# Group 2 — Homework II

Alessio Vacca
*Politecnico di Torino*
s288240@studenti.polito.it

Luca Benfenati
*Politecnico di Torino*
s286582@studenti.polito.it

Nicola Scarano
*Politecnico di Torino*
s287908@studenti.polito.it

First of all, we denote that, for all the different model versions proposed, the strict requirements on model size have brought us to exploit TFLite conversion and zlib compression. Configuration results are shown in table I and II.

## Exercise I - a regression task

***Version a)***

We choose the Multilayer Perceptron (MLP) as architecture since it has proven to be more suited for this regression task. We notice immediately that the hardest requirement to satisfy is the size of the model. In order to reduce it, we apply a structured pruning. Specifically we choose a very small width multiplier $\alpha$, thus reducing the number of units of each Dense layer. Then, we decide to apply magnitude-based pruning: after each training step, we remove the links with the smallest weights, resulting in an improvement both for the size (reducing it) and the accuracy (increasing it). Subsequently, we deploy post-training quantization (PTQ) to reduce model size with little degradation in accuracy: in particular, 8-bits integer weights-only is selected since guarantees better performance, even if the size reduction effect is less evident with respect to the weights and activations one. With this technique, weights are stored and manipulated as int8, instead of float32.

***Version b)***

The increase in the output steps may influence the performance of the model, leading us to reconsider the entire network architecture. We find that the Convolutional Neural Network (CNN) is more robust with respect to parameter reduction when considering a 9-output steps inference. In addition, we remove a Flatten and a Dense layer from the original architecture, reducing its size without experiencing any relevant loss in performances. For the optimization pipeline, we set $\alpha$ to a slightly larger value than before, since the size requested is slightly larger. Note that, in this case, structured pruning reduces the number of filters of each convolutional layer. While for quantization, we find an 8-bits integer weights and activations PTQ to achieve better results. This kind of quantization exploits a generator function that provides a small dataset to calibrate the range of all arrays in the model to optimize it.

## Exercise II - a classification task

The demanding requirements in terms of accuracy have brought us to choose the Mel-frequency cepstral coefficients (MFCCs) in the preprocessing to extract features from our mini-speech commands audio, since they have shown better performances compared to the Short Time Fourier Transform (STFT).

***Version a)***

The absence of a latency constraint and the relatively large size required allow us to use MFCCs in preprocessing step with the default parameters and to the deploy light optimization techniques. As model architecture, DS-CNN shows us good performances and a reduced size with respect to the basic CNN. To further reduce the size of our model, we consider pruning and quantization techniques. The most suitable optimization pipeline, that allows us to reduce size without too much loss in accuracy, is the magnitude-based pruning followed by a 16-bits float weights-only PTQ (thus weights are stored as float16, instead of float32).

***Version b)***

The constraint on latency pushes us to rethink the preprocessing step of our pipeline. In particular, to reduce the total latency, we decide to *down-sample* our signals to 8kHz. A fine-tuning phase on the MFCCs' parameters is then necessary to regain the accuracy lost during the resampling. No changes in the model architecture are needed, while instead we significantly change the optimization step: in particular, we deploy structural pruning step together with the magnitude-based one. Concerning the quantization step, we decide to employ the full 8-bits integer weights and activation quantization.

***Version c)***

For this version we need to change only the optimization step. Since the latency constraint is unchanged, knowing that the latency is mainly affected by the preprocessing phase, we decide to leave this phase as in version (b). Concerning the optimization phase, the $\alpha$ of the structured pruning is severely decreased to 0.3 to get the model compliant with the size constraint. To cope with the consequent decrease in accuracy, the *final sparsity* of magnitude-based pruning is reduced and quantization is relaxed, using a much lighter 16-bits float weights-only.

We denote that, with the same model structure, we were able to reduce the model size by 5 times, experiencing a relatively small decrease in accuracy through the use of preprocessing and optimization techniques alone. To conclude, we highlight that the latency constraint is strongly influenced by the preprocessing and quantization phase, while pruning plays a crucial role in accuracy and model size.

TABLE I
PARAMETERS OF DIFFERENT MODEL VERSIONS OF EXERCISE 1

| | **architecture** | **optimization** | | | **training time** | **Results** | | |
|---|---|---|---|---|---|---|---|---|
| | | Structured pruning | Magnitude based pruning | Quantization | Epochs of training | MAE(T) | MAE(H) | TfLite size [kB] |
| **Version a** | MLP | $\alpha$: 0.03 | init_spars: 0.20<br>final_spars: 0.40<br>init_step: 2<br>final_step: 20 | weight-only | 31 | 0.268 | 1.199 | 1.35 |
| **Version b** | CNN | $\alpha$: 0.04 | init_spars: 0.00<br>final_spars: 0.25<br>init_step: 2<br>final_step: 20 | weight + activation | 50 | 0.666 | 2.468 | 1.68 |

TABLE II
PARAMETERS OF DIFFERENT MODEL VERSION OF EXERCISE 2

| | **preprocessing** | **architecture** | **optimization** | | | **training time** | **Results** | | |
|---|---|---|---|---|---|---|---|---|---|
| | MFCC | | Structured pruning | Magnitude based pruning | Quantization | Epochs of training | Total latency [ms] | Accuracy | TfLite size [kB] |
| Version a | default | DS-CNN | / | init_spars: 0.25<br>final_spars: 0.75<br>init_step: 2<br>final_step: 20 | weight-only | 25 | / | 0.921 | 125.42kB |
| Version b | sampling rate: 8000<br>mel bins: 16<br>frame length: 320<br>frame step: 160 | DS-CNN | $\alpha$: 0.7 | init_spars: 0.40<br>final_spars: 0.75<br>init_step: 2<br>final_step: 22 | weight + activation | 20 | 36.99 | 0.919 | 49.02kB |
| Version c | sampling rate: 8000<br>mel bins: 16<br>frame length: 320<br>frame step: 160 | DS-CNN | $\alpha$: 0.3 | init_spars: 0.15<br>final_spars: 0.35<br>init_step: 2<br>final_step: 30 | weight-only | 25 | 33.64 | 0.917 | 24.77kB |

## APPENDIX

Commands used to measure the latency:

```
b) python kws_latency.py --model ./models/Group2_kws_b.tflite --rate 8000 --bins 16 --length 320
                         --stride 160 --mfcc
c) python kws_latency.py --model ./models/Group2_kws_c.tflite --rate 8000 --bins 16 --length 320
                         --stride 160 --mfcc
```