Team 25 - Homework 2

Machine Learning for IoT
Politecnico di Torino

I. Classifier Training and Model Optimization

Before starting the optimization phase, we have evaluated the performance of the reference model using the reference Mel-Spectrogram hyper-parameters. It obtained an accuracy of 96% on test set requiring approximately a training of 40 minutes throughout a total of 20 epochs. In addition to not achieving the required accuracy, the reference model occupied too much memory, ~ 450Kb.
Starting from that, the next step has consisted of changing the audio preprocessing. Instead of analyzing the input audio via Mel Spectrogram, we passed to the Mel Frequency Cepstral Coefficient (MFCC) that have led us to better results, but far way from our objective.

A. *Hyperparamter tuning*

Before focusing on the model and its architecture, the attention has been directed towards the audio **preprocessing hyperparameters**. As first thing, the *frame length* value has been changed from 0.04 (reference value) to 0.032 justified from the fact that using power-of-two values significantly increases the performance in terms of latency due to the way the FFT algorithms are designed. The same reasoning has been also applied to the *frame step* value. Differently from VAD, in KWS it is important to have overlapping frames since the task requires an higher level of contextual information and the system must be able to capture temporal dynamics that are less impactful in VAD scenarios. Regarding the *number of Mel Bins*, a tuning by hand has been implemented exploiting the fact that a larger number of Mel Bins leads to a more detailed frequency information, but at the same time could lead to overfitting since the model may become too specific to the training data. With respect to the *upper frequency*, differently from the VAD, the parameter has been set to a larger value. It is a consequence of the fact that in KWS could be also crucial to capture not only fundamental frequencies to distinguish the words. Finally, for the *number of MFCC* a brief paper research has been conducted, finding that a common practice is to set its value between 13 and 20. In addition, being strongly task-specific a tuning by hand has been performed.

In the following table, the pre-processing hyper-parameters values are showed.

Parameter	Selected Value
Frame Length (s)	0.032
Frame Step (s)	0.016
Number of Mel bins	16
Lower Frequency (Hz)	20
Upper Frequency (Hz)	4000
Number of MFCC	20

The last step done before focusing on the model optimization, has been the analysis of the **training hyperparameters**. For each of them, a tuning by hand has been implemented to get the initial value and then we optimize using grid search. It has been observed that there is a link between weight-based pruning and the learning rate. Hence, the learning rate was also updated in the model optimization phase.

Parameter	Selected Value
Batch Size	20
Initial Learning rate	0.01
Final Learning rate	0.001,
Epochs	20

B. *Model Optimization*

Adapting the hyperparameter values to our task was not enough for meeting all the three constraints. Therefore, the strategy has been focusing on the model analyzing one constraint at a time. Starting from the ***model size***, to reduce the size of the model, the number of filters in the Conv2D layers has been significantly decreased. Through a hyperparameter tuning, it has been found out that 32 filters in each convolutional layer was the right compromise for reducing substantially the model size while still leading to a good level of accuracy.
Since the constraint on the model size was not yet satisfied the next two steps have consisted of substituting the canonical 2D convolutional layers with the more efficient pair composed by Depth-Wise and Point-Wise Conv2D layers and applying weight-based pruning during the training of the model. With respect to weight pruning, it was applied after 10% of the training steps using a linear decay, with the initial and end sparsity set to 0.10 and 0.90, respectively. The final sparsity value was determined by starting from a lower value and incrementally increasing it until a significant drop in performance was observed.
These two changes have allowed the reduction of the number of model parameters, meeting the model size requirement, 25 KB.
Once satisfied one of the three requirements, the attention has shifted to the next one: the ***model accuracy***. The previous mentioned steps have led to a light model but with an high level of overfitting that has a great impact on the test set accuracy. In order to overcome this problem some stabilizers have been added to the model. In detail, the decision landed on adding Dropout and Max Pooling layers.

The final model architecture resulting from the analysis consisted of three main blocks composed by *Convolutional layers*, *max-pooling*, *batch normalization*, *ReLU activation* and *dropout*. At the end a *global average pooling* layer consolidates spatial information, and the final layers consist of a *dense layer* with two units (for binary classification) followed by a *softmax activation function*. Note that the second and third blocks contain the Depth-wise convolution layers instead of the canonical ones.

C. *Results*

The mentioned architecture prioritizes efficiency, leveraging depthwise separable convolutions for a lighter model with a lower inference time. Batch normalization, Dropout and Max-Pooling aids stability mitigating overfitting. In the following table are showed the final results.

Accuracy	Model Size	Zipped Model Size	Latency Savings
0.995	24KB	11KB	55%