Training & Deployment of a “Up/Down” Keyword Spotter

1 Discovery of Optimal Hyperparameters

To meet the constraints of **model size <50 KB**, **accuracy >99.4%**, and **latency < 40ms**, we proceed as following:

Pre-processing hyperparameters:

Mel-Frequency Cepstral Coefficients (MFCCs) were adopted as feature extraction methods due to their ability to capture essential spectral components relevant for keyword spotting, thereby enhancing the accuracy and efficiency of the analysis. Based on this method, the following search for each hyperparameter was conducted:

#mel_bins: although feature extraction is based on MFCC, the number of Mel bins affects the computation of the Mel Spectrogram, from which the MFCC coefficients are derived. The range [10-40] was explored, and an intermediate value equal to 21 was selected to minimize the loss in accuracy, while simplifying the representation, reducing computational complexity and model size.

#mfcc_coefficients: since the number of MFCCs must be less or equal than the number of mel bins, the search was performed within the range from 10 to the number of mel bins found above. A value sufficiently high of 14 has been selected, in order to avoid excessive simplification of the feature space and a subsequent loss in accuracy. However, it was not the maximum possible, as that would have resulted in exceeding the constraints on size and latency.

frame length: the values were explored within the range of 0.08 ms to 0.064 ms, considering that the FFT achieves optimal computational efficiency of Nlog(N) when N is a power of two. An intermediate value of 0.032 ms was selected, as it enabled higher accuracy levels.

frame-step: a value equal to half the frame length was selected, corresponding to a 50% overlap between consecutive frames. This ensures minimal information loss between frames while maintaining a manageable feature extraction workload, thereby keeping latency low.

lower frequency: a search within the range 20-80 Hz has been conducted, selecting an intermediate value of 40Hz, allowing to capture low-frequency features like speech, while minimizing the inclusion of irrelevant noise.

upper frequency: upper frequencies were examined in the range of 2000-8000 Hz, with the most favorable results obtained at 5000 Hz. This frequency range enables the capture of fine details, while filtering out irrelevant information

The final configuration of the pre-processing hyperparameters is reported in the table below (Table 1):

Preprocessing Type	Sampling Rate (Hz)	Frame Length (s)	Frame Step (s)	# Mel Bins	Frequency Range (Hz)	# Coefficients
MFCC	16000	0.032	0.016	21	40 - 5000	14

Table 1: configuration of Pre-processing Hyperparameters

Model Training Hyperparameters:

With regard to the training parameters, the ones listed below have been adopted:

batch_size: a size of 20 provides a sufficient number of training samples to prevent overfitting, while also avoiding excessively long training times.

epoch: a number of epochs equal to 20 makes the model converge efficiently without overfitting, maintaining a balance between accuracy and latency.

learning rate and *end learning rate:* a learning rate of 0.01 ensures a quick convergence in the early stages of training, essential for achieving high accuracy within 20 epochs. A smaller rate could have slowed convergence, preventing optimal performance. The gradual decrease to 1.e-5 helps fine-tune the model's weights, supporting high accuracy (above 99.4%) while avoiding overfitting.

width multipliers: width multipliers were explored within the range of 0.25-0.75. By reducing the width multiplier, the number of filters (and thus the number of weights) is decreased, which in turn reduces the model size and computation time. We found that a value of 0.4 effectively reduces both model size and latency without significantly compromising accuracy.

initial sparsity: initial sparsity indicates that only 10% of the weights are pruned at the start of training. This avoids destabilizing the training process during the crucial initial learning phases. In this way the model is prevented from rapidly losing important information that might be relevant for future learning steps.

final sparsity: the final sparsity was searched in the range 70%-90%, a value of 82% was found allowing to compress the model enough to respect the constraint, but preserving the performance in accuracy.

begin step: starting pruning at 10% of the total steps allows the model to learn a meaningful representation before introducing sparsity. This makes it possible to maintain high performance (accuracy) in the early stages of training, preventing the

introduction of sparsity from overly reducing the model's capacity in the initial phases.

end step: the pruning process concludes at 90% of the total training steps. Ending pruning earlier allows the model to fine-tune its performance after most of the sparsity has been applied. This-ensures to achieve a balance between reducing model size and preserving accuracy.

The table below (Table 2) provides a summary of the previously mentioned and presents the parameters used:

Hyper Parameters	Optimal Values
Batch Size	*20*
Epochs	*20*
Learning rate	*0.01*
End learning rate	*1e-5*
Width multipliers	*0.04*
Initial sparsity	*0.10*
Final Sparsity	*0.82*
Begin step	*0.10*
End step	*0.90*

Table 2: Optimal values of Model Training Hyperparameters

2 Model Architecture and Optimizations

The model is a lightweight **Convolutional Neural Network (CNN)** built with Depth-Wise Separable Convolutions (DS-CNN) and designed for efficient keyword spotting on resource-constrained devices. It is specifically trained to recognize the words "up" and "down" with high accuracy and efficiency, meeting the constraints of accuracy >99.4%, latency <40 ms, and model size <50 KB.

Key Features:

- **Feature Extraction**: the initial convolutional layers process MFCC features to identify temporal and spectral patterns of the audio input.
- **Fully Connected Layers**: these layers map the extracted features into higher-level representations.
- **Softmax Output Layer**: final classification of the input into the "up" or "down" categories.

Optimizations:

- **Width Multipliers**: these scale the number of filters in the convolutional layers, reducing the model's capacity to balance computational efficiency with accuracy. Adjusting the multiplier ensures the model remains compact while achieving high performance, making it suitable for IoT devices.
- **Magnitude-Based Pruning**: during training, weights with smaller magnitudes are gradually pruned. This technique reduces model size and computational demands without compromising its ability to distinguish the target keywords. The pruning process is carefully controlled to retain high-accuracy predictions throughout training.
- **Model Compression:** to ensure the model adheres to the specified size constraints, it has been compressed using a ZIP format. This format, indeed, leads to a significant drop in model size, making the model suitable for the use on memory-constrained devices like the Raspberry Pi.

This architecture is compact, fast, and accurate, ensuring compliance with the constraints on size, latency, and accuracy. Its design and optimizations make it ideal for deployment on resource-constrained devices.

3. Results

Final Results	
Accuracy	*99.5%*
Model Size	*40.9 KB*
Total Latency	*27.1 ms*

Table 3: Final values obtained with the selected parameters and model optimizations

Comments on Results

Overall, the results obtained in the project shows a successful optimization of the model, which is able to meet the constraints over the model size, accuracy, and latency.

1. **Accuracy:** the model achieved an accuracy of 99.5%, surpassing the required threshold of 99.4. This high level of accuracy shows that the feature extraction technique (MFCC), combined with the choice of training parameters and model design, helped ensure that the model captured the relevant features of speech while avoiding overfitting.

   Considering the nature of the keyword spotting task, it is significant not only to accurately recognize the target word but also to minimize false positives and false negatives.

   The final level of accuracy was largely influenced by adjusting the **frame length** and **frame step**. By selecting a frame length of 0.032 seconds and a frame step of 0.016 seconds (50% overlap), the system achieves an optimal balance between capturing sufficient temporal detail and minimizing information loss between frames. These parameters were critical in ensuring the feature extraction process provided high-quality input for the model.

   Additionally, tuning the number of **Mel bins** (set to 21) and **MFCC coefficients** (set to 14) allowed us to fine-tune the representation of the audio data. The chosen values reduced the dimensionality of the feature space while retaining the essential spectral features necessary for accurate classification. This approach minimized the risk of overfitting, while ensuring the model could capture the nuances of the "up" and "down" keywords effectively.

2. **Model size:** the reduction in size was achieved through optimization techniques such as pruning and width multipliers, which allowed the model to be compressed without significantly compromising accuracy. The final model size (40.9 KB) is below the 50 KB limit, allowing the deployment on resource-constrained devices like the Raspberry Pi.

3. **Latency:** the model achieved a latency of 27.1 ms, which is below the 40 ms threshold. This performance is crucial for keyword spotting systems, where response time is key to ensuring reliability and interactivity.

Further considerations and impact of Pruning technique on final results

In general, pruning techniques played a major role in optimizing the model to meet the constraints of size, latency, and accuracy. By removing less important weights during training, pruning provided the following advantages:

- **Model Size Reduction**: by eliminating unnecessary weights, the model size was reduced while retaining the core capabilities for data processing.
- **Improved Latency**: the reduction in the number of parameters lowered the computational demands, speeding up inference times and improving the system's responsiveness.
- **Preservation of Accuracy**: pruning was carried out in a controlled manner, ensuring that key features of the model were preserved and maintaining high performance even after compression.

In summary, pruning optimized the model to be more efficient and faster, without sacrificing accuracy, making it suitable for use on resource-constrained devices and real-time applications.