MUSICNN: PRE-TRAINED CONVOLUTIONAL NEURAL NETWORKS FOR MUSIC AUDIO TAGGING

Jordi Pons
Music Technology Group
Universitat Pompeu Fabra
jordi.pons@upf.edu

Xavier Serra
Music Technology Group
Universitat Pompeu Fabra
xavier.serra@upf.edu

EXTENDED ABSTRACT

Pronounced as "musician", the *musicnn* library contains a set of pre-trained musically motivated convolutional neural networks [4, 5] for music audio tagging:

https://github.com/jordipons/musicnn

This repository also includes some pre-trained vgg-like baselines [2]. These models can be used as out-of-the box music audio taggers, as music feature extractors, or as pre-trained models for transfer learning.

These models are trained with two different datasets: the MagnaTagATune dataset (the MTT of 19k training songs [3])[1] and the Million Song Dataset (the MSD of 200k training songs [1])[2].

Which pre-trained models are available? Although the main focus of the library is to release pre-trained musically motivated convolutional neural networks, we also provide several vgg-like models[3] (as baselines for comparison). A high-level depiction of the *musicnn* architecture[4] is depicted in the following figure:

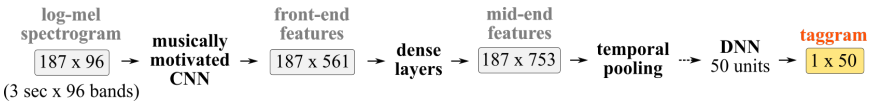

Figure 1: The *musicnn* architecture: a musically motivated convolutional neural network [4, 5].

The following models are available: `MTT_musicnn`, `MSD_musicnn`, `MSD_musicnn_big`, `MTT_vgg`, and `MSD_vgg`. The `MTT` models are trained with the MagnaTagATune dataset, and the `MSD` models are trained with the Million Song Dataset. Given that the Million Song Dataset contains more training data, we also provide a larger *musicnn* model: `MSD_musicnn_big`. Architectural details are accessible online[4][3].

What's the *musicnn* library for? Out-of-the-box music audio tagging. From within python, one can estimate the top-10 tags by simply running:

```
from musicnn.tagger import top_tags
top_tags('music_file.mp3', model='MTT_musicnn', topN=10)
```

[1] The MagnaTagATune 50-tags vocabulary: *guitar, classical, slow, techno, strings, drums, electronic, rock, fast, piano, ambient, beat, violin, vocal, synth, female, indian, opera, male, singing, vocals, no vocals, harpsichord, loud, quiet, flute, woman, male vocal, no vocal, pop, soft, sitar, solo, man, classic, choir, voice, new age, dance, male voice, female vocal, beats, harp, cello, no voice, weird, country, metal, female voice, choral.*

[2] The Million Song Dataset 50-tags vocabulary: *rock, pop, alternative, indie, electronic, female vocalists, dance, 00s, alternative rock, jazz, beautiful, metal, chillout, male vocalists, classic rock, soul, indie rock, mellow, electronica, 80s, folk, 90s, chill, instrumental, punk, oldies, blues, hard rock, ambient, acoustic, experimental, female vocalist, guitar, hip-hop, 70s, party, country, easy listening, sexy, catchy, funk, electro, heavy metal, progressive rock, 60s, rnb, indie pop, sad, house, happy.*

[3] An in-depth depiction of *vgg* architecture and its feature-maps is accessible online via a Jupyter Notebook: https://github.com/jordipons/musicnn/blob/master/vgg_example.ipynb

[4] An in-depth depiction of the *musicnn* architecture (musically motivated CNN) and its feature-maps is accessible online via a Jupyter Notebook: https://github.com/jordipons/musicnn/blob/master/musicnn_example.ipynb

From the command-line, one can also print the top-N tags on the screen (top) or save them to a file (bottom):

```
python -m musicnn.tagger music.au --model 'MTT_musicnn' --topN 10 --print
python -m musicnn.tagger audio.wav -m 'MTT_vgg' --topN 5 --save out.tags
```

What's the *musicnn* library for? Music feature extraction. Out of the extractor, see the example below, one gets the output of the model (the taggram and its associated tags) and all the intermediate representations of it (we refer to those as features). The features are packed in a dictionary and, for the *musicnn* models, you can extract `timbral`, `temporal`, `cnn1`, `cnn2`, `cnn3`, `mean_pool`, `max_pool`, and `penultimate` features[4]. For the *vgg* models, you can extract `pool1`, `pool2`, `pool3`, `pool4`, and `pool5` features[3].

```
from musicnn.extractor import extractor
output = extractor(file, model='MTT_musicnn', extract_features=True)
taggram, tags, features = output
```

What's the *musicnn* library for? Transfer learning. Our pre-trained deep learning models can be fine-tuned, together with an output neural-network that acts as a classifier, to perform any other music task. To assess the utility of our embeddings, we build SVM classifiers on top of several pre-trained models that act as music feature extractors. The tools used to run this simple transfer learning experiment are accessible online:

`https://github.com/jordipons/sklearn-audio-transfer-learning`

We report accuracy results on the test set of the GTZAN (fault-filtered) dataset, and our processing pipeline consists of "feature extraction" + 128 PCA + SVM. The feature extraction can be based on VGGish audioset features (77.58% accuracy), OpenL3 audioset features (74.65% accuracy), `MTT_musicnn` features (71.37% accuracy), `MTT_vgg` features (72.75% accuracy), or `MSD_musicnn` features (77.24% accuracy). Note that our MSD pre-trained models outperform the MTT ones. Besides, the `MSD_musicnn` achieves similar results than the VGGish audioset features (that are trained with a much larger dataset: 2M audios).

How did you train *musicnn* models? The code we employed to train the models above is also accessible:

`https://github.com/jordipons/musicnn-training`

These models achieve state-of-the-art performance on the MagnaTagATune dataset: `MTT_musicnn` (90.69 ROC-AUC / 38.44 PR-AUC) and `MTT_vgg` (90.26 ROC-AUC / 38.19 PR-AUC). But also for the Million Song Dataset: `MSD_musicnn` (88.01 ROC-AUC / 28.90 PR-AUC), `MSD_musicnn_big` (88.41 ROC-AUC / 30.02 PR-AUC) and `MSD_vgg` (87.67 ROC-AUC / 28.19 PR-AUC).

But the `musicnn-training` framework also allows to implement other models. For example, a similar architecture than *musicnn* but with an attention-based output layer (instead of the temporal pooling layer) can achieve 90.77 ROC-AUC / 38.61 PR-AUC on the MagnaTagATune dataset — and 88.81 ROC-AUC / 31.51 PR-AUC on the Million Song Dataset. You can find further details about this new architecture online.

ACKNOWLEDGMENTS

This work was partially supported by the Maria de MaeztuUnits of Excellence Programme (MDM-2015-0502) — and we are grateful for the GPUs donated by NVidia.

REFERENCES

[1] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR*, 2011.

[2] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. In *ISMIR*, 2016.

[3] Edith Law, Kris West, Michael I Mandel, Mert Bay, and J Stephen Downie. Evaluation of algorithms using games: The case of music tagging. In *ISMIR*, 2009.

[4] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *International Workshop on Content-Based Multimedia Indexing (CBMI)*, 2016.

[5] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik Schmidt, Andreas Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *ISMIR*, 2017.