Efficient Computing for Artificial Intelligence
Lab 4 – Optimization

<u>Exercise 1:</u> Efficient Layers: Depthwise Separable Convolutions

1.1 Review the slides **18-21** in ***ECAI_06-Part-II_Optimization_II***.

1.2 In Deepnote, clone the notebook *6.1 - Lab3 - KWS Training* into a new notebook called *7.1 – Training with DS-CNN* to train a CNN with depthwise (DConv.) separable (SConv.) convolutions (DS-CNN). The model specifications are given in the following table:

Layer Type	Inp. Channels	Out. Channels	Kernel Size	Stride	Padding	Bias
2D Conv.	1	128	3×3	2	0	No
Batch Norm.	128	128	—	—	—	—
ReLU	—	—	—	—	—	—
2D DConv.	128	128	3×3	1	1	No
Batch Norm.	128	128	—	—	—	—
ReLU	—	—	—	—	—	—
2D SConv.	128	128	1×1	1	0	No
Batch Norm.	128	128	—	—	—	—
ReLU	—	—	—	—	—	—
2D DConv.	128	128	3×3	1	1	No
Batch Norm.	128	128	—	—	—	—
ReLU	—	—	—	—	—	—
2D SConv.	128	128	1×1	1	0	No
Batch Norm.	128	128	—	—	—	—
ReLU	—	—	—	—	—	—
GAP	128	128	—	—	—	—
Linear	128	*# classes*	—	—	—	Yes

1.3 Train a first version of the DS-CNN using the following configurations:

Preprocessing Hyperparameters:
- Sampling rate: 16000Hz.
- STFT frame length: 40ms.
- STFT frame overlap: 50%.
- # of mel bins: 40.
- Mel lower frequency: 20Hz.
- Mel upper frequency: 4000Hz.
- # MFCC: 40

Training Hyperparameters:
- Batch size: 32.
- Optimizer: Adam.
- Learning Rate: 0.001.
- # of training steps: 2000.

1.4 Repeat the training & evaluation flow using different hyperparameters values:
- STFT frame length ∈ [10, 50] ms. Try also with power-of-two values like 8ms, 16ms, 32ms.
- STFT frame step such that overlap ∈ {0%, 25%, 50%, 75%}.

- # of mel bins ∈ [10, 40].
- Mel lower frequency ∈ [20, 80] Hz.
- Mel upper frequency ∈ [2000, 8000] Hz.
- # of MFCCs ∈ [10, 40].

1.6 In Deepnote, evaluate the accuracy and size of the generated *ONNX* pipelines using the ONNX Runtime. Comment the collected results.

1.7 Deploy the ONNX pipelines to the Raspberry Pi and measure the inference latency. Comment the collected results.

<u>Exercise 2:</u> Magnitude-Based Weights Pruning

2.1 Review the slides **71-83** in ***ECAI_06-Part-II_Optimization_II***.

2.2 In Deepnote, clone and complete the notebook *7.2 – Lab4 – Training with Weight Pruning* to train the CNN model of *Lab3* using magnitude-based weights pruning.
Set the following pruning hyperparameters:

- Start pruning: 499 (start pruning at this training iteration)
- End pruning: 1499 (stop pruning at this training iteration)
- Prune amount: 0.1 (percentage of connection to prune)
- Prune every steps: 100 (apply pruning every *N* steps)

2.3 Repeat the training & evaluation flow with different values of *prune amount* and different pre-processing hyperparameters. For the training hyperparameters, use the same setup of *Exercise 1.*

2.4 In Deepnote, evaluate the accuracy and size (before and after ZIP compression) of the generated *ONNX* pipelines using the ONNX Runtime. Comment the collected results.

2.5 Deploy the ONNX pipelines to the Raspberry Pi and measure the inference latency. Comment the collected results.

<u>Exercise 3:</u> Node-Based Pruning

3.1 Review the slides **121-130** in ***ECAI_06-Part-II_Optimization_II***.

3.2 In Deepnote, clone and complete the notebook *7.3 – Lab4 – Training with Node-Based Pruning* to train the CNN model of *Lab3* with node-based pruning using masks.

3.3 Repeat the training & evaluation flow with different values of *regularization strength* and different pre-processing hyperparameters.

3.4 In Deepnote, evaluate the accuracy and size of the generated ONNX pipelines using the ONNX Runtime. Comment the collected results.

3.5 Deploy the ONNX pipelines to the Raspberry Pi and measure the inference latency. Comment the collected results.

Exercise 4: **Post-Training Quantization**

4.1 Review the slides **163-195** in ***ECAI_06-Part-II_Optimization_II***.

4.2 In Deepnote, clone and run the notebook to quantize the CNN model of *Lab3* using post-training quantization.

4.3 In Deepnote, evaluate the accuracy and size of the quantized ONNX model using the ONNX Runtime. Comment the collected results.

4.4 Deploy the quantized ONNX model to the Raspberry Pi and measure the inference latency. Comment the collected results.

4.5 Compare the results of the four 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?