Zafar Rafii, Erling Wold, and Richard Boulderstone

How to Design a Cheap Music Detection System Using a Simple Multilayer Perceptron With Temporal Integration

We show how to design a cheap system for detecting when music is present in audio recordings. We make use of a small neural network consisting of a simple multilayer perceptron (MLP) along with compact features derived from the mel spectrogram by means of temporal integration.

Scope

Temporal integration is the process of combining information over time, for example, by computing statistics over a sequence of spectra in audio [1], [2]. We experiment with common statistics and compare the performances of various small MLPs, with one another and with more complex models, on two large music detection datasets. We note that this article does not claim to propose a state-of-the-art music detection model but, rather, to demonstrate how anybody can design their own efficient system from scratch using simple ideas and with limited resources.

Relevance

An efficient music detection system can be the first step to other complex tasks, such as audio segmentation, music identification, or instrument classification, and can further lead to some important applications, such as automatic content recognition and digital rights management. In this era of tech giants with immense resources and massive models, we wish to promote the use of practical and low-cost ideas that could potentially encourage small labs and individual researchers to develop their own systems in an efficient and sustainable manner. The model presented in this article is efficient enough to be trained without requiring expensive GPUs, simple enough to be implemented from scratch at inference time without depending on external deep learning libraries, and small enough to be deployed on a lightweight device.

Gael Richard acted as the associate editor for this article and approved it for publication.

Digital Object Identifier 10.1109/MSP.2024.3424578

Prerequisites

A basic knowledge of audio analysis, statistics, and deep learning is required to understand this article, in particular, concepts such as the mel spectrogram, moments/percentiles, and the MLP. For more information regarding music detection and temporal integration, the reader is referred to [3] and [1] and [2], respectively.

Problem statement

You are a researcher in a small company tasked with designing a system for detecting when music is present in audio streams, for example, in a live streaming service or an online video platform. You need the system to be as cheap as possible, as it is meant to be integrated in a small tool kit, and you have limited access to GPUs. You also wish to be able to easily implement the system in the environment of your choice without depending on external libraries (e.g., PyTorch or TensorFlow), to save space and avoid potential compatibility issues.

While a number of pretrained models can be found online, including ones that can be fine-tuned [e.g., YAMNet (https://github.com/tensorflow/models/tree/master/research/audioset/yamnet)], they are typically too large (up to millions of parameters) to fit on lightweight devices. Various techniques have been proposed to compress large models into smaller ones, but they do not always guarantee leading to models that are sufficiently small and still accurate. Thanks to the constant development of deep learning libraries, anyone can now build, train, and deploy their own model. While using such libraries for training is pretty much unavoidable, they are not necessarily required at inference time, as one can directly implement the learned weights and calculations of the predictions themselves, especially if the model is simple. And the simplest neural network to implement from scratch is arguably the MLP, which is basically a series of matrix multiplications with nonlinear functions in between.

Audio models typically build on some 2D time-frequency representation, such as the mel spectrogram. While it may be possible to design a system that takes 2D features as inputs without being too heavy, for example, using a convolutional neural network (CNN), we instead propose to

 1053-5888/24©2024IEEE

make use of a simpler MLP. Feeding an MLP directly with 1D spectrogram frames would, however, be inefficient; a better idea would be to combine a series of frames into a more compact 1D feature, which could then be used with an MLP. As it happens, temporal integration is a well-known technique for combining a sequence of features into a single one, for example, by computing statistics over time [1], [2].

Prior work

Temporal integration

Temporal integration is the process of combining a sequence of short-time feature vectors into a single feature vector [1], [2]. It has been widely used, especially in earlier works, to design simple but robust features for various audio tasks, such as music detection [4], [5], music genre classification [1], or musical instrument classification [2]. In addition to producing something more compact, the resulting features typically capture some relevant temporal properties leading to better classification performances [1], [2]. Temporal integration was oftentimes performed using basic statistics, such as the mean and the variance, but also skewness and kurtosis [2], [4] or filter banks [1]. The integration was generally computed over a spectral feature, such as the mel spectrogram or the mel frequency cepstral coefficients (MFCCs), but also over the zero-crossing rate (ZCR) [2], [4], [5] or the spectral centroid [2], [5]. The resulting features were then fed into various classical machine learning models, such as Gaussian mixture models [1], [2], [5] or support vector machines [2].

Music detection

Music detection is the task of finding whether music is present in an audio signal, be it isolated or overlapped. Early works experimented with a number of handcrafted features and classical machine learning models, including in a real-time setting. Features included the ZCR, spectrum-based features, MFCCs, and various combinations of those, oftentimes followed by temporal integration [4], [5]. More recent works focused on data-driven approaches, typically using CNNs, recurrent neural networks (RNNs), or hybrid models, such as convolutional RNNs (CRNNs) [3], [6]. With enough data, such models would learn to generalize better, but they would rarely be cheap enough to fit on lightweight devices.

Similar works

Few recent works have mentioned the potential of using temporal integration with neural networks for cheap music detection, albeit experimenting with small and private datasets in a limited reproducible manner. Khan and Al-Khatib used feature selection on various features after temporal integration and tested several models for music and speech classification. They showed that a combination of several integrated features with an MLP, including the ZCR, spectral flux, and MFCCs, lead to the best results when tested on their own small dataset [7]. Seyerlehner et al. tested several features, with and without temporal integration, and models, with and without smoothing the predictions, for music detection in TV broadcasts [8]. While temporal integration did not seem to help in their case, smoothing the predictions did, especially when using an MLP. Agüera y Arcas et al. presented a low-power music recognition system that included a music detector using a mel spectrogram and a small CNN [9]. Their tests showed that their detector could get a decent true positive rate (TPR) with a low false positive (FP) rate. Vrysis et al. compared different combinations of features, temporal integration, and models, including the first four moments, the ZCR, and MFCCs, and an MLP and CNNs, for a music, speech, and other classification task [10]. They tested their models on their own small dataset and showed that relatively small models could achieve good results, especially when combining multiple features and temporal integration or even directly using a small CNN.

Solution

System

We wish to design a music detection system that is cheap to train, simple to implement, small to store, fast to compute, and sufficiently accurate. With this goal in mind and inspired by previous works, we propose to use a very simple MLP that consists of a single hidden layer of 256 units, a rectified linear unit (ReLU) activation, and a dropout with a rate of 0.5, followed by an output layer with a single unit and a sigmoid activation. After learning the weights, the model can be easily implemented from scratch as two matrix multiplications with a ramp function in between and a sigmoid function at the end.

We then propose to compute the first four moments, i.e. the mean, variance, skewness, and kurtosis, over nonoverlapping 10-s segments of the mel spectrogram, after taking its $\log(x + 1)$; Figure 1 illustrates such temporal integration. We pick the first four moments, as they are standard statistics and can summarize a distribution well; they are also very simple to implement and very cheap to compute. We can also use percentiles, such as the 50th (median), 25th, 75th, and so on, which are slightly more expensive but more robust. We chose the mel spectrogram, as it is a time-frequency representation widely used in numerous audio tasks; we take its $\log(x + 1)$ to map the values into a more manageable positive decibel range. We decided to use nonoverlapping 10-s segments to keep the segmentation straightforward and because 10 s is short enough for an acceptable detection rate and long enough to contain sufficient temporal information. We note that, regardless of the segment duration, the integrated feature will have the same size and should be statistically consistent as long as the duration is long enough to include sufficient information and short enough not to average the information too much.

Experiments

We propose to test our system on two large and publicly available music detection datasets. MUSAN (https://

www.openslr.org/17/) consists of 109 h of audio downloaded from various sources and separated into music, noise, and speech subsets [11]; it is the largest dataset listed for training in the latest music detection task of the Music Information Retrieval Evaluation Exchange (https://www.music-ir.org/mirex/wiki/2021:Music_Detection). TVSM (https://zenodo.org/record/7025971) is a more recent and larger dataset, which consists of precomputed features from 1,608.1 h of audio sampled from various TV broadcasts and separated into three subsets, one small, with human annotations of music and speech presence (suggested for testing), and two large, with approximate annotations derived from TV metadata and a pretrained model (suggested for training) [3].

For each audio file in MUSAN, we compute the $\log(x + 1)$ mel spectrogram using Librosa (https://librosa.org) with the default parameters, i.e., a sampling frequency of 22.05 kHz, a hop length of 512 samples, and 128 mel bands. TVSM already comes with precomputed mel spectrograms, with the same parameters but at a sampling frequency of 16 kHz. We then compute the first four moments over nonoverlapping 10-s segments of the log mel spectrogram for both datasets and also five different percentiles, namely, the fifth, 25th, 50th, 75th, and 95th.

We randomly split all the files in MUSAN into 80% for training and 20% for testing, while we use the two automatically annotated subsets of TVSM for training and the human-annotated subset for testing. We build our simple MLP in Keras (https://keras.io/) using different combinations of statistics, namely, one (mean), two (mean and variance), three (mean, variance, and skewness), and four moments (mean, variance, skewness, and kurtosis), and one (50th), three (25th, 50th, and 75th), and five percentiles (fifth, 25th, 50th, 75th, and 95th). We compile the different models using an Adam optimizer with a learning rate of 1e-4 and a binary cross-entropy loss and train them for 100 epochs. We note that, given the simplicity of the models, we are able to train them without using any GPUs. We train them on a Linux server using two Intel Xeon Gold 5122 CPUs at 3.6 GHz, with four cores per socket and two threads per core. The training takes around 3 s per epoch on MUSAN and around 165 s per epoch on TVSM for each model.

To compare with models which could directly use 2D features, we also build two simple CNNs, one for the same 10-s segments of the log mel spectrogram and one for 10-s segments of MFCCs (both without temporal integration). We compute the MFCCs for MUSAN using Librosa with the default parameters, i.e., 20 coefficients at 22.05 kHz, while the MFCCs are also precomputed for TVSM, with the same parameters but at a sampling frequency of 16 kHz. We build our CNNs using three convolutional layers and two fully connected layers. The convolutional layers have 32, 64, and 64 filters with kernel sizes of 3×3, each followed by a ReLU activation, a dropout with a rate of 0.5, and a maximum pooling of size 2×2 (we omit the last maximum pooling for the MFCC model because of the shorter height of the feature). The first fully connected layer has 64 units, with a ReLU activation, and a dropout with a rate of 0.5, while the last one (the output layer) has a single unit with a sigmoid activation. The CNNs are

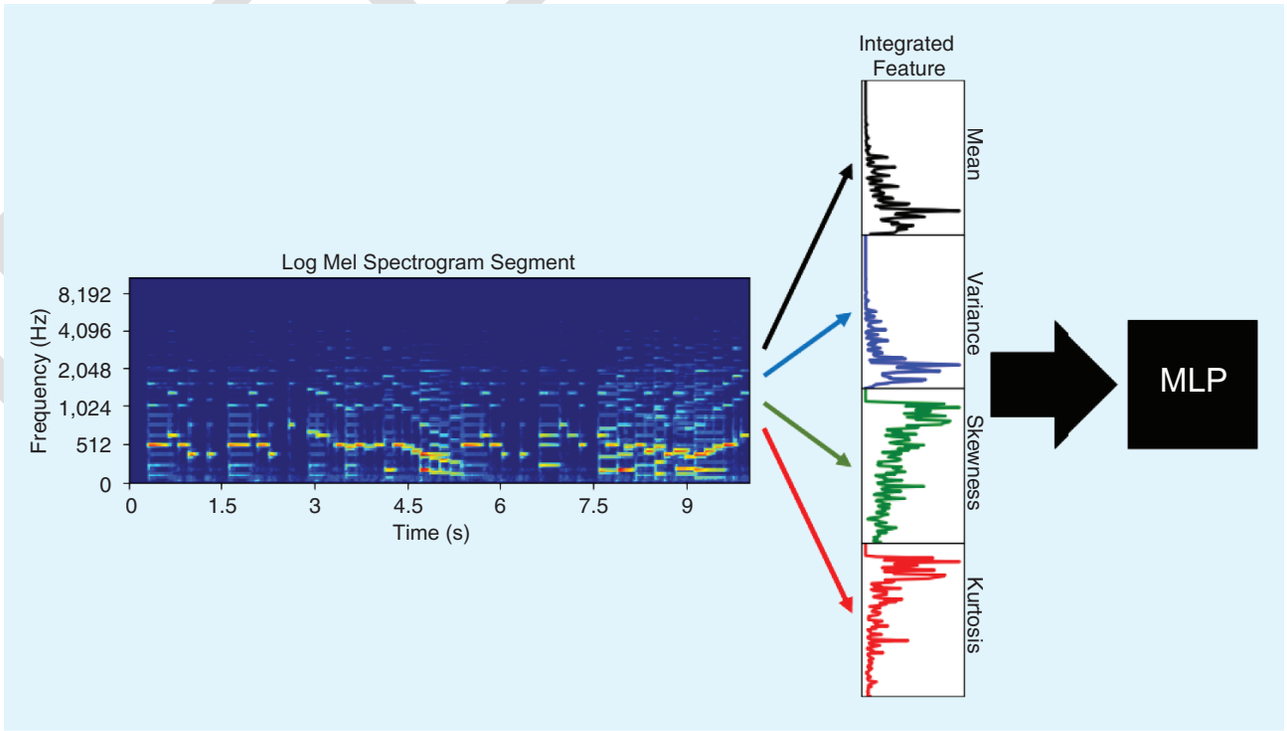

FIGURE 1. The temporal integration over a 10-s segment of a log mel spectrogram by computing the first four moments (mean, variance, skewness, and kurtosis) and concatenating them into a 1D feature that can then be fed into an MLP.

compiled and trained in Keras similarly to the MLPs. We note that, in this case, we use GPUs to train the models. We train them on a Linux server using one Tesla T4 with 16 GB, two Nvidia TITAN RTX graphics cards with 24 GB, and three Nvidia GeForce RTX 2080 Ti graphics cards with 11 GB. The training takes around 20 and 25 s per epoch on MUSAN and around 185 and 280 s per epoch on TVSM for the MFCC and log mel spectrogram models, respectively.

To have an external baseline to compare our models to, we also run three publicly available and larger third-party models. The InaSpeechSegmenter (Ina) (https://github.com/ina-foss/inaSpeechSegmenter) is a CNN-based system that can segment audio into music, speech, and noise [12]; train-synth-audio-seg (TSAS) (https://github.com/satvik-venkatesh/train-synth-audio-seg/) is a more recent system based on a CRNN, which was trained on synthesized radio broadcast audio [6]. YAMNet is a popular classification system based on the MobileNet v1 architecture, which can predict 521 audio event classes from the AudioSet-YouTube corpus (https://research.google.com/audioset/), including music.

Results

Figure 2 displays the receiver operating characteristic (ROC) curves for MLPs using one, two, three, and four moments (logmel_1*m*, 2*m*, 3*m*, 4*m*) and one, three, and five percentiles (logmel_1*p*, 3*p*, 5*p*) over the log mel spectrogram. As we can see, we can achieve good results using simple statistics and cheap MLPs, although the models struggled more with TVSM, probably because it contains mostly overlapped music and the annotations for the training set are approximate. For MUSAN, the best models achieved almost 99% of TPs for 1% of FPs, while for TVSM, they achieved almost 90% of TPs for 10% of FPs, which is still reasonable for such a simple and cheap system. As expected, using a single statistic did not lead to the best results, but surprisingly, using four moments also led to poorer results; since the kurtosis is not a robust measure, it could have been influenced by outliers in the data, which somehow hurt the temporal integration here. On the other hand, percentiles are robust statistics; they were shown to do better than the moments, with five percentiles doing slightly better than three, at the cost of being slightly more expensive. We note that additional tests using different segment durations showed similar results as long as the duration was not too short or too long; for example, 5- or 10-s segments led to pretty much the same performances, while 1- or 30-s segments led to poorer performances (too little or too much information to integrate).

Figure 3 provides the ROC curves for MLPs using two moments and three percentiles over the log mel spectrogram (logmel_2*m* and logmel_3*p*; solid lines), CNNs using the log mel spectrogram and the MFCCs without temporal integration (logmel_cnn and mfcc_cnn; dashed lines), and three third-party models. Note that Ina and TSAS generate time annotations that we translated into equivalent 10-s binary labels (hence, their ROC curves have a single data point), while YAMNet generates predictions but for shorter segments of 1 s. Note also that, since TVSM did not include audio files, we were not able to evaluate third-party systems on it. As we can see, a CNN using MFCCs directly gave results close to cheap MLPs, while, as expected, a CNN using the log mel spectrogram directly was able to achieve better results but at the cost of being much more expensive. We can also see that, compared to the larger third-party models, the cheap

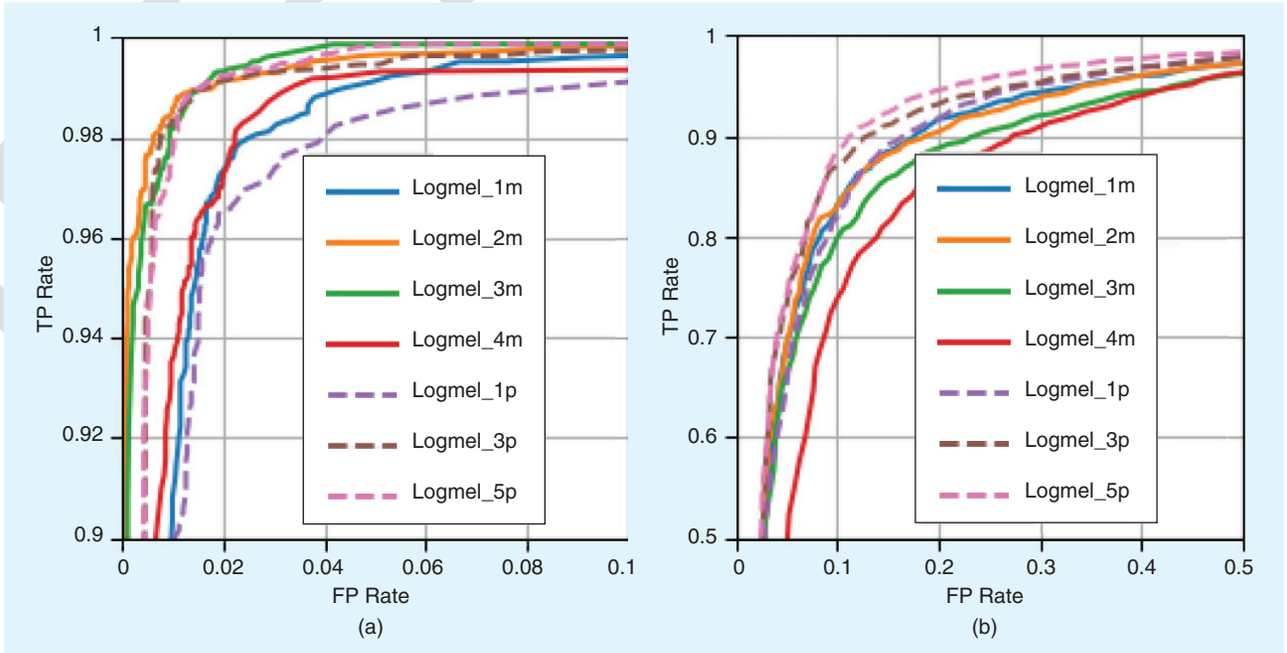

FIGURE 2. ROC curves (zoomed in) for MLPs using one, two, three, and four moments (solid lines) and one, three, and five percentiles (dashed lines) over the log mel spectrogram. (a) MUSAN. (b) TVSM.

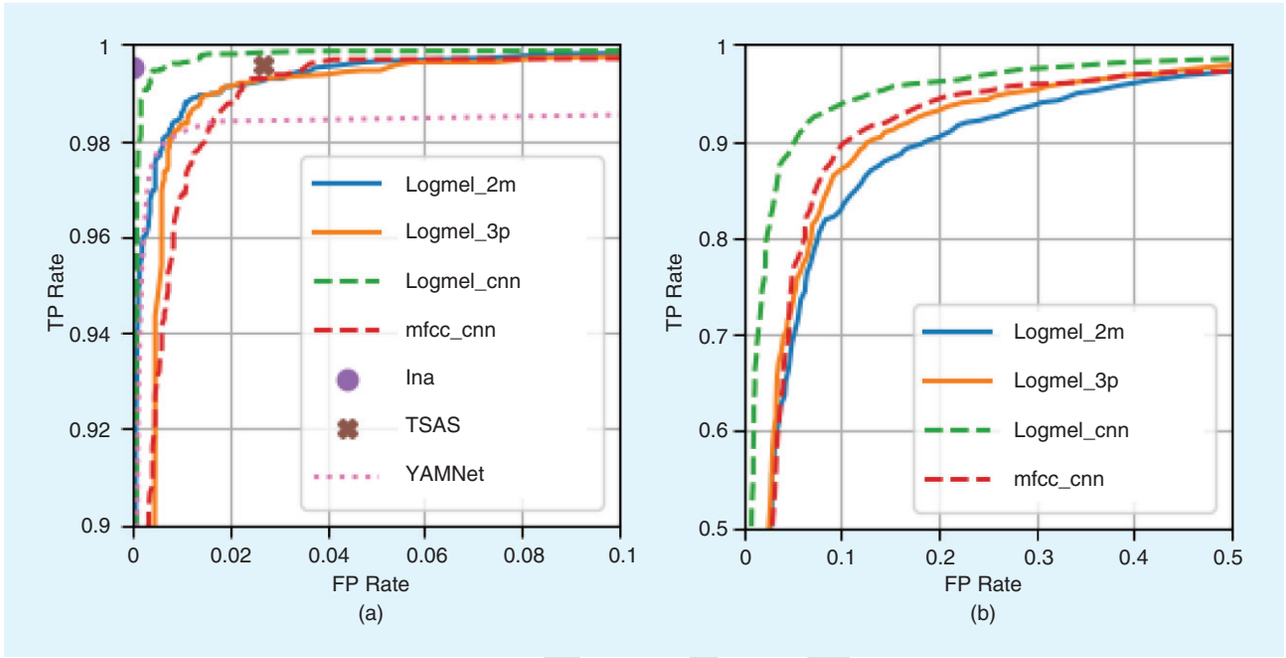

FIGURE 3. ROC curves for MLPs using two moments and three percentiles over the log mel spectrogram (solid lines), CNNs using the log mel spectrogram and the MFCCs directly (dashed lines), and three third-party models. (a) MUSAN. (b) TVSM.

Table 1. The F_1 score and area under curve for MLPs using two moments and three percentiles over the log mel spectrogram, CNNs using the log mel spectrogram and the MFCCs directly, and three third-party models.

	MUSAN		TVSM	
Model	**F_1↑**	**AUC↑**	**F_1↑**	**AUC↑**
logmel_2*m*	0.987	0.999	0.894	0.926
logmel_3*p*	0.986	0.997	0.91	0.939
logmel_cnn	0.995	0.999	0.939	0.967
mfcc_cnn	0.979	0.998	0.917	0.935
Ina	0.998	0.998	0.48	—
TSAS	0.984	0.985	0.88	—
YAMNet	0.973	0.989	—	—

AUC: area under curve.

Table 2. Sizes for MLPs using two moments and three percentiles over the log mel spectrogram, CNNs using the log mel spectrogram and the MFCCs directly, and three third-party models.

Model	Feature Size	Parameters	Model Size
logmel_2*m*	256	66,049	258 KB
logmel_3*p*	384	98,817	386 KB
logmel_cnn	128 × 431 = 55,168	3,037,761	11.59 MB
mfcc_cnn	20 × 431 = 8,620	481,857	1.84 MB
Ina	21 × 68 = 1,428	≈1.35 million	≈5.15 MB
TSAS	80 × 802 = 64,160	≈0.7 million	≈2.73 MB
YAMNet	64 × 96 = 6,144	≈3.7 million	≈14.45 MB

MLPs are able to get quite competitive results on MUSAN.

Table 1 lists the F_1 scores and the area under the curve for the same models shown in Figure 3. While we do not find the F_1 score very informative, as it gives equal importance to precision and recall, and recall can matter more in practice (here, we want a high TPR and do not mind few FPs), it was the measure reported for Ina and TSAS on TVSM in [3]. Note that, to compute the F_1 scores for our models (and YAMNet), we had to binarize the predictions, which we did using a threshold of 0.5. As we can see, for MUSAN, both measures are quite high, with little difference between the models. For TVSM, the measures are a bit lower but around the same for logmel_2*m*, logmel_3*p*, and mfcc_cnn and a little higher for logmel_cnn, confirming what we saw with the ROC curves. The relatively low F_1 scores for Ina and TSAS are probably because they were reported for shorter segments of 0.01 s, which can be more prone to FPs.

Table 2 provides the feature size, number of parameters, and model size for the same models in Figure 3 and Table 1. As we can see, compared to the CNNs and third-party models, which have large features and models, the MLPs use small features, have under 100,000 parameters, and need less than 1 MB. This means that they will take little space to deploy and can be trained without the need for GPUs, while the features can be easily stored and transferred around.

What we have learned

We showed that we can easily design a cheap but effective music detection system by using a simple MLP with few parameters and small features derived by means of temporal integration with basic statistics. The proposed system is efficient enough to be trained without GPUs, simple enough to be implemented from scratch without depending on external libraries, and small enough to be deployed on a lightweight device. For clarity, we focused on few temporal integration schemes and model

parameterizations, but we note that other designs and combinations are possible (e.g., different time-frequency representations, statistics, and/or segment durations), leaving room for improvement for deriving more efficient systems, especially if more resources are available.

Authors

Zafar Rafii (z_rafii@audiblemagic.com) received his Ph.D. degree in electrical engineering and computer science from Northwestern University. He is a research scientist at Audible Magic, Los Gatos, CA 95032 USA. His research interests include multiple aspects of audio analysis, with a focus on source separation and audio identification. He is an associate editor for *IEEE Transactions on Audio, Speech and Language Processing (TALSP)* and the *EURASIP Journal on Audio, Speech, and Music Processing (JASMP)* and a reviewer for several journals and conferences including the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), and the International Society for Music Information Retrieval (ISMIR). He is a Senior Member of IEEE.

Erling Wold (e_wold@audiblemagic.com) received his Ph.D. degree from the University of California, Berkeley, in electrical engineering and computer science, specifically, signal processing for computer music. He is the chief scientist and technologist at Audible Magic, Los Gatos, CA 95032 USA. His research interests include signal processing for audio and music. He is a Member of IEEE, the Association for Computing Machinery, American Mathematical Society, and Opera America.

Richard Boulderstone (r_boulderstone@audiblemagic.com) received his M.Sc. degrees in statistics and advanced computer technology from Birkbeck, University of London. He is a research fellow and principal software engineer at Audible Magic, Los Gatos, CA 95032 USA. His research interests include software development, computer vision, digital libraries, and IT architecture.

References

[1] A. Meng, P. Ahrendt, J. Larsen, and L. K. Hansen, "Temporal feature integration for music genre classification," *IEEE Trans. Audio Speech Lang. Process.*, vol. 15, no. 5, pp. 1654–1664, Jun. 2007, doi: 10.1109/TASL.2007.899293.

[2] C. Joder, S. Essid, and G. Richard, "Temporal integration for audio classification with application to musical instrument classification," *IEEE Trans. Audio Speech Lang. Process.*, vol. 17, no. 1, pp. 174–186, Jan. 2009, doi: 10.1109/TASL.2008.2007613.

[3] Y.-N. Hung, C.-W. Wu, I. Orife, A. Hipple, W. Wolcott, and A. Lerch, "A large TV dataset for speech and music activity detection," *EURASIP J. Audio Speech Music Process.*, vol. 2022, no. 1, Dec. 2022, Art. no. 21, doi: 10.1186/s13636-022-00253-8.

[4] J. Saunders, "Real-time discrimination of broadcast speech/music," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Process.*, vol. 2, Atlanta, GA, USA, May 1996, pp. 993–996, doi: 10.1109/ICASSP.1996.543290.

[5] E. Scheirer and M. Slaney, "Construction and evaluation of a robust multifeature speech/music discriminator," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Process.*, vol. 2, Munich, Germany, 1997, pp. 1331–1334, doi: 10.1109/ICASSP.1997.596192.

[6] S. Venkatesh, D. Moffat, and E. R. Miranda, "Investigating the effects of training set synthesis for audio segmentation of radio broadcast," *Electronics*, vol. 10, no. 7, pp. 1–20, Mar. 2021, doi: 10.3390/electronics10070827.

[7] M. Kashif Saeed Khan and W. G. Al-Khatib, "Machine-learning based classification of speech and music," *Multimedia Syst.*, vol. 12, no. 1, pp. 55–67, Apr. 2006, doi: 10.1007/s00530-006-0034-0.

[8] K. Seyerlehner, T. Pohle, M. Schedl, and G. Widmer, "Automatic music detection in television productions," in *Proc. Int. Conf. Digit. Audio Effects*, Bordeaux, France, Sep. 2007.

[9] B. A. y Arcas et al., "Now playing: Continuous low-power music recognition," in *Proc. Conf. Neural Inf. Process. Syst.*, LongBeach, CA, USA, Nov. 2017.

[10] L. Vrysis, N. Tsipas, I. Thoidis, and C. Dimoulas, "1D/2D deep CNNs vs. temporal feature integration for general audio classification," *J. Audio Eng. Soc.*, vol. 68, nos. 1–2, pp. 66–77, Jan. 2020, doi: 10.17743/jaes.2019.0058.

[11] D. Snyder, G. Chen, and D. Povey, "MUSAN: A music, speech, and noise corpus," 2015, *arXiv:1510.08484v1*.

[12] D. Doukhan, J. Carrive, F. Vallet, A. Larcher, and S. Meignier, "An open-source speaker gender detection framework for monitoring gender equality," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Calgary, AB, Canada, Apr. 2018, pp. 5214–5218, doi: 10.1109/ICASSP.2018.8461471.

SP

An efficient music detection system can be the first step to other complex tasks, such as audio segmentation, music identification, or instrument classification.

In this era of tech giants with immense resources and massive models, we wish to promote the use of practical and low-cost ideas.

Thanks to the constant development of deep learning libraries, anyone can now build, train, and deploy their own model.

Temporal integration is a well-known technique for combining a sequence of features into a single one, for example, by computing statistics over time.

We wish to design a music detection system that is cheap to train, simple to implement, small to store, fast to compute, and sufficiently accurate.

The model presented in this article is efficient enough to be trained without requiring expensive GPUs.