AUDIO-BASED DETECTION OF EXPLICIT CONTENT IN MUSIC

Andrea Vaglio[†⋆] *Romain Hennequin*[†] *Manuel Moussallam*[†]
Gaël Richard[⋆] *Florence d'Alché-Buc*[⋆]

[†]Deezer Research & Development
[⋆] LTCI, Télécom Paris, Institut Polytechnique de Paris
research@deezer.com

ABSTRACT

We present a novel automatic system for performing explicit content detection directly on the audio signal. Our modular approach uses an audio-to-character recognition model, a keyword spotting model associated with a dictionary of carefully chosen keywords, and a Random Forest classification model for the final decision. To the best of our knowledge, this is the first explicit content detection system based on audio only. We demonstrate the individual relevance of our modules on a set of sub-tasks and compare our approach to a lyrics-informed oracle and an end-to-end naive architecture. The results obtained are encouraging with a F1-score of 67% on a industrial scale explicit content dataset.

Index Terms— Explicit content detection, keyword spotting, lyrics transcription, CTC training, music information retrieval

1. INTRODUCTION

For over three decades [1], a parental advisory label has been found on musical recordings when they include explicit content (*e.g.* lyrics potentially unsuitable for children). As of today, this labelling is mainly done manually following guidelines [2]. This process is slow and hard to scale to industrial-size catalogs. Existing automatic approaches are scarce and rely on the availability of the lyrics in text format.

Lyrics transcriptions could be obtained from audio using singing voice recognition algorithms. However, although *Automatic Speech Recognition* (ASR) methods have recently shown impressive progress [3], singing voice transcription still raises challenging issues [4]. First, in comparison to speech, singing voice characteristics are often more varied. The pitch, the pronunciation and the vowels duration can fluctuate greatly. Second, the accompaniment can be considered as a highly correlated noise with a level comparable to the signal of interest. A pre-processing step of voice separation is generally used to improve results [5], although still not on par with those obtained on a capella singing [6]. Third, until recently [7], no open dataset was available to train statistical models at scale.

When lyrics are available, explicit content detection can be approximately achieved through assessing the presence of words from a fairly small specialized dictionary. In fact, as proven in [8], state-of-the-art deep neural network algorithms perform just slightly better than dictionary-based methods with suitable keywords. This suggests that detecting a set of carefully chosen keywords directly on the audio signal is a good proxy to perform the explicit detection task in the general case. Keyword Spotting in audio is an actively studied task, achieving high performances on speech signals [9]. A few attempts to transfer them to singing voice have been proposed [10, 11]. One existing approach relies on a keyword-filler *Hidden Markov Model* (HMM) algorithm [11]. This method presents several limitations. First, expert knowledge is required for the tedious task of creating a pronunciation dictionary. Second, model training requires synchronized annotations, at the phoneme level, between audio and text. Since no readily accessible dataset exists for polyphonic music, such annotations are generated with an acoustic model trained on speech, by aligning textual lyrics to music, leading to sub-optimal model performance.

In this paper, we address the task of explicit musical content detection from audio only. Namely, given a piece of music, we aim at classifying an audio recording as either *explicit* or *non-explicit*. Our proposed work is, to the best of our knowledge, the first audio-based detection system of explicit content in music. Our approach is based on an *Audio-To-Character* (A2C) recognition model recently proposed for singing voice transcription [12] and a keyword spotting model associated with a dictionary of carefully chosen keywords in relation to the explicit detection task. In [13], authors have demonstrated the better performance of this architecture over the keyword filler. To the best of our knowledge, it is the first time that such an architecture is used for *Keyword Spotting* (KWS) on singing voice. The key advantages of this method are that it requires no expert knowledge and is usable with unsynchronized annotations. The decoding is directly performed on the output of the A2C and, contrary to end-to-end KWS approach like in [14], new keywords can be added easily to the dictionary without retraining the model. Finally,

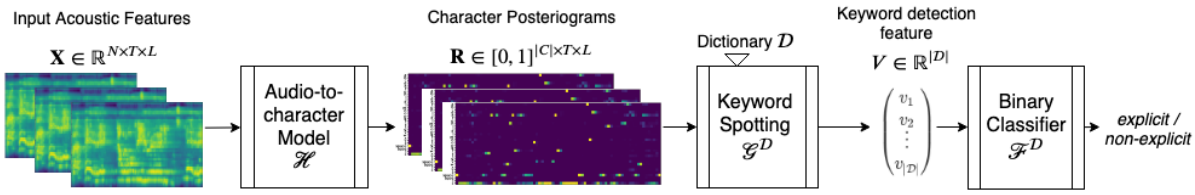

Fig. 1. General overview of the proposed modular explicit content detection system

the *explicit* label is inferred by a binary classifier using the output of the keyword spotting system. In this study, we restrict ourselves to recordings with English lyrics.

2. PROPOSED METHOD

A general overview of our system is given in Figure 1. A monophonic song S is sliced into L segments of equal size. It is worth noting that L may be different for each song since it depends on the song duration. The system takes as input an acoustic feature tensor $\mathbf{X} \in \mathbb{R}^{N \times T \times L}$ built from S, with T the number of temporal frames per segment and N the features dimension. Given a dictionary $\mathcal{D}$, the predictive model $\mathscr{L}^{\mathcal{D}}$ is the composition of three modules:

$$\mathscr{L}^{\mathcal{D}}(\mathbf{X}) = \mathscr{F} \circ \mathscr{G}^{\mathcal{D}} \circ \mathscr{H}(\mathbf{X}) \tag{1}$$

For a given input tensor $\mathbf{X}$, the audio-to-character module $\mathscr{H}$ outputs a 3D-tensor $\mathbf{R}$. Given X_ℓ, the matrix of acoustic features extracted from the ℓ^{th} segment, each coefficient $r_{i,j,\ell}$ provides an estimation of the posterior probability of c_i, the i^{th} character being uttered at t_j, the j^{th} frame. :

$$r_{i,j,\ell} = h_{i,j}(X_\ell) = \hat{P}(c_i, t_j \mid X_\ell), \tag{2}$$

for $1 \leq i \leq |C|$, $1 \leq j \leq T$, $1 \leq \ell \leq L$, C being the set of characters outputted by the model. It consists of 26 lowercase letters of the Latin alphabet, plus the word-boundary "space" token, the instrumental token "I", the apostrophe and the CTC blank token ϵ introduced in section 2.2. Note that $\mathscr{H}$ relies on the design of $h : \mathbb{R}^{N \times T} \rightarrow [0,1]^{|C| \times T}$, that takes a segment represented by an acoustic feature matrix and predicts a posteriogram on characters.

For a given input tensor $\mathbf{R} = \mathscr{H}(\mathbf{X})$, a vector V is outputted by the keyword spotting module $\mathscr{G}^{\mathcal{D}}$, whose each coefficient v_n gives an estimate of the (log) posterior probability of k_n, the n^{th} keyword of the dictionary $\mathcal{D}$, by averaging on all segments:

$$v_n = \underset{\ell=1,\ldots,L}{\text{mean}} \log \hat{P}(k_n \mid R_\ell), \tag{3}$$

where R_ℓ is the ℓ^{th} matrix in tensor $\mathbf{R}$.

For a given input vector $V = \mathscr{G}^{\mathcal{D}} \circ \mathscr{H}(\mathbf{X})$, an *explicit* label $\mathscr{L}^{\mathcal{D}}(\mathbf{X})$ is outputted by the binary classifier $\mathscr{F}$ discriminating the content of the song S as *explicit* or *not-explicit*. Among these three modules, only modules $\mathscr{H}$ and $\mathscr{F}$ require to be learned. Learning $\mathscr{H}$ boils down to learning model h which can be done using a dataset of annotated segments $\{(X^i, u^i)_{i=1}^{n_{seg}}\}$ where X^i is a matrix of acoustic features describing a segment and u^i is the corresponding sequence of characters. Learning $\mathscr{F}$ requires to apply the pre-processing $\mathscr{G}^{\mathcal{D}} \circ \mathscr{H}$ to the training dataset $\{(\mathbf{X}^i, y^i)_{i=1}^{n_{songs}}\}$ containing songs annotated by *explicit/non explicit* labels.

2.1. Audio-to-character recognition

The model h on which A2C module $\mathscr{H}$ relies is implemented with a *Convolutional Recurrent Neural Network* (CRNN) trained with a *Connectionist Temporal Classification* (CTC) algorithm. The *Recurrent Neural Network* (RNN) CTC has been successfully applied to ASR [3]. To reduce the dimension of features and accelerate training, we use additional convolutional layers. For RNN layers, we choose bidirectional *Long Short-Term Memory* (LSTM), so that outputs at each frame depend of the entire input sequence [15].

The CTC algorithm [16] allows to train RNN models without aligned annotations. To do that, a "blank" symbol (noted ϵ) is introduced to represent a non-emission token. Any character, including ϵ, can be emitted at each frame by the model. The total probability of the output character sequence is maximized using the CTC algorithm by marginalizing over all possible alignments for a given input. The objective function being differentiable, the network is trained with back-propagation through time. More details for CTC algorithm are given in [16].

2.2. Keyword spotting

Following the work of [13], we implement $\mathscr{G}^{\mathcal{D}}$ as a CTC-based decoding function. For a given searched keyword k, we consider k' which is the keyword k with an ϵ at the beginning, end, and between every character to allow the use of ϵ during decoding. A decoding network of size $|k'| \times T$ is constructed from k'. The goal of the decoding function is to find the path in the network that maximize the CTC scoring for the keyword k'. To do that, we define network's node $\alpha_{s,j}$ as the CTC score of the sub-sequence $k'_{1:s}$ after j frame. A forward-backward algorithm can be used to compute efficiently $\alpha_{s,j}$ scores, by merging together paths that reach the same node. $\alpha_{s,j}$ is then computed recursively from α's of the previous

frame. Only transitions between blank and non-blank characters, and between pair of distinct non-blank characters are allowed. As ϵ at the beginning and end of the sequence is optional, there are two valid starting nodes and two final nodes. The coefficients α's are initialized as follows:

$$\alpha_{s,0} = P(k'_s, t_0 \mid X_\ell) \text{ for } s \in \{0, 1\} \text{ and } \alpha_{s,0} = 0, \forall s > 1$$

Recursion is given by:

$$\begin{aligned} \alpha_{s,j} &= (\sum_{\tau=0}^{\tau=1} \alpha_{s-\tau,j-1}) P(k'_s, t_j \mid X_\ell), \text{if } k'_s \in \{\epsilon, k'_{s-2}\} \\ \alpha_{s,j} &= (\sum_{\tau=0}^{\tau=2} \alpha_{s-\tau,j-1}) P(k'_s, t_j \mid X_\ell), \qquad \text{otherwise} \end{aligned} \tag{4}$$

Finally, keyword probability is given at each step j by:

$$P(k, t_j) = \alpha_{|k'|-1,j} + \alpha_{|k'|-2,j} \tag{5}$$

We consider the detection score s of the keyword k to be the maximum of the keyword probability over all time step:

$$s(k) = \max_{j=1,\dots,T} P(k, t_j) \tag{6}$$

We found empirically that with these computation rules, we can only find keywords at the beginning of segments. In practice, keyword probabilities after the first sung word in the recording are artificially low. To prevent this and allow the keyword detector to be fired at any time, we choose to reinitialize the first node at each time step:

$$\alpha_{0,j} = P(k'_0, t_j \mid X_\ell) \tag{7}$$

As naive CTC scoring is numerically unstable, computations are done in log-space using the log-sum-exp trick [17].

3. EXPERIMENTS

3.1. Recognition model

The main component h of the A2C model $\mathscr{H}$ is trained with the DALI dataset introduced in [7]. DALI contains 5358 audio tracks with time-aligned lyrics at paragraph, line and word levels. It is composed of varied western genres (*e.g.* rock, rap and electronic). Tracks are downsampled to 16 kHz and converted to mono. Vocals from each song are then extracted using Spleeter [18]. For each song, training samples are generated by segmenting the track using a window of 5 seconds with a hop size of 2.5 seconds. The character sequence associated with a segment is created by concatenating all words whose start position are within the segment. In case no words start within a segment, we generate a token "I" (for "Instrumental"). Character sequences are transformed to fit the set of characters C outputted by the model: each character sequence is converted to lower-case and non-valid characters are discarded. Finally, we make an artist aware split [19] between train, validation and test dataset of 70%-1%-14%. We respectively obtain datasets of 384, 5 and 63 hours of music.

The model h is composed of 2 convolutional layers, followed by 3 layers of bidirectional LSTM and a dense layer. For each input sample, values of mel-scale log filterbanks coefficient and energy plus deltas and double-deltas are extracted. A Hann window of 32 ms with a step size of 16 ms is used. For the convolutional part, the size of filters and max-pooling are respectively 3×3 and 2×3. The number of features map of each layer is 32. Each recurrent layer has a dimension equal to 256. A dropout of 0.1 is applied between recurrent layers. The last layer is a single affine transformation followed by a softmax function which outputs the probabilities of characters from vocabulary C. The model is trained using the CTC loss implementation of [20]. The loss is minimized using ADAM with a learning rate of 10^{-4}, a batch size of 32, 4000 training epochs with 250 steps per epoch. We use validation-based early stopping. For transcription, classic beam search decoding [15] is used, using a beam width of 100.

Finally, we obtain a *Character Error Rate* (CER) of 47.41% on the test dataset. This is on par with results reported by [12] on a different dataset. An example of posteriogram R_ℓ inferred using the trained network is pictured in Figure 2. R_ℓ consists of a sequence of spikes, associated with detected characters, separated by ϵ character.

3.2. Keyword spotting

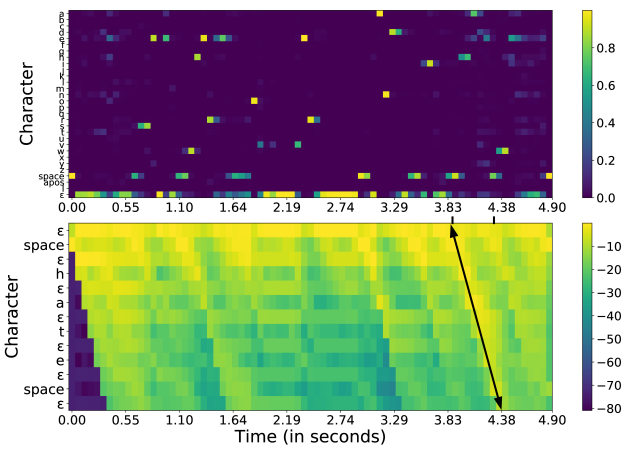

Fig. 2. A positive sample for keyword "hate". Top: Posteriogram R_ℓ inferred by acoustic model $\mathscr{H}$. Bottom: Decoding matrix composed of coefficients α. Ground truth: "to see we're over and i hate when". Transcription with beam search: "e se where over and i hae we". Decoding line and ground truth position of keyword are displayed in the figure

Following the work of [21] we select the keywords of $\mathcal{D}$

Metrics	Audio baseline	Our system	Lyrics baseline
Precision	.61 (.02)	.63 (.02)	.65 (.02)
Recall	.59 (.02)	.71 (.02)	.84 (.02)
F1-score	.60 (.02)	.67 (.02)	.73 (.02)

Table 1. Results for explicit detection task on the test set (standard deviation in parenthesis)

based on the explicit and not-explicit lyrics word distributions. To generate $\mathcal{D}$, we use the importance I defined in [8]. For a chosen keyword, I is computed as the ratio between frequency of the word in explicit and non-explicit lyrics. As in [8], we manually discard stop words, too common words, too rare words, onomatopoeia and abbreviations. The dictionary is constructed with 128 words with the highest importance.

Performance of decoding with dictionary $\mathcal{D}$ are assessed on DALI test set. 75% of $\mathcal{D}$ keywords, with at least one occurrence in the test dataset, have a *Area Under Curve* (AUC) of *Receiver Operating Characteristic* (ROC) curve greater than 0.81. Being the first time such metrics are computed for keyword spotting in the singing case, we cannot compare it to other results. Since these values are significantly higher than random, the feature vector V carry some information on the presence of keywords in $\mathcal{D}$. An example of decoding is displayed in Figure 2. The example is "positive", as the searched keyword is indeed present in the ground truth character sequence. A decoding line is visible in the figure. Position of "space" character delimiting the decoding line (3.94s to 4.28s) are quite close to the ground truth position of the word (3.9s to 4.29s). This result suggests that the acoustic model $\mathscr{H}$ correctly uses the "space" character and is able to find words position. This is consistent with results found for lyrics-to-audio alignment [12].

3.3. Explicit lyrics detection

To train $\mathscr{F}$, we use a private dataset. Songs are either labelled *explicit* or *non-explicit*. We discard tracks also present in DALI to avoid overfitting. We notice that the music genre distribution of explicit tracks and non-explicit tracks, is very different: rap is strongly over-represented in explicit tracks (40% of all tracks), but not in the non-explicit ones (few percents). To avoid creating an explicit content detection that rely mostly on the genre information, we sample both explicit and non-explicit tracks to obtain same genre distribution for the two types of songs. The complete dataset then consists of 2600 non-explicit tracks and 2530 explicit ones. Finally, we make an artist aware split [19] between training, validation, test of respectively 70%, 15%, 15%. We create another dataset the same way for dictionary creation. The dataset consists of 24250 non-explicit tracks and 24250 explicit ones. No songs are common between the two datasets.

Our model is compared to two baseline systems. The first one is a classic CRNN audio classifier. This architecture was successfully used in a variety of music classification tasks, such as genre recognition [22] or music emotion recognition [23]. Unlike our system, this classifier tries to directly infer *explicit* labels from audio in an end-to-end manner. The model is composed of 4 convolutional layers, followed by 1 gated recurrent units layer and a dense layer. For each input sample, values of mel-scale log filterbanks coefficient are extracted using a Hann Window of 48 ms with a step size of 48 ms. The model is trained using a binary cross-entropy loss which is optimized using an Adadelta optimizer and a batch size of 1. The model is trained for 3000 epochs with 450 steps per epoch. We use validation-based early stopping. The second baseline is a dictionary lookup based on lyrics as in [8]. Given the dictionary $\mathcal{D}$, this method classifies a song as *explicit* if its lyrics contain at least one of the keywords in $\mathcal{D}$ and as *non-explicit* otherwise. Unlike our system, this baseline is informed by lyrics at test time. As such, this baseline can be considered as an oracle (*e.g.* providing an upper bound for performance) for our task of detecting explicit content from audio only.

For $\mathscr{F}$, we use a Random Forest classifier [24]. Hyperparameters of the classifier are tuned using a first step of random search and a second step of grid search. Number of keywords of the dictionary, for our model and for the lyrics baseline, are tuned on the validation dataset. We report precision, recall and F1-score for the explicit class. Since explicit content might be sensitive to certain audiences, emphasis is put, at highest F1-score, on the system maximizing the recall. We use this rule to choose our "best" parameters on the validation dataset. The "best" number of keywords is 128 for the lyrics baseline and 32 for our model. Baselines and our system with their "best" parameters are evaluated on the test dataset. Results are reported in Table 1. Scores reached by the lyrics baseline are similar to those found in [8]. Performance of a naive audio baseline on this challenging task is significantly outperformed by our modular approach. While yet not equivalent to a lyrics-informed scenario, these results are encouraging and show the validity of the proposed method. Performances of these systems are still insufficient to be deployed without human oversight. In [8], authors argue that explicit detection is an inherently hard task. They propose using these systems as tools to help annotators making the final labelling.

4. CONCLUSION

We address the novel task of explicit musical content detection from audio only. Despite the task being challenging, our proposed modular approach yield promising results. Moreover, the system's decisions can be explained in terms of specific keyword presence probability which is a desirable property given the sensitivity of the task. Future works will investigate keyword decoding augmentation with a character level language model as in [25].

5. REFERENCES

[1] National Public Radio, "'parental advisory' labels — the criteria and the history," https://www.npr.org/sections/therecord/2010/10/29/130905176/you-ask-we-answer-parental-advisory---why-when-how, 2019, [Online; accessed October 18, 2019].

[2] British Phonographic Industry, "Parental advisory guidelines," https://www.bpi.co.uk/media/1047/parental-advisory-guidelines.pdf, 2019, [Online; accessed October 18, 2019].

[3] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International Conference on Machine Learning (ICML)*, 2016, pp. 173–182.

[4] Anna Mesaros, *Singing Voice Recognition for Music Information Retrieval*, Ph.D. thesis, Tampere university of technology, 2012.

[5] Annamaria Mesaros, Tuomas Virtanen, and Anssi Klapuri, "Singer identification in polyphonic music using vocal separation and pattern recognition methods," in *In Proc. IEEE International Society for Music Information Retrieval Conference (ISMIR)*, 2007, pp. 375–378.

[6] Annamaria Mesaros and Tuomas Virtanen, "Automatic recognition of lyrics in singing," *EURASIP Journal on Audio, Speech, and Music Processing*, 2010.

[7] Gabriel Meseguer-brocal, Alice Cohen-hadria, and Geoffroy Peeters, "Dali : a Large Dataset of Synchronized Audio , Lyrics and Notes , Automatically Created Using Teacher-Student Machine Learning Paradigm," in *IEEE International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[8] Michael Fell, Elena Cabrio, Michele Corazza, and Fabien Gandon, "Comparing Automated Methods to Detect Explicit Content in Song Lyrics," in *Recent Advances in Natural Language Processing (RANLP)*, Varna, Bulgaria, Sept. 2019.

[9] Raphael Tang and Jimmy Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5484–5488.

[10] Hiromasa Fujihara, Masataka Goto, and Jun Ogata, "Hyperlinking lyrics : A method for creating hyperlinks between phrases in song lyrics," *Proceedings of the International Conference on Music Information Retrieval*, pp. 281–286, 2008.

[11] Anna Marie Kruspe, *Application of Automatic Speech Recognition Technologies to Singing Doctoral Thesis*, Ph.D. thesis, University Fraunhofer, apr 2018.

[12] Daniel Stoller, Simon Durand, and Sebastian Ewert, "End-to-end Lyrics Alignment for Polyphonic Music Using an Audio-to-Character Recognition Model," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[13] Kyuyeon Hwang, Minjae Lee, and Wonyong Sung, "Online Keyword Spotting with a Character-Level Recurrent Neural Network," *Arxiv*, pp. 1–10, 2015.

[14] Guoguo Chen, Carolina Parada, and Georg Heigold, "Small-footprint keyword spotting using deep neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. may 2014, pp. 4087–4091, IEEE.

[15] Alex Graves and Navdeep Jaitly, "Towards End-To-End Speech Recognition with Recurrent Neural Networks," *Internation Conference on Machine Learning (ICML)*, vol. 32, no. 1, pp. 1764–1772, 2014.

[16] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," *ACM/IEEE Supercomputing Conference (SC)*, vol. 148, pp. 369–376, 2006.

[17] Awni Hannun, "Sequence modeling with ctc," *Distill*, vol. 2, no. 11, pp. e8, 2017.

[18] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam, "Spleeter: A fast and state-of-the art music source separation tool with pre-trained models," Late-Breaking/Demo ISMIR 2019, November 2019, Deezer Research.

[19] Arthur Flexer, "A closer look on artist filters for musical genre classification," *World*, vol. 19, no. 122, pp. 16–17, 2007.

[20] Yann Soullard, Cyprien Ruffino, and Thierry Paquet, "CTC-Model: Connectionist Temporal Classification in Keras," 2018.

[21] Jayong Kim and Y Yi Mun, "A hybrid modeling approach for an automated lyrics-rating system for adolescents," in *European Conference on Information Retrieval (ECIR)*. Springer, 2019, pp. 779–786.

[22] 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)*. IEEE, 2017, pp. 2392–2396.

[23] Miroslav Malik, Sharath Adavanne, Konstantinos Drossos, Tuomas Virtanen, Dasa Ticha, and Roman Jarina, "Stacked convolutional and recurrent neural networks for music emotion recognition," *arXiv preprint arXiv:1706.02292*, 2017.

[24] Tin Kam Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*. IEEE, 1995, vol. 1, pp. 278–282.

[25] Yanzhang He, Rohit Prabhavalkar, Kanishka Rao, Wei Li, Anton Bakhtin, and Ian McGraw, "Streaming small-footprint keyword spotting using sequence-to-sequence models," in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 474–481.