SPEECH COMMANDS - REPORT

By : Thibault Lahire (thibault.lahire@student.isae-supaero.fr)

1 Classification of segmented voice commands

Before answering the questions asked, we must explain some choices we made when we built the mel-filterbanks features. As pre-emphasis parameter, we set 0.97, since it emphasizes more the high frequencies than 0.6. The time window is 25 ms, which corresponds approximately to 2 times the pitch pulse period.

Moreover, we do not flatten the speech features vectors, as proposed in the provided code. We prefer to keep storing these features as 2-dimensional vectors in order be able to process them in 2D-convolutional networks. The padding function has also been modified to capture information of interest. Several tests have been performed to obtain that $max\_len = 100$ is the best value of max_len.

From question 1.1 to 1.4, the size of the training set is 9000, and the classifier is a logistic regression regularized by $\alpha = 0.1$. This parameter should also be fine tuned, but we do not discuss it since we will not use logistic regression in the second part and we are just explaining how things work for now. It is important to mention that we begin with :

Default values	**nfilt**	**ncep**	**do_deltas**	**do_deltasdeltas**	**norm across chan**	**norm per chan**
melfbank	20	0	False	False	True	False
mfcc	20	8	False	False	True	False

Question 1.1. Since we know that men speak on average with a frequency of 125 Hz, women with a frequency of 210 Hz, and children with a frequency of 300 Hz, and since there are no more information (according to Shannon's theory) in the range $[F_s/2; F_s]$, with $F_s = 16$ kHz (sampling frequency), we take the following frequency range : [10Hz; 8000Hz]. Indeed, increasing the lower bound of this interval leads to a decrease in the amount of information on the signal. Here are three meaningful results :

Range	**Accuracy on train set**	**Accuracy on valid set**	**Accuracy on test set**
[10Hz; 8000Hz]	38.3 %	35.1 %	32.0 %
[10Hz; 4000Hz]	34.4 %	31.2 %	29.7 %
[4000Hz; 8000Hz]	13.6 %	9.3 %	9.4 %

Question 1.2. We set the frequency range to [10Hz; 8000Hz] and we study the influence of the number of filters for the mel-log filterbanks. Computing filter banks means applying triangular filters on a Mel-scale to the power spectrum to extract frequency bands. The Mel-scale aims to mimic the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies. The more filters we have, the tighter the frequency bands. A trade-off has to be found, since a very tight band might contain very few information. According to our results and to what can be read in the literature, we select nfilt = 40.

nfilt	**Accuracy on train set**	**Accuracy on valid set**	**Accuracy on test set**
10	32.6 %	31.0 %	29.7 %
20	38.3 %	35.1 %	32.0 %
30	40.3 %	35.4 %	32.2 %
40	41.4 %	36.4 %	32.8 %
50	42.9 %	35.6 %	33.9 %

Then, we study the influence of the number of cepstral coefficients for the MFCC, with nfilt = 40. As cepstral features are computed by taking the Fourier transform of the warped logarithmic spectrum, they contain information about the rate changes in the different spectrum bands. Cepstral features are interesting for us because they are able to separate the impact of the source and the filter in a speech signal. Indeed, in the cepstral domain, the influence of the vocal cords (source) and the vocal tract (filter) in a signal can be separated since the low-frequency excitation and the formant filtering of the vocal tract are located in different regions in the cepstral domain. It can be read that 12 to 20 cepstral coefficients are typically optimal for speech analysis, even if higher order coefficients could bring more spectral details. Selecting a large number of cepstral coefficients results in more complexity in the models. We choose ncep = 20.

ncep	**Accuracy on train set**	**Accuracy on valid set**	**Accuracy on test set**
8	44.3 %	39.3 %	39.3 %
10	47.8 %	40.6 %	41.6 %
13	47.8 %	40.6 %	41.6 %
15	55.4 %	41.0 %	41.6 %
20	61.1 %	42.3 %	41.6 %
25	65.4 %	41.2 %	40.4 %
35	74.3 %	39.4 %	41.0 %

Question 1.3. Setting the number of cepstral coefficients to 20, we now study the influence of the delta and delta_delta for the MFCC. In pronunciation, context and dynamic information are important. Articulations, like stop closures and releases, can be recognized by the formant transitions. Characterizing feature changes over time provides the context information, that's why it is interesting to use at least the first derivative. Even if it contradicts our results, we continue with do_delta = True.

do_deltas	**do_deltas_deltas**	**Accuracy on train set**	**Accuracy on valid set**	**Accuracy on test set**
False	False	61.1 %	42.3 %	41.6 %
True	False	62.1 %	39.1 %	37.9 %
False	True	60.9 %	39.3%	37.8%
True	True	61.0 %	37.4 %	35.5 %

Question 1.4. Setting the two derivatives to True and False, we study the influence of two types of normalization : across channels and per channel. We understand "across channels normalization" as centering and scaling each speech feature dimension by computing mean and variance across the whole dataset, and "per channel normalization" as computing means and variances for each sample. The advantage of the second one is an adaptability to the conditions of the recording, which are not necessarily the same for the training and for the testing (i.e. the user may use our system in environments unseen during the training). However, according to our results, we obtain better performance with a normalization across channels.

Across channels normalization	**Per channel normalization**	**Accuracy on train set**	**Accuracy on valid set**	**Accuracy on test set**
True	False	62.1 %	39.1 %	37.9 %
False	False	28.9 %	27.6 %	25.0 %
False	True	66.9 %	33.8%	34.1%

From now on, we only consider a normalization across channels, and we study the addition of noise. To do so, the voice command signal and the noise should not be simply added, because some noise are louder than voice commands. Summing these signals makes the command inaudible. To decide how much noise to add, we use the ratio of the RMS of the two signals. Augmenting the training set of 30 % led to an accuracy of 58.6 % on the training set, 40.1 % on the validation set, and 36.6 % on the test set. Hence augmenting the training set is an interesting pre-processing step, however this is something we generally do when we lack of training examples. We finally study the influence of the size of the training set (without augmentation or noise addition). As we could have guessed, we obtain better results when the training set is large (because many situations have been encountered during the training, yielding in a model with a better generalization). However, the bigger the training set, the longer the training.

Size of the training set	**Accuracy on train set**	**Accuracy on valid set**	**Accuracy on test set**	**CPU time**
4500	78.4 %	36.4 %	32.3 %	- -
9000	62.1 %	39.1 %	37.9 %	-
18000	54.8 %	42.9 %	42.0 %	+
27000	51.4 %	46.0 %	47.3 %	++

Question 1.5. For computational time reasons, we restrict ourselves to a training set of 9000 samples, augmented by 30 %. We do not try to fine tune the logistic regression classifier, but we introduce two new architectures : a fully-connected network, and a 2D convolutional neural network. We test these two architectures on the previously tune MFCC and also directly on mel-log filterbanks. For the 2D CNN, we preprocess the data so that they have the shape of an image. As presented below, our best model is the 2D CNN on mel-log filterbanks. This is not surprising, since 2D CNN are designed to recognize patterns. Indeed, when a word is said, it can be found at different locations on the signal. A CNN easily captures the interesting part and analyzes it, whereas the fully connected network has more difficulties. Note also that the kernels are not square, and have been designed to facilitate the retrieval of information.

Network	**Method**	**Accuracy on train set**	**Accuracy on valid set**	**Accuracy on test set**
MLP	melfbank	76.6 %	68.8 %	70.0 %
CNN	melfbank	94.2 %	84.6 %	86.8 %
MLP	MFCC	91.0 %	60.3 %	61.9 %
CNN	MFCC	86.4 %	74.0 %	74.0 %

Question 1.6. For the rest of this practical assignment, we will use a CNN on melfilterbanks, since this model achieves (approximately) 85 % on the test set. The confusion matrix on Fig. 1 shows that the harder words to classify are *three* and *tree*, which are often confused, as well as *go* and *no*, which are often confused too.

We conclude this first part with three important remarks. From question 1.1 to 1.4, the parameters have been optimized sequentially. If we think in terms of minimizing a cost function where the variables would be the hyperparameters we fined tune, we have to remark that we did a descent coordinate by coordinate, which is known to be irrelevant. This can explained some weird results, such as the ones we obtain for the delta and delta_delta study. Moreover, for these study, a seed has been used to get rid off randomness. If we remove the seed, the results have a very high variance. Finally, we mention that some files in the commands folder have the wrong label.

2 Classification of segmented voice commands

Question 2.1. The prior probability of each word to be equal is approximated when we train the discriminator with a balanced training set, fixing a given number of examples per class. The corresponding line is `elif train_labels.count(label) < nb_ex_per_class`.

Question 2.2. S, D, I, and N are non-negative quantities, so WER>0. However, S+D+I may be larger than N (we may have an arbitrary number of insertions for example), so it is possible to have WER>100.

Question 2.3. In the example present in the provided notebook, we had `go marvin one right stop` for the true sentence and `up marvin one on sheila` for the predicted one, resulting in a WER of $0.6 = 3/5$. Indeed, there were 3 substitutions out of 5 words ("go" $\leftarrow$ "up", "right" $\leftarrow$ "on", and "stop" $\leftarrow$ "sheila").

Question 2.4. We note w_k the k-th word of the sequence. The Bigram approximation formula of the language model is $\mathbb{P}(w_k|w_{k-1},...,w_1,w_0) = \mathbb{P}(w_k|w_{k-1})$.

Question 2.5. The entries of the transition matrix storing the bigrams are $(\mathbb{P}(w_k = j|w_{k-1} = i))_{i,j}$. A simple way to compute these probabilities would be to count the bigrams and divide by the sum of the row:

$\frac{\#(i,j)}{\#i}$. However, we have a problem if the bigram (i, j) does not appear in the training sample. We solved this issue with a 1-Laplace smoothing, i.e. adding 1 to every bigram counts. The estimate of the probability becomes: $\frac{\#(i,j)+1}{\#i+|V|}$ with $|V|$ the size of the vocabulary.

Question 2.6. If we increase N, we obtain a more accurate model of the language (advantage), but the transition matrix (or tensor if $N > 2$) has M^N (M the number of words) coefficients,[1] and the space complexity increases exponentially with N (drawback). Moreover, the transition tensor will be very sparse, which can be seen as a drawback or an advantage if we know how to handle efficiently sparse tensors.

Question 2.7. We note M the number of words we consider, B the beam size and L the length sequence. The time complexity of the beam search algorithm is $\mathcal{O}(LBM\log(BM))$ assuming that the sorting algorithm has a complexity of $\mathcal{O}(n\log(n))$. Note that, since B is small, it is also possible to directly search the highest values without sorting the arrays, which gives a complexity of $\mathcal{O}(LBM)$. The space complexity of this algorithm is $\mathcal{O}(BM)$.

Question 2.8. We note $v_k(j)$ the probability to be in state j at step k, p_L the language model and p_A the acoustic model. We have : $v_k(j) = \max\{v_{k-1}(j') \times p_L(j|j')\} \times p_A(j|k)$, where we take the maximum over all the states j'. $p_L(j|j')$ is given by the transition matrix and $p_A(j|k)$ is our discriminator. The time complexity of the Viterbi algorithm corresponds to the complexity of the "forward step", i.e. $\mathcal{O}(LM^2)$. Its space complexity is $\mathcal{O}(LM)$ for a bigram language model.

Question 2.9. Viterbi algorithm is an exact method, whereas beam-search is not. So we expect to obtain better results with Viterbi. However, in practice, beam-search is slightly better than Viterbi, which is surprising. As expected, we observe experimentally that the greedy decoder yields the worst results, since there is no language model for this decoder. For the results below, we used 1-Laplace smoothing.

Decoder	**WER on train set**	**WER on test set**
Greedy	42.2 %	37.3 %
Beam search	20.3 %	16.9 %
Viterbi	22.7 %	19.1 %

Question 2.10. The systematic errors I have remarked are a confusion between "sheila" and "marvin", and also between "cat" and "tree". See the notebook for more examples of failure.

Question 2.11. To face rare seen words (or sequence of words), we have implemented 1- and 2- Laplace smoothing, and λ-smoothing presented in class. Note that the λ-smoothing we have implemented uses 1-Laplace smoothing. The effects on the WER are summarized in the table below. In order to face Out Of Vocabulary words, one could change the model so that there is no probability distribution as output but rather a score. If the score is too low, then the word is classified as OOV.

Decoder Viterbi	**WER on train set**	**WER on test set**
1-Laplace smoothing	22.7 %	19.1 %
2-Laplace smoothing	27.7 %	19.9 %
λ smoothing ($\lambda = 0.7$)	27.7 %	23.2 %

Question 2.12. To optimize jointly an acoustic model and a language model, a possibility is to perform end-to-end speech recognition with a LSTM (or other efficient RNN-based models).

[1]This is an approximation. For our bigram model, we have 31×30 coefficients.

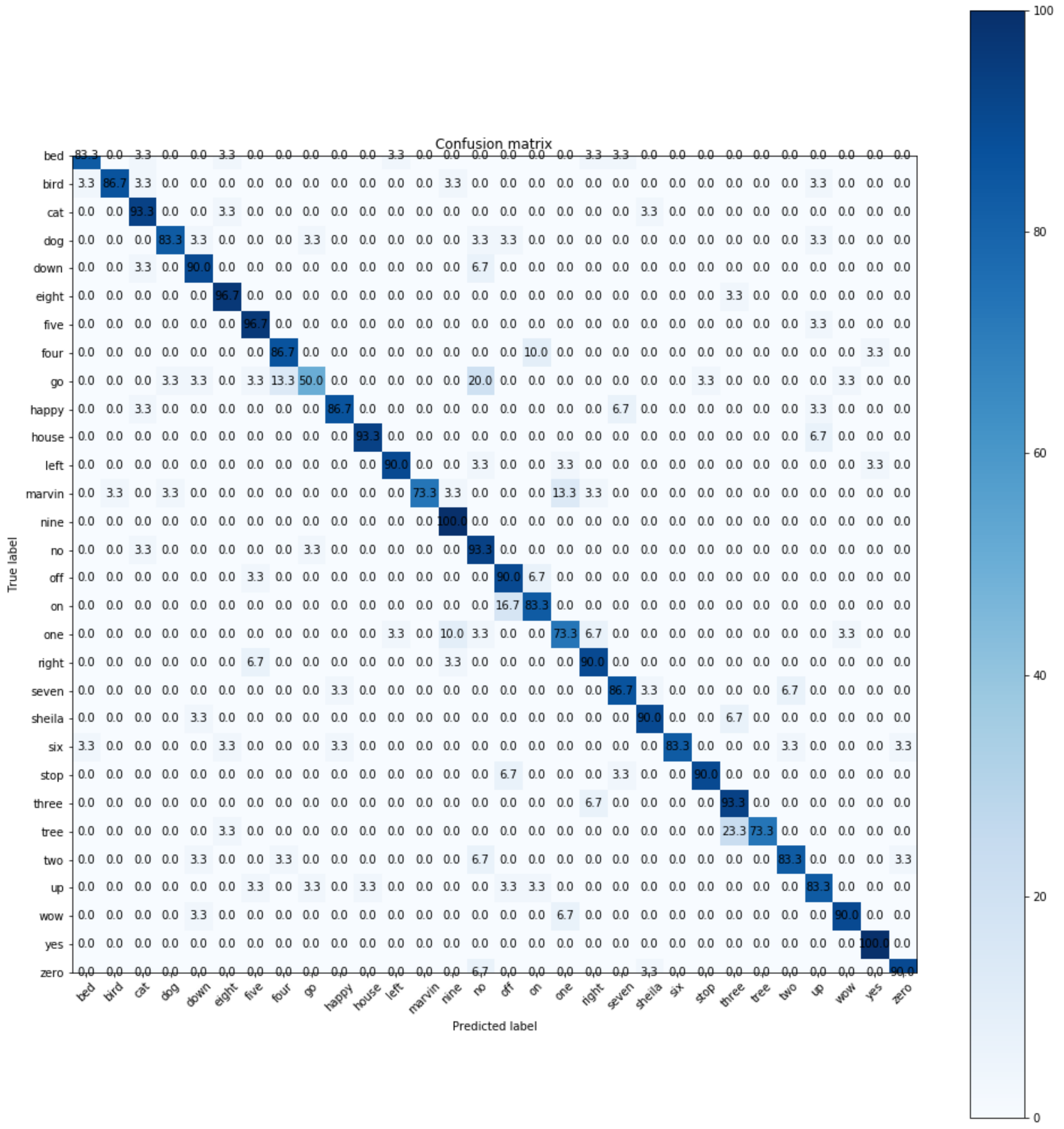
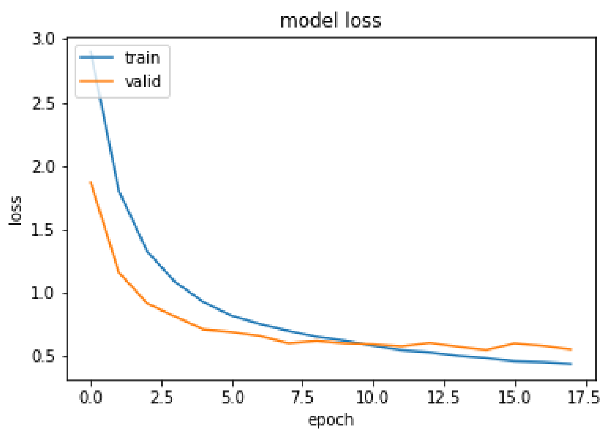


Figure 1: Normalized confusion matrix for the cnn model trained directly on mel-log filterbanks



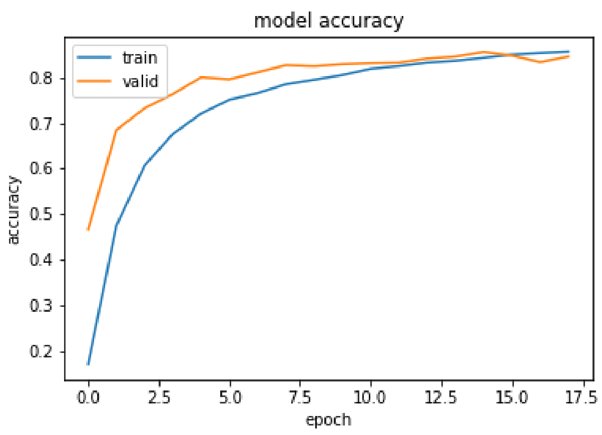


Figure 2: Loss and accuracy of the cnn model over epochs