**EMBEDDED VOICE COMMAND SYSTEM**

Radu-Ilie HEREȘANU

Military Technical Academy Ferdinand I, Boulevard George Coşbuc, Bucharest, Romania

Corresponding author email: radu.heresanu@mta.ro

***Abstract***

*In this paper, I use convolutional neural network architectures established in image classification to improve the speech signal classification performance in activation words. The goal is to develop a command system that is vocally actuated by a predefined set of words. To achieve this goal, the voice signal was transformed into MFCC plus Delta feature matrices, which we can consider as images, which were then used to train a model to classify these activation words. By treating the voice signal as an image, it was very advantageous to implement and compare three architectures that are famous for this type of task – VGG16, AlexNet and SqueezeNet which achieved outstanding results in competitions such as ImageNet ILSVRC. From my own analysis, I found that a quantization of the model brings advantages both in terms of performance and prediction speed, the model being reduced in size, the range of values in which the calculations are made being much more suitable for an edge device. Using previous studies, I also tested an augmentation of the characteristics, but the performance was not substantially improved.*

***Key words****: quantization, augmentation, MFCC, VGG16, AlexNet, SqueezeNet.*

# 1. INTRODUCTION

Currently, deep learning is successfully applied in various fields, including automatic speech recognition (ASR), where the main research objective is to design the best possible network architectures, for example, DNNs, CNNs, RNNs and end-to-end models. In particular, I adapted three convolutional neural networks used in image classification with the aim of being able to classify certain words spoken by a speaker. Unlike products like Siri, Alexa and Google Assistant, this system uses the device's voice recognition in a different way. Instead of relying solely on connecting to the cloud infrastructure, this system uses technology built into the device to recognize keywords and perform certain simple tasks without the need for an Internet connection. In the context of using an edge device, model size and classification accuracy play a critical role. Therefore, fulfilling these conditions requires the use of techniques such as model quantization, feature augmentation, and the utilization of a representative, balanced, and easily adaptable dataset.

# 2. CONTENT

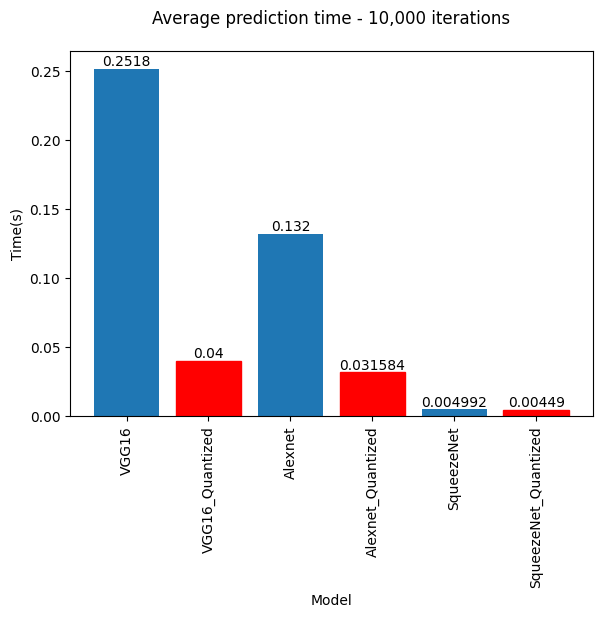
The data set used was Google Speech Commands V2. It contains 105,836 wav audio files divided into 36 classes. The pre-processing and feature extraction involved the choice of activation words, the extraction of MFCC and Delta MFCC features, the augmentation of the feature matrices with random masking bands, and finally the division of the data set into train, validation and test. Following the performance measurements, it was found that the total number of activation words (grouped in one or more classes) should be equal to the number of non-activation samples (noise/other words). Modifications were made to the CNNs in order to be able to work with images with the shape (32,98) and to improve their performance. It has been observed that for major changes such as the type of optimization/activation function, the number of convolutional layers, the drop-out percentage, the accuracy and the loss are negatively affected. The main results obtained with these architectures are very good on the data set used. Accuracy is above 90% in almost all cases. Comparing each CNN, I found that for an edge device, a model with fewer parameters, such as SqueezeNet which has 50 times fewer parameters than Alexnet and 100 times fewer than VGG16, lends itself better to embedded processing and an acceptable true match ratio. In order to decrease the prediction latency and the size of the models I applied post-training quantization. This transformation reduced the complexity of the models and the representation of the parameters (weights, activations and other variables) so that the calculations were simplified. Using the tensorflow library, I have four types of quantization available: dynamic range, float16, integer, integer quantization with int16 activations. Each type of quantization has its role, depending on the device used, but a more accurate classification I found in the official documentation and is presented in a table below. I mainly used dynamic range quantization. For example, to quantize 8-bit values:

(1)

(2)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Arhitecture | H5 | TFLite | TFLite Quantized | Train | | Validation | | Test | |
| Loss | Acc. | Loss | Acc. | Loss | Acc. |
| VGG16 | 118MB | 59MB | **14,9MB** | 0.545 | 0.9722 | 0.7137 | **0.9366** | 0.4198 | **0.957** |
| AlexNet | 77MB | 33MB | **9.64MB** | 0.018 | **0.9941** | 0.291 | **0.9475** | 0.2876 | **0.9593** |
| SqueezeNet | 1.1MB | 488kB | **156KB** | 0.104 | 0.9687 | 0.5713 | 0.8899 | 0.5746 | 0.9146 |

(The dimensions of the models and their performances)



(Average prediction time – 10.000 iterations)

|  |  |  |
| --- | --- | --- |
| Technique | Benefits | Hardware |
| Dynamic range quantization | 4x smaller, 2x-3x speedup | CPU |
| Full integer quantization | 4x smaller, 3x+ speedup | CPU,EdgeTPU, Microcontrollers |
| 16x8 integer quantization | 3-4x smaller, 2x-3x+ speedup | CPU,possibly Edge TPU, Microcontrollers |
| Float16 quantization | 2x smaller, GPU acceleration | CPU, GPU |

(The differences between the various quantization techniques)

# 3. CONCLUSIONS

The results of this study show that current deep learning networks can be successfully modified and designed to perform a wide range of tasks, including ASR applications on low-computing embedded devices.

It has been confirmed that a key aspect to enable speech recognition (as well as other machine learning solutions) on a system on chip and embedded devices is finding the right trade-off between network performance and computational load.