Machine Learning for IoT - Politecnico di Torino

Homework 2 report

Paolo Aberto
StudentID : s278098

Lorenzo De Nisi
Student ID: s276545

Carmine De Stefano
Student ID: s278176

1 Multi-Step Temperature and Humidity Forecasting

After implementing a WindowGenerator class returning 6 output steps, we tried to train the models. We settled on a weights-only quantization, and after tweaking the pruning parameters we found two versions of the CNN model that were able to respect the given constraints. The parameters used for the 2 models were the final sparsity (%) at which pruning ends, and the width multiplier α.

We adopted a callback function to stop the training when the validation set performances stopped to improve. We adopted the standars Keras checkpoint approach, and we monitored the mean of the two MAE metrics for the checkpoint.

In the following table we can see the results for the two models:

Quantization	**Final sparsity**	α	**Size**	**Compr. size**	**TempMAE**	**HumMAE**
Weights only	0.745	0.12	5246 kB	1992 kB	0.4812 °C	1.786 %
Weights only	0.75	0.07	4480 kB	1689 kB	0.4795 °C	1.879 %

Table 1: Metrics for the final models obtained (version A and B)

2 Keyword Spotting

For keyword spotting on the mini speech command dataset, we used the DS-CNN model with weights-only quantization, obtaining the models in the table below:

Quantiz.	**F.S.**	α	**Size**	**Compr.**	**Inf. Latency**	**Tot. Latency**	**Test acc.**
Weights only	0.7	0.9	77248 kB	21764 kB	9.03ms	66.71ms	0.91375
Weights only	-	0.3	26960 kB	23233 kB	1.16ms	58.72ms	0.91
Weights only	-	0.4	40320 kB	34831 kB	4.52ms	21.07ms	0.9175

Table 2: Metrics for the final models obtained (version A, B and C)

As before, we adopted a callback function to monitor the accuracy metric on the validation set. In order to obtain the performances requested in version B, we used a quite low parameter α, to reduce inference computation. For version C instead, we switched to STFT in order to dramatically reduce processing time.

For both models, compression has been perormed thanks to the zlib library. Final memory performances were calculated by considering the compressed model. For both tasks, we used the original architectures provided in the laboratory.