Machine Learning for IoT
Homework 2

Di Nepi Marco S277959 | Benedetto Irene S276200 | Falletta Alberto Maria S277971

1 Multi-Step Temperature and Humidity Forecasting

The goal of the exercise is to train and optimize models to perform multi-output/multi-step predictions under some constraints. The dataset used is Jena Climate Dataset, on which was performed a pre-processing step in order to obtain windows of 12 elements (for both temperature and humidity, so 24 in total): six of them for training and the following six as labels to be predicted.
Being in a multi-step prediction setting, a custom MAE has been designed: for every prediction the six obtained values are subtracted with the six label values. A mean is then performed according to each axis and the obtained values are then averaged with values from all batches.
At each epoch, the MAE returns two value, the absolute error on temperature and on humidity.

MLP1	MLP2
Flatten() Dense(alpha*128) Relu() Dense(12) Reshape(OutputShape)	Flatten() Dense(alpha*128) Relu() Dense(alpha*128) Relu() Dense(12) Reshape(OutputShape)

Table 1: Models used for version A and B

Starting from the MLP suggested in Lab03, the output has been reshaped in order to match the shape [None, 6, 2] and, in addition, for version A only it can be noticed that a dense layer has been removed (allowing the model to be smaller in size and satisfying the constraints once compressed).
Furthermore, the training hyperparameters have been modified as showed in table 2, in order to obtain an higher starting accuracy before the applied optimizations.
Since the tflite model size without optimizations is around 70kb, a 35x reduction is needed, and to do so, Magnitude Based Pruning (for both versions, with initial sparsity of 0.3 and PolynomialDecay applied from the 5th until the 15th epoch), Structured Pruning and Post-training Quantization have been used.
The parameters (width scaling factor alpha and final sparsity) have been found through a grid search. The benefits of these methods come from the reduction of the precision (passing from float32 to int8) and to the augmented sparsity of the model, that can therefore be compressed with zlib.

Version	Description	Training Parameters	Alpha	Sparsity	Size	MAE T	MAE H
A	MLP1	LR=0.1 (divided by 10 at epochs [10,50,60]) Batch size = 512 Epoch = 70	0.3	0.9	1.9kb	0.47°C	1.76%
B	MLP2	LR=0.02 Batch Size = 32 Epochs = 20	0.1	0.85	1.68kb	0.52°C	1.85%

Table 2: Hyperparameters and results for each version

2 Keyword Spotting

The aim of the script is to train models for keyword spotting on the original Mini Speech Command Dataset, keeping track not only of accuracy and size constraints but also latency. We used the DS-CNN suggested in Lab03 instead of CNN or MLP, since it is better in both accuracy and size. Starting from this model, without optimizations the tflite model size is around 550kb, respectively a 22x, a 16x and a 12x reduction is needed, in order to satisfy size constraints. In version A, a Magnitude Based Pruning (with initial sparsity of 0.3 and PolynomialDecay applied from the 5th until the 25th epoch), Structured Pruning and Post-training Quantization have been used. The parameters (scaling factor and final sparsity) have once again been found through a grid search. In versions B and C the models are allowed to be slightly bigger, we therefore decided not to apply Magnitude Based Pruning since its only advantages are in size and not in latency. The optimization applied in the two versions is mainly Structured Pruning, that allows to have a small models resulting in a much faster inference. In particular, in version B we computed MFCC representations, since the constraint is related to the inference time only and not the preprocessing time, obtaining also an higher margin in accuracy. To reduce the total latency for version C, we used STFT (since the main bottleneck is given by the computation of MFCC) with resized spectrograms. Feeding the network with higher resolution images allows to gain some accuracy points and meet the constraint.

DS-CNN
Conv2D(filters=int(256*alpha))
BatchNormalization()
ReLU()
DepthwiseConv2D()
Conv2D(filters=int(256*alpha))
BatchNormalization()
ReLU()
DepthwiseConv2D()
Conv2D(filters=int(alpha*256))
BatchNormalization()
ReLU()
GlobalAvgPool2D()
Dense(OutputShape)

Table 3: The same model for all three the versions has been used

Version	Training	Alpha	Sparsity	Size	Accuracy	Mode	Inference	Total Latency
A	Epoch = 30, LR = 0.02 (divided by 10 at epochs [20,25])	0.75	0.9	23.1kb	0.93	MFCC	-	-
B	Epoch = 30, LR = 0.02 (divided by 10 at epochs [20,25])	0.3	None	26.4kb	0.93	MFCC	1.17ms	-
C	Epoch = 30, LR = 0.02 (divided by 10 at [20,25]), resize=[56, 56], Frame_length=256 Frame_step=128	0.3	None	26.4kb	0.91	STFT	-	23ms

Table 4: Hyperparameters and results for each version