# Machine Learning for Internet of Things 2024-2025 Homework2, Group 4

Tanguy Marie Yvan Dugas du Villard, Muhammad Nouman Siddiqui, Shadi Mahboubpardahi
Student id: s321277, s329112, s329057
s321277@studenti.polito.it, s329112@studenti.polito.it, s329057@studenti.polito.it
*Politecnico di Torino*

## I. Methodology

### A. Optimization

To find a model compliant with the latency, accuracy and size constraints, we first defined our optimization strategies: Weight pruning, weight clustering and quantization. We decided at first to prune weight based on magnitude, this strategy showed sufficient results. The model is trained aware of this pruning for up to 50 epochs (if no early stopping occurs). Then, the model is trained for 2 more epochs with clustering optimization, whose number of clusters is a hyper-parameter. Finally, the weights are quantized during the conversion in tensorflow lite objects. We realized that a conversion from float32 to float16 was enough to ensure compliance with the latency and size constraints without degrading the accuracy.

### B. Architecture

After a few steps of exploration, we found out that our optimization method allowed us to build models with up to 200 kB of parameters. Any bigger model would not be compressed below 50 kB. Based on this information, we built different architectures. All of them include a few convolutional layers, followed by a batch normalization. Some architectures use also several pools. All architecture must end with a dense layer, having two output neurons with a softmax activation, as we are working with binary classification. The number of filters in the convolutional layers is a hyperparameter.

### C. Preprocessing

For this type of task, two preprocessing methods could be applied. The first one relies on using the Mel spectrogram, while the second uses the MFCC coefficients. We tried both methods by performing a grid search on the hyperparameters. This grid search has been carried out on several architectures and for both methods.

## II. Preprocessing hyperparameters

The model we trained uses the MFCC coefficients as inputs. These coefficients are computed with the parameters Tab. I.

| Parameter | Value | Unit |
|---|---|---|
| Frame length | 0.064 | Second |
| Frame step | 0.032 | Second |
| Number of mel bins | 100 | Number |
| Lower frequency | 20 | Hertz |
| Upper frequency | 1000 | Hertz |
| No. of MFCC coefficients | 10 | Number |

TABLE I
Preprocessing parameters

## III. Training hyperparameters

The final model has been trained with the parameters Tab. II.

| Parameter | Value |
|---|---|
| Batch size | 256 |
| Initial learning rate | 0.01 |
| Final learning rate | 1e-5 |
| Initial sparsity | 0 |
| Final sparsity | 0.6 |
| Epoch of start pruning | 5 |
| Epoch of end pruning | 45 |
| Number of clusters | 8 |
| Number of filters | (16, 32, 32) |

TABLE II
Training hyperparameters

## IV. Model architecture and optimizations

The model is made of 3 blocks of convolutional layers followed by a batch normalization and a ReLU activation, then a global average pool, and finally a dense layer with two output cels and a softmax activation for classification.

As explained by the methodology, we applied weight pruning based on magnitude during the training of the model. The size of the pruned blocks is 1 by 1. After training, weights are clustered in 8 different clusters. Finally, the model is quantized by using weights stored as float16 instead of float32.

## V. Model performances

The performance of the model is reported Tab. III.

| Constraint | Constraint | Result |
|---|---|---|
| Accuracy | $> 99.4\%$ | 99.5% |
| Size | $< 50$ kB | 32 kB |
| Latency | $< 40$ ms | $27.5 \pm 8.6$ ms |

TABLE III
Performance of the model

## VI. Discussion

### A. Accuracy

Among a vast amount of models trained during the carrying of the homework, only one model achieved the requested test accuracy of 99.5 %. This threshold has been hard to reach and the reproduction of the training, even with the same parameters, is not possible. As most models reach easily 90% of accuracy on the test dataset, only a few reached more than 95%.

### B. Size

The tflite model reaches a size of 32 kB while the same model compressed as a zip reaches 8.3 kB. This small size allows it to be more easily stored and more quickly run on the edge device. To achieve this compression, the TensorFlow model, whose parameters reach more than 110 kB, has been weight-pruned, and its weights have been clustered and quantized.

### C. Latency

As the submitted model is small, it can run very fast. In fact, less than 1 millisecond is required to run the model on the edge device. Most of the computation time comes from the computation of the MFCC coefficients.