Deep Neural Networks Approaches for Music Tagging

Emanuele Dalla Longa
emanuele.dallalonga@mail.polimi.it

Salvatore D'Amicis
salvatore.damicis@mail.polimi.it

Matteo Romeo
matteo2.romeo@mail.polimi.it

Davide Urzino
davide.urzino@mail.polimi.it

Abstract

In this paper we present three different approaches to multilabel music tagging, with features such as genre, instrumentation etc.

We use the Area Under the Receiver Operating Characteristic curve as main performance evaluation metric to compare the models.

Two models reproduce previous results. One uses the raw waveform as input and is trained end-to-end; the other one is based on the Mel spectrogram (a higher level, handcrafted representation), and trains a shallower network over it, achieving the same performance.

We also introduce a novel approach using LSTMs, which have seen widespread success for sequence classification problems, it achieved comparable performances.

Our best result lead to an average AUC among classes of 0.894, which is in line with the result of the previous works.

1. Introduction

Music classification is a hot topic nowadays, since both the number of services providing music content and the size of the content collections are growing: being able to autotag music can be a precious tool for companies. Music tags provide high level information about a song characteristic: mood (sad, anger, happy), genre (rock, pop) and instrumentation (guitar, violin), and for such reason tags are used for recommendation systems and retrieval tasks.

The purpose of the paper is to explore and compare the capabilities of DCNNs and RNNs in music classification tasks, trying both different architectures and different input space representations.

There are two representations of the input space regarding audio signals that will be considered:

- Raw waveforms – time $\times$ samples, 1D representation of a sound.
- Mel spectrogram – time $\times$ Mel-coefficients, 2D representation of a sound obtained by a transformation which mimics the human perception behavior.

Regarding raw waveforms, our work built on a previous paper [1], where the DCNN model was trained splitting songs in multiple frames of the same length, while our work tried to achieve the same goal using the entire songs as the model input, producing very similar results in terms of AUC.

Regarding Mel spectrogram representation, two different kinds of architectures are proposed: DCNNs and RNNs.

DCNNs are the most popular in music classification tasks: the input is a 2D matrix (time $\times$ Mel coefficients) and the architecture of the model follows the typical structure of a DCNN used for image classification tasks [2]. Using hand crafted features, a shallower network can be used without having to learn the lower level features during end-to-end training. We were able to train a model with performances able to equal previous results.

Furthermore, we wanted to explore the capabilities of RNNs (in particular, LSTMs), to model sequential patterns. We were able, using a simple model, to reach performances comparable to those of the existing ones, showing how promising this approach can be.

Finally, we discuss how we measured the performances of the models.

2. Related work

The raw waveform based approach has not been considered very much in the past with respect to the Mel spectrogram. Our work started from the one of Jongpil Lee et al., who managed to obtain a model with 0.9055 AUC [1]. This approach had been previously discussed by Dieleman and Schrauwen [3], who were not able to obtain the same results, attributing this to the impossibility to find a non-linearity function who could replace the log-based amplitude compression in the spectrogram. However, they noticed the capability of the network to learn some relevant features.

The model proposed by Choi et al. [4] uses deep, fully convolutional neural networks with Mel-spectrogram inputs

to deal with music auto-tagging tasks. This work shows the quality of DCNNs with respect to more classical methods based on STFTs and MFCCs.

Another interesting work (one of the few using RNNs, actually) is described in [5] and shows the capabilities of Recurrent Neural Networks in real time genre classification tasks, even if is a slightly different problem (real time, single label).

3. Proposed approach

Regarding the pure convolutional approaches, their effectiveness is well established. Our goal is to verify the replicability of the previous works and compare their results with our LSTM proposal.

3.1. Why LSTMs

DCNNs are state-of-the art models, but they exhibit problems in learning features on multiple scales; music tags can be either related to the whole song (e.g., "rock", "pop", "orchestral") or to a single segment (e.g., "female voice", "guitar solo"). The rationale behind using LSTMs is that these models are able to tune their memory length based on what they identify in the data. Stacks of LSTMs can learn features on a large number of samples. Also, we wanted to exploit the sequential nature of musical works: music is by all means a language, with all the corresponding rules, and since LSTMs are state-of-the-art models in learning this kind of structure, we thought about providing a kind of embedding, the Mel spectrogram, to the network, as it was a sequence of words.

4. Experiments

In this section, we will describe the dataset used and discuss the different architectural choices tested for both raw and Mel spectrogram input representation.

4.1. Dataset

In our experiments, we use the MagnaTagATune Dataset. It is composed by 25860 songs, annotated with 190 different tags. The dataset is highly unbalanced: if we consider the first 50 most popular tags, the first one (guitar) appear 4851 times while the 50th (choral) appears only 490 times. Analyzing the different tags, we noticed that some of them were pretty similar and redundant, so we decided to aggregate some of them (e.g. `classical`, `clasical` and `classic` have been merged in a single tag, `classical`) reducing the number of tags to 125. Then, after the aggregation, we have kept only the 50 most popular classes with a coverage of 90% of the label over the entire dataset (Figure 1)

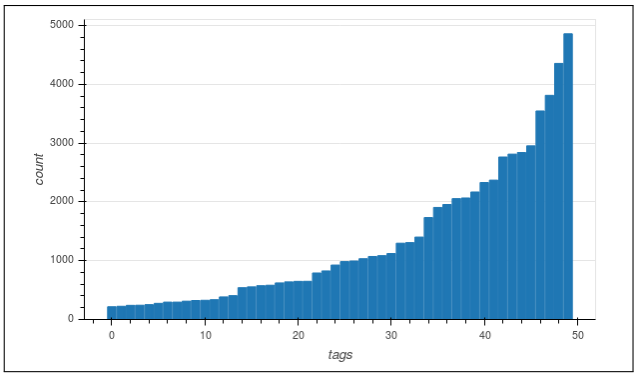


Figure 1: Top-50 aggregated tags distribution

The songs are stored in `.mp3` format with a sample rate of 16kHz, which we have converted in `.wav` format with the same sample rate. These songs have been directly used to train the raw waveform model, while for the Mel spectrogram we have processed the songs to create a 2D array for each one. To build the mel spectrogram for the DCNN we have used a FFT window length of 2048 samples and an hop size of 1792 samples. For the LSTM approach, we tested different parameters described in the following paragraph.

4.2. Raw input representation

We replicated the architecture by J. Lee et al [1]. In their experiments they used an architecture based on an high number of layer with very small filter sizes: this choice should be able to address log-scale amplitude compression and phase-invariance. Their proposed architecture takes as input frames of 59049 samples and averages the output predictions on frames of the same song to obtain the final outcome. We decided to test the same model using the entire song as input, which obviously lead to an higher number of parameters, as shown in Table 1

4.3. Mel spectrogram input representation

4.3.1 DCNN

The proposed model consists of 2D convolutions and pooling layers, to take the local harmonic structure into account. Batch Normalization has been applied after each convolution layer and dropout regularization after max pooling layers (Table 2).

4.3.2 RNN (LSTM)

For the LSTM, we use stacked LSTMs followed by 1D convolutional layers. As input, we use different Mel spectrograms generated with different parameters (in addition to

Sample CNN architecture 465984 (29.124s) as input			
layer	stride	output	# of params
conv 3 - 128	3	155328x128	1024
conv 3 - 128	1	155328x128	49792
maxpool 3	3	51776x128	
conv 3 - 256	1	51776x256	99584
maxpool 3	3	17258x256	
conv 3 - 256	1	17258x256	197888
maxpool 3	3	5752x256	
conv 3 - 256	1	5752x256	197888
maxpool 3	3	1917x256	
conv 3 - 256	1	1917x256	197888
maxpool 3	3	639x256	
conv 3 - 256	1	639x256	197888
maxpool 3	3	213x256	
conv 3 - 256	1	213x256	197888
maxpool 3	3	71x256	
conv 3 - 512	1	71x512	
maxpool 3	3	23x512	395776
dropout 0.5	-	23x512	
sigmoid	-	50	588850
Total params			2.1×10^6

Table 1: Sample CNN architecture

Mel Spectrogram architecture (128, 261) as input			
layer	stride	output	# of params
conv (3,3) - 32	(1,1)	126x256x32	
maxpool (2,2)	(2,2)	63x129x32	448
dropout 0.1	-	63x129x32	
conv (3,3) - 64	(1,1)	61x127x64	
maxpool (2,2)	(2,2)	30x63x64	18752
dropout 0.1	-	30x63x64	
conv (3,3) - 64	(1,1)	28x61x64	
maxpool (2,2)	(2,2)	14x30x64	37184
dropout 0.1	-	14x30x64	
conv (3,3) - 64	(1,1)	12x28x6	
maxpool (2,2)	(2,2)	6x14x64	37184
dropout 0.1	-	6x14x64	
sigmoid	-	50	268850
Total params			362418

Table 2: Mel Spectrogram architecture

the ones used for the DCNNs), as described in the following section.

LSTM architecture (227,128) as input		
layer	output	# of params
LSTM(100)	(227,200)	183200
LSTM(100)	(227,200)	240800
conv 3 - 50	(225,50)	30050
conv 3 - 50	(223,50)	7550
conv 3 - 50	(221,50)	7550
sigmoid	50	552550
Total params		1.02×10^6

Table 3: LSTM architecture

4.4. A novel approach using LSTMs

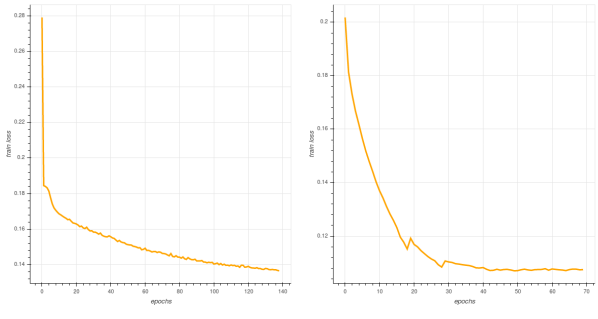
4.4.1 Architecture Design

Following Deep Learning best practices, we started from the simplest LSTM network, using a single network followed by a dense layer. We found that using only the last output was not enough to extract meaningful data from the network, so we used whole sequence outputs. We found this to be working quite well, but this required a huge number of parameters in the dense layer. Since learned features should be time invariant, the dense layer was replaced by a parameter-sharing section: 3 layers of 1D convolutions, with 50 filters, one for each tag to detect. We found that adding more parameters did not significantly impact the performance of the network; instead, we found the input sequence length lead to significant improvements when it was reduced. In fact, by inspecting the weights, we found that on longer sequences they quickly saturated.

We also found that the model seemed unable to overfit data. Even when increasing the parameter number, increasing the number of levels or reducing the dropout rate on the dense level, the validation loss did not seem to grow after hundreds of epochs and little reduction in learning rate (Figure 2a, 2c). This is in contrast with what happened with the other models, as shown in Figure 2b and 2d, where the effect of overtraining becomes apparent: before an early stopping restarts the training from the best model so far, we observe a simultaneous train loss decrease and validation loss increase.

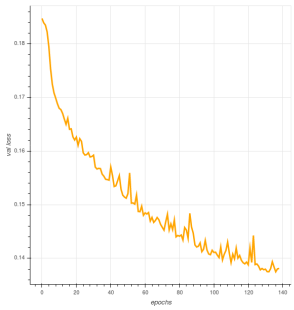
4.4.2 Tuning the Mel transform

To reduce the sequence length we had to tune the stride. We obtained a 0.861 AUC using 2048 samples as FFT length and a longer stride of 2560 samples. This was not ideal, since parts of the song were consequentially ignored. So, we modified the input to the network, splitting each song in 4 segments, all with the same labels, and averaging the

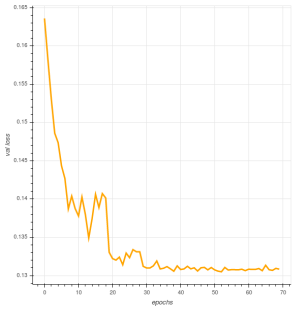


(a) LSTM Train loss

(b) Raw CNN Train loss



(c) LSTM validation loss



(d) Raw CNN validation loss

Figure 2: Loss functions for each training epoch

predictions of the segments for the same song during the testing phase. This allowed us to gain almost a 1% increase, reaching our best performances of 0.87 AUC.

4.5. Optimization

The task is a multi-class, multi-label classification problem, so we use the binary cross-entropy as loss function. We use the SGD optimizer for all the experiments, with an initial learning rate of 0.01 and a momentum of 0.9. In addition, to improve the learning capabilities of the network, we use a two level learning rate decay:

- **Local decay** – the learning rate is updated after each batch of training, and reset at the beginning of each epoch

$$\mu_{i+1}^{(LD)} = \frac{\mu_i^{(LD)}}{1 + local\_decay * iterations} \quad (1)$$

- **Global decay** – we use the *early stopping* technique, where the loss on the validation set is monitored, and when it does not improve for a certain number of epochs, the learning rate is reduced and the training is restarted from the best performing checkpoint:

$$\mu_{i+1}^{(GD)} = \mu_i^{(GD)} (global\_decay)^{(n\_early\_stops+1)} \quad (2)$$

4.6. Performance evaluation

Choosing a metric to evaluate the model is not trivial, since we're facing a multiclass multilabel classification task with an unbalanced dataset. Tags are sparse and most metrics do not help to evaluate the model. For example, if we consider the accuracy as a metric, it's easy to reach really high performance due to the sparsity of the dataset: the model basically predicts zero for every tag.

So, according to the literature, and to be able to compare results with previous works, we decided to use the Area Under the Receiver Operating Characteristic curve (*AUC* from now on). This metric takes into account different thresholds distributed on a linear space.

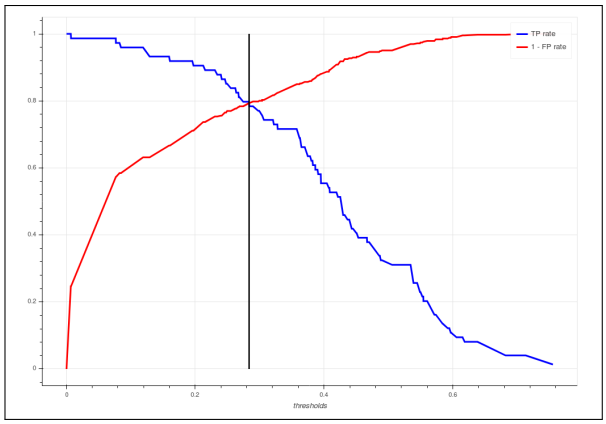


Figure 3: Example of threshold selection on `classical` tag

AUC We compute the predictions (which are real numbers between 0 and 1) and, for each threshold, and we compute the rate of True Positives and False Positives. A good model with high confidence should generate predictions which are really close to the correct value, which is either zero or one. Thus, changing the threshold would not influence the prediction, which would be correct for most thresholds. In this case, the AUC is maximum.

Choosing the best threshold After training, we select the threshold with the same Recall (TP_Rate) and Specificity ($1 - FP\_Rate$), which is the intersection between the TP Rate $\times$ Threshold curve and the FP Rate $\times$ Threshold curve, so as not to prioritize one or the other (an example is shown in Figure 3).

4.7. Results and discussion

In Table 4 are presented the results of our experiments. It is evident that the performances among the approaches are not radically different, the main difference being in computational requirements. The raw approach is more resource demanding, even if the preprocessing part is not required, and does not appear worthy given that computationally eas-

Result of our approaches			
model	AUC	TPR	FPR
Mel Spectrogram - 2D CNN	0.894	0.827	0.173
Sample CNN - 1D CNN	0.880	0.812	0.187
LSTM	0.870	0.800	0.203

Table 4: Result of each model, with TPR and FPR computed using the best threshold for each class

ier models, using Mel input, have comparable, if not better performances. The LSTM model is highly experimental, and our purpose here is only to show its potential.

4.7.1 Performance on external data

We have also tested our model on recent songs not in the MagnaTagATune dataset, leading to coherent, although noisier, results. For example, we're sure you are all familiar with the popular EDM hit Astronomia by Tony Igy. Here we publish the predicted labels:

```
trance   dance     loud
 male     fast   electro
         techno
```

While these are the predictions for AC/DC's Back in Black first 30 seconds:

```
spanish   bass   country
  jazz     pop     male
  rock   guitar   drum
```

While we are surprised that Brian Johnson's falsetto is correctly recognized as `male`, still we kind of disagree with the network identification of `spanish` music elements.

These are the tags for Lynyrd Skynyrd's Free Bird guitar solo:

```
hard   heavy   sitar
loud   india    fast
rock  guitar
```

Interestingly, the `sitar` tag is probably recognized because of its similarity in timbre with the guitar. `india` is probably learned as being strongly associated with it.

5. Conclusion

We observe that all performances are close to the state of the art, and that performances on commercial music seem to be poorer, probably indicating that a breakthrough improvement would need higher quality data. Also, we observe how simple, traditional models seem to tackle the problem well.

Further improvements on the LSTM model can be made by exploring more parameter combinations for the Mel spectrogram, or using different hand crafted features. The architecture can be further improved by means of ablation studies to add complexity and handle the likely underfitting we have observed.

References

[1] J. Lee, T. Kim, J. Park, and J. Nam, "Raw waveform-based audio classification using sample-level CNN architectures," *CoRR*, vol. abs/1712.00866, 2017. [Online]. Available: http://arxiv.org/abs/1712.00866

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[3] "End-to-end learning for music audio," in *International Conference on Acoustics Speech and Signal Processing ICASSP*. IEEE, 2014, pp. 6964–6968. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6854950

[4] K. Choi, G. Fazekas, and M. B. Sandler, "Automatic tagging using deep convolutional neural networks," in *ISMIR*, 2016.

[5] P. Kozakowski and B. Michalak, "Music genre recognition," Oct 2016. [Online]. Available: http://deepsound.io/music_genre_recognition.html