Processing steps for classification

# Data preparation and quick execution

Organize the folders wav (with waveform files) and the software (with file execute\_all.py) in the same folder.

Edit the file execute\_all.py to choose the options.

Execute:

python execute\_all.py

# DNN steps for mel spectrogram assuming Prof. Meyer’s dataset

1) Start with folders wav and whistled-speech in the same folder.

07/22/2022 10:30 AM <DIR> wav

07/22/2022 06:07 PM <DIR> whistled-speech

2) The wav folder has the files in format wav and named as:

...

07/22/2022 10:30 AM 85,786 Ricardo\_caya\_1.wav

07/22/2022 10:30 AM 104,578 Ricardo\_caya\_2.wav

07/22/2022 10:30 AM 104,126 Tini-caba\_1.wav

07/22/2022 10:30 AM 105,034 Tini-caba\_2.wav

07/22/2022 10:30 AM 133,558 Tini-caca\_1.wav

...

3) Choose the name for a general folder in a parent folder. The default is “../general”. Create the file with list of wav files and their labels. Also, note the histograms. David has 30 words and JoseLuiz only 7.

python automation/create\_label\_file.py ../wav ../general

Created output folder ../general

Label Histogram:

['caba', 'cada', 'caga', 'caca', 'capa', 'casa', 'cata', 'caya']

[15 16 16 22 15 15 13 15]

Speaker Histogram:

['David', 'Fuyi', 'Idir', 'Joel', 'JoseLuiz', 'Ricardo', 'Tini']

[30 19 16 21 7 18 16]

Wrote file ../general\wavs\_labels.csv

Wrote file ../general\labels\_dictionary.json

The files ../general/ and ../general\labels\_dictionary.json will be used in all simulations, with different frontends and ML models.

4) Choose the features (magnasco, mel, stft) and the dimensions D (frequency) and T (time), compose the output folder name with them and the name of the file with the features. For instance:

python feature-extraction\general\_frontend.py --D 100 --T 60 --output\_dir ..\mel\_D100T60 --features mel --normalization minmax --log\_domain

...

cata from Tini-cata\_2.wav

caya from Tini-caya\_1.wav

caya from Tini-caya\_2.wav

Wrote file ..\mel\_D100T60\features\_melD100T60.hdf5

Considering original features

Frequency dimension range = 100 100

Time dimension range = 298 744 <=== HOW TIME VARIED FOR EACH EXAMPLE

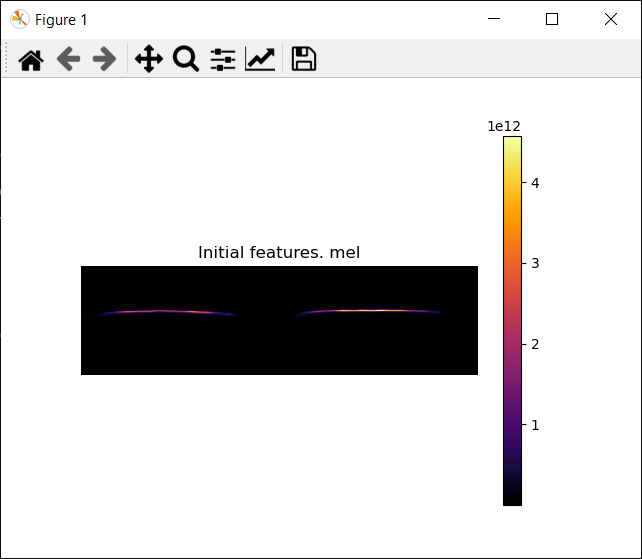
Two files are written in the output folder:

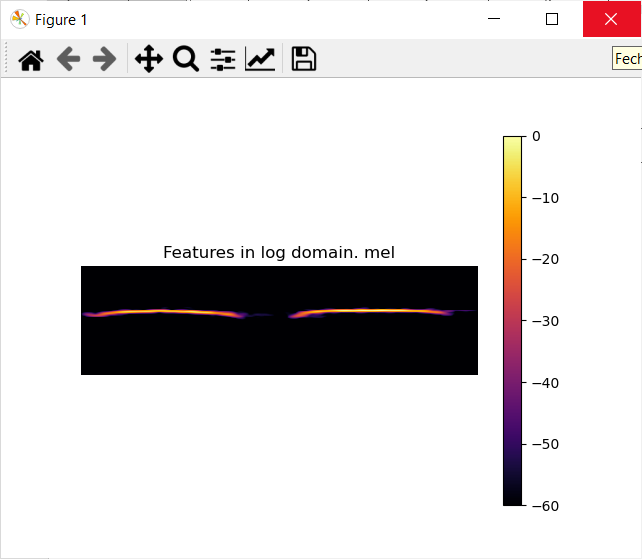
07/22/2022 07:46 PM 132 commandline\_args.txt

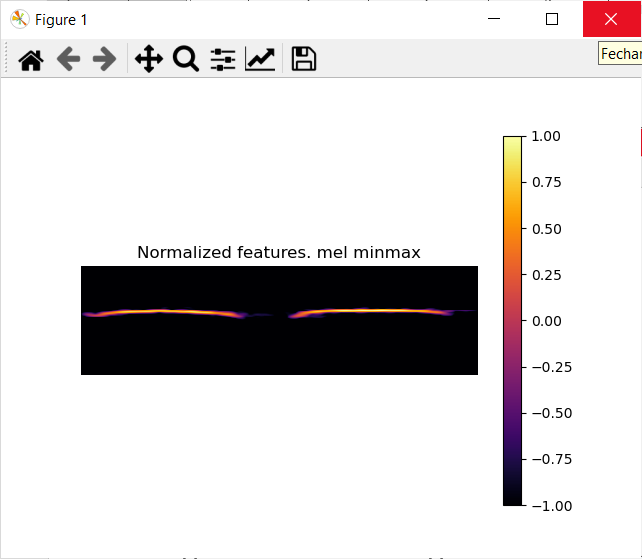
07/22/2022 07:46 PM 6,098,556 features\_melD100T60.hdf5

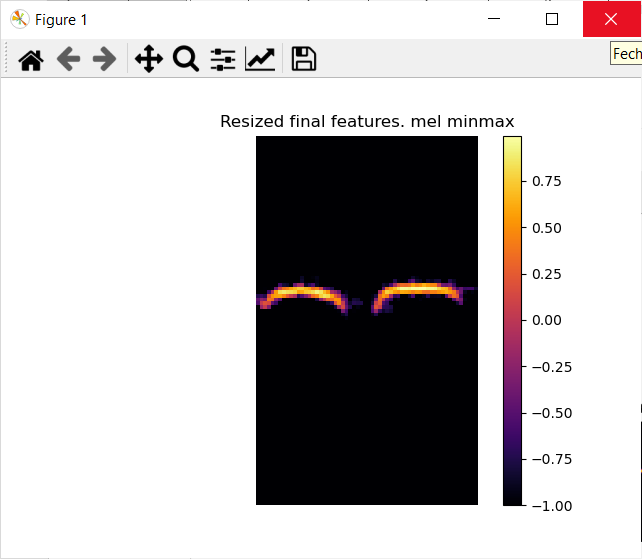
commandline\_args.txt has the command that was used.

You can observe the features being processed in the main 4 stages using the --show\_plot option. For instance:

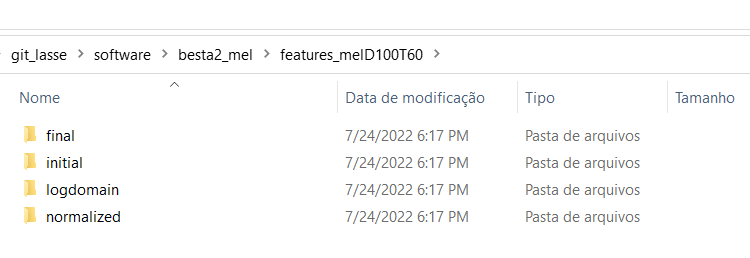








Obs: if you use the option --save\_plots, then the script will save PNG files with the initial, final and intermediate features. They will be located in different folders and there will be one PNG per wav file as described below:



5) Study the features you created:

python feature-extraction\signal\_statistics.py --input\_file ..\mel\_D100T60\features\_melD100T60.hdf5

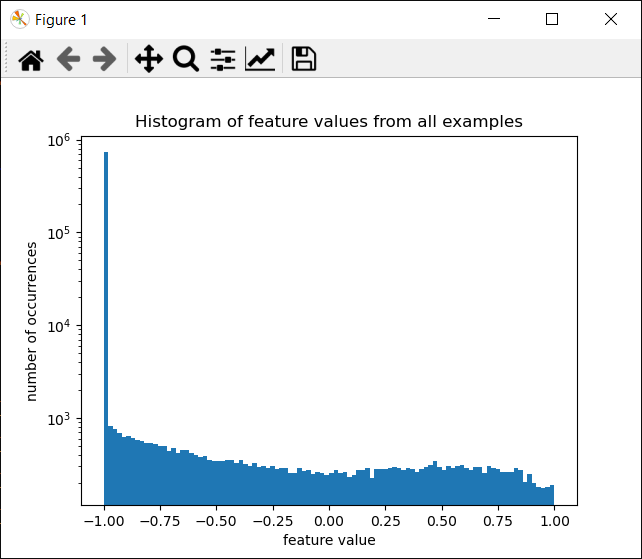
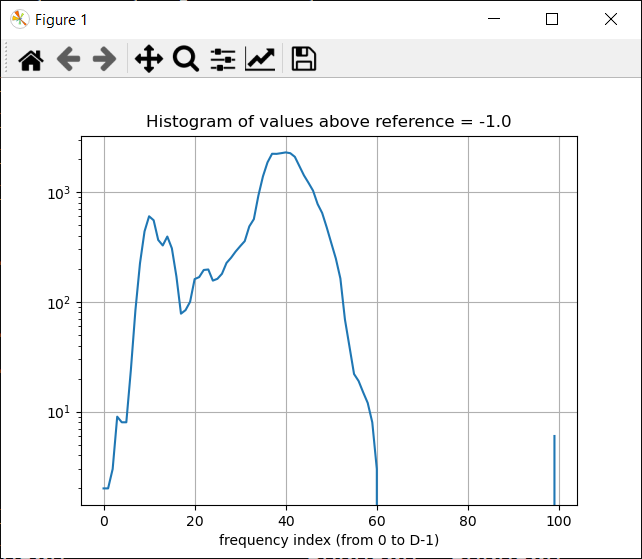


Figure below shows that when the frequency index is larger than 60, there is no useful information.



For each frequency dimension, number of occurrences above threshold

[ 2 2 3 9 8 8 24 84 224 438 601 553 366 326

393 307 170 78 84 100 161 168 195 197 156 162 180 226

252 287 321 357 486 566 929 1388 1862 2229 2225 2257 2294 2256

2091 1721 1417 1212 1028 777 644 476 343 248 163 69 39 22

19 15 12 8 3 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 6]

The frequency indices below are non-informative and could be removed by cut\_features.py:

(array([61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,

78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94,

95, 96, 97, 98], dtype=int64),)

6) Say we decide to keep only the frequency range with indices from 6 to 60, use

*The syntax has changed. It now requires --input\_folder*

python feature-extraction\cut\_frequencies.py --input\_file ..\mel\_D50T60\features\_melD50T60.hdf5 --Dmin 3 --Dmax 29 --input\_folder ..\mel\_D50T60\features\_melD50T60\features\_no\_resizing

python feature-extraction\cut\_frequencies.py --input\_file ..\mel\_D100T60\features\_melD100T60.hdf5 --Dmin 6 --Dmax 60

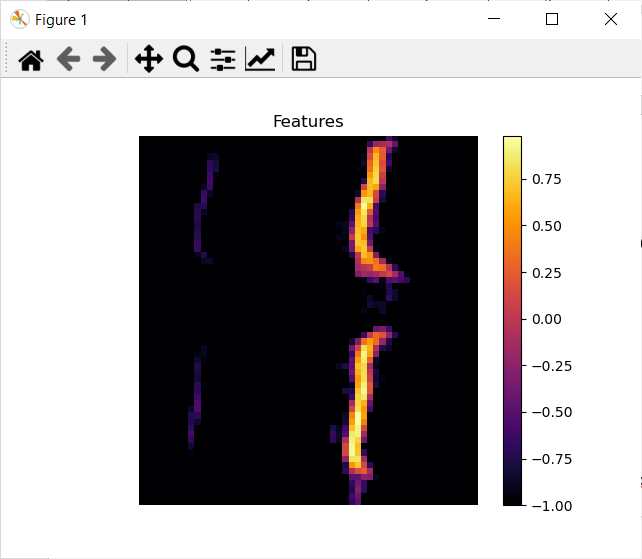
Original D= 100 . New D= 55

Wrote file ..\mel\_D100T60\features\_melD100T60\_cutD6\_60.hdf5 with data of dimension (127, 60, 55)

Note that D was reduced from 100 to 55.

7) Observe the new features using the --show\_plot option of signal\_statistics.py

python feature-extraction\signal\_statistics.py --input\_file ..\mel\_D100T60\features\_melD100T60\_cutD6\_60.hdf5 --show\_plot



Recall that we use the transpose, such that the dimension of the features is time x frequency, not frequency x time as a typical spectrogram.

8) Now train the ML model with mixed speakers:

python machine-learning\convnet\_classifier\_hdf5.py --input\_file ..\mel\_D100T60\features\_melD100T60\_cutD6\_60.hdf5

Epoch 19/20

90/90 [==============================] - 3s 31ms/step - loss: 2.0722 - accuracy: 0.1667 - val\_loss: 2.0396 - val\_accuracy: 0.2727

Epoch 20/20

90/90 [==============================] - 3s 31ms/step - loss: 2.0708 - accuracy: 0.2222 - val\_loss: 2.0381 - val\_accuracy: 0.1818

1/1 [==============================] - 0s 403us/step - loss: 2.1382 - accuracy: 0.0385

Wrote file ..\mel\_D100T60\nn\_model\_DanielTrueDisjointFalse\_features\_melD100T60\_cutD6\_60.hdf5

8.1) In case you want to continue the training of a DNN file that has been saved, use:

python machine-learning\convnet\_classifier\_hdf5.py --input\_file ..\mel\_D100T60\features\_melD100T60\_cutD6\_60.hdf5 –continue\_training

9) Evaluate the model using mixed speakers

python machine-learning\evaluate\_classifier.py --input\_file ..\mel\_D100T60\features\_melD100T60\_cutD6\_60.hdf5 --input\_model ..\mel\_D100T60\nn\_model\_DanielFalseDisjointFalse\_features\_melD100T60\_cutD6\_60.hdf5

The software is capable of indicating the error and match for each example of the test set:

ERROR: For Fuyi\_caya\_2.wav , model predicted cada

ERROR: For Joel\_cata\_1.wav , model predicted cada

ERROR: For David\_capa\_1.wav , model predicted cada

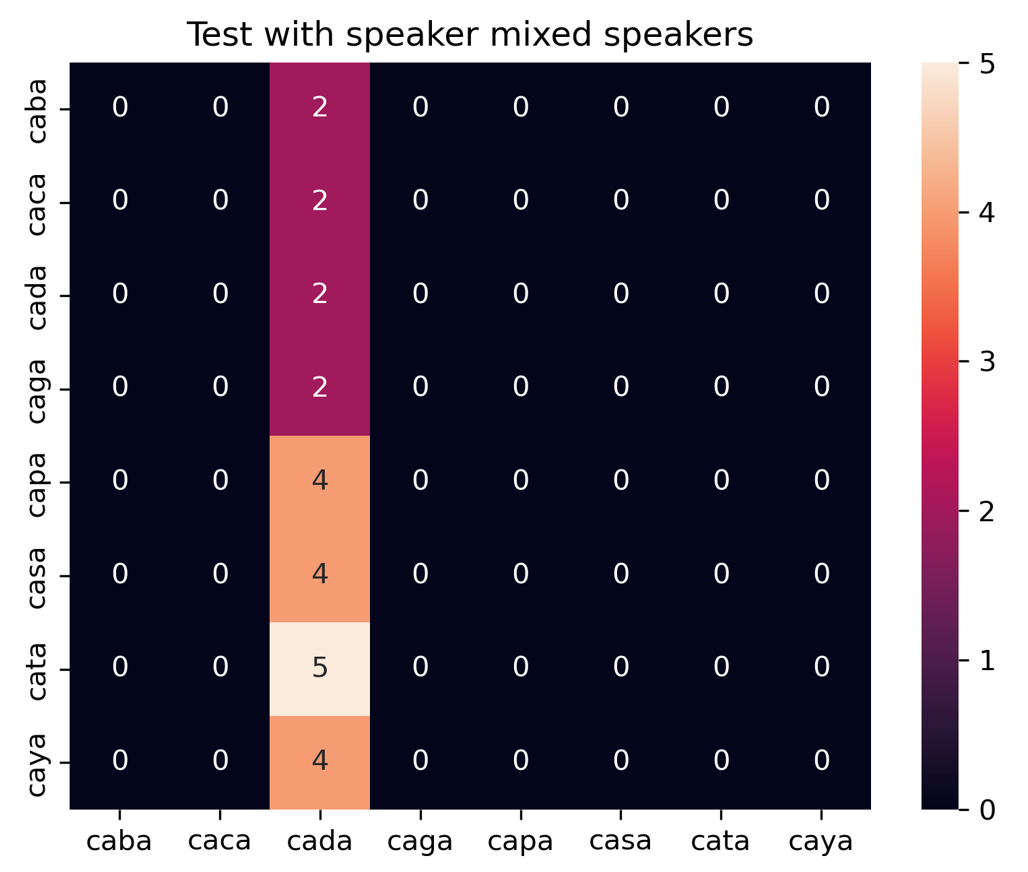
ERROR: For Tini-caya\_1.wav , model predicted cada

ERROR: For Ricardo\_cada\_1.wav , model predicted cada

Misclassification error = 92.0 %

Wrote PNG file ..\mel\_D100T60\nn\_model\_DanielTrueDisjointFalse\_features\_melD100T60\_cutD6\_60\_mixed\_speakers\_conf\_matrix.png

And creates a confusion matrix:



10) Now train the ML models with disjoint speakers. This creates a NN model for each test speaker, with the training of the model using all examples of all other speakers.

python machine-learning\convnet\_classifier\_hdf5.py --input\_file ..\mel\_D100T60\features\_melD100T60\_cutD6\_60.hdf5 --disjoint\_speakers

99/99 [==============================] - 3s 30ms/step - loss: 2.0920 - accuracy: 0.1111 - val\_loss: 2.1188 - val\_accuracy: 0.1667

1/1 [==============================] - 0s 994us/step - loss: 2.0932 - accuracy: 0.1250

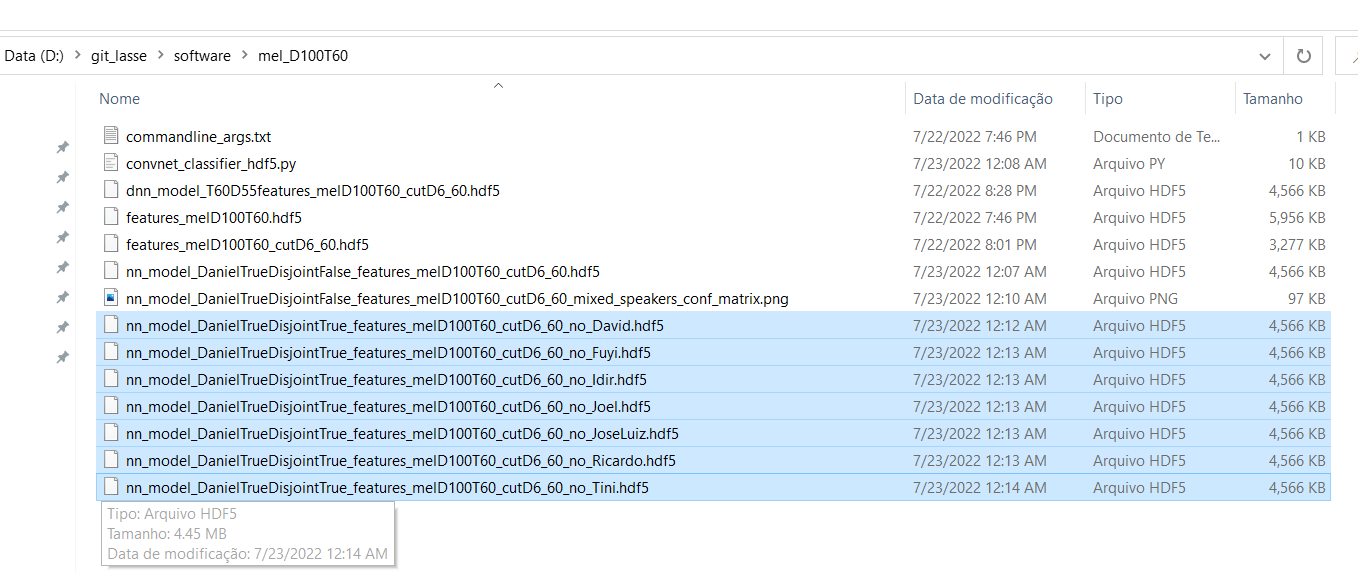
Wrote file ..\mel\_D100T60\nn\_model\_DanielTrueDisjointTrue\_features\_melD100T60\_cutD6\_60\_no\_Tini.hdf5

['David' 'Fuyi' 'Idir' 'Joel' 'JoseLuiz' 'Ricardo' 'Tini']

Test scores

[[2.115396499633789, 0.06666667014360428], [2.079620838165283, 0.10526315867900848], [2.1285505294799805, 0.125], [2.085265874862671, 0.095238097012043], [2.0331625938415527, 0.1428571492433548], [2.151801586151123, 0.1111111119389534], [2.0932211875915527, 0.125]]

Note that models in blue below:



11) Evaluate the model using disjoint speakers

python machine-learning\evaluate\_classifier.py --input\_file ..\mel\_D100T60\features\_melD100T60.hdf5 --input\_model ..\mel\_D100T60\nn\_model\_DanielTrueDisjointTrue\_features\_melD100T60\_cutD6\_60 --disjoint\_speaker

CORRECT: Tini-casa\_1.wav , model predicted casa

CORRECT: Tini-casa\_2.wav , model predicted casa

ERROR: For Tini-cata\_1.wav , model predicted casa

ERROR: For Tini-cata\_2.wav , model predicted casa

ERROR: For Tini-caya\_1.wav , model predicted casa

ERROR: For Tini-caya\_2.wav , model predicted casa

Misclassification error = 87.5 %

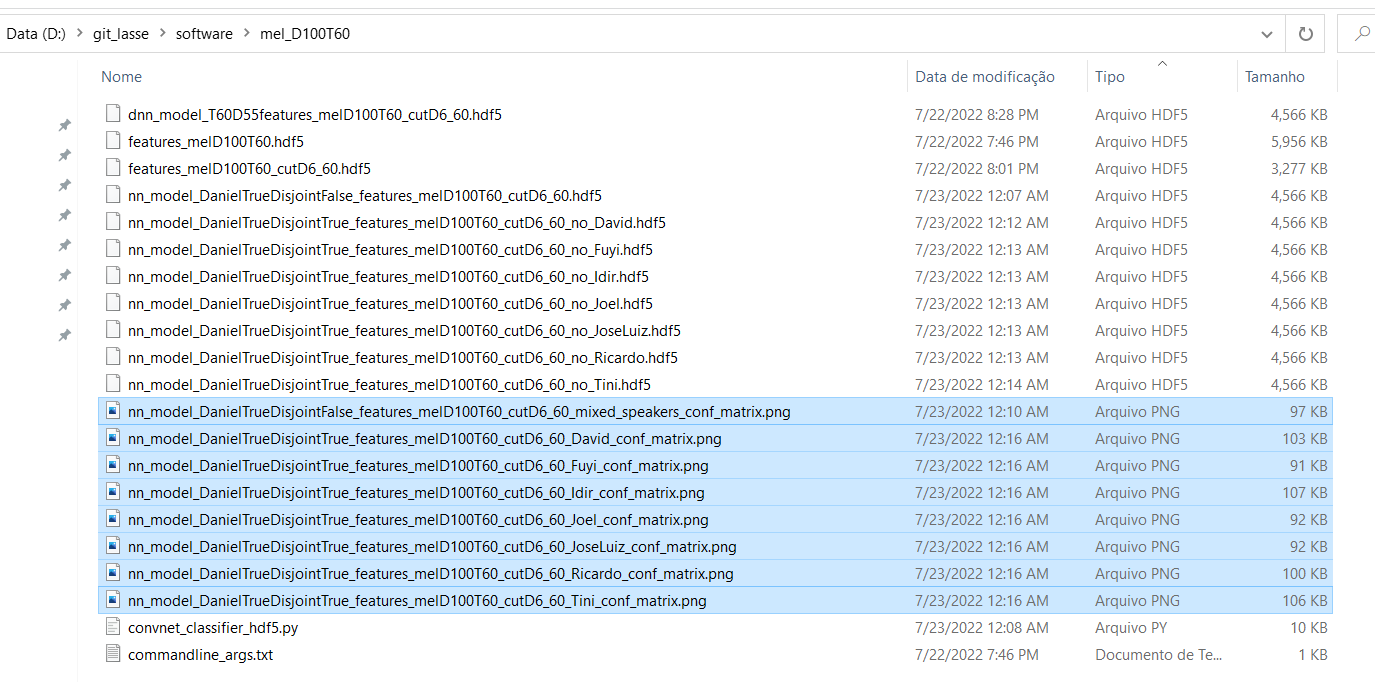
Wrote PNG file ..\mel\_D100T60\nn\_model\_DanielTrueDisjointTrue\_features\_melD100T60\_cutD6\_60\_Tini\_conf\_matrix.png

['David' 'Fuyi' 'Idir' 'Joel' 'JoseLuiz' 'Ricardo' 'Tini']

All error rates in %:

[96.66666666666667, 89.47368421052632, 87.5, 90.47619047619048, 85.71428571428571, 88.88888888888889, 87.5]

Note all confusion matrices as PNG below:



# DNN steps for Magnasco’s reassigned spectrogram

Using Magnasco’s:

python feature-extraction\general\_frontend.py --D 100 --T 60 --output\_dir ..\magnasco\_D100T60 --features magnasco --normalization maggie --log\_domain

leads to

cata from Tini-cata\_2.wav

caya from Tini-caya\_1.wav

caya from Tini-caya\_2.wav

Wrote file ..\magnasco\_D100T60\features\_magnascoD100T60.hdf5

Considering original features

Frequency dimension range = 370 370

Time dimension range = 310 775

python feature-extraction\signal\_statistics.py --input\_file ..\magnasco\_D100T60\features\_magnascoD100T60.hdf5

For each frequency dimension, number of occurrences above threshold

[ 0 0 0 0 0 1 4 10 6 4 2 3 3 7 2 3 2 2

6 7 5 6 12 50 93 202 280 318 154 114 159 122 119 70 21 24

36 51 61 47 90 35 23 23 26 33 42 40 33 52 43 66 79 110

256 303 414 417 352 352 535 366 317 263 206 170 139 146 78 96 45 18

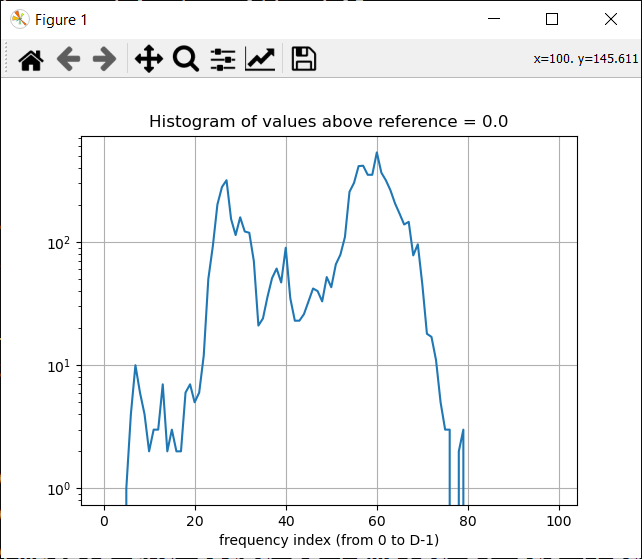
17 11 5 3 3 0 2 3 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0]

The frequency indices below are non-informative and could be removed by cut\_frequencies.py:

(array([ 0, 1, 2, 3, 4, 77, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90,

91, 92, 93, 94, 95, 96, 97, 98, 99], dtype=int64),)



python feature-extraction\cut\_frequencies.py --input\_file ..\magnasco\_D100T60\features\_magnascoD100T60.hdf5 --Dmin 23 --Dmax 74

Original D= 100 . New D= 52

Wrote file ..\magnasco\_D100T60\features\_magnascoD100T60\_cutD23\_74.hdf5 with data of dimension (127, 60, 52)

python machine-learning\convnet\_classifier\_hdf5.py --input\_file ..\magnasco\_D100T60\features\_magnascoD100T60\_cutD23\_74.hdf5

Epoch 20/20

90/90 [==============================] - 3s 30ms/step - loss: 2.0741 - accuracy: 0.2000 - val\_loss: 2.0635 - val\_accuracy: 0.1818

1/1 [==============================] - 0s 0s/step - loss: 2.0853 - accuracy: 0.0769

Wrote file ..\magnasco\_D100T60\nn\_model\_DanielTrueDisjointFalse\_features\_magnascoD100T60\_cutD23\_74.hdf5

python machine-learning\convnet\_classifier\_hdf5.py --input\_file ..\magnasco\_D100T60\features\_magnascoD100T60\_cutD23\_74.hdf5 --disjoint\_speakers

Epoch 2/2

99/99 [==============================] - 3s 33ms/step - loss: 2.0812 - accuracy: 0.1313 - val\_loss: 2.0668 - val\_accuracy: 0.2500

1/1 [==============================] - 0s 995us/step - loss: 2.0809 - accuracy: 0.1250

Wrote file ..\magnasco\_D100T60\nn\_model\_DanielTrueDisjointTrue\_features\_magnascoD100T60\_cutD23\_74\_no\_Tini.hdf5

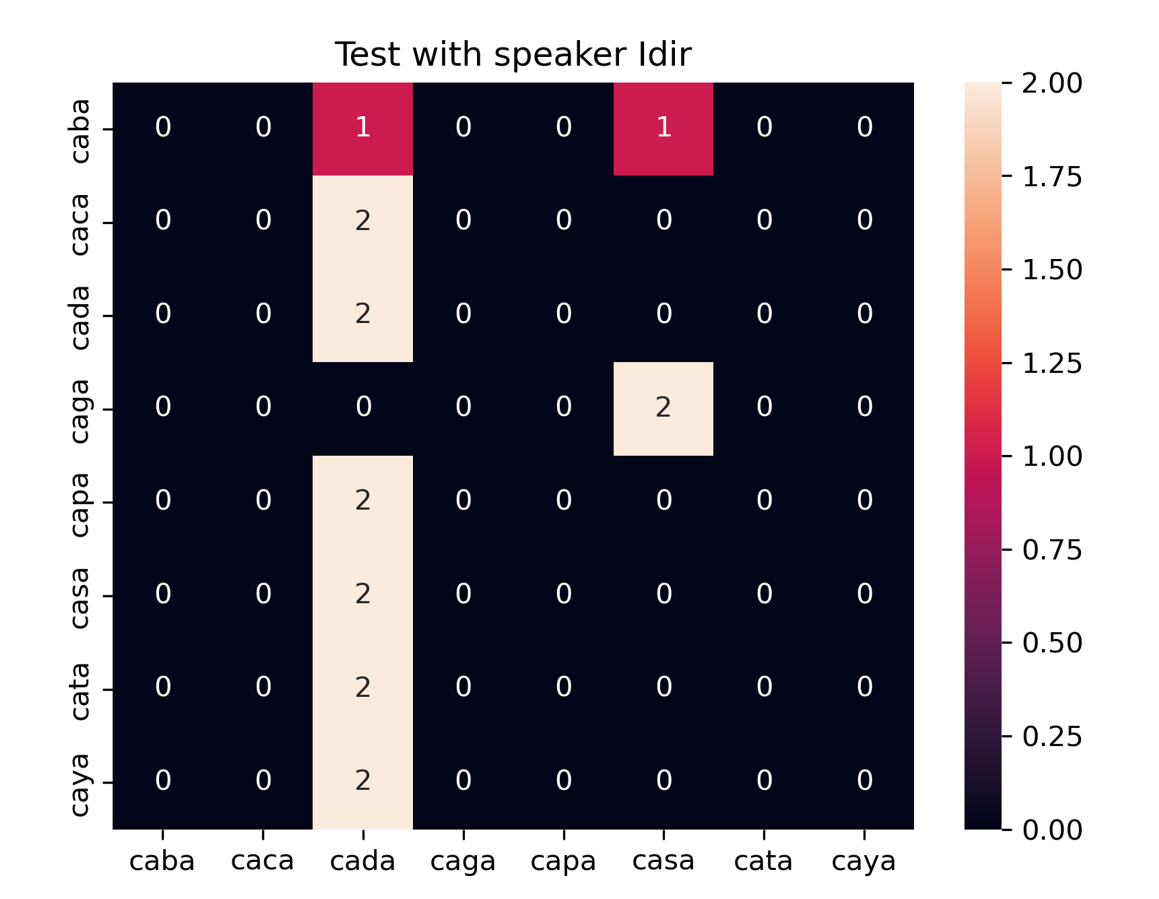
['David' 'Fuyi' 'Idir' 'Joel' 'JoseLuiz' 'Ricardo' 'Tini']

Test loss (score[0]) and accuracy (score[1]):

[[2.084789514541626, 0.06666667014360428], [2.0811362266540527, 0.10526315867900848], [2.0801424980163574, 0.125], [2.0701024532318115, 0.1428571492433548], [2.0805180072784424, 0.0], [2.0807478427886963, 0.0555555559694767], [2.080935001373291, 0.125]]

python machine-learning\evaluate\_classifier.py --input\_file ..\magnasco\_D100T60\features\_magnascoD100T60\_cutD23\_74.hdf5 --input\_model ..\magnasco\_D100T60\nn\_model\_DanielTrueDisjointTrue\_features\_magnascoD100T60\_cutD23\_74.hdf5 --disjoint\_speakers

Creates confusion matrices as PNG files:



# Using several classifiers

After running the front end:

python machine-learning\other\_classifiers.py --input\_file ..\mel\_D100T60\features\_melD100T60\_cutD6\_60.hdf5

# Some (not all) steps using **variable dimension**

8) python machine-learning\variable\_dim\_encoder\_clusterer.py --input\_folder ..\output\_magnasco\_D120\_T110\features\_magnascoD120T110\features\_no\_resizing

python machine-learning\variable\_dim\_encoder\_clusterer.py --input\_folder ..\mel\_D50T60\features\_melD50T60\features\_no\_resizing\_cutD3\_29

This script does not use “resize” but matrices with dimension imposed by the shortest example (with respect to time duration).

create the models, especially the "encoder" that converts the T x D matrices into a vector of dimension LATENT (we used 128).

variable\_dim\_encoder\_clusterer.py provides support to arrays with variable-length (T varies). It looks at the smallest T, called here Tmin, and works with subarrays of dimension Tmin x D.