Nelly Gaillard, s282613
Giuseppe Galilei, s295620
Beatrice Alessandra Motetti, s287618

Exercise 1

In exercise 1, we used the preprocessing pipeline already developed for Lab3, creating time-series windows of length 6 as inputs of the neural network and considering two different features, temperature and humidity. For what concerns the models choice, whose architectures are reported in Table 2, we decided to calibrate the parameters in order to minimize the size of the TFLite model keeping the accuracy inside the requested boundaries. We chose the MLP model because it showed to have the best trade-off between accuracy and resources usage.

In particular, we observed that a smaller net (i.e. with a strong weight scaling, corresponding to a low alpha) helps to increase the accuracy, but it does not allow to exploit the pruning benefits in terms of tflite file size. Therefore, although the base models turned out to respect all the constraints only by tuning the parameter alpha, we tried to further reduce the size of the tflite models, allowing a small worsening in the accuracy metrics. We report the results and the related parameters in Table 1. To both versions we applied weights quantization and pruning with PolynomialSchedule, starting with 0 sparsity. In the two cases, we applied pruning to a pre-trained base model, therefore the number of epochs and the learning rate refers to the second training slot.

Table 1: Selected configuration of parameters

Version	**out-steps**	**batch_size**	α	**LR**	**sparsity**	**TFLite size**	**T_MAE**	**Rh_MAE**
A	3	256	0.25	1e-3	0.65	1.099 kB	0.26 °C	1.16 %
B	9	256	0.4	5e-4	0.65	1.700 kB	0.61 °C	2.43 %

Exercise 2

As in Exercise 1, we exploited the same data preparation pipeline developed for Lab3. The features extraction strategy which did yield, in our specific case, the best results is the one based on the Mel-frequency cepstral coefficients.

We chose as model architecture for the three versions the DS-CNN (Table 3), since we found it to be the one with the best trade-off between size and accuracy. We set a variable Learning rate, which is reduced whenever the accuracy on the validation set does not increase for 2 epochs, having a minimum LR equal to 1e-04.

We applied structured pruning by means of channel-based pruning, i.e. we reduced the number of filters of the convolutional layers by a factor α. We have also performed, for version C only, unstructured pruning with a PolynomialSchedule, having 0.3 as initial sparsity and 0.65 as final sparsity. Finally, we applied post-training weights quantization to all the models to further reduce the size of the model on disk. We set up a grid search ({α: {0.1 $\rightarrow$ 1, step = 0.05}, do pruning: {True, False}, final sparsity: {0.5 $\rightarrow$ 0.9, step = 0.1}}) to find out the best parameters combination, which is the one reported in Table 4. We have noticed how setting the frame length of the STFT equal to a power of 2 considerably increases the computation speed, thus we have set it to 512, with a frame step of 256. The commands exploited to measure the total latency of the models are reported in Table 5.

Table 2: Model Architectures for Exercise 1.

Model A	**Model B**
Flatten()	Flatten()
Dense(alpha*32)	Dense(alpha*16)
Dense(3*2)	Dense(alpha*32)
Reshape()	Dense(9*2)
	Reshape()

Table 3: Model Architecture for Exercise 2.

Model
Conv2D(filters = α*256)
BatchNormalization()
ReLU()
DepthwiseConv2D()
Conv2D(filters = α*256)
BatchNormalization()
ReLU()
DepthwiseConv2D()
Conv2D(filters = α*256)
BatchNormalization()
ReLU()
GlobalAvgPool2D()
Dense(8)

Table 4: Selected configuration of parameters.

Version	**batch size**	α	**final sparsity**	**Accuracy**	**TFLite size**	**Latency**
A	32	0.8	/	0.9275	83.399 kB	30.26 ms
B	32	0.4	/	0.9225	36.018 kB	29.18 ms
C	32	0.4	0.65	0.9137	24.567 kB	29.05 ms

Table 5: Commands to measure the total latency of the models.

Version	**Command**
A	`python kws_latency.py --model ./Group7_kws_a.tflite --mfcc --length 512 --stride 256`
B	`python kws_latency.py --model ./Group7_kws_b.tflite --mfcc --length 512 --stride 256`
C	`python kws_latency.py --model ./Group7_kws_c.tflite --mfcc --length 512 --stride 256`