This document provides information for using the scripts to reproduce the experiments done for the project. Note that all scripts have been tested with a Python 2 interpreter

1. Data Preparation

1. Go here to download the MagnaTagATune dataset: http://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset. Download the Clip metadata in the CSV format and the Audio Data. Unzip all the audio data

2. The audio data is in MP3, but needs to be converted to WAV data. First the correct folder structure to store the WAV data is necessary. The shell script datapreparation/create_wav_folders.sh can do this. The script data_preparation/convert.py will do the actual conversion. It requires the pydub package: https://github.com/jiaaro/pydub. Edit the script to point it to the folder with mp3s, the folder for wavs and to the downloaded CSV file. It will also output a new CSV file that contains only the clips for which audio was available and that was successfully converted to MP3. Where this file is created can also be edited in the script.

3. Two scripts are provided to split the data in training and test sets. Both scripts make use of the numpy/scipy packages: https://www.scipy.org/

The script data_preparation/create_test_train_songs_mixed.py will create a test and training set in which clips from the same song can be in both sets.

The script data_preparation/create_test_train_songs_separated.py will create test and training sets in which clips from the same song will only occur in either the test or training set. Both scripts will output the test and training set as CSV files. The scripts will need to be edited so they point to the CSV file generated by the convert.py script in the previous step. Where the test and training CSV files are placed can also be edited in the scripts.

The exact splitting by these scripts is random. The exact data sets used for the experiments are also included. Datasets/test_songs_mixed.csv and datasets/train_songs_mixed.csv are test and training sets in which clips from the same song occur in both sets. Datasets/test_songs_separated.csv and datasets/train_songs_separated.csv are the sets in which clips from the same song only occur in either one of them.

2. Recurrent Convolutional Network

All scripts in this section require:

- numpy/scipy packages: https://www.scipy.org/
- Keras: https://keras.io/#installation.
- A Keras backend like tensorflow: https://www.tensorflow.org/install/

They can be found in the recurrent convolutional network folder. Some editing of the scripts in this section will be necessary. All parameters that may need to be edited are shown at the top of the scripts and commented with an explanation. Some more information on all scripts follows below.

1. Training the network

Training the network can be done with the recurrent_convolutional.py script. This script will need to be edited to point it to the training set CSV generated or downloaded in the previous section and to the folder containing the songs in WAV format. The network parameters itself can also be tweaked at the top at this script. This include how large the segmentation window should be, the stride of this

window, how many epochs to train and the filter sizes and number of LSTM nodes. The batch size has a big effect on memory usage. You can try to lower it if the script crashes due to running out of memory. After every epoch the weights are saved to the disk.

2. Evaluating the network

The accuracy of all epochs for the test and training set can be computed with the evaluate_recurrent_convolutional.py script. It expects the weights generated by the training script in the same folder. The script will need to be pointed to the train and test CSVs used to train the network and the songs in wav format. The network parameters can also be edited in this script and should be the same as the ones in the recurrent_convolutional.py script that computed the weights. As evaluating the training set can take a lot of time, this can be turned off. Results are also outputted in text files.

3. Generate top K results

The top K accuracy on the test set can be computed for 1 specific file of weights generated by the training script with the script ktop_recurrent_convolutional.py. The script also needs to be edited to point to the test and train CSVs (train CSVs still necessary as this was used to map the artist to an index during training) and the songs in wav format. Network and segmentation parameters should again match the ones used to train the network. Results will also be stored as text file.

4. Genre Inspection.

To do the genre inspection on the test set, the clips tagged with genres are necessary. This is the file datasets/clips_with_genres.csv. The script to do the genre inspection is genres_recurrent_convolutional.py This script needs to be edited to point to the clips with genres CSV, the test CSV, the same train CSV used during training, and the wav format songs. Network and segmentation parameters should again match those used during training. The script does the genre inspection on a single set of weights. Which one will also need to be edited in the script.

3. Convolutional Network

All scripts in described in this section are found in the convolutional network folder. They are used in the same way and have the same requirements as all the scripts mentioned in section 2. The difference is that these scripts will train and evaluate the original pure convolutional networks instead of the recurrent convolutional networks. The scripts are:

- convolutional.py: Trains the convolutional network
- evaluate_convolutional.py: Evaluates for all epoch computed by the training script. Will output results for both averaged predictions and predictions for each segment individually
- ktop_convolutional.py: Will compute top-K results for all K on a single set of weights
- genres_convolutional.py: Will do genre inspection on a single set of weights. Requires clips_with_genres.csv