Hybrid Quantization for Vocal Commands Recognition on Microcontroller

Ivana Guarneri, Viviana D'Alto, Danilo Pau, Marco Lattuada
STMicroelectronics

Introduction

Automatic Speech Recognition (ASR) systems achieve today remarkable results thanks to the exploitation of high performing hardware and sophisticated deep neural network architectures.
Powerful ASR solutions need high memory storage capacities and high-power processing capabilities. The challenge is to deploy ASR complex algorithms in hardware platforms with limited resources.
This work proposes an AI-based vocal commands recognition system to be executed directly on a Microcontroller Unit (MCU).
The target hardware platform is the STM32L496G microcontroller running @80MHz with 1MB of FLASH and 320KB of RAM.
The aim of this work is to design a Deeply Quantized Neural Network (DQNN) by applying to a Deep Neural Network (DNN) different schemes of training aware quantization exploiting Google QKeras framework [1], thus evaluating the resulting memory and power saving versus the drop of accuracy.
The final system will execute the audio pre-processing step and the NN inference classification (Figure 1) on the MCU.

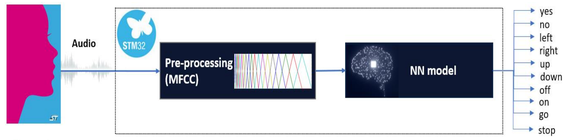

Figure 1

Dataset

The neural model was trained and tested on Google Speech Commands Dataset [2], which includes 30 different words for a total of 65.000 1-sec audio files.
Each word has got about 2000 samples. The dataset is released under Creative Commons BY 4.0 and it is constantly updated by the community. Words are pronounced by different speakers (both male and female subjects), using microphones placed at various distances.

Audio pre-processing

The neural network receives the Mel Filter Cepstral Coefficients (MFCC) as inputs. The pipeline used to extract the MFCC coefficients is showed in Figure 2.

Figure 2

The MFCC are classic audio features widely used in Speech Recognition processing, since they use a bank of Mel Filters, i.e., triangular filters able to mimic the non-linear human ear perception of the sound. Furthermore, at the end of MFCC computation a Discrete Cosine Transform removes the redundancy in the signal, representing it with fewer number of parameter than the original signal.

Audio buffer handling

The audio signal is sampled at 16KHz, it is buffered in 512 audio samples with an overlap of 256 samples. For each frame, the first 38 audio Mel Frequency Cepstral Coefficients are extracted and organized in a (38 x 55) matrix continuously updated with new columns added to the queue in FIFO (First In First Out) mode, as shown in Figure 3.

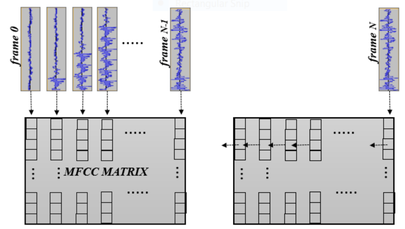

Figure 3

The MFCC matrix is the input of the Neural Network.
Before the training step, the audio features are normalized according to the equation (A) to have data with standard normal distribution.

$$x' = \frac{x-\mu}{\sigma} \qquad \text{(A)}$$

Where x is the original feature vector, μ and σ are the mean and the standard deviation of the training set respectively. Normalization is useful to speeds-up the convergence of the neural model during the training. The computed (μ, σ) are used to normalize the audio features also before the inference model execution.

Deep Neural Network architecture

The proposed deep neural network architecture recognizes 10 vocal commands taken from the Google dataset [2]. The network consists of six convolutional layers followed by three fully connected layers (see Figure 4).

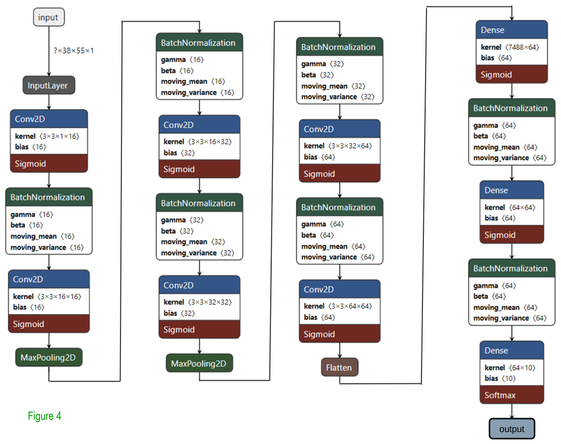

Figure 4

Keras model

At first the neural network was designed in floating point 32bits and trained using the Keras framework [3]. The floating-point model reached an accuracy of 94% on the test set, but the memory requirements (FLASH=2.12MB; RAM=149.20KB) exceed the memory available in the selected target platform. Figure 5 highlights that the dense layer (id = 14) impacts the most on FLASH requirements, as it uses the 86.2% of the total amount of memory required by the entire pipeline.

```
Complexity report per layer - macc=20,496,832 weights=2,225,384 act=144,384 ram_io=8,400
------------------------------------------------------------------------------------------
id   name                  c_macc                    c_rom                     c_id
------------------------------------------------------------------------------------------
0    conv2d                ||                 3.1%   |                  0.0%   [0]
1    batch_normalization   |                  0.3%   |                  0.0%   [1]
2    conv2d_1              ||||||||||||||||   25.3%   |                  0.4%   [2]
4    batch_normalization_1 |                  0.1%   |                  0.0%   [3]
5    conv2d_2              ||||||||           12.3%   |                  0.8%   [4]
6    batch_normalization_2 |                  0.2%   |                  0.0%   [5]
7    conv2d_3              ||||||||||||||||   23.9%   |                  1.7%   [6]
9    batch_normalization_3 |                  0.0%   |                  0.0%   [7]
10   conv2d_4              ||||||||           10.9%   |                  3.3%   [8]
11   batch_normalization_4 |                  0.1%   |                  0.0%   [9]
12   conv2d_5              ||||||||||||||     21.4%   ||                 6.6%   [10]
14   dense                 ||                 2.3%   ||||||||||||||||  86.2%   [11]
14   dense_nl              |                  0.0%   |                  0.0%   [12]
16   dense_1               |                  0.0%   |                  0.7%   [13]
16   dense_1_nl            |                  0.0%   |                  0.0%   [14]
17   batch_normalization_6 |                  0.0%   |                  0.0%   [15]
18   dense_2               |                  0.0%   |                  0.1%   [16]
18   dense_2_nl            |                  0.0%   |                  0.0%   [17]
Creating txt report file C:\Users\guarneri\.stm32cubemx\network_analyze_report.txt
elapsed time (analyze): 2.234s
Analyze complete on AI model
Required Ram or Flash size for network is bigger than available Ram or Flash
```

Figure 5

QKeras 8-bit aware training quantization

In a second step, an 8-bit quantization was applied by testing different configurations to find the best QKeras quantizers parameter's setting. Final model performs on the test set an accuracy of 92%, while the final memory size has been reduced to a quarter of the original FP32 model. The 8-bit model has been validated with X-Cube-AI [4], it requires a FLASH = 0.48MB and a RAM = 127.4 KB.

QKeras hybrid aware training quantization

A final, more aggressive quantization has been applied through QKeras by binarizing a part of the KWS network. The hybrid model is composed of 8-bit Conv2D layers and two Dense binary layers. The binarization allows a further reduction of memory by achieving 90% of accuracy by using the Adam optimizer.

Hybrid pipeline learning curves

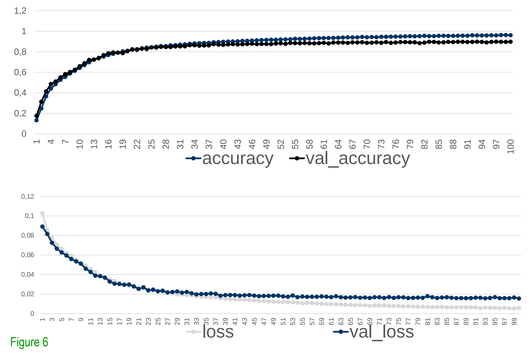

Figure 6

Results

The configuration details of the proposed KWS pipeline are reported in TABLE1. The network recognizes ten words, in future tests a data augmentation will be performed. The "unknown" class will be also included in the training step.

TABLE1	KWS proposed pipeline
Input net size	38X55
Output num classes	10 commands
Quantization schemes(*)	• QKeras 8-bit • QKeras 8-bit and 1bit
DNN topology	CNN

(*) only weights

Details of accuracy, memory requirements and complexity expressed in MACC (Multiply And ACCumulate) are reported in TABLE2. Obtained results show that the hybrid quantization can be a good strategy to strongly reduce the model FLASH size by maintaining an acceptable accuracy. The original FP32 proposed pipeline needed 2.12 MB of flash which has been reduced to 0.076 MB after applying a hybrid quantization scheme designed to binarize the more impacting layer on memory size.

TABLE2	ACCURACY %	FLASH MB	RAM KB	MACC M
KWS proposal FP32 (Keras)	94	2.12	149.20	20.5
KWS proposal 8-bit (QKeras)	92	0.48	127.4	10.5
KWS proposal hybrid1-8 bit(QKeras)	90	0.076	127.4	10.5

Conclusions

The KWS is a voice interaction application which, running in "always-on" mode, need to be characterized of a low power consumption, a low latency and, at the same time, high accuracy for an adequate real time user experience.
In these work is proposed a hybrid quantization scheme to deeply quantize a KWS DNN by applying a binarization to the more impacting layers. Results show that the DQNN KWS pipeline reaches an acceptable accuracy of 90% by requiring a FLASH of 0.076MB. Future experiments will further shrink the model size to reduce the memory requirements and the MCU's overload by quantizing also the bias and the activations of the binary pipeline's section. The binarization will be also extended to the convolutive layers and the drop of accuracy will be analyzed. Residual connections and various learning rate schedulers will be tested.

References

[1] GitHub - google/qkeras: QKeras: a quantization deep learning library for Tensorflow Keras
[2] Warden, P. (2018) Speech commands: A dataset for limited-vocabulary speech recognition. In arXiv preprint arXiv:1804.03209, 2018
[3] Keras: the Python deep learning API
[4] X-CUBE-AI - AI expansion pack for STM32CubeMX - STMicroelectronics