I performed **2 tonic classification tasks in the mridangam\_stroke dataset**, which you can find in 2 different folders and with different notebooks and .csv data files.

1 has been performed with low level features, and it has been useful for me to get more familiar with Python and the libraries used (you can see I commented

Many lines of code, had a look at the documentation, ecc.). The changes in the code are mainly done in order to create different models, trained on the same essentia.MusicExtractor() low level features

But different ground-truth class labels (tonics rather than strokes types/names).

With the other I experimented a bit, by using other features (TONAL features from essentia.MusicExtractor()), which I thought could be more efficient for classifying tonics.

It turns out they are not :) :)

Other changes include for example the number of input nodes in the neural network model (in order to reflect the fewer features extracted - 25 rather than 45), and of course different values for re-sampling the data

In the pre-processing step (in order to have balanced data and not to produce bias in the models).

I also tried to create another different Neural Network model for tonic classification (trying to improve the accuracy), because it seemed weird to me that the batch (batch size) dimension was ignored when creating the Sequential model (but was then fed as an argument to the .ft() method…). Also, I saw a “None” dimension in the printed model summary.

After looking at the documentation and here <https://stackoverflow.com/questions/47240348/what-is-the-meaning-of-the-none-in-model-summary-of-keras> it turns out that the batch size dimension is “automatically” created and dealt with (when creating the model, we don’t have to think about it, that is, we have to only think to create a Neural Network structure for only 1 training example).

I also performed an **instrument classification task on the TinySOL dataset**, using low level features. I lowered the number of “default” epochs (see comments in the code).

Also, by using feature reduction/metric learning (Sound\_classification.ipynb file), I tried to reduce the number of features and hence the number of nodes in the input layer of the Neural Network model (by extension, also reducing the number of hidden layers). By only selecting 10 features (from feature n. 50 to feature n. 60) I obtained a model with an accuracy around 0.5. By selecting 40 features (less than a half of the original 84 features) I obtained a model with accuracy of 0.86 (against 0.95 with 84 input features). All these choices were of course guided by the Linear Discriminant Analysis projections.

For all notebooks, I provided the accompanying data.csv file, so that you don’t have necessarily to recompute the features.

Matteo