END-TO-END LEARNING FOR MUSIC AUDIO TAGGING AT SCALE

Jordi Pons[1,2], **Oriol Nieto**[2], **Matt Prockup**[2], **Erik Schmidt**[2], **Andreas Ehmann**[2], **Xavier Serra**[1]
[1]Music Technology Group, Universitat Pompeu Fabra, Barcelona.
[2]Pandora Media Inc., Oakland.

ABSTRACT

The lack of data tends to limit the outcomes of deep learning research – specially, when dealing with end-to-end learning stacks processing raw data such as waveforms. In this study we make use of musical labels annotated for 1.2 million tracks. This large amount of data allows us to unrestrictedly explore different front-end paradigms: from assumption-free models – using waveforms as input with very small convolutional filters; to models that rely on domain knowledge – log-MEL spectrograms with a convolutional neural network designed to learn temporal and timbral features. Results suggest that, while spectrogram-based models surpass their waveform-based counterparts, the difference in performance shrinks as more data is employed.

1. MUSIC AUTO-TAGGING AT SCALE

The music auto-tagging task consists in automatically estimating the musical attributes of a song. Many approaches have been considered for this task (mostly based on *feature extraction + model* [4]), with recent publications showing promising results using deep learning [1–3]. In this work we confirm this trend, and we study how different deep learning architectures scale with a large music collection. To this end: we compare our model with a traditional method based on *feature extraction + model* [4], and we train our models with one million tracks annotated by musicologists. We divide deep learning models into two parts: front-end and back-end. The front-end is the part of the model that interacts with the input signal to map it into a lower-dimensional latent-space, and the back-end is the part which predicts the output given the representation obtained by the front-end. We report results for two different front-ends: one for processing waveforms, and another for spectrograms. We aim to study how superior/inferior waveform-based architectures are when compared to spectrogram-based ones given the unprecedented amount of data is used for training – 1M tracks for training, 100k for validation, and 100k for test. Experiments below share the same back-end, which enables a fair comparison among front-ends. Models' implementation details and further information about those is available online [1].

Shared back-end. It is conformed by three CNN layers (with 512 filters each and two residual connections), two pooling layers and a dense layer – see Figure 1 (*Bottom-left*). We introduced residual connections in our model because our intention was to explore very deep architectures, to take advantage of the large data available. Although adding more residual layers did not improve our results, we observed that adding these residual connections stabilized learning while slightly improving performance. The used CNN filters [1] are *(i)* computationally efficient and *(ii)* shaped such that all extracted features are considered across a reasonable amount of temporal context (note the $7 \times M'$ filter shapes, representing *time* $\times$ *all features*). We also make a drastic use of temporal pooling: firstly, down-sapling *x2* the temporal dimensionality of the feature maps; and secondly, by making use of a global pooling layer with mean and max statistics. The global pooling strategy allows for variable length inputs to the network, which we aim to further explore in future work. Finally, a dense layer with 500 units connects the pooled features to the output.

Waveform front-end. We make use of the *sample-level* front-end proposed by Lee *et al.* [2], which is composed of a stack of seven CNNs (3x1 filters), batch norm and max pool layers – see Figure 1 (*Top-left*). Each layer has 64, 64, 64, 128, 128, 128 and 256 filters, respectively. By hierarchically combining small-context representations and making use of max pooling, the *sample-level* front-end yields a feature map of an audio segment of 15 seconds (down-sampled to 16kHz), which is further processed by the previously described back-end.

Spectrogram front-end. Firstly, audio segments are converted to log-MEL magnitude spectrograms (15 seconds and 90 MEL bins) and normalized to have zero-mean and one-var. Secondly, we propose using vertical and horizontal filters explicitly designed to facilitate learning the timbral and temporal representations present in spectrograms [3]. Note in Figure 1 (*Right*) that the proposed front-end is a single-layer CNN with many filter shapes that are grouped into two branches [3]: *(i)* top branch – timbral features; and *(ii)* lower branch – temporal features. The top branch is designed to capture pitch-invariant timbral features that are occurring at different time-frequency scales in the spectrogram. Pitch invariance is enforced via enabling CNN filters to convolve through the frequency

[1] https://github.com/jordipons/music-audio-tagging-at-scale-models

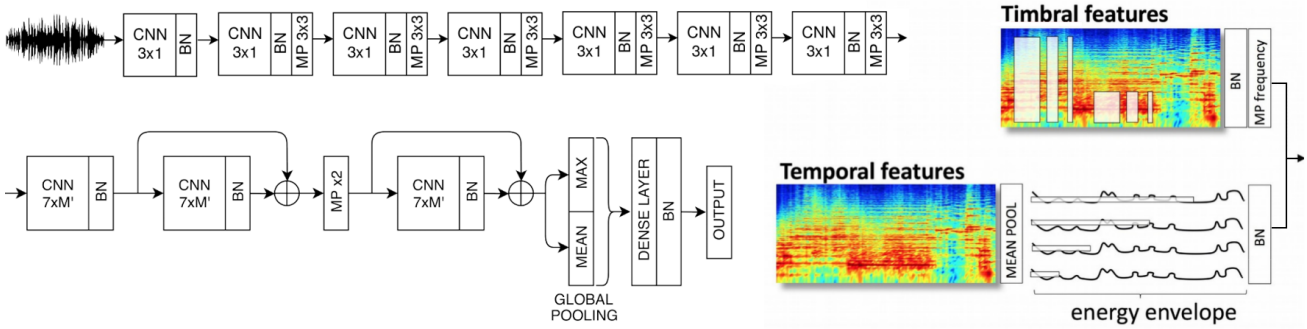

Figure 1. ***Bottom-left*** – back-end. ***Top-left*** – waveform front-end. ***Right*** – spectrogram front-end with two branches. ***Definitions*** – *M'* stands for the vertical axis of the feature map, *BN* for batch norm, and *MP* for max-pool.

Models	*#training examples*	*ROC AUC*	*PR AUC*	$\sqrt{MSE}$	Δ*ROC AUC*	Δ*PR AUC*	Δ $\sqrt{MSE}$	training time
GBT+features [4]	1.2M	91.61%	54.27%	0.1569	-	-	-	-
Waveform	1M	91.54%	57.86%	0.1501	0.6%	1.49%	0.0021	<2 weeks
Spectrogram	1M	**92.14%**	**59.35%**	**0.1480**				
Waveform	500k	91.23%	56.15%	0.1537	0.54%	1.75%	0.0044	≈ 1 week
Spectrogram	500k	91.76%	57.90%	0.1493				
Waveform	100k	89.16%	49.25%	0.1591	0.97%	2.83%	0.0049	few days
Spectrogram	100k	90.13%	52.08%	0.1542				

Table 1. ***Center*** – performance measurements of the baseline (GBTs+features [4]) and the studied models (waveform and spectrogram front-ends) when considering different training set sizes. ***Right*** – performance difference (Δ) between spectrogram and waveform models for each metric. For ROC-AUC and PR-AUC, the higher the score the better; for $\sqrt{MSE}$, the lower the better. All models have $\approx$5.5M parameters.

domain, and via max-pooling the feature map across its vertical axis. The lower branch is meant to learn temporal features, designed to efficiently capture different time-scale representations by using several filter shapes. These CNN filters operate over an energy envelope (not directly over the spectrogram) obtained via mean-pooling the frequency-axis of the spectrogram. The output of these two branches is merged, and the previously described back-end is used for *going deep*.

Results. Since we aim song-level predictions, several estimations are done per song (via sliding a window of 15 seconds) and then averaged – results are in Table 1. The spectrogram-based model trained with 1M tracks achieves better results than the baseline in every measurement. Furthermore, note that the waveform model trained with 1M tracks also achieves remarkable results. For most measurements it achieves better results than the baseline, but it always achieves worse results than the spectrogram model. However, a closer inspection of the results reveals that the differences between spectrogram and waveform models shrink as more data is used for training.

2. CONCLUSIONS

The two presented models are based on two conceptually different design principles. The first is based on a waveform front-end, and no domain knowledge inspired its design. The assumptions of this model are reduced to its minimum expression: raw audio is set as input, and the used CNN does minimal assumptions over the structure of the data due to its set of very small filters. For the second model, with a spectrogram front-end, we make use of domain knowledge to propose intuitive improvements for our model. The proposed models are capable of outperforming the proposed (strong) baseline, and our results confirm that spectrogram-based architectures are still superior to waveform-based models – see Table 1 (*Center*). However, the gap between waveform-based and spectrogram-based models is reduced when training with more data – see Table 1 (*Right*).

3. ACKNOWLEDGMENTS

This work is partially supported by the Maria de Maeztu Units of Excelence Programme (MDM-2015-0502).

4. REFERENCES

[1] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *ICASSP*, 2014.

[2] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. *arXiv:1703.01789*, 2017.

[3] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *CBMI*, 2016.

[4] Matthew Prockup, Andrew J. Asman, Fabian Gouyon, Erik M. Schmidt, Oscar Celma, and Youngmoo E. Kim. Modeling rhythm using tree ensembles and the music genome project. *ICML Workshop*, 2015.