Homework 2 - Team 13

1st Fabrizio Battiloro
s317786@studenti.polito.it

2nd Lorenzo Mossotto
s318971@studenti.polito.it

3rd Arash Daneshvar
s314415@studenti.polito.it

I. Methodology Adopted

To identify the optimal hyper parameters while adhering to the constraints outlined in the homework, we proceeded in three different steps.

Firstly, we fine-tuned the audio preprocessing parameters, building on the analysis conducted in the previous homework. We made slight adjustments to some parameters and increased the batch size to 25 in the training arguments.

Next, we adjusted the initial network architecture and sparsity. The original model, being too deep, was ineffective for our task. By adding a layer and gradually reducing filters from 128 to 16, we improved both accuracy and model weight. Notably, greater filter reduction enhanced model generalisation. We also increased final sparsity to 0.85

The final step involves the conversion and compression phase. After finding a model that met our accuracy constraints and was sufficiently lightweight, we used the TFLiteConverter module and applied default optimisation. This process resulted in an exceptionally lightweight final model, as seen in the results paragraph.

Pre-processing Args	Value
Sampling Rate	16000
Frame Length in s	0.032
Frame Step in s	0.01
Num Mel bins	20
Lower Frequency	20
Upper Frequency	6000

TABLE I: Pre-processing hyper-parameters

Training Args	Value
Batch Size	25
Initial Learning Rate	0.01
End Learning Rate	1e-5
Epochs	80

TABLE II: Training hyper-parameters

II. Model architecture and Optimization

The model architecture comprises a Convolutional Neural Network (CNN) structured using the Keras Sequential API. It consists of several Conv2D layers with 16 filters each, utilizing varying kernel sizes and strides, followed by BatchNormalization and ReLU activation functions. GlobalAveragePooling2D is employed to condense spatial dimensions, leading to a Dense layer mapping features to the number of classes via Softmax activation. The model integrates Batch Normalization and ReLU activation for stability and non-linearity, while TensorFlow Model Optimization is used for weight pruning. Pruning gradually reduces less significant weights' impact, optimizing model efficiency through magnitude-based weight reduction, facilitated by a PolynomialDecay schedule, starting at 0.2 sparsity and progressing to a defined final sparsity level over training steps. This approach aims to streamline model size and computational demands while retaining classification performance.

III. Results

As shown in the table below, our model satisfies all the constraints we aimed to achieve. In particular, as mentioned earlier, we observed that reducing the number of filters led to better accuracy and a lighter weight. To meet the accuracy goal, we added an additional layer, resulting in an accuracy consistently ranging from 99% to 100% in some cases. The model's size is remarkably small, with a zipped file of only 4,305 KB, allowing us to achieve a weight saving greater than 82.5%, compared to the initial 25 KB constraints. Lastly, the reduction in the number of filters also led to a significant decrease in latency, achieving a 66.8% savings.

Metrics	Value
Accuracy	99%
TFLite Size	11,883 KB
Zip TFLite Size	4,305 KB
Total Latency Savings	66,8%

TABLE III: Results