Automatic playlist generation using Convolutional Neural Networks and Recurrent Neural Networks

Rosilde Tatiana Irene*, Clara Borrelli+, Massimiliano Zanoni+, Michele Buccoli+, Augusto Sarti+
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano,
piazza Leonardo da Vinci 32 - 20133 Milano, Italy
Email: *rosildetatiana.irene@mail.polimi.it; +{name.surname}@polimi.it

***Abstract*—Nowadays, a great part of music consumption on music streaming services are based on playlists. Playlists are still mainly manually generated by expert curators, or users, process that in several cases is not a feasible with huge amount of music to deal with. There is the need of effective automatic playlist generation techniques. Traditional approaches to the problem are based on building a sequence of music pieces that satisfies some manually defined criteria. However, being the playlist generation a highly subjective procedure, to define an a-priori criterion can be an hard task in several cases. In this study we propose an automatic playlist generation approach which analyses hand-crafted playlists, understands their structure and evolution and generates new playlists accordingly. We adopt Recurrent Neural Network (RNN) for the sequence modelling. Moreover, since the representation model adopted to describe each song is determinant and is also connected to the human perception, we take advantages of Convolutions Neural Network (CNN) to learn meaningful audio descriptors.**

***Index Terms*—automatic playlist generation, deep learning, machine learning, music recommendation, music organization**

I. Introduction

In today music fruition scenario, where the most of music consumption happens through music streaming, playlists have an important role. From one side, playlists are one of the most valuable solutions for both users and content providers to aggregate songs, from the other side they are an effective way for music discovery. A great amount of users, indeed, relies on playlists to discover new music they may like. For example *Discover Weekly* or *Release Radar* playlists by Spotify, reached nearly 100 millions followers, with a consequent exponential growing of interests by the music industry.

As far as content providers is concerned, most of the available playlists are still manually created by curators. Even if we underline the importance of the involvement of experts, the manual creation process has some important issues [1]. First, it is a highly time consuming and challenging task, due either to the required level of music expertise and to the vast size of the available music contents. Secondly, it does not allow to generate playlists that are specifically personalised on single users preferences. Finally, manual creation tends to promote well-known song, since curators will hardly reach little know musicians. These are the reasons why automatic playlist generation has become one of the most challenging and interesting application of the MIR research field.

A playlist can be defined as a sequence of songs aggregated in order to satisfy a specific criterion. Traditionally the problem has been addressed by *similarity-based* approaches, whose intent is to construct playlists which are as much homogeneous as possible with respect to a given set of features [2] [3]. However, users are more interested in the evolution within the sequence of songs rather then their uniformity [4]. In order to take into account the user preferences, other methods exploit collaborative filtering techniques [5] [6]. Nonetheless, these approaches are not always applicable when it comes to recommend new items, an issue known as *cold-start problem* [7] [8].

However, it is very hard to define an effective set of criteria that is valid for a general application since they can vary by task to task and they can be either very objective (uniformity of the genre) or very subjective (user taste). For this reasons, according to McFee *et al.* [9], to learn the criteria from the analysis of human-crafted playlists would result in more effective and general methods. At this purpose, McFee *et al.* exploits methods used in language modeling and [10] [11] uses Markov chains to model the transitions between songs through a probabilistic formulation.

To better match the human playlist creation process the use of techniques devoted to emulate the human thinking would be more effective. For this reason, some studies using Deep Neural Networks in the context of automatic playlist generation have been presented [12]. In [8] the authors use Recurrent Neural Networks (RNN) for playlist continuation by the analysis of the last song in the sequence. [13] uses Convolutional Neural Network (CNN) to learn a meaningful representation of the song and to classify the piece as belonging or not belonging to a specific playlist.

In this study we evolve the works in [8] [13]. Being an analysis of manually-crafted playlists, in order to mimic our perception, we use CNN to learn a feature representation for each song. We then provide these features as input for an RNN devoted to model the tracks relationships underlying their corresponding playlist's structure, which defines the playlist evolution. Unlike [8], in the playlist continuation task our approach take advantages of long-term analysis. This means that the whole playlist affects the prediction of a new song rather than just the tail.

II. Problem Statement

Playlists are defined as sequences of songs described trough a specific *Background Knowledge* all belongings to a music

978-9-0827-9703-9/19/$31.00

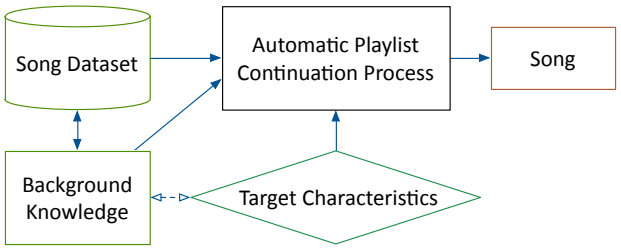


Fig. 1: Automatic playlist generation model.

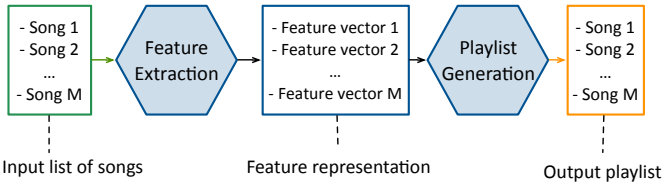


Fig. 2: The complete schema of the automatic playlist generation process.

library and grouped according to a given criterion or *Target Characteristic*.

The Target Characteristics are the properties the playlist should have in order to be considered "well-built". These characteristics are really dependent on the purpose of creation of the playlist. Examples of Target Characteristics are constraints on genre or mood of the music, match to one or more keywords, homogeneity or novelty with user's listening history. Of course, these characteristics can be objective or strongly subjective, hence it is not always straightforward to define them in formal terms.

The Background Knowledge is a set of descriptors of the considered songs that allows to verify if the given criterion, i.e. the Target Characteristic, has been respected. This information can be of different type, like audio signal features, metadata or usage data, and its formulation will be dependent on the specific playlist creation purposes. For example, if the creation purpose refers to the audio content, like constant rhythm intensity, the Background Knowledge should be comparable content-based descriptors, like rhythmic features extracted from audio signal of the tracks. If instead the criterion is the year of release of the songs, the Background Knowledge can be extracted from metadata of the songs, that usually includes this information. The Background Knowledge should be designed in order to be enough expressive to allow the verification of satisfaction of Target Characteristics. Regarding content-based audio features, to enhance the effectiveness of the description is possible to combine classic signal processing with deep learning architectures, like Convolutional Neural Networks. These methods, that will be presented in details in the next Section, allow to obtain a representation of the music track which spans different levels of abstraction and is able to reveal significant patterns.

Given this framework, we can define the task of playlist continuation as shown in Figure 1. Formally a playlist of length L can be expressed as a set $\mathscr{P} = (s_0, s_1, \ldots, s_{L-1})$ where s_i is the i-th song of the playlist. Given $\mathscr{P}$, the goal of playlist continuation is to predict $\hat{s}_L$ such that the Target Characteristics are respected by the extended playlist as much as possible. Playlist generation problem can be seen as a specific case of playlist continuation. Indeed, a new playlist can be generated by having one seed song, hence starting with $L = 1$, and by applying playlist continuation in a iterative fashion until reaching the playlist desired length M.

In this work we are interested in designing a playlist generation or continuation process such that the Target Characteristics match the one that would be generated by human playlist creation process. In particular we try to embed the decision-making of curators in the model and replicate the trend present in human-generated playlist. The solution proposed employs a deep learning architecture, Recurrent Neural Networks with Long Short-Term Memory cells, that is able to model evolution of sequences over time. This allows to avoid a formal definition of Target Characteristics by training the neural network on hand-crafted playlists. The details on the RNN and LSTM used in this work will be introduced in the next Section.

III. Proposed Approach

Our method for automatic playlist generation involves two steps: feature extraction from audio content and actual automatic playlist generation, as shown in Figure 2.

On the one side, the feature extraction stage aims at extracting a representation of the aforementioned Background Knowledge and hence providing a high-level descriptor of the audio content that is meaningful for the playlist generation task. On the other side, the playlist generation step aims at finding new items from their feature representation that follows the same Target characteristics.

Feature Extraction. In order to model a playlist generation, we need to extract a high-level representation of music, which is close to the way people think of music, such as genre, mood, or natural language description of the music. Such representation however would need to address several tasks, i.e. genre recognition, mood recognition and auto-tagging.

In [14] Keunwoo Choi *at al.* present a five-layer Convolutional Neural Networks (CNN) combined with transfer learning that is effective to predict mood, genre and audio events. In particular, the (pooled) output after each layer provides a compact, multi-level and flexible descriptor for audio content. For this reason, we use their network for our work. For an in depth description of the network structure and training methodology, please refer to [14].

In order to use the feature extraction network, we first compute a time-frequency representation of the audio signal, namely the mel-spectrogram, which is a perceptually-relevant version of the spectrogram and has been proven to be effective with deep neural networks [14].

As a result of the feature extraction stage, we compute a F-feature vector $\mathbf{s}_i \in \mathbb{R}^F$ for each song s_i in the playlists.

Playlist generation. The playlist generation stage is composed of two steps: first we train a model to recognise the pattern within a sequence of songs; and once this model is trained, we use it to generate new items, corresponding to the playlist continuation.

Given an input playlist $\mathscr{P} = (s_0, s_1, \ldots, s_{L-1})$, we extract the feature representation and end up with the input matrix $\mathbf{P} = [\mathbf{s}_0, ..., \mathbf{s}_{L-1}] \in \mathbb{R}^{L \times F}$. The first step is concerned with modeling the structure and the evolution of the input sequence in order to predict a coherent playlist which can also be employed as a suitable continuation. The output of this step is a sequence of feature vectors that are the best candidate for the continuation.

A playlist can be seen as a sequence of elements. For this reason, in this study, we use a RNN for modeling the playlist sequences and carry out the song prediction step. A RNN aims at modeling the sequence $\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_{L-1}$ by modeling the distribution of each step of the sequence $P(\mathbf{s}_i|\mathbf{s}_{i-1}, \boldsymbol{\theta})$, with θ the parameters of the model. As its name suggests, by recurrently unwrapping the formula, we can model each element of the sequence as a function of its previous elements:

$$\begin{aligned} P(\mathbf{s}_i|\mathbf{s}_{i-1}, \boldsymbol{\theta}) &= P(\mathbf{s}_i|P(\mathbf{s}_{i-1}|\mathbf{s}_{i-2}, \boldsymbol{\theta}), \boldsymbol{\theta}) \\ &= \cdots = P(\mathbf{s}_i|\mathbf{s}_{i-1}, \mathbf{s}_{i-2}, \ldots, \mathbf{s}_0), \end{aligned}$$

where in the last step we neglected the nested notation for the sake of clarity.

In particular, we employ a two-layer RNN architecture based on long short-time memory (LSTM) cells, which involve several gates that help the model understand what to "remember" and to "forget" about the past samples, through an internal state. LSTM cells are also effective to address the RNN training issues related to the *vanishing gradient* and *exploding gradient* phenomena [15]. We apply a dropout layer after each LSTM layer to regularize training and make it more robust.

Given the input playlist, the RNN emulates its evolution by predicting a sequence of feature vectors which reflects the pattern in the input playlist. Formally, given an input playlist $\mathscr{P}$ of length L defined as $\mathscr{P} = (s_0, s_1, \ldots, s_{L-1})$ the RNN is used to predict a playlist $\hat{\mathscr{P}}$ of the same length which is an estimate of all the songs' continuations:

$$\hat{\mathscr{P}} = (\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_{L-1}, \hat{s}_L) \ . \tag{1}$$

The last song $\hat{s}_L$ can be employed as a continuation for $\mathscr{P}$ since it has been predicted by following the input playlist's composition logic. This process can be repeated iteratively to generate a playlist of any desired length M.

In the training phase, we provide a set of input sequences $\mathbf{X}$ and a set of ground-truth sequences Y as the ground truth for the output. We use $\mathbf{X} = [\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{L-2}]$ as input playlist and $\mathbf{Y} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{L-1}]$ as desired output. This training data allows the model to learn the dependencies between the several playlist's tracks as

$$\hat{\mathbf{Y}} = [\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, \ldots, \hat{\mathbf{s}}_{L-1}] = f([\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{L-2}]; \Theta) \ , \tag{2}$$

where Θ are the network parameters.

The network is trained to minimize the mean squared error between $\hat{\mathbf{Y}}$ and $\mathbf{Y}$, hence making the network prediction closer to the ground truth.

Once the training phase is complete we can use the network in a generative way to receive as input the feature representation of the songs in a generic playlist and predict a sequence of feature vectors which are the best candidate for continuing it.

However, such a feature vector may not correspond to an actual track that the network has seen during training. This behaviour is in fact desirable, since we want the approach to be scaled and generalized to new content inserted in the music library $\mathcal{S} = \{s_1, ..., s_N\}$, which is an extremely common scenario in the music streaming services.

Therefore, given a generic predicted feature vector $\hat{\mathbf{s}}_i$, we continue the playlist with the song s_j such that $\mathbf{s}_j$ is the closest point to $\hat{\mathbf{s}}_i$ in $\mathcal{S}$ according to the Euclidean distance, which is coherent with the choice of the MSE as loss function during training.

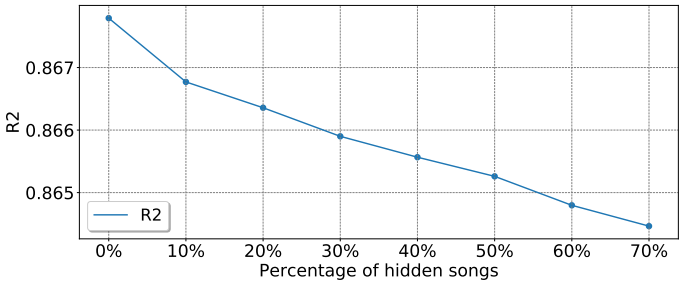
Finding the closest point in a F-dimensional space with a large amount of items, such as the current music libraries, is a computational expensive process if performed via a linear scan. Several approaches have been proposed to optimize this process. In our experiments, we employ a *r-tree* structure, which creates overlapping regions of samples and organizes them as a K-branch tree, achieving logarithmic complexity in the number of elements [16].

IV. Experimental setup and evaluation

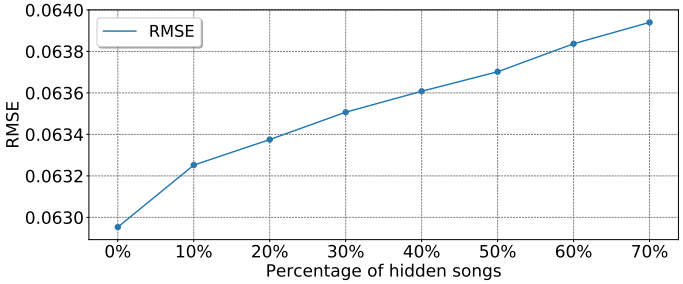
For our experiments we employed the *Art of the Mix-2011* (AotM-2011) dataset [17]. AotM-2011 is a collection of 101,343 hand-crafted playlists meta-data acquired from the Art of the Mix platform, with an average 19.6 song for each playlist. The playlists contain a total of 642,118 unique songs by 227,703 unique artists. We have employed the Spotify Web APIs in order to extract 30-second audio excerpts representative of the song sampled at 48KHz. Since the audio excerpts were not available for every song in the AotM-2011, we discarded the playlists with one or more missing audio content, leading to a total number of 44,819 fully-matched playlists. We extracted the mel-spectrogram for each song by computing the STFT with $N_{FT} = 512$ and $H = 256$ and a triangular mel-filterbank with 128 bands logarithmically distributed on the frequency axis. This processing phase has been implemented using *kapre* [18] to exploit GPU acceleration.

Each mel-spectrogram is then provided as input for the CNN architecture, which extracts a feature vector representation for the signal of dimension 1 x F, where $F = 160$ is the total number of features. The resulting vectors are aggregated into a matrix $\mathscr{P}$ of dimension L x F, where L is the playlist's length. This process is repeated for every playlist available in the dataset.

The produced matrixes are used to to train, validate and test the RNN model. The RNN consists of two LSTM layers each characterized of 512 neurons. In order to avoid overfitting, we

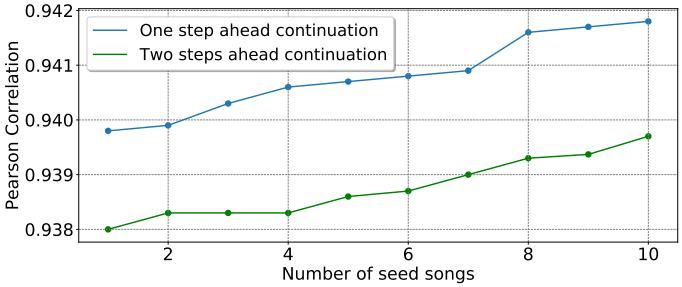


(a) R^2 score over percentage of hidden songs.

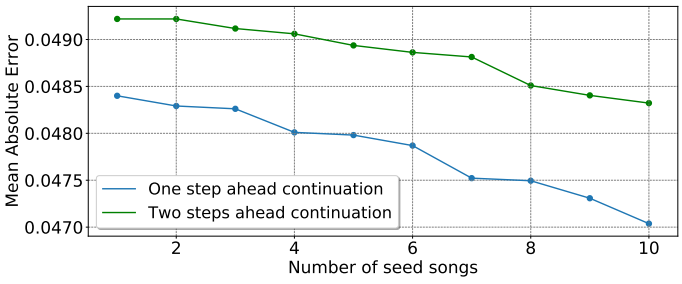


(b) $RMSE$ over percentage of hidden songs.

Fig. 3: Playlist generation with hidden songs test case results



(a) Pearson correlation ρ over number of seed songs.



(b) MAE over number of seed songs

Fig. 4: Playlist continuation with seed songs test case results.

have added a dropout layer with a dropout factor of 0.2 after each LSTM layer, i.e. randomly setting to 0 a portion of the previous cells outputs equal to their 20% during the training phase. The networks implemented using *Keras* library [19] with *TensorFlow* as back-end.

We have trained the RNN model using the 80% of the 44819 playlists that compose the dataset.

Each playlist of the training set has been divided into its correspondent input playlist $\mathbf{X} = (s_0, s_1, \ldots, s_{L-2})$ and desired output playlist $\mathbf{Y} = (s_1, s_2, \ldots, s_{L-1})$. The aim of our model in the training phase is to generate a prediction $\hat{\mathbf{Y}} = (\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_{L-1})$ which is as close as possible to the desired output $\mathbf{Y}$. To measure the prediction error between $\hat{\mathbf{Y}}$ and $\mathbf{Y}$ we have employed the *mean squared error* (MSE) as loss function. For the network's parameters estimation, we have employed the *Adam* optimization algorithm [20].

The testing phase has been computed using the remaining 20% portion of the dataset. We have employed six metrics to evaluate the quality of the prediction provided by our model. The metrics can be divided in two groups: performance metrics and distance metrics. The *performance metrics* aim to define how much the model is effective in understanding and reproducing the playlist evolution. To perform this evaluation we have chosen the *coefficient of determination* - (R^2) [14], the *explained variance* [21] and the *Pearson correlation* [5].

The *distance metrics* are employed to compute the accuracy of the model by evaluating the distance between s_i and its prediction $\hat{s}_i$. The chosen metrics are the *root mean square error* [22], the *mean absolute error* [23] and the *median absolute error* [24].

With the goal to evaluate our method in playlist continuation scenario, we used two test methodologies: *playlist generation with hidden songs* and *playlist continuation with seed songs*.

Playlist generation with hidden songs. The first methodology aims to measure the correctness of the model prediction $\hat{\mathbf{Y}}$ with respect to the ground truth playlist $\mathbf{Y}$ when an increasing portion of the input playlist $\mathbf{X}$ is hidden to the model. In other words, by iteratively provide as input a shorter version of $\mathbf{X}$, we decrease the provided amount of information. This allows us to evaluate the robustness of the method. To carry out this test we have defined a *hidden songs percentage* β varying from 0% to 70%. In order to compare $\hat{\mathbf{Y}}$ with the ground truth $\mathbf{Y}$, we add to the input playlist the last predicted feature vector, repeating the model prediction process until $\hat{\mathbf{Y}}$ shares the same length of $\mathbf{Y}$.

In order to provide an overall evaluation of the method, given a β we firstly performed the prediction for each playlist in the test set. We then retrieved the closest feature vectors employing the r-tree structure and finally we computed the six metrics to each playlist generated by the model with respect to its ground-truth. The overall evaluation is obtained as the average along the playlists for each metric.

For the sake of space we present here only the results related to R^2 and RMSE metrics (figures 3a and 3b). As expected, the effectiveness of the model decreases as the number of hidden songs increases. This is due to the fact that at each iteration of the test, i.e. for each value of β, we reduce the information provided to the model regarding the desired evolution and characteristics of the playlist. Nonetheless, the differential is very low ever for low β values.

Playlist continuation with seed songs. The second testing methodology is concerned with the evaluation of the playlist continuation performance of our model. Given a number of seed songs n_s varying from 1 to 10, for each input playlist $\mathbf{X}$

of the test dataset we provide the first n_s songs as input of the model, which outputs a playlist of the same length. From this output playlist we take the last predicted feature vector, which represents the song chosen as continuation for the input seed songs. From this prediction we firstly retrieve the closest feature vector in the dataset using the r-tree. Then, we compare this predicted feature vector $\hat{s}_{n_s-1}$ with the feature vector s_{n_s-1} at the same position in the ground truth Y. For the test purposes we have evaluated the model performance for the one prediction, namely the *one step ahead continuation* and two predictions, the *two steps ahead continuation*, repeating the test for a number of seed songs varying from 1 to 10. We present here only the results related to Pearson correlation ρ and the mean absolute error MAE (figures 4a and 4b. The increase of the amount of seed songs allows the model to better understand the playlist evolution, which causes an improvement of the model performance.

V. Conclusions

In this work we presented a content-based approach for the automatic playlist generation task. Our approach is based on a RNN that models the sequence of songs in a playlist as the evolution over time of its feature representation. The feature representation is learnt by means of CNNs.

The model is able to generate or continue playlists following the learned structure and evolution. We evaluated our approach using the Art of the Mix-2011 playlist data-set through quality and error metrics under two scenarios: playlist generation with hidden songs, playlist continuation with seed songs.

The model achieves good results over all the considered metrics, showing a good level of resilience to reduction of the input (first scenario) and the capability of increasing the scope of prediction to two steps ahead the current song (second scenario). In particular, the low decrease in terms of performance and error of the two steps ahead prediction makes us confident in the model ability of learning the long term dependencies between the songs in a playlist. With respect to [8] we did not experienced overfitting issues, possibly due to the larger size of the employed dataset.

The model is able to start a trend even starting by a single seed song; however in this case it is hard -even for a music curator- to understand the desired target characteristics of the playlist. In future work, we intend to investigate user's preferences allowing them to "steer" and customize the playlist generation. Moreover, we would like to use the prediction power of the model to sort songs in a collection, i.e., find the best way to present a set of songs in a sequence. This would improve the perceived quality of recommendation systems, such as the *Discover Weekly* playlist.

References

[1] G. Bonnin and D. Jannach, "Automated generation of music playlists: Survey and experiments," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, p. 26, 2015.

[2] C.-Y. Chi, R. T.-H. Tsai, J.-Y. Lai, and J. Y.-j. Hsu, "A reinforcement learning approach to emotion-based automatic playlist generation." in *Proceedings of the International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, 2010, pp. 60–65.

[3] M. Dopler, M. Schedl, T. Pohle, and P. Knees, "Accessing music collections via representative cluster prototypes in a hierarchical organization scheme." in *Proceedings of the 9th International Society for Music Information Retrieval (ISMIR) Conference*. ISMIR, 2008.

[4] D. Jannach, L. Lerche, and I. Kamehkhosh, "Beyond hitting the hits: Generating coherent music playlist continuations with the right tracks." in *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015, pp. 187–194.

[5] D. B. Hauver and J. C. French, "Flycasting: using collaborative filtering to generate a playlist for online radio." in *Proceedings of the First International Conference on Web Delivering of Music*. IEEE, 2001, pp. 123–130.

[6] Z. Chedrawy and S. S. R. Abidi, "A web recommender system for recommending, predicting and personalizing music playlists." in *Proceedings of the International Conference on Web Information Systems Engineering*. Springer, 2009, pp. 335–342.

[7] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation." in *Proceedings of Advances in Neural Information Processing Systems (NIPS) Conference*, 2013, pp. 2643–2651.

[8] K. Choi, G. Fazekas, and M. Sandler, "Towards playlist generation algorithms using rnns trained on within-track transitions," *arXiv preprint arXiv:1606.02096*, 2016.

[9] B. McFee and G. R. Lanckriet, "The natural language of playlists." in *Proceedings of 12th International Society for Music Information Retrieval (ISMIR) Conference*, vol. 11. ISMIR, 2011, pp. 537–542.

[10] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims, "Playlist prediction via metric embedding." in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, pp. 714–722.

[11] J. L. Moore, S. Chen, T. Joachims, and D. Turnbull, "Learning to embed songs and tags for playlist prediction." in *Proceedings of the 13th International Society for Music Information Retrieval (ISMIR)*, vol. 12. ISMIR, 2012, pp. 349–354.

[12] A. Vall, M. Quadrana, M. Schedl, and G. Widmer, "The importance of song context and song order in automated music playlist generation." *CoRR*, vol. abs/1807.04690, 2018.

[13] A. Vall, H. Eghbal-zadeh, M. Dorfer, M. Schedl, and G. Widmer, "Music playlist continuation by learning from hand-curated examples and song features: Alleviating the cold-start problem for rare and out-of-set songs." in *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*. ACM, 2017, pp. 46–54.

[14] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks." in *Proceedings of the 18th International Society of Music Information Retrieval (ISMIR) Conference*. ISMIR, 2017.

[15] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: The difficulty of learning longterm dependencies," 2001.

[16] A. Guttman, *R-trees: a dynamic index structure for spatial searching.* ACM, 1984, vol. 14, no. 2.

[17] B. McFee and G. R. Lanckriet, "Hypergraph models of playlist dialects." in *Proceedings of the 13th International Society for Music Information Retrieval (ISMIR) Conference*, vol. 12. ISMIR, 2012, pp. 343–348.

[18] K. Choi, D. Joo, and J. Kim, "Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras," *CoRR*, vol. abs/1706.05781, 2017.

[19] A. Gulli and S. Pal, *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[21] J. K. Roberts, J. P. Monaco, H. Stovall, and V. Foster, "Explained variance in multilevel models." *Handbook of advanced multilevel analysis*, pp. 219–230, 2011.

[22] J. A. F. Fernandes, "Automatic playlist generation via music mood analysis," Ph.D. dissertation, University of Coimbra, Faculty of Sciences and Technology, Department of Informatics Engineering, 2010.

[23] A. Huq, J. P. Bello, and R. Rowe, "Automated music emotion recognition: A systematic evaluation." *Journal of New Music Research*, vol. 39, no. 3, pp. 227–244, 2010.

[24] L. B. Sheiner and S. L. Beal, "Some suggestions for measuring predictive performance." *Journal of pharmacokinetics and biopharmaceutics*, vol. 9, no. 4, pp. 503–512, 1981.