Machine Learning for IoT

Lab 4 – Optimization

Exercise 1: Efficient Layers: Depthwise Separable Convolutions

1.1 In Deepnote, develop a convolutional neural network with depthwise separable convolutions (DS-CNN) following the specifications reported in Table I.

```
                              DS-CNN
Conv2D(filters=256, kernel_size=[3, 3], stride=[2, 2],
       use_bias=False, padding='valid')
BatchNormalization()
ReLU()
DepthwiseConv2D(kernel_size=[3, 3], stride=[1, 1],
                use_bias=False, padding='same')
Conv2D(filters=256, kernel_size=[1, 1], stride=[1, 1],
       use_bias=False)
BatchNormalization()
ReLU()
DepthwiseConv2D(kernel_size=[3, 3], stride=[1, 1],
                use_bias=False, padding='same')
Conv2D(filters=256, kernel_size=[1, 1], stride=[1, 1],
       use_bias=False)
BatchNormalization()
ReLU()
GlobalAveragePooling2D()
Dense(units=8)
Softmax()
```

Table I

1.2 Train different versions of the DS-CNN on the MSC dataset using the pre-processing techniques and hyperparameters listed in Table II. For the training hyperparameters, use the same setup of *LAB3*.

1.3 Convert each model version to the *TFLite* format and evaluate accuracy, memory, and latency. Plot the collected metrics to identify Pareto-efficient configurations. Comment the results.

Version	Features	Pre-processing Hyperparameters
#1	Log-Mel Spectrogram	Sampling rate: 16000Hz. STFT frame length: 40ms. STFT frame overlap: 50%. # of Mel bins: 40. Mel lower frequency: 20Hz. Mel upper frequency: 4000Hz.

#2	MFCCs	Sampling rate: 16000Hz. STFT frame length: 40ms. STFT frame overlap: 50%. # of Mel bins: 40. Mel lower frequency: 20Hz. Mel upper frequency: 4000Hz. of MFCCs: 10.

Table II

<u>Exercise 2:</u> Structured Pruning via Width Multiplier

2.1 In Deepnote, develop different versions of the CNN model of *LAB3* using different width multipliers α, e.g., $\alpha \in \{0.25, 0.5, 0.75\}$.

2.2 For each value of α, train different versions of the CNN on the MSC dataset using the pre-processing techniques and hyperparameters listed in Table II. For the training hyperparameters, use the same setup of *LAB3*.

2.3 Convert each model version to the *TFLite* format and evaluate accuracy, memory, and latency. Plot the collected metrics to identify Pareto-efficient configurations. Comment the results.

<u>Exercise 3:</u> Magnitude-Based Weights Pruning

3.1 In Deepnote, develop a Python notebook to train the CNN model of *LAB3* on the MSC dataset using magnitude-based weights pruning. Setup a *PolynomialDecay* pruning schedule with the following hyperparameters:

- Initial sparsity: 20%
- Final sparsity: 70%
- Begin step: 20% of the training
- End step: 80% of the training

3.2 Repeat the training with different values of final sparsity ∈ [70%, 95%] and the different pre-processing techniques of Table II. For the training hyperparameters, use the same setup of *LAB3*.

3.3 Convert each model version to the *TFLite* format, apply *ZIP* compression, and evaluate accuracy, memory (of the zipped *TFLite*), and latency. Plot the collected metrics to identify Pareto-efficient configurations. Comment the results.

3.4 Compare the results of the three exercises and answer the following questions:

- Which strategy reduces accuracy the least?
- Which strategy reduces memory the most?
- Which strategy reduces latency the most?
- How to maximize memory and latency savings with minimal impact on accuracy?