**Work description:**

I used a database of 4 different songs when each song has 4 different versions. The database stored in the “raw\_data” folder. Each song has its own folder (named after the song) in which stored the 4 versions of this specific song.

70% of the database was used for training and 30% was used for testing.

The GetData, get\_transformed\_data and the transform functions:

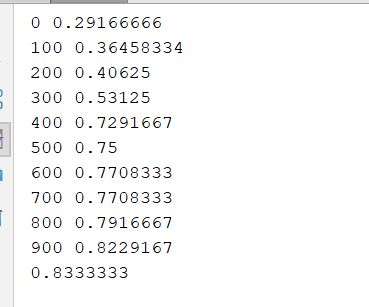
At the first run the program goes over all the files and transforms them to a single .npy file which saved in the “transformed\_data” folder as “list of lists”, the version’s labels are saved, as well according to their folder name, in another .npy file as “label”. In result, the “list of lists” contains an array with 16 cells of song data, each of them has 20 cells of sections and each of them has 5000+ cells of features (mfcc) and the ‘label” file contains an array with 320 cells – a label for every section – 20\*16=320. After the first run the program have the access to the files it saved previously so it doesn’t transform the data more than once.

The neural network is executed by the “train\_network” function which receives both the data – x and the labels – y. the weights, biases and the normalization function – leaky\_relu are being updated in each of the 1000 iterations. finely we perform, the sofmax, and the train step (0.1).   
in order to track the progress of the learning we calculate the correct prediction,the accuracy and the cross entropy which also will be updated and printed every iteration.

Main function:  
In order for the network to work I had to transform the data into an array when each of objects has to have the same shape , this was done by taking only the first 5000 cells of each of the 20 sections and place all of those sections next to each other in one row inside the x array (row 146 in code) which resulted an array with 320 cells total.  
then the data is being normalized (vector normalization) and the indexes of the x and y are being shuffled and divided to train and test purposes. Next, we encode the labels into a numeric form, so the network could understand. (first song – 1,0,0,0, second – 0,1,0,0 etc.)

I tried to change verity of parameters some of them got better results and some of them didn’t:

* Train step (row 112) – I started with 0.5 step and got 73% of accuracy I tried 0.1 and the results got better with accuracy of 82% then I tried 0.01 but this made the results much wars only 37% of accuracy.
* Normalization (row 156) – without it I got very bad results with accuracy of 29% and with is was a much better result 75%-84%.
* Hidden layer sizes (row 90) - I tried (100,100) and got a result of 72% and then tried 50 which improved the result to 83 of accuracy.

Finely I got a result of about 83% of accuracy.