Ex 1: For this exercise, we trained 3 distinct models (MLP, LSTM, CNN) and evaluated each of them on the test data using multiple outputs mean absolute error (MAE). We selected MLP after examining all models, as it was the most suitable for our task.

It was essential to reduce the model complexity, so we proceeded by applying 2 distinct techniques. The first was applying structured pruning to reduce the model complexity, using alpha for decreasing size and latency.

As for the second technique, we used post-training quantization for both weights only, and weight and activation for decreasing and satisfying the size constraint. However, this increases the MAE, so we should consider both aspects.
Upon examining the results, weight-only quantization was superior to weight and activation quantization.

The results we achieved are summarized in Table 1.

Version	alpha	Compressed file size(KB)	Decompressed file size(KB)	T MAE	H MAE
A	0.03	1.234	2.528	0.260	1.178
B	0.03	1.428	2.736	0.578	2.438

Table 1: Result for ex1

Ex 2: For this exercise, we used standard preprocessing of audios by applying STFTs followed by MFCCs. We decreased the number of mel-bins to 16 to decrease the latency. In addition, we increased the frame length and frame step to achieve the latency, remaining cautious of the fact that this can decrease the accuracy.
The MFCC representation of audios was used as image inputs to our DS-CNN model. We used this model because It has a lower number of parameters to adjust as compared to standard CNNs, hence we reduce the risk of overfitting.
In addition, DS-CNNs are computationally cheaper because they need fewer computations, which allows us to deploy faster convolution neural network models without losing much of the accuracy.

In order to satisfy all 3 constraints, we had to use quantization and pruning techniques.
Quantization dramatically reduces both the memory requirement and computational cost of using neural networks, which results in less size and latency, with negligible degradation in model accuracy.
Furthermore, we decided to use structured pruning for removing unimportant weights from the network in hopes of increasing inference speed and decreasing the model's size.
We can observe the complete results in Table 1, the accuracies for all versions in Fig 1, and differences of sizes before and after quantization in Fig 2.

Version	alpha	Frame length	Frame step	accuracy	Compressed file size(KB)	DeCompressed file size(KB)	latency
A	0.59	1000	350	93.125	48.97	57.7	58.89
B	0.6	1024	256	92.25	49.95	59	35.17
C	0.3	1024	256	91.75	23.45	28	27.18

Table 2: Result for ex2

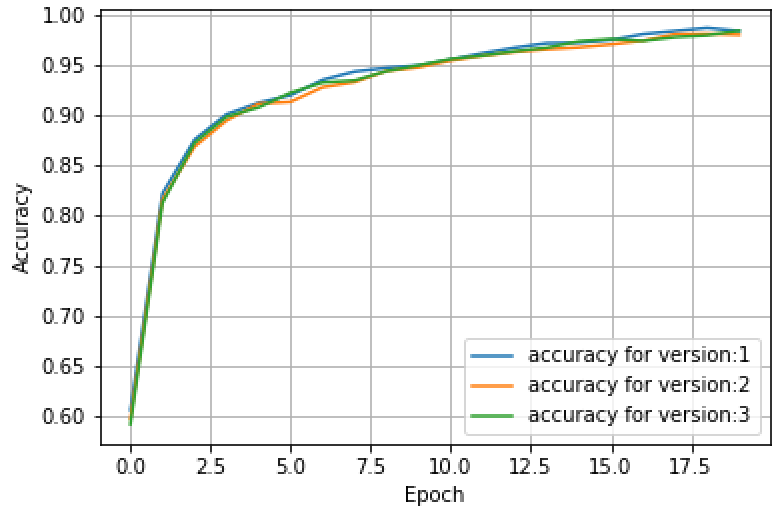

Figure 1: Trend of accuracy for versions

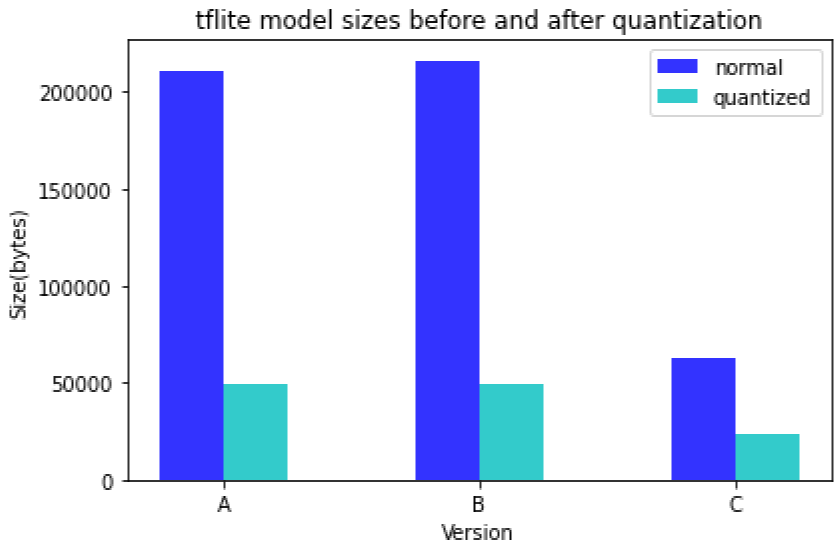

Figure 2: The model size before and after compression

<u>the command used to measure the latency:</u>

a) Python kws_latency.py –model Group19_kws_a.tflite.zlib –mfcc –length 1000 –stride 350 –bins 16
b) Python kws_latency.py –model Group19_kws_b.tflite.zlib –mfcc –length 1024 –stride 256 –bins 16
c) Python kws_latency.py –model Group19_kws_c.tflite.zlib –mfcc –length 1024 –stride 256 –bins 16