Using Quantized Neural Network for Speaker Recognition on Edge Computing Devices

Tongwei Dai1,*

1Computer Science Department, Viterbi School of Engineering, University of Southern California

*Corresponding author e-mail: tongweid@gmail.com

Abstract. Most successful CNN architectures are deep networks, and their intensive memory and processing requirements have made it difficult to deploy them to microcontrollers or other real-time systems in which memory footprint and power consumption may not be neglected. Consequently, many efforts have been made to adapt deep networks to such contexts, either through hardware specialization and optimization or through architectural modifications. This paper concentrates on the application of CNNs in the field of voice recognition and verification, marked by the works of Simonyan and Zisserman and their proposed VGGNet neural network. As the mentioned model is commonly trained on an audio-visual dataset known as VoxCeleb, this paper further evaluates and improves its performance on a challenging Chinese-speaking audio dataset collected in various media with little preprocessing. Quantization is used to reduce its memory usage and to gear it towards edge computing applications.

Keywords: Speaker Identification, Convolutional Neural Network, Edge Computing

1. Introduction

1.1. Background
The task of voice recognition can be divided into two categories, namely speaker identification and speaker verification. Speaker identification aims to identify the speaker from a predefined group of speakers (one-to-many), whereas speaker verification aims to verify whether a given utterance belongs to a speaker by comparing it against positive speech samples. The latter poses a one-shot learning problem, which is typically resolved by modifying a model to output embeddings instead of classification results. In essence, what the two types of voice recognition
require is a robust way of extracting from a voice sample the components that are most contributive to determining who the speaker is.

The early standard approaches to voice recognition rely on Gaussian Mixture Models (GMM) and low-dimensional input feature vectors, which are normally encoded in Mel Frequency Cepstral Coefficients (MFCC) and obtained from a fixed feature extraction pipeline. Some GMM-based methods have also seen promising results, such as i-vectors [1] and joint factor analysis (JFA) [2]. However, it has been observed that MFCC does not work well with real-world noises and does not exploit speaker-specific features (e.g. pitch information).

Active research is being conducted on incorporating deep neural networks (DNN) into systems based on hand-crafted features. An early experiment by Chen et al. used DNN to learn speaker-specific characteristics from MFCC and to improve upon MFCC [3] as an acoustic representation DNNs also played a pivotal role in the x-vector and i-vector systems, both of which are considered as state-of-the-art systems in speaker recognition.

It is not until recent years that neural networks are directly used to capture features. CNNs have been rather successful in performing cognitive tasks, most notably in image analysis. In traditional architectures of CNN, convolutional layers serve as feature extractors, scanning an input along its original dimensions and thereby preserving spatial information. The ability of convolutional layers to learn underlying patterns in the inputs eliminates the need of hand-crafted features and is more flexible in comparison, although handcrafted features can still complement the process in different ways. Additionally, the parameter sharing scheme drastically reduces the number of parameters and maintains invariants that are held in different patches of the input.

Lukic et al [4], experimented with using a simple neural network consisting of two convolutional layers to learn the feature vectors from raw spectrograms directly, casting away the conventional heavy pre-processing to create feature vectors. Such methods necessitate a large amount of data that are collected “in the wild” rather than in a controlled setting. An automated collection pipeline was then developed by Nagrani et al. [5] which utilizes YouTube as their primary source of audio samples. Using this pipeline, they curated a large-scale audio-visual dataset called VoxCeleb. With readily available data, they were able to achieve state-of-the-art results on VoxCeleb by using a ResNet variant called VGGVox.

1.2. Challenges and related works

There are two major challenges in deploying CNN-based models on embedded systems for real-world cognitive tasks. One challenge is that there are both intrinsic (e.g. accent, age, emotion) and extrinsic noises (e.g. reverberation, telephone effects, background sounds). Therefore, it is crucial to introduce noises into data during training. The usual practice of preprocessing data is adding random noise, which includes changing playback speed, adding reverberation, inserting background influences, etc.

Another challenge stems from the memory- and processing-intensive nature of DNNs. With the philosophy of edge computing gaining popularity in recent years, it is oftentimes the case that a successful model needs to be deployed onto an IoT (Internet of Things) device such as a security camera or a smart phone, where computation resources may be highly limited. Generally, the model is compressed to a level viable for the target execution environment. Therefore, minimizing the performance loss of the compressed model is crucial and has always been an active area of research.

Quantization is a general process of discretizing continuous values, which can be applied during the training or thereafter. In most cases, the real values (which are usually floating
points) are mapped to integer values either deterministically or randomly [6]. Since some integer formats (e.g. 4-bit, 8-bit, and 16-bit integers) occupy less space than floating points, a quantized network can save a significant amount of memory. A common deterministic quantization scheme is to use an affine transformation between a real value $r$ and an integer $i$, such as $r = a \times i + b$ (where $a$ and $b$ are real-valued constants) because this allows for efficient implementations of mathematical operations, notably matrix multiplication, in neural networks by only performing arithmetic operations on quantized values [7]. As operations on integers can be done far cheaply than those on floating points, an efficient implementation of such scheme could greatly relieve the processing burden brought by Floating Point Unit (FPU) and even enables the network to be deployed onto platforms without FPU.

Special backpropagation process, such as Binaryconnect [8], may also be adopted to directly train binary or ternary networks, and several successful schemes exist to reduce the size of a model trained with full precision. Pruning is also widely used, in which neural connections are pruned based on certain importance criteria [9].

1.3. Our contribution
The purpose of this paper is to gauge the viability of a quantized CNN for edge computing applications. The VGGVox model is used for our experiments. We begin by analyzing its performance on two datasets, with one mostly comprised of audio clips with considerable background noises and a baseline comprised of audio clips recorded in noised-reduced environments. This gives us a good approximation on how much worse the model would perform in real-world environments than it would in controlled environments. For the second part of the experiments, the trained model is quantized based on the affine mapping scheme mentioned above. The results of both experiments are detailed in the experiment section.

2. Model specification
Experiment The VGGVox network runs in a simple feed-forward manner. To account for utterances of different lengths, the average pool layer (Average Pool6) before the fully connected layers is adaptive, whose kernel size depends on the dimension of the input spectrogram. Batch normalization layers are inserted between convolutional layers in order to make the model less sensitive to fluctuations in weights. We used Rectified Linear Units (ReLU) as the activation function for each neuron in the network. As a reference, the architectural detail is shown in Table 1 and Figure 1.

![Figure 1. Architectural detail.](image-url)
Table 1. The CNN architecture used.

| Layer   | Filter Dimensions | Stride | Padding | Output Dimensions |
|---------|-------------------|--------|---------|------------------|
| Conv1   | 7×7×96            | 2×2    | 1       | 254×148×96       |
| Max Pool1 | 3×3              | 2×2    | 0       | 126×73×96        |
| Conv2   | 5×5×256           | 2×2    | 1       | 62×36×256        |
| Max Pool2 | 3×3              | 2×2    | 0       | 30×17×256        |
| Conv3   | 3×3×384           | 1×1    | 1       | 30×17×384        |
| Conv4   | 3×3×256           | 1×1    | 1       | 30×17×256        |
| Conv5   | 3×3×256           | 1×1    | 1       | 30×17×256        |
| Max Pool5 | 5×3              | 3×2    | 0       | 9×8×256          |
| Conv6   | 9×1×4096          | 1×1    | 0       | 1×8×4096         |
| Ave. Pool6 | 1×n             | 1×1    | 0       | 1×1×4096         |
| FC7     | 4096×1024         | -      | -       | 1×1024           |
| FC8     | 1024×m            | -      | -       | 1×m              |

For simplicity, batch normalization and ReLU layers are omitted. The variable m represents the number of classes. Note that the output dimensions given in the table are calculated with the assumption that the input audio clip is 3 seconds long and sampled at 16kHz. When the input clip length is left unrestricted, the kernel size (1×n) of the Average Pool6 layer varies based on the size of the input.

3. Classification performance on cn-celeb and zhvoice

3.1. Datasets

CN-Celeb is a dataset modelled after VoxCeleb with its data carefully chosen to simulate realistic settings [10]. The majority of VoxCeleb are clips taken from interviews which are normally conducted in an environment with little background interference. In comparison, CN-Celeb introduces environments with considerable background noises (e.g., singing, drama, advertisement, entertainment, etc.). The utterances of the same speaker may span a few genres, which aids in boosting the model’s robustness against noises during training. Another challenging aspect of CN-Celeb is that it contains extremely short utterances, with some being less than 1 second.

Zhvoice is in fact an agglomeration of eight small open-source datasets [11], which makes it about 3 times larger than CN-Celeb in terms of the total duration of utterances, but as it also has a lot more persons of interest (POIs), the average duration per person is similar for both datasets. Zhvoice is considered less challenging than CN-Celeb and is intended to be used as a control group for our experiment. The audio files in Zhvoice have undergone noise reduction and are not deliberately sourced from a wide range of genres. Additionally, as shown in Figure 2, CN-Celeb has a significantly higher variance in terms of total utterance duration per speaker, which indicates that the amount of data available to train for each speaker differs significantly, with a lot of speakers having a total length of utterance as short as 1 second.

The statistics of the two datasets can be found in Table 2.
Table 2. Magnetization as a function of applied field.

| Properties / Datasets          | CN-Celeb | Zhvoice |
|-------------------------------|----------|---------|
| Number of Audio Files         | 129,283  | 1,131,635 |
| Number of Speakers            | 1,000    | 3,249   |
| Total Audio Duration (hr)     | 272.419  | 889.75  |
| Ave. Duration Per Speaker (min)| 16.431  | 16.345  |
| Max Audio Clip Length (sec)   | 2277.9   | 42.2    |
| Min Audio Clip Length (sec)   | 0.4      | 0.1     |

Note that “Fig.” is abbreviated. There is a period after the figure number, followed by one space. It is good practice to briefly explain the significance of the figure in the caption.

3.2. Data preprocessing
Single-channel data streams from audio files are extracted at a fixed 16kHz sampling rates, which are then converted into spectrograms using a hamming window of 10ms step and 25ms width. All spectrograms are mean-normalized. During training and evaluation, the length of the audio clip is either padded or cropped to be exactly 3 seconds, whereas during inference, the spectrogram reflects the duration of the original audio clip. We ensure that lengthy clips (more than 2 minutes) are broken down into small pieces to reduce the likelihood of sampling the same spectrogram interval between batches.

3.3. Training and validation
A model is first trained from scratch on the CN-Celeb dataset, and the resultant weights are then used to initialize the other model to be trained on Zhvoice. The training configuration for both datasets is the same. Specifically, each dataset is divided into training, evaluation, and test sets by the ratio 8:1:1. Training loss is measured by the cross-entropy function. The traditional Stochastic Gradient Descent (SGD) is used for backpropagation, with a weight decay rate of 0.0005 and a learning rate which has an initial value of 0.001 and decays by a factor of 10 once every 20 epochs. After each epoch, the model being trained is evaluated on the validation set, and the weights that give rise to the best validation accuracy thus far are tracked. The training ends when the model hits a plateau and begins to overfit the training set (i.e. when validation loss starts to steadily increase while training loss is decreasing).

3.4. Evaluation metrics
The model’s performance is judged by its accuracy, namely the ratio between the number of correctly predicted speakers and the total number of speakers. In terms of negatives and positives, classification accuracy can be formulated as

\[
\frac{(TP + TN)}{N} = \frac{(TP + TN)}{(TP + TN + FP + FN)}
\]

where \(TP, TN, FP, FN\) stand for true positive, true negative, false positive, and false negative respectively. This metric should suffice for our experiment, considering the fact that there are a significant number of POIs to classify and the data in both datasets are not overwhelmingly...
asymmetrical.

3.5. Inference

The model was able to achieve a high classification accuracy on the Zhvoice dataset. However, just as performance degradation was observed with traditional systems (e.g. i-vector) on CN-Celeb [10], so too is a reduced classification accuracy with the VGGVox model. The classification accuracy on CN-Celeb is about 20 percent lower than that on Zhvoice. It may be sound to conclude that the noises that are only present in the CN-Celeb dataset, notably music, reverberation, and background speech, are the primary reason why there is a considerable performance drop. Additionally, the high variance in the data available to each speaker in CN-Celeb also likely contributed to the decrease in accuracy. This uneven focus of POIs can lead to some degree of overfitting, compromising the model’s ability to generalize to unseen data. For further reference, the model’s performance is tested when inputs are restricted to exactly 3 seconds (we did this by randomly sampling a 3-second interval from the spectrogram). As expected, this resulted in a slight reduction in accuracy because the longer the input clip, the more extracted information there would be for the model to aggregate using adaptive pooling. All the classification accuracies obtained in this experiment are listed in Table 3.

![Figure 2. A density plot over the total utterance length per speaker for both datasets.](image)

| Datasets/ Accuracies | Top-1 (3-sec input length) |
|----------------------|---------------------------|
| Zhvoice (Baseline)   | 97.23                     |
| CN-Celeb             | 75.41                     |

4. Quantization

Post-training quantization is chosen as our approach to compress the VGGVox model. The model is quantized using two quantization pipelines developed by Facebook, FBGEMM for the x86 platform and QNNPACK for the ARM platform. FBGEMM quantizes the layers on a per-channel basis, whereas QNNPACK does it on a per-tensor basis. For both pipelines, FP32 is mapped to INT8 linearly on a given granularity, and in order to determine the optimal scale and
offset values of the linear mapping, the original FP32 model was run on a calibration dataset (for which we just used 1,000 batches from the test set). The quantized VGGVox produced by both pipelines is about 3 times smaller in size and 1.5 times faster in inference than the FP32 counterpart. The comparison results are presented in Table 4. In particular, using QNNPACK, the compressed model is able to achieve a classification accuracy comparable to the original model.

5. Conclusion and discussion
As with i-vector and x-vector systems, VGGVox has a noticeable accuracy drop on CN-Celeb, a dataset that simulates real-world scenarios, while it is able to achieve a near-perfect accuracy with noise-reduced voice recordings (Zhvoice). Our experiment reinforces the fact that it is challenging to make a voice recognition systems robust against noises. The first experiment shows the best accuracies that can be achieved under the specified training setting through hyper-parameter tweaking. That said, we did not attempt to alter the architecture of VGGVox, and this is certainly a point of interest for future research.

The post-training quantization on VGGVox yielded reasonable memory and computational efficiency; it is reasonable in the sense that it could operate within the hardware constraint of most real-time and embedded systems without significant performance degradation. We only explored INT8 as the target precision, which tends not to incur significant accuracy loss and is the minimum bandwidth supported on most devices, and little attempt was made to quantize the model to lower precisions (e.g. INT4, ternary, and binary). Moreover, during the calibration stage, the two quantization backends collect histograms of activations to optimize the linear mapping, but there are many novel accuracy-loss reducing techniques to be tested, such as Hessian-aware quantization recently proposed by [12]. Thus, it is very possible that the model could be further compressed, but this calls for future experiments to see if the loss in performance merits the gain in memory usage.

In conclusion, while the memory and computational demand of the VGGVox model could be effectively reduced without a significant performance cost, much remains to be done to optimize a CNN for various environments before it is deployed for general-purpose voice recognition tasks.

Table 4. The performance of the two quantized models produced by the FBGEMM and the QNNPACK libraries.

|                      | Original | FBGEMM | QNNPACK |
|----------------------|----------|--------|---------|
| Top-1 on Zhvoice     | 97.23    | 89.59  | 95.80   |
| Memory Size (MB)     | 68.94    | 20.02  | 20.18   |
| Inference Time (ms)  | 192.3    | 132.5  | 129.4   |
| Inference Speedup    | N/A      | 1.45x  | 1.49x   |

Inference time is the average time it takes for the model to output a result for a single spectrogram input. The absolute CPU inference time is measured on the hardware specified in the Experiment Setup section.

References
[1] Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., & Ouellet, P. (2011). Front-End Factor Analysis For Speaker Verification. IEEE Transactions on Audio Speech and Language Processing, (pp. 788-798).
[2] Kenny, P. (2005). Joint factor analysis of speaker and session variability: Theory and algorithms. Montreal.

[3] Chen, K., & Salman, A. (2011). Learning Speaker-Specific Characteristics. IEEE Transactions on Neural Networks, 1744-1756.

[4] Lukic, Y., Vogt, C., Dürr, O., & Stadelmann, T. (2016). Speaker identification and clustering using convolutional neural networks. 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP) (pp. 1-6). Vietri sul Mare, Italy: IEEE.

[5] Nagrani, A., Chung, J., & Zisserman, A. (2017). Voxceleb: a large-scale speaker identification dataset. Interspeech, (pp. 2616-2620).

[6] Guo, Y. (2018). A Survey on Methods and Theories of Quantized Neural Networks. coRR.

[7] Jacod, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Kalenichenko, D. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (pp. 2704-2713).

[8] Bengio, Y., Courbariaux, M., & David, J.-P. (2015). BinaryConnect: Training Deep Neural Networks with binary weights during propagations. Neural Information Processing Systems 28, 3123-3131.

[9] Yann LeCun, J. S. (1989). Optimal brain damage. Advances in Neural Information Processing Systems 2 (pp. 598-605). Denver, Colorado, USA: Morgan Kaufmann.

[10] Fan, Y., Kang, J., Li, L., Li, K., Chen, H., Cheng, S., Wang, D. (2019). CN-CELEB: a challenging Chinese speaker recognition dataset. arXiv preprint arXiv:1911.01799.

[11] Kuang, DD. (2020). zhvoice. Retrieved from Github: https://github.com/KuangDD/zhvoice.

[12] Dong, Z., Yao, Z., Gholami, A., Mahoney, M., & Keutzer, K. (2019). HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), (pp. 293–302).