MUSICNN FOR DMX LIGHTING SOFTWARE AUTOMATION

Pol Mor-Puigventós
LIACS, Leiden University
Niels Bohrweg 1, 2333CA, The Netherlands
p.mor.puigventos@umail.leidenuniv.nl

ABSTRACT

We present an on-line song sections detector, identifying the changes of verse and beginnings and conclusions of the chorus for a song. We use the *musicnn* convolutional network [1] and some original experimental features based on the output tags of the network to predict switches among the song's sections. This information is used to launch different lighting configurations on a DMX lighting controller, achieving continuous audio processing for automated operation of lighting fixtures on a venue.

The code can be found on the author's Github. [1]

1. INTRODUCTION

We are interested in achieving an automatic behaviour of the lighting controller during the venues. Generally, the operator performs the initiations and ceasing of the previously prepared scenes. In our case, employing the online processing of the music played, we can detect the patterns that the lighting operator would use to perform the scene changes. For this task, we focus on a ConvNet predicting the song's sections and a lighting software that can operate autonomously.

When it comes to finding the network, music information retrieval tasks such as music classification and regression can be tackled through convolutional neural networks (ConvNets). However, in the musical domain, there are few datasets available with annotated pieces of audio, making it difficult for researchers to employ methods like ConvNets to solve specific audio-related problems. A possible solution to this lack of task-specific datasets is transfer learning. It is possible to train a deep neural network with the major actual datasets to solve a general classification task. Next, any researcher looking for a more domain-specific audio classification can fine-tune this network adding layers on top, the needed output and re-train for a few epochs with a small dataset oriented to the new task.

[1] https://github.com/poldemort/Musicnn-for-DMX

On the other hand, lighting fixtures controllers for venues such as concerts or clubs always require a technician that installs and patches the fixtures on the expected DMX channels. Later, the operator must define scenes where all the fixtures behave together using a lighting console or a computer with a lighting software and a USB-DMX converter. Finally, during the venue, the operator performs the initiations and ceasing of the previously prepared scenes, with the possibility of combining them, following the music rhythms and patterns.

2. RELATED WORK

2.1 Convolutional neural networks for audio tagginng

Multiple works have been published since the early 2010s to predict tags in audio samples [2]. Later, some were trained on the Million Song dataset [3]. These were able to classify songs automatically, tagging them using deep convolutional neural networks [4] [5] [6].

Many publications have been centred on the sentiment analysis of the studied songs taking into account the temporal variations of these sentiments. Using different sound features [7] or performing continuous music mood regression evaluating arousal and valence online per timestep [8].

An example of transfer learning for music classification and regression tasks is the work from Choi and others [9] where a big network is trained and later fine-tuned for six different tasks separately. On the other hand, *musicnn* [1] is proposed with a set of ConvNets for music audio tagging. *musicnn* not only predicts the labels of a song (like techno, drums, male vocals...) but also the temporal evolution of these possible 50 tags. These musically motivated convolutional networks (in the *musicnn* case also evaluating temporal features) offer a wide range of possibilities if are used as transfer learning to solve a different task [10] [11].

2.2 Lighting controllers and operation

The Digital Multiplex protocol (DMX or DMX512) is widely used for controlling lighting dimmers, fixtures and special effects devices. A unidirectional signal is sent from the lighting controller to the fixtures in series, transmitting asynchronous serial data at 250 kbit/s where each fixture is patched to take one or more information slots at each frame to modify its behaviour, see the technical standard [12]. This signal can be originated on a lighting controller de-

signed for this use [13] or from a computer with lighting software and a USB-DMX converter [14].

3. SETTING UP THE ENVIRONMENT

For simplicity, we first prepare the environment using Conda [15] on an Ubuntu machine as we do not manage to install it initially on a Windows machine due to package incompatibilities. We install the needed packages for the environment for the *musicnn* network with the the packages `python` 3.7.13, `tensorflow` 1.15.0 and `librosa` 0.9.1. The network was created in these conditions and multiple methods are deprecated in actual versions. Later, we want to replicate the environment on our Windows machine.

Several options are available for copying an environment to a different machine, those are cloning the environment, creating a snapshot of the environment, creating a list of packages or exporting a YAML file to be used later in the creation of a new environment.

Only the last option can be used when the operative systems of the machines to export and import the environment are different. So, we use the exported YAML file to create a new environment on Windows. However, we find the following package dependencies:

- librosa 0.9.1 depends on 1.17.0<=numpy
- musicnn 0.1.0 depends on 1.14.5<=numpy<1.17
- numba 0.55.1 depends on 1.18<=numpy<1.22

A priori, the situation depicted above is impossible to solve as the combination of requirements of `librosa` and *musicnn* can't be satisfied by any version of `numpy`. We solve the situation by removing the packages `librosa` and `numba` from the list on the YAML file while keeping `numpy` 1.16.6. Later, we install them manually without any errors. The YAML file can be found on the author's Github.

4. METHODS

4.1 *musicnn* model

We use the *musicnn* model `MTT_musicnn` trained on the MagnaTagATune dataset to analyze the temporal evolution of the output tags. We adapt the output of the features extractor, to obtain the values of the labels every half second without overlap. The extractor calculates the spectrogram with the library `librosa` [16] and generates a forward pass of the ConvNet using the library `tensorflow` [17].

An example of the output of the model is plotted in Figure 4. In this case, we analyze 75 seconds of a song, but the network can handle online processing and outputs the tags distributions every half a second, for example.

With this, we decide to examine the output of the ConvNet at every timestep, maintaining a small buffer with the last few tag values. So, we generate online actions to modify the lighting scenes given both the present timestep t and some recent past tags $t-n$. The action releases are defined as follows:

4.1.1 Total energy drop below a threshold

At every timestep t we sum the values of all the tags. This way, we obtain the total energy E of that audio (in terms of the generated tags) at t. We calculate the absolute value of $E(t) - E(t-1)$ and check if that difference is above a constant threshold defined in previous experiments. When this happens, we decide a scene switch to the ID=2 from Table 1.

4.1.2 Top-5 tags change

At each t we generate an array of the tags with the highest values. At the next timestep, we check the differences in this Top-5 array. When all the tags change from $t-1$ to t we can expect a change of section on the song as different instruments are used or there is a silent period. When this happens we output the scene switch to the ID=3 from Table 1.

4.1.3 A top tag drops below threshold

Also, using the Top-5 activated tags per timestep we check if one of them drops. There are two conditions to satisfy: (1) in $t-1$ the tag has to belong to the top-5 with an energy higher than a threshold, (2) at instant t, this same tag has

Figure 1. Scene type 1, activated by the keyboard input 'w'.

Figure 2. Scene type 5, activated by the keyboard input 'v'.

ID	Scene features	Arousal	Activation keys		
1	static heads, PARs full on, static RBG	low	q	w	e
2	moving heads, intermittent PARs, static RGB	medium	r	t	y
3	intermittent moving heads, intermittent PARs, static RGB	medium	a	s	d
4	wide-angle moving heads, static PARs, shapes in RGB	medium	f	g	h
5	stroboscopic moving heads, intermittent PARs, shapes in RGB	high	z	x	c
6	wide-angle moving heads, intermittent PARs, stroboscopic shapes in RGB	high	v	b	n

Table 1. Definition of the scenes to launch during the execution

to drop to below another threshold. This can mean that one of the principal instrument lines start or stop, leading to a possible change of the song. We output a scene switch to the ID=4 from Table 1.

4.1.4 Changes in the vocals or no-vocals tags

In previous experiments we detect the high activation of the tags *vocals* and *no vocals*. As these tags are very often accurately generated by the model, we decide to use them to detect changes in the songs. When the vocals start or stop, we generally find a switch in the song. We keep a buffer of two timesteps and calculate the mean values of both tags at $t-2$ and $t-1$. If any of the classes drops or rises from a threshold at instant t compared to the computed past averages, we output a scene change to the ID=5 from Table 1.

4.1.5 Peak detection

For the last detection routine, we simply check for the peaks per channel above a threshold. If a channel is strongly activated it is generally because of a starting section of a song. For instance, see Figure 4 for the tag *rock* with a peak value (in yellow colour) at around second 54. In this case, the chorus of the song is starting. We would tag this with the ID=6. [2]

4.2 Global routine

For our demo, the first timestep of a song is always labelled with the scene type ID=1, to start the song with a low-arousal scene. Likewise, the last timestep gets the keyboard key switching to the *full on* scene, where all the fixtures are static and project white light, exhibiting the end of the show.

In addition, it is common to predict multiple changes on the audio tags one after another through the multiple detectors within the span of a few seconds. For this reason, there is a control over the global scene switches, obviating the scene changes that are proposed by the algorithm just after another scene change. We keep a minimum of 5 seconds between lighting changes to ensure the stability of the performance.

It is important to mention that all the features depend on heuristics, especially because a threshold has to be defined in advance or a defined number of tags have to change to release a new scene. Every scene type from 1 to 6 progressively generates higher arousal and we ordered the song changes detectors accordingly. For instance, we observe that the total energy drop of the song happens when *soft* changes in the song are felt. In contrast, when we find peaks in specific tags, we find high-arousal moments of the songs.

4.3 Adapting the output for lighting automation

We are interested in detecting the changes in verse and the start and end of the chorus. Also, silent or arousal instants. These features motivate the lighting operators to unleash different pre-defined scenes for the lighting fixtures.

Depending on the setup, the fixtures employed are different. In our experiments, we use 8 moving heads and 8 PAR LED on the different trusses hanging from the ceiling. Also, there are 36 LED fixtures on the floor.

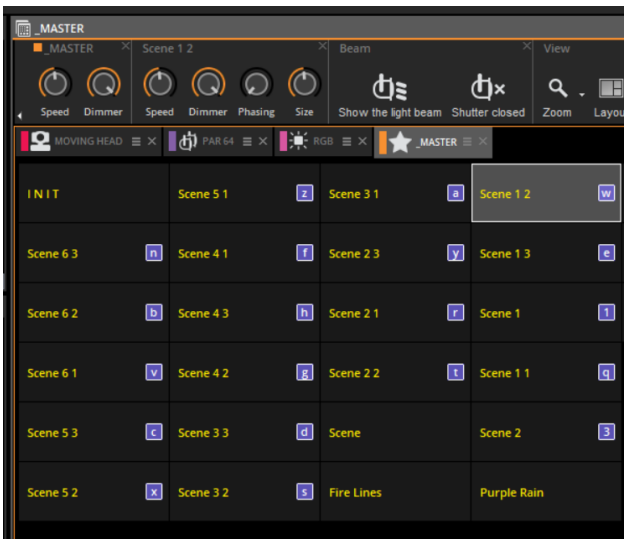

Figure 3. Scenes selection panel of the SunLite software with the predefined scenes for our experiments. Every prepared scene is linked to a letter of the keyboard.

We predefine 18 scenes with different intensities, colours, *gobo* [3] and patterns. These scenes are linked to an activation keyboard key as can be seen in Figure 3. Once a key is pressed, this scene is executed. In Table 1, we show the six different main configurations, each one of them has three different possible activation keys, with changes in the

[2] On the example, there is no resulting activation on the second 54 in Figure 4. This is because the start of the chorus has been detected previously by the Top-5 tags change, three seconds before, generating a scene of type 3.

[3] The word *gobo* stands for the projected shape from the moving head fixtures.

colours, *gobo* and on/off patterns. This way, we display more variety of scenes, even if there are just six scene models. See Figures 1 and 2 for low and high arousal scenes respectively. Note that the static screenshots do not generate the same sensations as when observing the simulation on a video.

5. RESULTS

The obtained results are hard to measure. We define the scene changes detectors while experimenting with audio samples of contemporary rock music of bands such as Bastille, Crystal Fighters, Tame Impala, or The Strokes, for instance. We also define the thresholds for the algorithm following the general behaviour of these songs.

As an example, we plot the tags of the song Pompeii by Bastille generated by *musicnn* in the upper image of Figure 4. We observe that at 18 seconds, multiple tags are activated with the change from the introduction part to the start of the first verse. At around 32 seconds we observe a second pattern of activations, depicting the transition to the second verse. Later, at around 50 seconds, the chorus starts with continuous activation of the *rock* class and a more modest presence of the label *loud*. The chorus finishes after one minute and 13 seconds after the start of the song, as we clearly observe in the figure. All these song sections are predicted in the output of our algorithm, activating the different scene types values, see the lower row labelled as "Switches" in Figure 4.

However, when changing the domain of the audio samples used, the performance of the detector decreases, still proposing scene switches during the songs but with a behaviour closer to being random. We experiment less capacity to detect changes in electronic and dance music which generally play the same notes generated with a synthesizer. Also, on music that is more than 10 years old, that has been mastered with less sharp sounds, causing very different activations on the output of the ConvNet.

For example, when looking at Figure 5, we show the tags distribution and the decisions of our predictor in terms of song sections for the song The Nights of Avicii. We observe multiple patterns in the upper image and some activations in the output of the algorithm. The change of the song from the verse to the chorus occurs at 32 seconds the second part of the chorus starts at around 39 seconds. Any of them are predicted by our algorithm due to the low generalisation capacity.

6. CONCLUSIONS

We can affirm that it is possible to detect the changes of verse and the starting and ending of the chorus using ConvNets. We adapt the output of *musicnn* obtaining excellent results on the songs that belong to the domain of the ones that have been used for calibration while not finding the song's sections on songs of other genres. We consider this issue addressable with normalizations of the parameters and a wider amount of detectors.

On the other hand, we now consider it feasible to use a ConvNet like *musicnn* for transfer learning to solve this task. We should label several songs with their sections and train and fine-tune some convolutional layers on top of *musicnn*. This way, online continuous music section detection would be possible and much more robust than the algorithm we developed.

7. LIMITATIONS

The network has been trained on the MagnaTagATune dataset, which might be unbalanced among the genres' songs and release years, producing a possibly biased classifier for our task.

Our implementation is calibrated on a narrow amount of songs which causes poor performance of the predictor for songs with different patterns.

8. FUTURE WORK

The defined thresholds are defined in previous experiments generalising for all the songs used during the experimentation process. We observe a high variability of the optimal values for these thresholds among different songs. Especially when these songs belong to very different genres or their master has been produced in very different ages. We propose a dynamic normalization of these tags based on the last timesteps of the song to adapt them to each song.

We have not found literature on predicting song sections or on having available datasets for that matter. We now prove that this is possible using networks like *musicnn* and applying some techniques on top. So, a fully convolutional network can solve this task and perform online song section prediction usable for lighting operators to decide the behaviour of their fixtures in real-time applications.

9. REFERENCES

[1] J. Pons and X. Serra, "musicnn: Pre-trained convolutional neural networks for music audio tagging," 2019. [Online]. Available: https://arxiv.org/abs/1909.06654

[2] E. Law, K. West, M. Mandel, M. Bay, and J. Downie, "Evaluation of algorithms using games: The case of music tagging." 01 2009, pp. 387–392.

[3] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," 2011.

[4] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6964–6968.

[5] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," 2016. [Online]. Available: https://arxiv.org/abs/1606.00298

[6] K. Choi, G. Fazekas, and M. Sandle, "Explaining deep convolutional neural networks on music classification,"

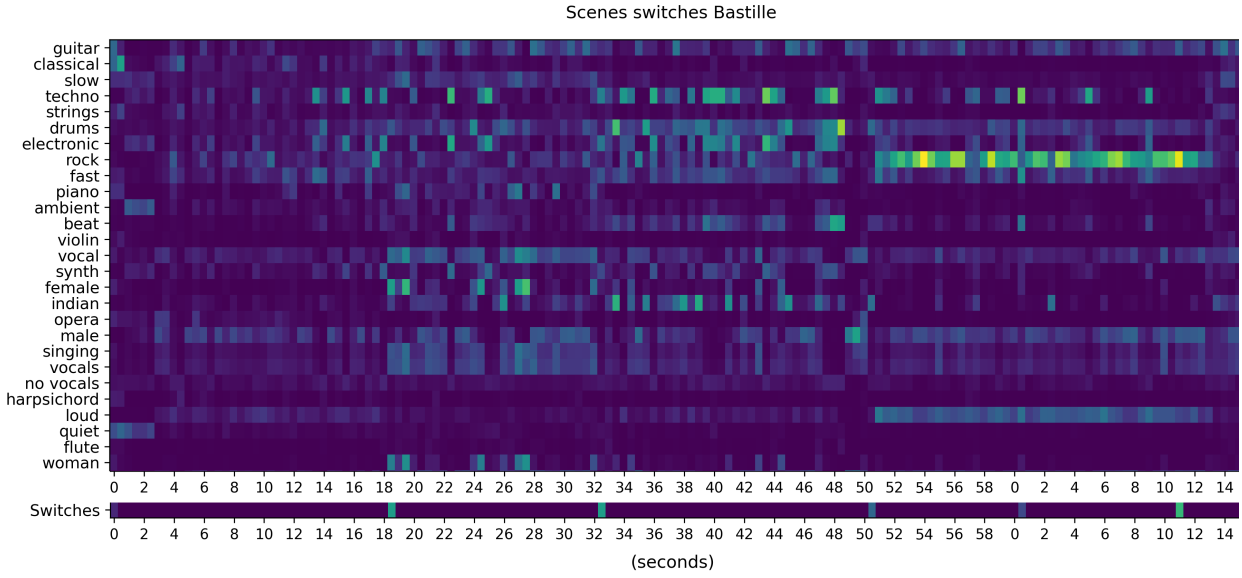

Figure 4. The upper plot is the tags extraction of the song Pompeii from Bastille. For simplicity, only 27 of the 50 tags are plotted and for just the first 75 seconds of the song.The second plot is the output of our algorithm proposing the scenes switching for the song.

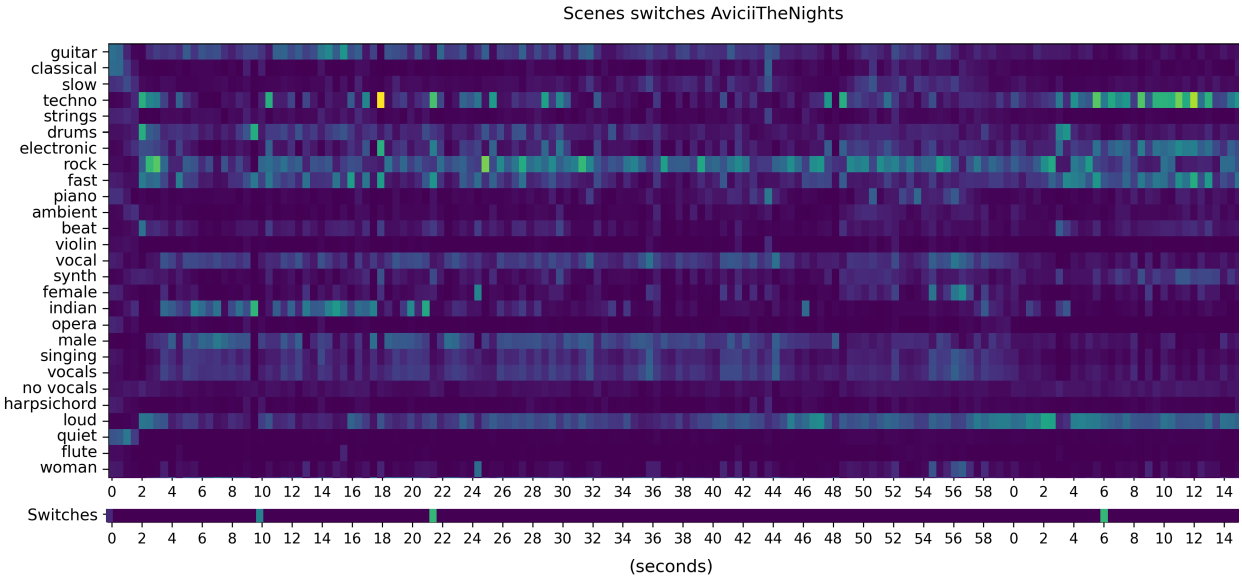

Figure 5. The upper plot is the tags extraction of the song The Nights from Avicii. For simplicity, only 27 of the 50 tags are plotted and for just the first 75 seconds of the song. The second plot is the output of our algorithm proposing the scenes switching for the song.

2016. [Online]. Available: https://arxiv.org/abs/1607.02444

[7] E. Coutinho and A. Cangelosi, "A neural network model for the prediction of musical emotions," *Advances in cognitive systems*, vol. 71, p. 333, 01 2010.

[8] F. Weninger, F. Eyben, and B. Schuller, "On-line continuous-time music mood regression with deep recurrent neural networks," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5412–5416, 2014.

[9] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," 2017. [Online]. Available: https://arxiv.org/abs/1703.09179

[10] J. Pons, T. Lidy, and X. Serra, "Experimenting with musically motivated convolutional neural networks," in *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, 2016, pp. 1–6.

[11] J. Pons and X. Serra, "Designing efficient architectures for modeling temporal features with convolutional neural networks," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2472–2476, 2017.

[12] K. G. Ruling *et al.*, "American National Standard ANSI E1.11-2008," https://tsp.esta.org/tsp/documents/docs/ANSI-ESTA_E1-11_2008R2018.pdf, Accessed on 01/06/2022.

[13] A. Limited, "Avolites: Creative visual control," https://www.avolites.com/, Accessed on 01/06/2022.

[14] G. Nicolaudie, "Sunlite dmx lighting software for pc," https://www.sunlitepro.com/, Accessed on 01/06/2022.

[15] "Conda:system-level binary package manager," 2015. [Online]. Available: https://docs.conda.io/

[16] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[17] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/