and biases for speech recognition neural network.

bits). For this precision, the error rates are almost the same as that of the floating point implementation.

5 Hardware Architecture

5.1 Neural Network Coprocessor Architecture

Most of the recently proposed neural network hardware accelerators are designed for large networks and target very high performance [24, 25]. However, keyword detection, when used as a front-end trigger in mobile devices, is required to be always on and thus has a very tight power budget. Even high computational applications such as speech recognition, have a power budget of less than 100mW.

Figure 11 shows the top-level architecture for keyword detection. It consists of a neural network (NN) co-processor that is connected to a microprocessor system through a bus interface. Such an organization allows for the control intensive pre- and post-processing to be performed on the general purpose micro-processor, while the computation and memory-access intensive neural network processing is offloaded to the co-processor. The hardware architecture of

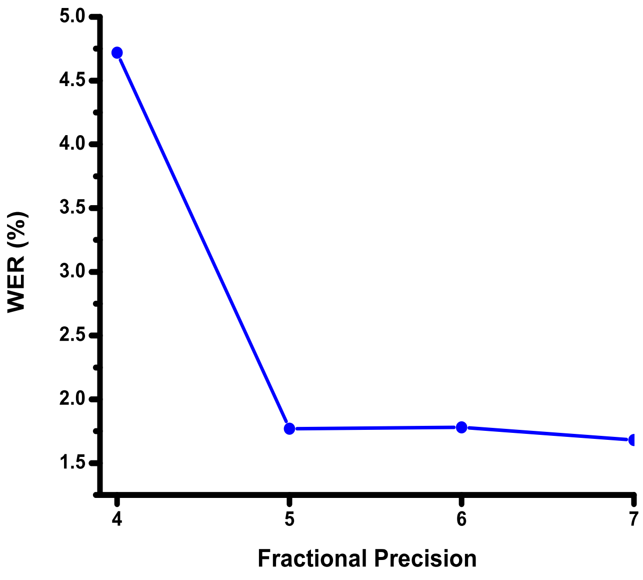


Figure 9 Effect of fractional precision of weights on WER for speech recognition network.

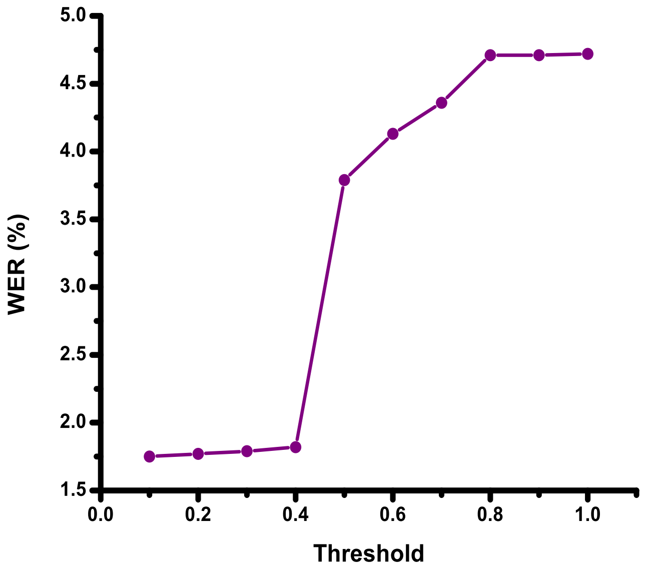


Figure 10 Effect of weight pruning threshold on WER for speech recognition network.

Table 2 WER and memory requirements of floating and fixed-point implementations for speech recognition network.

Architecture	WER	Memory
Floating Point	1.65%	19.53MB
Fixed Point (Q2.3)	1.77%	3.66 MB
Weight Pruning	1.81%	3.63 MB

the co-processor consists of one Processing Element (PE), three register files, a central memory for weights and one finite state machine (FSM) scheduler. While the neural network hardware architectures in [26, 27] employ a group of PEs in the form of a systolic array, our design uses only one PE because of the relatively low throughput requirement of the application (47 million multiplications per second for the keyword detection application).

As shown in Fig. 12, a PE contains one multiply accumulate (MAC) unit, a rectified linear (ReLU) module, and a sigmoid module. The ReLU module is implemented by a comparator. The sigmoid function can be implemented using a Look-Up Table (LUT). Both ReLU and sigmoid are included since some DNNs may favor one over the other. Three register file based FIFOs are used to hold input and output neurons. Every 10ms, 13 new inputs are shifted into the 512-word FIFO serving as a circular input buffer. Two 512-word FIFOs store outputs from 2 hidden layers with 512 neurons each or 1 hidden layer with 1024 neurons when combined. Although the capacity of the FIFOs are designed to be maximum 512, users can configure the active FIFO depth according to the size of the neuron network. The scheduler has the capability to turn off and skip the unused FIFO registers at run time. This flexibility minimizes the energy overhead of using the 512 deep FIFOs for realizing smaller sized neural networks such as the keyword detection network.

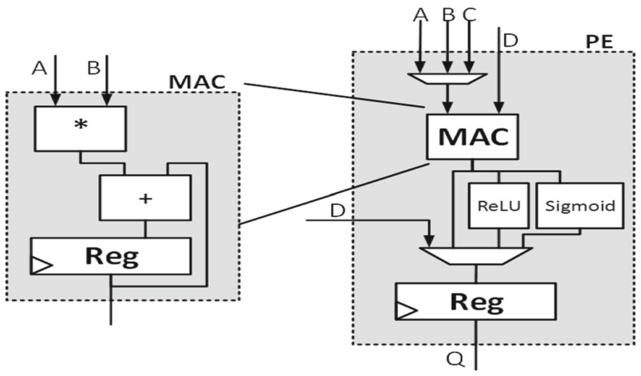


Figure 12 Internal architecture of the processing element (PE) and multiply-accumulate unit (MAC).

Network weights are stored in a 1MB memory block made by the 8T SRAM compiler. 8T SRAM cell is chosen over 6T because it is more reliable under lower supply voltages. To use the memory efficiently, six 5-bit weights are attached together to occupy one wordline in memory. Notice that we use a 1MB Synapsis Memory, which is larger than the 256KB memory used for the keyword detection architecture in [15]. This memory size is chosen so that both the keyword detection (the entire neural network) and a single layer of the speech recognition application can share the same co-processor architecture. Because of the relatively large memory requirement of the application, the chip space and power is dominated by this 1MB memory.

To meet the workload requirement of keyword detection application, a 50MHz clock frequency is used. Since a total of 474k MAC operations are required per classification, for a frame rate of 100 frames/second, 47.4 million MAC operations have to be done per second. In our architecture, MAC is a single cycle operation at least 47.4 MHz clock frequency is required, and so we round it and operate at 50 MHz frequency. The proposed accelerator minimizes power consumption by scaling down the supply voltage to

Figure 11 Top level architecture of the feedforward neural network for keyword detection.

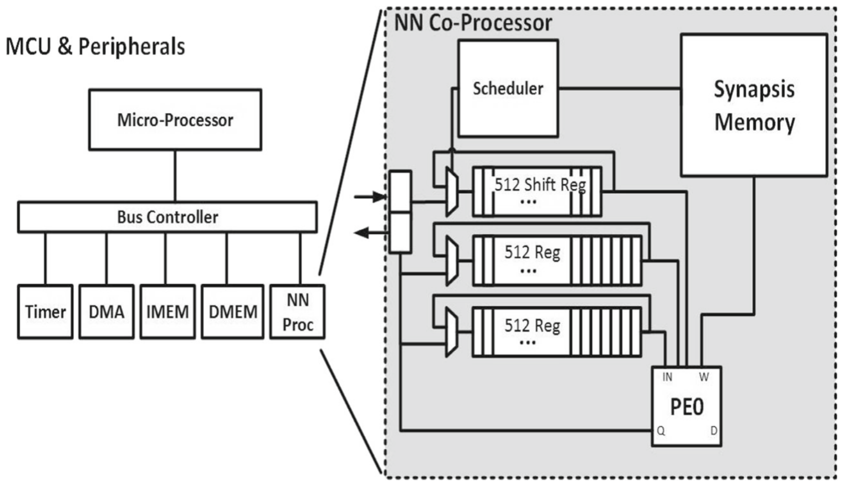


Table 3 Area and power estimation for keyword detection network.

	Area μm^2	Power mW
Scheduler	2,855	0.176
Register	108,870	0.45
PE	1,548	0.11
SRAM	3,096,215	10.56

Table 4 Area Comparison for keyword detection (Unit: μm^2).

	Q2.2 w/ pruning	Floating-point w/o pruning
Register	108,870	139,353
PE	1,548	6,211
SRAM	3,096,215	24,769,726
TOTAL	3,206,633	24,915,290

Table 5 Power Comparison for keyword detection(Unit: mW).

	Q2.2 w/ pruning	Floating-point w/o pruning
Register	0.45	0.6
PE	0.11	2.14
SRAM	10.56	76.4
TOTAL	11.12	79.14

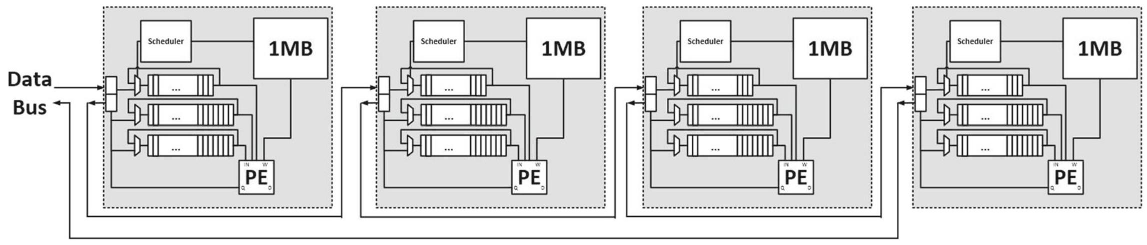


Figure 13 Top level architecture of the speech recognition neural network.

Table 6 Area and power estimation for speech recognition network.

	Area μm^2	Power mW
Scheduler	11,420	2.8
Register	435,480	7.2
PE	6,192	1.76
SRAM	12,384,863	40.2
Total	12,837,955	51.96

near threshold values (e.g. 0.5V-0.6V). Based on the synthesis results, the area and power estimation of each component at 0.6V is given in Table 3. SRAM consumes the biggest portion of the total power. Because of the low workload of the keyword detection application, only 2 % of the SRAM power is active power and the rest 98 % comes from leakage. In Tables 4 and 5, the current design (Q2.2 with pruning) is compared with floating point implementation without pruning. The results show that 5-bit fixed-point for weight representation and node pruning decisions are very effective in saving area and power.

5.2 Concatenated Neural Network Coprocessors for Speech Recognition

The computation demand of speech recognition is significantly higher than that of keyword detection, about 13X more multiplications. Our solution is (i) to concatenate four co-processors as shown in Fig. 13, and map each layer onto a co-processor, and (ii) clock each co-processor 4 times faster than when used for keyword spotting. Concatenation of four of these coprocessors in a daisy-chain structure enables them to work in a pipelined fashion with each coprocessor serving one layer in the 4-layer speech recognition neural network. The 1MB Synapsis Memory in a co-processor is sufficient to support a single layer of the speech recognition neural network.

The operations of the four co-processors is coordinated by one central controller. Each neural network co-processor communicates with another using its input/output bus ports. The final output softmax is sent back to CPU for HMM processing. The area and power estimation of speech recognition neural network is shown in Table 6. The SRAM power has a 4 fold increase due to the fact that ASR data storage is almost 4 times the memory capacity of a single co-processor. The power of Scheduler/Register/PE power in each co-processor increases 4 fold due to the increased clock frequency. However its contribution to the total power is quite small.

6 Conclusion

Fully connected, feed-forward, fixed point neural network architectures were presented for two key speech applications, namely, keyword detection and speech recognition. The neural network for the keyword detection system consists of 2 hidden layers, with 400 neurons per layer while the network for speech recognition is much larger with 4 hidden layers and 1024 neurons per layer. Techniques were developed to represent the weights and biases with minimum number of bits to reduce the memory footprint while minimally affecting the detection/recognition performance. For keyword detection, where 10 keywords were selected from the RM corpus, experimental results show that there is only a marginal loss in performance when the weights are stored in Q2.2 (5 bits) format. The total memory required in this case is approximately 200 KB (compared to 1.2MB if the weights were represented by 32 bit floating point), making it highly suitable for resource constrained hardware devices. On the larger speech recognition network, the memory reduction is even more significant. Here the memory size dropped from 19.53MB to 3.66MB when the weights are represented in Q0.5 (6 bits) format. An additional 1 %–15 % reduction can be obtained by representing insensitive weights by lower precision. Finally, a co-processor architecture using a single multiplier and a memory bank was used to implement the keyword detection network. Four of these coprocessors were chained to support pipelined processing for speech recognition. The area and power results in TSMC 40nm node demonstrate that the proposed networks are highly suitable for deployment in resource-constrained hardware platforms. As future work, to further reduce power consumption, we propose to exploit the memory access pattern and put some of the memory banks that are idle into a retention mode by lowering the VDD voltage (e.g. 0.3V-0.4V) of the cell array.

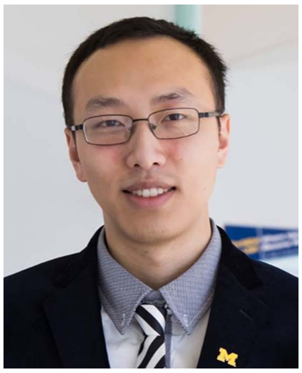
References

1. Su, D., Wu, X., & Xu, L. (2010). GMM-HMM acoustic model training by a two level procedure with gaussian components determined by automatic model selection. In *Proceedings of IEEE international conference on acoustics speech and signal processing* (pp. 4890–4893).
2. Sha, F., & Saul, L.K. (2006). Large margin Gaussian mixture modeling for phonetic classification and recognition. In *Proceedings of IEEE international conference on acoustics speech and signal processing* (pp. 265–268).
3. Miller, D.R., Kleber, M., Kao, C.-L., Kimball, O., Colthurst, T., Lowe, S.A., Schwartz, R.M., & Gish, H. (2007). Rapid and accurate spoken term detection. In *8th annual conference of the international speech communication association* (pp. 314–317).
4. Parlak, S., & Saraclar, M. (2008). Spoken term detection for Turkish broadcast news. In *Proceedings of IEEE international conference on acoustics, speech and signal processing* (pp. 5244–5247).
5. Mamou, J., Ramabhadran, B., & Siohan, O. (2007). Vocabulary independent spoken term detection. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 615–622).
6. Rohlicek, J.R., Russell, W., Roukos, S., & Gish, H. (1989). Continuous hidden Markov modeling for speaker-independent word spotting. In *Proceedings of IEEE international confernce on acoustics speech and signal processing* (pp. 627–630).
7. Rose, R.C., & Paul, D.B. (1990). A hidden Markov model based keyword recognition system. In *Proceedings of IEEE international conference on acoustics speech and signal processing* (pp. 129–132).

8. Wilpon, J., Miller, L., & Modi, P. (1991). Improvements and applications for key word recognition using hidden Markov modeling techniques. In *Proceedings of IEEE international conference on acoustics speech and signal processing* (pp. 309–312).
9. Silaghi, M.-C., & Bourlard, H. (1999). Iterative posterior-based keyword spotting without filler models. In *Proceedings of the IEEE automatic speech recognition and understanding workshop* (pp. 213–216).
10. Silaghi, M.-C. (2005). Spotting subsequences matching an HMM using the average observation probability criteria with application to keyword spotting. In *Proceedings of the national conference on artificial intelligence*, (Vol. 20 pp. 1118–1123).
11. Dahl, G.E., Yu, D., Deng, L., & Acero, A. (2011). Large vocabulary continuous speech recognition with context-dependent DBN-HMMs. In *Proceedings of IEEE international conference on acoustics speech and signal processing* (pp. 4688–4691).
12. Chen, G., Parada, C., & Heigold, G. (2014). Small-footprint keyword spotting using deep neural networks. In *Proceedings of IEEE international conference on acoustics speech and signal processing* (pp. 4087–4091).
13. He, G., Sugahara, T., Miyamoto, Y., Fujinaga, T., Noguchi, H., Izumi, S., Kawaguchi, H., & Yoshimoto, M. (2012). A 40 nm 144 mw VLSI processor for real-time 60-kword continuous speech recognition. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *59*(8), 1656–1666.
14. Price, M., Glass, J., & Chandrakasan, A. (2015). A 6mw 5,000 word real-time speech recognizer using WFST models. *IEEE Journal of Solid State Circuits*, *50*(1), 102–112.
15. Shah, M., Wang, J., Blaauw, D., Sylvester, D., Kim, H.-S., & Chakrabarti, C. (2015). A fixed-point neural network for keyword detection on resource constrained hardware. In *IEEE workshop on signal processing systems (SiPS)* (pp. 1–6).
16. Price, P., Fisher, W.M., Bernstein, J., & Pallett, D.S. (1988). The DARPA 1000-word resource management database for continuous speech recognition. In *Proceedings of IEEE international conference on acoustics speech and signal processing* (pp. 651–654).
17. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., & et al. (2011). The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. no. EPFL-CONF-192584.
18. Rath, S.P., Povey, D., Veselỳ, K., & Cernockỳ, J. (2013). Improved feature processing for deep neural networks. In *INTERSPEECH* (pp. 109–113).
19. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., & et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97.
20. Miao, Y. (2014). Kaldi+ PDNN: building DNN-based ASR systems with Kaldi and PDNN. arXiv:1401.6984.
21. Venkataramani, S., Ranjan, A., Roy, K., & Raghunathan, A. (2014). axNN: energy-efficient neuromorphic systems using approximate computing. In *Proceedings of the international symposium on low power electronics and design* (pp. 27–32).
22. Bradley, A.P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145–1159.
23. Morris, A.C., Maier, V., & Green, P.D. (2004). From WER and RIL to MER and WIl: improved evaluation measures for connected speech recognition. In *INTERSPEECH* (pp. 2765–2768).
24. Park, S., Bong, K., Shin, D., Lee, J., Choi, S., & Yoo, H.-j. (2015). a1.93TOPS/w scalable deep learning/inference processor with tetra-parallel mimd architecture for big-data applications. In *IEEE international solid-state circuits conference-(ISSCC) digest of technical papers* (pp. 1–3).
25. Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., & Temam, O. (2014). Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM Sigplan Notices*, *49*(4), 269–284.
26. Moons, B., & Verhelst, M. (2016). A 0.3-2.6 TOPS/W, precision-scalable processor for real-time large-scale conv nets, arXiv:1606.05094.
27. Chen, Y.-H., Krishna, T., Emer, J., & Sze, V. (2016). Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *IEEE international solid state circuits conference (ISSCC)* (pp. 262–264).

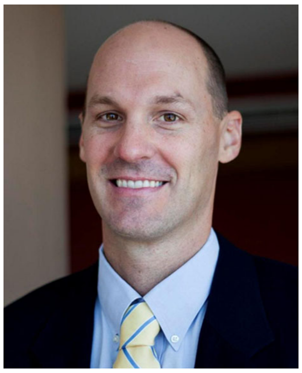
Mohit Shah completed his PhD in Electrical Engineering from Arizona State University in 2015. His primary research areas include speech-based emotion recognition, automatic speech recognition, natural language processing and machine learning. Currently, he works on developing computational methods for understanding low-resource Indian languages.

Sairam Arunchalam received the B.Tech degree in electrical and electronics engineering from the National Institute of Technology, Calicut, India in 2013 and the M.S. in Computer Engineering degree from Arizona State University in 2016. His research interest include deep learning, speech recognition and machine learning.



Jingcheng Wang (S'15) received the B.S. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2014, where he is currently working toward the M.S. and Ph.D. degrees in electrical engineering. He received a Dwight F. Benton Fellowship in 2015 from University of Michigan. His research interests are in high-speed and low-power VLSI circuits and systems.

David Blaauw received his B.S. in Physics and Computer Science from Duke University in 1986, and his Ph.D. in Computer Science from the University of Illinois, Urbana, in 1991. After his studies, he worked for Motorola, Inc. in Austin, TX, where he was the manager of the High Performance Design Technology group. Since August 2001, he has been on the faculty at the University of Michigan where he is a Professor. He has published over 500 papers and holds 50 patents. His work has focussed on VLSI design with including near-threshold and subthreshold design for ultra low power mm-scale sensor nodes. He was the Technical Program Chair and General Chair for the International Symposium on Low Power Electronic and Design. He was also the Technical Program Co-Chair of the ACM/IEEE Design Automation Conference and a member of the ISSCC Technical Program Committee. He is an IEEE Fellow.



Dennis Sylvester received a PhD from the University of California, Berkeley and is Professor of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. He has published over 450 articles along with one book and several book chapters, and holds 31 US patents. His research interests include the design of millimeter-scale computing systems and energy efficient near-threshold computing. He is co-founder of Ambiq Micro, a fabless semiconductor company developing ultra-low power mixed-signal solutions for compact wireless devices. He is an IEEE Fellow.

Hun Seok Kim (S'10 – M'11) received the B.S. degree in electrical engineering from Seoul National University, South Korea, and the M.S. and Ph.D. degrees in electrical engineering from the University of California at Los Angeles (UCLA), Los Angeles, CA, USA. He was a Technical Staff Member with Texas Instruments Inc. from 2010 to 2014, where he served as an industry liaison for university projects funded by the Semiconductor Research Cooperation and Texas Instruments Inc. He is currently an Assistant Professor at the University of Michigan, Ann Arbor, MI, USA. He holds nine granted patents and has eleven pending applications in the areas of digital communication, signal processing, and low- power integrated circuits. His research focuses on system novel algorithms and efficient VLSI architectures for low-power/high-performance signal processing, wireless communication, computer vision, and machine learning systems. He is a recipient of fellowships from the Ministry of Information and Telecommunication, South Korea, Seoul National University, and UCLA.

Jae-sun Seo received the B.S. degree from Seoul National University in 2001, and the M.S. and Ph.D. degree from the University of Michigan in 2006 and 2010, respectively, all in electrical engineering. He spent graduate research internships at Intel circuit research lab in 2006 and Sun Microsystems VLSI research group in 2008. From January 2010 to December 2013, he was with IBM T. J. Watson Research Center, where he worked on energy-efficient integrated circuits for high-performance processors and cognitive computing chips. In January 2014, he joined ASU as an assistant professor in the School of ECEE. During the summer of 2015, he was a visiting faculty at Intel Circuits Research Lab. His research interests include efficient hardware design of machine learning / neuromorphic algorithms and integrated power management. Mr. Seo was a recipient of Samsung Scholarship from 2004 to 2009, and received a IBM outstanding technical achievement award in 2012. He serves on the technical program committee for International Symposium on Low Power Electronics and Design (ISLPED) since 2013, and on the organizing committee for International Conference on Computer Design (ICCD) since 2015.

Chaitali Chakrabarti received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 1984, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, in 1986 and 1990, respectively. She is a Professor with the School of Electrical Computer and Energy Engineering, Arizona State University (ASU), Tempe, and a Fellow of the IEEE. Her research interests are in low power embedded systems design, and VLSI architectures and algorithms for signal processing and communications. She is currently an Associate Editor of the Journal of VLSI Signal Processing Systems and on the Senior Editorial Board of IEEE Journal on Emerging and Selected Topics in Circuits and Systems.