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

Jordi Pons⋆† **Oriol Nieto**† **Matthew Prockup**† **Erik Schmidt**† **Andreas Ehmann**† **Xavier Serra**⋆
⋆ Music Technology Group, Universitat Pompeu Fabra, Barcelona
† Pandora Media Inc., Oakland, CA

ABSTRACT

The lack of data tends to limit the outcomes of deep learning research, particularly when dealing with end-to-end learning stacks processing raw data such as waveforms. In this study, 1.2M tracks annotated with musical labels are available to train our end-to-end models. This large amount of data allows us to unrestrictedly explore two different design paradigms for music auto-tagging: assumption-free models – using waveforms as input with very small convolutional filters; and models that rely on domain knowledge – log-mel spectrograms with a convolutional neural network designed to learn timbral and temporal features. Our work focuses on studying how these two types of deep architectures perform when datasets of variable size are available for training: the MagnaTagATune (25k songs), the Million Song Dataset (240k songs), and a private dataset of 1.2M songs. Our experiments suggest that music domain assumptions are relevant when not enough training data are available, thus showing how waveform-based models outperform spectrogram-based ones in large-scale data scenarios.

1. INTRODUCTION

One fundamental goal in music informatics research is to automatically structure large music collections. The music audio tagging task consists of automatically estimating the musical attributes of a song – including: moods, language of the lyrics, year of composition, genres, instruments, harmony, or rhythmic traits. Thus, tag estimates may be useful to define a semantic space that can be advantageous for automatically organizing musical libraries.

Many approaches have been considered for this task (mostly based on feature extraction + model [1, 22, 26]), with recent publications showing promising results using deep architectures [5, 9, 14, 21]. In this work we confirm this trend by studying how two deep architectures conceived considering opposite design strategies (using domain knowledge or not) perform for several datasets – with one of the datasets being of an unprecedented size: 1.2M songs. Provided that a sizable amount of data is available for that study, we investigate the learning capabili-

ties of these two architectures. Specifically, we investigate whether the architectures based on domain knowledge overly constrain the solution space for cases where large training data are available – in essence, we study if certain architectural choices (e.g., using log-mel spectrograms as input) can limit the model's capabilities to learn from data. The main contribution of this work is to show that little to no model assumptions are required for music auto-tagging when operating with large amounts of data.

Section 2 discusses the main deep architectures we identified in the audio literature, section 3 describes the datasets used for this work, section 4 presents the architectures we study, and section 5 provides discussion about the results with conclusions drawn in section 6.

2. CURRENT DEEP ARCHITECTURES

In order to facilitate the discussion around the current audio architectures, we divide deep learning models into two parts: front-end and back-end – see Figure 1. The front-end is the part of the model that interacts with the input signal in order to map it into a latent-space, and the back-end predicts the output given the representation obtained by the front-end. In the following, we present the main front- and back-ends we identified in the literature.
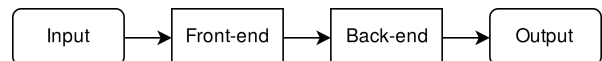

Figure 1. Deep learning pipeline.

Front-ends. These are generally comprised of convolutional neural networks (CNNs) [5, 9, 20, 21, 27], since these can learn efficient representations by sharing weights [1] along the signal. Front-ends can be divided into two groups depending on the used input signal: waveforms [9, 14, 27] or spectrograms [5, 20, 21]. Further, the design of the filters can be either based on domain knowledge or not. For example, one leverages domain knowledge when a front-end for waveforms is designed so that the length of the filter is set to be as the window length of a STFT [9]. Or for a spectrogram front-end, it is used vertical filters to learn timbral representations [12] or horizontal filters to learn longer temporal cues [25]. Generally, a single filter shape is used in the first CNN layer [5, 9, 12, 25], but some recent works report performance gains when using several filter shapes in the first layer [4, 18–21, 27]. Using many filters promotes a richer feature extraction in the first layer, and facilitates leveraging domain knowledge for designing the filters' shape. For example: a

---

[1] Which determine the learned feature representations.

waveform front-end using many long filters (of different lengths) can be motivated from the perspective of a multi-resolution time-frequency transform [2] [27]; or since it is known that some patterns in spectrograms are occurring at different time-frequency scales, one can intuitively incorporate many (different) vertical and/or horizontal filters in a spectrogram front-end [18–21]. To summarize, using domain knowledge when designing models allows us to naturally connect the deep learning literature with previous signal processing work. On the other hand, when domain knowledge is not used, it is common to employ a deep stack of small filters, e.g.: 3×1 as in the sample-level front-end used for waveforms [14], or 3×3 filters used for spectrograms [5]. These models based on small filters make minimal assumptions over the local stationarities of the signal, so that any structure can be learned via hierarchically combining small-context representations. These architectures with small filters are flexible models able to potentially learn any structure given enough depth and data.

Back-ends. Among the different back-ends used in the audio literature, we identified two main groups: *(i)* fixed-length input back-end, and *(ii)* variable-length input back-end. The generally convolutional nature of the front-end allows it to process different input lengths. Therefore, the back-end unit can adapt a variable-length feature map to a fix-sized output. The former group of models *(i)* assume that the input will be kept constant – examples of those are front-ends based on feed-forward neural-networks or fully-convolutional stacks [5,9]. The second group *(ii)* can deal with different input-lengths since the model is flexible in at least one of its input dimensions – examples of those are back-ends using temporal-aggregation strategies such as max-pooling, average-pooling, attention models or recurrent neural networks [23]. Given that songs are generally of different lengths, these types of back-ends are ideal candidates for music processing. However, despite the different-length nature of music, many works employ fixed-length input back-ends (group *i*) since these architectures tend to be simpler and perform well [5,9,21].

3. DATASETS

We study how different deep architectures for music auto-tagging perform for 3 music collections of different sizes:

1) The MagnaTagATune (MTT) dataset is of $\approx$ 26k music audio clips of 30s [11]. Predicting the top-50 tags of this dataset is a popular benchmark for auto-tagging.

2) Although the Million Song Dataset (MSD) name indicates that 1M songs are available [2], audio files with proper tag annotations (top-50 tags) are only available for $\approx$ 240k previews of 30s. This dataset constitutes the biggest public dataset available for music auto-tagging, making these data highly appropriate for benchmarking.

3) A private dataset consisting of 1M songs for training, 100k for validation, and 100k for test [3] is available for this study. The 1.2M-songs dataset has 139 track-level human-expert annotations that can be summarized as follows:

· *Meter tags* denote different sorts of musical meters (e.g., triple-meter, cut-time, compound-duple, odd).
· *Rhythmic feel tags* denote rhythmic interpretation (e.g., swing, shuffle, back-beat strength) and elements of rhythmic perception (*e.g.*, syncopation, danceability).
· *Harmonic tags*: major, minor, chromatic, etc.
· *Mood tags* express the sentiment of a music audio clip (e.g., if the music is angry, sad, joyful).
· *Vocal tags* denote the presence of vocals and timbral characteristics of it (e.g., male, female, vocal grittiness).
· *Instrumentation tags* denote the presence of instruments (e.g., piano) and their timbre (e.g., guitar distortion).
· *Sonority tags* detail production techniques (e.g., studio, live) and overall sound (e.g., acoustic, synthesized).
· *Basic genre tags*: jazz, rock, rap, latin, disco, etc.
· *Subgenre tags*: jazz (e.g., cool, fusion, hard bop), rock (e.g., light, hard, punk), rap (e.g., east coast, old school), world music (e.g., cajun, indian), classical music (e.g., baroque period, classical period), etc.

Other large (music) audio datasets exist: the Free Music Archive (FMA: $\approx$106k songs) [8] and Audioset ($\approx$2.1M audios) [10]. Since previous works mainly used the MTT and MSD [5,14], we employ these datasets to assess the studied models with public data. Despite our interest in using FMA, for brevity, we restrict our study to 3 datasets that already cover a wide range of different sizes. Finally, Audioset is not used since most of its content is not music.

4. THE ARCHITECTURES UNDER STUDY

After an initial exploration of the different architectures introduced in section 2, we select two models based on opposite design paradigms: one for processing waveforms, with a design that does minimal assumptions over the task at hand; and another for spectrograms, with a design that heavily relies on musical domain knowledge. Our goal is to compare these two models for providing insights in whether domain knowledge is required (or not) for designing deep learning models. This section provides discussion around our architectural choices and introduces the *basic* configuration setup – which is also accessible online. [4]

The waveform model was selected after observing that the sample-level front-end (using a deep stack of 3×1 filters) was remarkably superior to the other waveform-based front ends – as shown in the original paper [14]. This result is particularly compelling because this front-end does not rely on domain-knowledge for its design. Note that raw waveforms are fed to the model without any pre-processing, and the small filters considered for its design make no strong assumptions over the most informative local stationarities in waveforms. Therefore, the sample-level can be seen as a problem agnostic front-end that has the potential to learn any audio task provided that enough depth and data are available. Given that a large amount data is available for this study, the sample-level front-end is of particular interest due to its strong learning potential: its solution space is not constrained by severe architectural choices relying on domain knowledge.

---

[2] The Constant-Q Transform [3] is an example of such transform.
[3] Test & validation sets are kept the same throughout the experiments for a fair evaluation. All used partitions are stratified and artist-filtered.

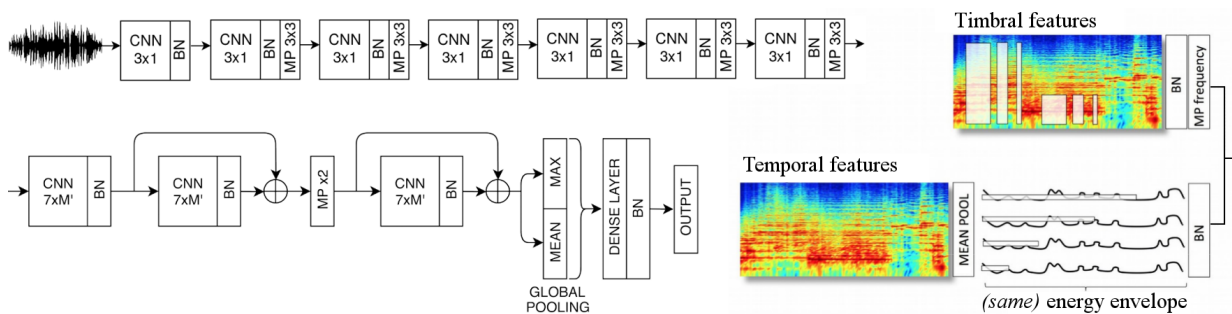[4] https://github.com/jordipons/music-audio-tagging-at-scale-models

Figure 2. *Bottom-left* – back-end. *Top-left* – waveform front-end. *Right* – spectrogram front-end. *Definitions* – M' stands for the feature map's vertical axis, *BN* for batch norm, and *MP* for max-pool.

On the other hand, when experimenting with spectrogram front-ends, we found domain knowledge intuitions to be valid guides for designing deep architectures. For example, front-ends based on *(i)* many vertical and horizontal filters in the first layer were consistently superior to front-ends based on *(ii)* a single vertical filter – as shown in recent publications [4, 18–20]. Note that the former front-ends *(i)* can learn spectral and (long) temporal representations already in the first layer – which are known to be important musical cues; while the latter *(ii)* can only learn spectral representations. Moreover, we observed that front-ends based on a deep stack of 3×3 filters were achieving equivalent performances to the former front-end *(i)* when input segments were shorter than 10s – as noted in the literature [21]. But when considering longer inputs (which yielded better performance), the computational price of this deeper model increases: longer inputs implies having larger feature maps in every layer and therefore, more GPU memory consumption. For that reason, we refrained from using a deep stack of 3×3 filters as a front-end – because our 12GBs of VRAM were not enough to input 15s of audio when using a back-end. Hence, making use of domain knowledge also provides guidance for minimizing the computational cost of the model – since by using a single layer with many vertical and horizontal filters, one can efficiently capture the same receptive field without paying the cost of going deep. Finally, note that front-ends using many vertical and horizontal filters in the first layer are an example of deep architectures relying on (musical) domain knowledge for their design.

After considering the previous discussion, we select the sample-level front-end as main part of our assumption-free model for waveforms; and we use a spectrogram front-end with many vertical and horizontal (first-layer) filters for the model designed considering domain knowledge. Experiments below share the same back-end, which enables a fair comparison among the previously selected front-ends. Unless otherwise stated, the following specifications are the ones used for the experiments – throughout the document, we refer to these specifications as the *basic* configuration:

Shared back-end. It consists of three CNN layers (with 512 filters each and two residual connections), two pooling layers and a dense layer – see Figure 2 (*Bottom-left*). We introduced residual connections in our model to explore very deep architectures, such that we can take advantage

of the large data available. Although adding more residual layers did not drastically improve our results, we observed that adding these residual connections stabilized learning while slightly improving performance [16]. The used 1D-CNN filters [9] are computationally efficient and shaped such that all extracted features are considered across a reasonable amount of temporal context (note the 7×M' filter shapes, representing *time×all features*). We also make a drastic use of temporal pooling: firstly, down-sampling *x2* the temporal dimensionality of the feature maps; and secondly, by making use of global pooling with mean and max statistics. The global pooling strategy allows for variable length inputs to the network and therefore, such a model can be classified as a "variable-length input" back-end. Finally, a dense layer with 500 units connects the pooled features to a sigmoidal output.

Waveform front-end. It is based on a sample-level front-end [14] composed of seven: 1D-CNN (3×1 filters), batch norm, and max pool layers – see Figure 2 (*Top-left*). Each layer has 64, 64, 64, 128, 128, 128 and 256 filters. For the 1.2M-songs dataset, we use a model with more capacity having nine layers with 64, 64, 64, 128, 128, 128, 128, 128, 256 filters. By hierarchically combining small-context representations and making use of max pooling, the sample-level front-end yields a feature map for an audio segment of 15s (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 96 mel bins [17]) and normalized to have zero-mean and unit-var. Secondly, we use vertical and horizontal filters explicitly designed to facilitate learning the timbral and temporal patterns present in spectrograms [19–21]. Note in Figure 2 (*Right*) that the spectrogram front-end is a single-layer CNN with many filter shapes that are grouped into two branches [19]: *(i)* top branch – timbral features [21]; and *(ii)* lower branch – temporal features [20]. 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 domain, and via max-pooling the feature map across its vertical axis [21]. Note that several filter shapes are used to efficiently capture many different time-frequency patterns: 7×86, 3×86, 1×86, 7×38, 3×38

1.2M-songs Models	train size	ROC AUC	PR AUC	$\sqrt{MSE}$
Baseline	1.2M	91.61%	54.27%	0.1569
Waveform	1M	**92.50%**	**61.20%**	**0.1465**
Spectrogram	1M	92.17%	59.92%	0.1473
Waveform	500k	91.16%	56.42%	0.1504
Spectrogram	500k	91.61%	58.18%	0.1493
Waveform	100k	90.27%	52.76%	0.1554
Spectrogram	100k	90.14%	52.67%	0.1542

Table 1. 1.2M-songs average results (3 runs) when using different training-set sizes. Baseline: GBTs+features [22].

and 1×38 [5] – to facilitate learning, e.g.: kick-drums (with small-rectangular filters of 7×38 capturing sub-band information for a short period of time), or string ensemble instruments (with long vertical filters of 1×86 which are capturing timbral patterns spread in the frequency axis). The lower branch is meant to learn temporal features, and is designed to efficiently capture different time-scale representations by using several long filter shapes [20]: 165×1, 128×1, 64×1 and 32×1. [6] These filters operate over an energy envelope (not directly over the spectrogram) obtained via mean-pooling the frequency-axis of the spectrogram. By computing the energy envelope in that way, we are considering high and low frequencies together while minimizing the computations of the model – note that no frequency/vertical convolutions are performed, but 1D (temporal) convolutions. Thus, domain knowledge is also providing guidance to minimize the computational cost of the model. The output of these two branches is merged, and the previously described back-end is used for going deeper. For further details, see its online implementation.[4]

Parameters. 50% dropout before every dense layer, ReLUs as non-linearities, and our models are trained with SGD employing Adam (with an initial learning rate of 0.001) as optimizer. We minimize the MSE for the 1.2M-songs dataset, but we minimize the cross entropy for the other datasets. During training our data are converted to audio patches of 15s, but during prediction one aims to consider the whole song. To this end, several predictions are computed for a song (by a moving window of 15s) and then averaged. Although our models are capable of predicting tags for variable-length inputs, we use fixed length patches since in preliminary experiments we observed that predicting the whole song at once yielded worse results than averaging several patch predictions. In future work we aim to further study this behavior, to find ways to exploit the fact that the whole song is generally available.

5. EXPERIMENTAL RESULTS

5.1 1.2M-songs dataset

Experimental setup. As a baseline, we use a system consisting of a music feature extractor (in essence: timbre, rhythm, and harmony descriptors) and a model based on gradient boosted trees (GBT) for predicting each of the tags [22]. By predicting each tag individually, one aims
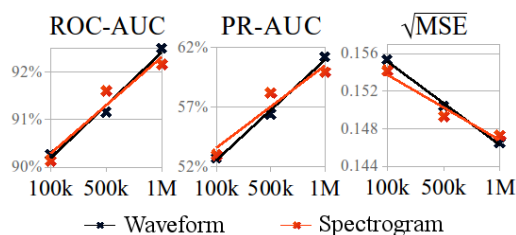

Figure 3. Linear regression fit on the 1.2M-songs results.

to turn a hard problem into multiple (hopefully *simpler*) problems. A careful inspection of the dataset reveals that, among tags, two different data distributions dominate the annotations: *(i)* tags with bi-modal distributions, where most of the annotations are zero, which can be classified; and *(ii)* tags with pseudo-uniform distributions that can be regressed. [7] A regression tag example is *acoustic*, which indicates how acoustic a song is – from zero to one, zero being an electronic music song and one a string quartet. And a classification tag example can be any genre – for example, most songs will not be cataloged as *rap* since the dataset is large and its taxonomy contains dozens of genres. We use two sets of performance measurements: ROC-AUC [8] and PR-AUC [8] for the classification tags, and error ($\sqrt{MSE}$ [8]) for the regression tags. ROC-AUC can lead to over-optimistic scores in cases where data are unbalanced [7]; given that classification tags are highly unbalanced, we also consider the PR-AUC metric since it is more indicative than ROC-AUC in these cases [7]. For ROC-AUC and PR-AUC, the higher the score the better – but for $\sqrt{MSE}$, the lower the better. Studied spectrogram and waveform models are set following the *basic* configuration – and are composed of 5.9M and 5.5M parameters, respectively. Given the unprecedented size of the dataset, we focus on how these models scale when trained with different amounts of data: 100k, 500k, or 1M songs. Average results (across 3 runs) are shown in Table 1 and Figure 3.

Quantitative results. Training the models with 100k songs took a few days, with 500k songs one week, and with 1M songs less than two weeks. The deep learning models trained with 1M tracks achieve better results than the baseline in every metric. However, the deep learning models trained with 100k tracks perform worse than the baseline. This result confirms that deep learning models require large datasets to clearly outperform strong methods based on feature-design – although note that large datasets are generally not available for most audio tasks. Moreover, the biggest performance improvement *w.r.t.* the baseline is seen for PR-AUC, which provides a more informative picture of the performance when the dataset is unbalanced [7]. In addition, the best performing model is based on the waveform front-end – being capable of outperforming the spectrogram model in every metric when trained with 1M songs. This result confirms that waveform sample-level front-ends have a great potential to learn from large data, since their solution space is not constrained by any se-

---

[5] Each filter shape has 16, 32, 64, 16, 32 and 64 filters, respectively.
[6] Each filter shape has 16, 32, 64 and 128 filters, respectively.

[7] Note that all output nodes are sigmoidal – i.e., we treat classification tags as regression tags for simplicity's sake.
[8] ROC: Receiver Operating Characteristic. PR: Precision Recall. AUC: Area Under the Curve. MSE: Mean Squared Error.

vere architectural choice. On the other hand, the architectural choices defining the spectrogram front-end might be constraining the solution space. While these architectural constraints are not harmful when training data are scarce (as for the 100k/500k songs results or in prior works [24]), such a strong regularization of the solution space may limit the learning capacity of the model in scenarios where large training data are available – as for the 1M songs results. One can observe this in Figure 3, where we fit linear models to the obtained results to further study this behavior. When 100k training songs are available: trend lines show that spectrogram models tend to perform better. However, when 1M training songs are available: the lines show that waveform models outperform the spectrogram ones. It is worth mentioning that the observed trends are consistent throughout metrics: ROC-AUC, PR-AUC, and $\sqrt{MSE}$. Finally, note that there is room for improving the models under study – e.g.: one could address the data imbalance problem during training, or improve the back-end via exploring alternative temporal aggregation strategies.

Qualitative results. Since it is the first report of a deep music tagging model trained with such a large dataset, we also perceptually assess the quality of the estimates. To this end, we compared the predictions of one of our best performing models to the predictions of the baseline, and to the human-annotated ground-truth tags. Some interesting examples identified during this qualitative experiment are available online. [9] First, we observed that the deep learning model is biased towards predicting the popular tags (such as *lead vocals*, *English* or *male vocals*). Note that this is expected since we are not addressing the data unbalancing issue during training. And second, we observe that the baseline model (which predicts the probability of each tag with an independent GBT model) predicts mutually exclusive tags with high confidence – e.g., it predicted with high scores: *East Coast* and *West Coast* for an East Cost rap song, or *baroque period* and *classic period* for a Bach aria. However, the deep learning model (predicting the probability of all tags together) was able to better differentiate these similar but mutually exclusive tags. This suggests that deep learning has an advantage when compared to traditional approaches, since these mutually exclusive relations can be jointly encoded within the model.

5.2 MagnaTagATune (MTT) dataset

Experimental setup. State-of-the-art models are set as baselines, and we use the same (classification) performance metrics as for the 1.2M-songs dataset: ROC-AUC and PR-AUC – note that the MTT labels are binary. One of the baseline results (the SampleCNN [14] with 90.55 ROC-AUC) was computed using a slightly different version of the MTT dataset – which only includes songs having more than 1 tag and lasting more than 29.1 seconds. As a result, this cleaner version of the MTT dataset is of ≈21k songs instead of ≈26k. Although this dataset cleans out potential noisy annotations, we decided to use the original dataset to easily compare our results with former works. Thus, to fairly compare our models with

the SampleCNN, we reproduce their work considering the original dataset – achieving a score of 88.56 ROC-AUC. Given that less noise is present in the SampleCNN dataset, it seems reasonable that their performance is higher than the one obtained by our implementation.

The MTT experiments can be divided in two parts: waveform and spectrogram models – see Tables 2 and 3. Due to the amenable size of the dataset (every MTT experiment lasts < 5h), it is feasible to run a comprehensive study investigating different architectural configurations. Specifically, we study how waveform and spectrogram architectures behave when modifying the capacity of their front- and back-ends. For example, the experiment "# filters ×1/2" in Table 2 consists of dividing the number of filters available in the waveform front-end by two. This means having 32, 32, 32, 64, 64, 64 and 128 filters, instead of the 64, 64, 64, 128, 128, 128 and 256 filters in the *basic* configuration. We also apply this methodology to the spectrogram front-ends, and we add/remove capacity to them by increasing/decreasing the number of available filters. After running the front-end experiments with a fixed back-end (following the *basic* configuration: 512 CNN filters, 500 output units), we select the most promising ones to proceed with the back-end study – for waveforms: "# filters ×2", [10] and for spectrograms: "# filters ×1/2". Having now a fixed front-end for every experiment, we modify the capacity of the back-end via changing the number of filters in every CNN layer (512, 256, 128, 64) and changing the number of output units (500, 200). Since the *basic* configuration leads to relatively big models for the size of the dataset, these experiments explore smaller back-ends. The inputs for the MTT are set to be of 3s, since longer inputs yield worse results [15, 21].

Quantitative results. The waveform and spectrogram models we study outperform the proposed baselines – which represent the current state-of-the-art. Further, performance is quite robust to the number of parameters of the model. Although the best results are achieved by models having higher capacity, the performance difference between small and large models is minor – what means that relatively small models (which are easier to deploy) can do a reasonable job when tagging the MTT music. Finally: spectrogram models perform better than waveform models for this small public dataset – which aligns with previous works using datasets of similar size [20,21]. Consequently, these results confirm that domain knowledge intuitions are valid guides for designing deep architectures in scenarios where training data are scarce.

5.3 Million Song Dataset (MSD)

Experimental setup. State-of-the-art models are set as baselines, and we use the same (classification) performance metrics as for the 1.2M-songs dataset: ROC-AUC and PR-AUC – note that the MSD labels are binary. These experiments aim to validate the studied models with the biggest public dataset available. Models are set following the *basic* configuration, and results are shown in Table 4.

---

[9] http://www.jordipons.me/apps/music-audio-tagging-at-scale-demo

[10] "# filters ×2" front-end was selected instead of "# filters ×4", because it performs similarly with less parameters.

MTT dataset	ROC	PR	#
Waveform models	AUC	AUC	*param*

State-of-the-art results – with our own implementations			
SampleCNN [14] [13]	90.55	-	2.4M
SampleCNN (reproduced)	88.56	34.38	2.4M
Dieleman et al. [9]	84.87	-	-
Dieleman et al. (reproduced)	85.58	29.59	194k

How much capacity is required for the front-end?			
# filters ×4	**89.05**	**34.92**	11.8M
# filters ×2 (selected)	88.96	34.74	7M
# filters ×1	88.9	34.18	5.3M
# filters ×1/2	88.69	33.97	4.7M
# filters ×1/4	88.47	33.89	4.4M

How much capacity is required for the back-end?			
# filters in every CNN layer - # units in dense layer			
64 CNN filters - 500 units	88.57	33.99	1.3M
- 200 units	<u>88.94</u>	<u>34.47</u>	1.3M
128 CNN filters - 500 units	88.82	34.62	1.8M
- 200 units	88.81	34.6	1.7M
256 CNN filters - 500 units	88.95	34.27	3.1 M
- 200 units	88.59	34.39	2.9M
512 CNN filters - 500 units	<u>88.96</u>	<u>34.74</u>	7M
- 200 units	88.3	34.05	6.7M

Table 2. MTT results: waveform models.

Quantitative results. The spectrogram model outperforms the waveform model for this public dataset – having $\approx$ 200k training songs. Furthermore, the spectrogram model performs equivalently to 'Multi-level & multi-scale' [13], which is the best performing method in the literature – denoting that musical knowledge can be of utility to design models for the MSD. Additionally, the waveform model performs worse than other waveform-based models that also employ sample-level front-ends. Such performance decrease could be caused because *(i)* SampleCNN methods [14,15] average ten [11] estimates for the same song to compensate for possible faults in song-level predictions, while our method only averages two – predicting consecutive patches of 15s; or *(ii)* because the major difference between SampleCNN and the waveform model is that the latter employs a global pooling strategy that could remove potentially useful information for the model. Besides, the best performing waveform-based model ('SampleCNN multi-level & multi-scale' [15]) also achieves lower scores than the best performing spectrogram-based ones. Considering the outstanding results we report when the waveform model is trained with 1M songs, one could argue that the lack of larger public datasets is limiting the outcomes of deep learning research for music auto-tagging – particularly when dealing with end-to-end learning stacks processing raw data such as waveforms.

---

[11] Since MSD audios are of 30s, ten tag estimates per song can be obtained via running the model with consecutive patches of 3s.

[13] Result computed with a different MTT version, see section 5.2.

[14] Reproduced using 96 mel bands instead of 128 as in [21].

MTT dataset	ROC	PR	#
Spectrogram models	AUC	AUC	*param*

State-of-the-art results – with our own implementations			
VGG - Choi et al. [5]	89.40	-	22M
VGG (reproduced)	89.99	37.56	450k
Timbre CNN [21]	89.30	-	191k
Timbre CNN (reproduced) [14]	89.07	34.92	220k

How much capacity is required for the front-end?			
# filters ×1/8	90.08	37.18	4.4M
# filters ×1/4	90.12	37.69	4.6M
# filters ×1/2 (selected)	**90.40**	**38.11**	5M
# filters ×1	90.31	37.79	5.9
# filters ×2	90.07	37.29	7.6M

How much capacity is required for the back-end?			
# filters in every CNN layer - # units in dense layer			
64 CNN filters - 500 units	90.03	36.98	277k
- 200 units	<u>90.28</u>	<u>37.55</u>	222k
128 CNN filters - 500 units	90.16	37.61	617k
- 200 units	<u>90.28</u>	<u>37.69</u>	524k
256 CNN filters - 500 units	90.18	37.98	1.6M
- 200 units	90.06	37.16	1.4M
512 CNN filters - 500 units	<u>90.40</u>	<u>38.11</u>	5M
- 200 units	89.98	37.05	4.7M

Table 3. MTT results: spectrogram models.

6. CONCLUSIONS

This study presents the first work describing how different deep music auto-tagging architectures perform depending on the amount of available training data. We also present two architectures that yield results on par with the state-of-the-art. These architectures are based on two conceptually different design principles: one is based on a waveform front-end, and no domain knowledge inspired its design; and the other, with a spectrogram front-end, makes use of (musical) domain knowledge to justify its architectural choices. While our results suggest that models relying on domain knowledge play a relevant role in scenarios where no sizable datasets are available, we have shown that, given enough data, assumption-free models processing waveforms outperform those that rely on musical domain knowledge.

MSD	ROC	PR	#
Models	AUC	AUC	*param*
Waveform *(ours)*	87.41	28.53	5.3M
SampleCNN [14]	88.12	-	2.4M
SampleCNN multi-level & multi-scale [15]	**88.42**	-	-
Spectrogram *(ours)*	**88.75**	**31.24**	5.9M
VGG + RNN [6]	86.2	-	3M
Multi-level & multi-scale [13]	**88.78**	-	-

Table 4. MSD results. ***Top*** – waveform-based models. ***Bottom*** – spectrogram-based models.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] Yann Bayle, Pierre Hanna, and Matthias Robine. Revisiting autotagging toward faultless instrumental playlists generation. *arXiv preprint arXiv:1706.07613*, 2017.

[2] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[3] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[4] Ning Chen and Shijun Wang. High-level music descriptor extraction algorithm based on combination of multi-channel cnns and lstm. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[5] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[6] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[7] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *International Conference on Machine Learning (ICML)*. ACM, 2006.

[8] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[9] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[10] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[11] Edith Law, Kris West, Michael I Mandel, Mert Bay, and J Stephen Downie. Evaluation of algorithms using games: The case of music tagging. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2009.

[12] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems*, 2009.

[13] Jongpil Lee and Juhan Nam. Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. *IEEE signal processing letters*, 24(8):1208–1212, 2017.

[14] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. *Sound and Music Computing Conference (SMC)*, 2017.

[15] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification. *Applied Sciences*, 8(1):150, 2018.

[16] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[17] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. 2004.

[18] Huy Phan, Lars Hertel, Marco Maass, and Alfred Mertins. Robust audio event recognition with 1-max pooling convolutional neural networks. *arXiv preprint arXiv:1604.06338*, 2016.

[19] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6. IEEE, 2016.

[20] Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[21] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. *European Signal Processing Conference (EUSIPCO2017)*, 2017.

[22] 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. *Machine Learning for Music*

Discovery Workshop at the International Conference on Machine Learning (ICML), 2015.

[23] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, 2016.

[24] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. Learning the speech front-end with raw waveform cldnns. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[25] Jan Schluter and Sebastian Bock. Improved musical onset detection with convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[26] Mohamed Sordo, Cyril Laurier, and Oscar Celma. Annotating music collections: How content-based similarity helps to propagate labels. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2007.

[27] Zhenyao Zhu, Jesse H Engel, and Awni Hannun. Learning multiscale features directly from waveforms. *arXiv preprint arXiv:1603.09509*, 2016.