Team 18 - Homework 2

Andrea Parolin
Politecnico di Torino
Torino, Italia
s291462@studenti.polito.it

MohammadSadegh Radmehr
Politecnico di Torino
Torino, Italia
s301623@studenti.polito.it

Sara Ferrua
Politecnico di Torino
Torino, Italia
s292946@studenti.polito.it

TABLE I: MFCCs Parameters

Parameter	**Values**	**Impact on performance**
downsampling_rate	16000	same as input
frame_length_in_s	[16 , 25, 32, 40] $\times 10^{-3}$	longer is better
frame_step _in_percentage	[0%, 40%, 50%, 60%]	shorter is better
num_mel_bins	[10, 20, 40]	fewer is better
lower_frequency	20	/
upper_frequency	[4000, 8000]	/
num_coefficients	[10, 20, 40]	fewer is better

A. Describe the methodology adopted to discover the hyper-parameters compliant with the constraints of 1.1

The widespread use of MFCCs (mel-frequency cepstral coefficients) for audio keyword recognition has been shown to be an effective approach. By using MFCCs, we were able to take advantage of the benefits of this representation of sound, including its ability to capture the relevant spectral characteristics of a sound while reducing the dimensionality of the data. This made it easier to train our model and improve its performance by also reducing the size.

In comparison *Log mel spectrograms* are a simpler representation of the spectrum of a sound in the logarithmic frequency scale, which can make it difficult to directly compare different phonemes or to train a complex model. It is used to detect if the recorded audio is silent or not as in the previous exercise. By using MFCCs instead, we were able to avoid these potential pitfalls and develop a more effective model for audio keyword recognition.

MFCCs are used in the pre-processing phase to represent the shape of the vocal tract in the short-time power spectrum.This method is applied to the test, train, and validation sets and latter on to the recorder audio. The possible parameters required for the MFCCs method are listed in Table I.

To find possible candidate for MFCCs parameters of Table I, the following assumption are made:

- *frame_lenght_in_s*: the signal has been framed into 20-40ms frames, which provide a sufficient number of samples for a reliable spectral estimate without introducing significant changes to the signal. If the frames were much shorter, we would not have enough samples to obtain a reliable estimate. Conversely, if the frames were longer, the signal would change too much throughout the frame to provide a meaningful analysis.
- *frame_step _in_percentage*: it is typically chosen to be in the range of 40% - 60% of the frame duration to balance the temporal and spectral resolution and it is in use in most of KWS application. If there is 0 overlap between the frames, then adjacent frames will be completely independent of each other, which can cause important information to be lost.
- Considering lower and upper frequencies to be included in the mel spectrum, the lower frequency is considered as a fixed value (no impact in the accuracy or loss) in contrast , the model performs better with upper frequency values equals to 8000. Plotting the average frequencies for both labels confirmed our theory that most of the given keywords are between those ranges.
- Using fewer values for *num_mel_bins* and for *MFCC coefficients* can reduce the size of the input features, which can improve the speed of inference on an embedded device. The performance of a speech recognition system using an higher number of *num_mel_bins*, *MFCCs coefficients* was compared to one using only the first 10 Mel bins. Most of the information contained in 'stop' and 'go' keywords are in the first 10 dimension by graphically plotting, and according to this assumption it was found that the system using fewer Mel bins achieved similar accuracy to the full dimension.

TABLE II: Final pre-processing hyper-parameters

Parameter	**Value**
Feature used	MFCCs
downsampling_rate	16000
frame_length_in_s	32 $\times 10^{-3}$
frame_step_in_percentage	50%
# Mel bins	10
# lower_frequency	20
# upper_frequency	8000
# num_coefficients	10

B. Add a table that reports the pre-processing hyper-parameters of the final solution.

See Table II.

C. Describe the model architecture and the adopted optimizations.

The model is a convolutional neural network (CNN) built using the Keras API with TensorFlow as the backend. The final model 1 has a Sequential architecture and it's fed with an input tensor of dimensions [61, 10, 1].
The subsequent layers are:

- A Conv2D layer with 128 filters and a kernel size of 3x3, with a stride of 2 in both dimensions.
- A Batch Normalization layer followed by ReLU layer.
- A Dropout layer, which randomly sets some of the input tensor values to 0, with a probability of 0.1, to prevent overfitting.
- DepthwiseConv2D layer, which applies a different convolution filter to each input channel. This layer has a kernel size of 3x3 and a stride of 1 in both dimensions.
- A Conv2D layer with 32 filters with a stride of 1 in both dimensions and kernel size of 1x1 to reduce number of channels in the tensor output.
- Another BatchNormalization layer, ReLU layer.
- A GlobalMaxPool2D layer, which applies global max pooling to the tensor.
- A Dense layer with as many units as there are classes in the classification task, in our case 2.
- A Softmax layer to produce a tensor of probability.

Adam with *Polynomial Decay* is an used as optimization technique for training and *EarlyStopping* to stop training when a *val_loss* has stopped improving.

D. Add a table that reports the training hyper-parameters of the final solution.

Hyperband tuning algorithm (iterative approach that starts with a large range of possible values for each hyper-parameter and gradually narrows down the search space to find the best values) was used to search for the best values for:

- *filter_size*, *dropout rate*, and *use_bias* for the layers parameters (marked in bold in 1).
- *initial_learning_rate*, *decay_steps*[1] and *epochs*[2]

The same search is also explored for pre-processing reported in table II with overlap 0%.

See Table III, **bold** parameters are the chosen one.

E. Add a table that reports Accuracy, TFLite Size (in KB), and Total Latency (ms) of the final solution.

See IV.

[1]decay_steps = len(train$_{ds}$) $\times$ $epochs$
[2]EarlyStopping with patience equal 5

Algorithm 1 Model

tf.keras.layers.Input(shape=[61, 10, 1])
tf.keras.layers.Conv2D(**filters**=*conv_2d_1_filters*, kernel_size=[3,3], strides=[2,2], **use_bias**=*conv_2d_1_bias*, **padding**=*conv_2d_1_padding*)
tf.keras.layers.BatchNormalization()
tf.keras.layers.ReLU()
tf.keras.layers.Dropout(**rate**=*dropout_rate*)
tf.keras.layers.DepthwiseConv2D(kernel_size=[3,3],strides=[1, 1], **use_bias**=*dconv_2d_1_bias*,**padding**=*dconv_2d_1_padding*)
tf.keras.layers.Conv2D(**filters**=*conv_2d_2_filters*, kernel_size=[3,3], strides=[2,2], **use_bias**=*conv_2d_2_bias*, **padding**=*conv_2d_2_padding*)
tf.keras.layers.BatchNormalization()
tf.keras.layers.ReLU()
tf.keras.layers.GlobalMaxPool2D()
tf.keras.layers.Dense(units=2)
tf.keras.layers.Softmax()

TABLE III: Hyperparameters Grid

Preprocessing parameters	**Value**
Arguments	Table II with overlap 0%, **Table II**
Layer related parameter	**Value**
conv_2d_1_filters	16, 32, 64,**128**
conv_2d_1_bias	True, **False**
conv_2d_1_padding	same, **valid**
dconv_2d_1_bias	True, **False**
dconv_2d_1_padding	same, **valid**
conv_2d_2_filters	16, 32, **64**, 128
conv_2d_2_bias	True, **False**
conv_2d_2_padding	same, **valid**
dropout_rate	0.1, **0.2**, 0.3, 0.4
Model hyper-parameters	**Value**
initial_learning_rate	0.01, **0.001**
epochs	20, 40, **60**
EarlyStopping	**Enabled**, Not enabled

TABLE IV: Results

Constraint	**Results**
Accuracy	100%
Model size	14.2KB
Preprocessing Latency	6.3ms
Model Latency	0.2ms
Total Latency	5.5ms

F. Comment on the reported results.

In this report, we proposed a model for keyword spotting that achieved 100% accuracy this particular dataset. However, we observed that the model's performance was slightly worse when using the built-in microphone[3], likely due to the presence of background noise. Using a wired microphone to reduce background noise improved the model's performance, indicating that the quality of the input audio is an important factor in the model's performance. This is to be expected, given that the dataset used for training and evaluation was very clean and did not contain any background noise. We also observed that the way audio was recorded affected the model's accuracy. In our experiments, audio recordings were made every second, which meant that a particular word could be partially overlapped across multiple recordings. This is different from the way the keywords were presented in the dataset, where each keyword was contained in a single entry. This difference in the input data may have contributed to the slight decrease in performance when using the recorded audio.

[3]Macbook Pro 2020