Juhan Nam, Keunwoo Choi, Jongpil Lee,
Szu-Yu Chou, and Yi-Hsuan Yang

# Deep Learning for Audio-Based Music Classification and Tagging

*Teaching computers to distinguish rock from Bach*

©ISTOCKPHOTO.COM/TRAFFIC_ANALYZER

Over the last decade, music-streaming services have grown dramatically. Pandora, one company in the field, has pioneered and popularized streaming music by successfully deploying the Music Genome Project [1] (https://www.pandora.com/about/mgp) based on human-annotated content analysis. Another company, Spotify, has a catalog of over 40 million songs and over 180 million users as of mid-2018 (https://press.spotify.com/us/about/), making it a leading music service provider worldwide. Giant technology companies such as Apple, Google, and Amazon have also been strengthening their music service platforms. Furthermore, artificial intelligence speakers, such as Amazon Echo, are gaining popularity, providing listeners with a new and easily accessible way to listen to music.

While music-streaming services have made a huge volume of music accessible to users, the enormous size of the service catalogs has created the challenge of finding among so many choices the songs that fit users' tastes. A general approach to this issue has been collaborative filtering, which predicts songs of potential interest based on previous usage data, such as play history and song rating. Although collaborative filtering effectively retrieves songs and accommodates personalized recommendations, its performance is hampered by such issues as popularity bias and the cold-start problem, the challenge of recommending new music to users [2]. The content-based approach is often regarded as a supplementary solution to those problems. Pandora radio is a representative example as it retrieves songs by exploiting the similarities of song descriptors, such as genre, mood, instruments, and vocal quality. However, high-quality manual annotation is costly and not scalable, suggesting a need for better ways to automate classification of music content. As a result, much attention in the field of music information retrieval (MIR) over the last few years has centered on finding ways to automate the process of classifying music genre and mood and tagging music. Hereafter, this article will use the term *music classification and tagging* as a general expression for tasks that involve taking music audio data as input and automatically annotating them with a certain form of semantic label.

*Digital Object Identifier 10.1109/MSP.2018.2874383*
*Date of publication: 24 December 2018*

The focus of a survey paper on music classification and tagging in 2011 [3] revealed the previous trends in the field. Most of the 149 papers surveyed therein were based on the "conventional" machine-learning framework, which involves a pipeline of feature extraction and classifier learning. The features were mostly manually designed to succinctly represent acoustic or musical characteristics given the task. However, recent breakthroughs using deep neural networks have shifted the paradigm to learning representations in an end-to-end manner, which has opened the era of deep learning [4], [5]. This method has been applied to various tasks in MIR as well [6]. For several reasons, researchers have been especially active in exploring the problems of music classification and tagging. First, music classification and tagging tasks annotate audio clips at a track level (i.e., segments lasting several seconds or longer), and the audio clips are typically represented as two-dimensional (2-D) image-like data, such as mel-spectrograms. This is similar to the way images are classified, which means the technique may be borrowed and applied in the field of music classification. Second, an essential ingredient of deep learning is the availability of large data sets. One is the Million Song Data Set (MSD), which was introduced in 2011 [7]. MSD has facilitated large-scale training of deep neural networks for music classification and tagging tasks. Last, successful efforts to automate music classification have drawn interest from the music-streaming service industry, leading to investment in research resources to develop advanced content-based approaches [1], [8].

While the latest developments in other domains have inspired parallel developments in music, it is still necessary to take into consideration the specific properties of music signals when developing deep-learning methods for music classification and tagging. This article, an up-to-date tutorial-like survey, reviews the representative deep-network designs tailored for music classification and tagging, the best practices found thus far, the applications to music services and other MIR tasks, and, finally, the limitations and open issues that still need to be addressed.

**It is still necessary to take into consideration the specific properties of music signals when developing deep-learning methods for music classification and tagging.**

## From feature engineering to end-to-end learning

Humans classify or annotate music based on diverse characteristics extracted from the audio signals. For example, a heavily distorted electric guitar sound with growling vocals is a good indication of metal music. Swing rhythms, syncopation, and chromatic comping by polyphonic instruments (e.g., piano or guitars) are obvious cues that the music is jazz. Translating these acoustic and musical features into numerical representations that computers can interpret is the essence of music classification and tagging. This usually involves a series of computation steps that convert audio content into a time–frequency representation, extract discriminative features, summarize them over time, and repeat the feature extraction and summarization until the proper category for the music can be determined. The way of improving each feature extraction step to achieve the best performance has evolved with advances in learning algorithms from hand engineering based on domain knowledge to end-to-end learning. Humphrey et al. [9] explained the transition in a unified deep architecture model where multiple blocks of affine transformation, nonlinear function, and optional pooling operation are pipelined. Figure 1 illustrates four different feature representation approaches in their framework. In reviewing the evolution of such approaches,

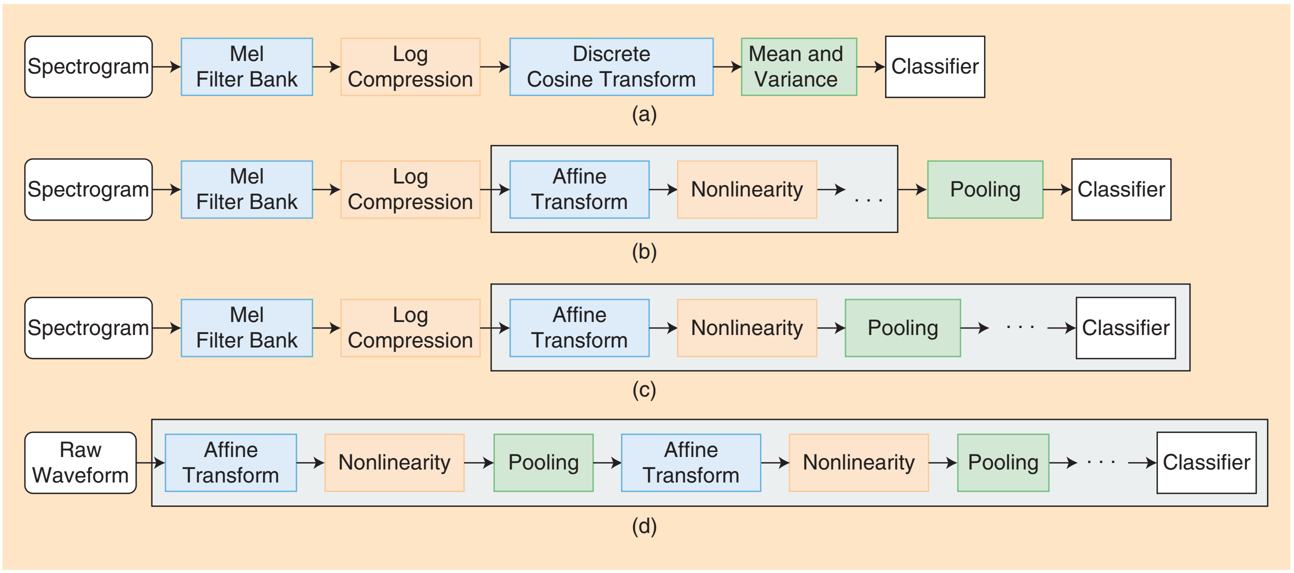

**FIGURE 1.** The transition of feature representation for music classification: (a) feature engineering [mel-frequency cepstral coefficients (MFCCs)], (b) low-level feature learning, (c) convolutional neural networks, and (d) end-to-end learning. The blocks inside the black lines indicate that they are learned by the algorithms.

we first separate them into two classes: feature engineering and feature learning.

### *Feature engineering*

A single line of melody can be arranged and performed in any of a variety of genres or moods of music, depending on the choice of instruments, chord progressions, rhythms, dynamics, and other musical elements. Considering the generative process in creating music, an intuitive approach to music classification and tagging would require features on each axis of the musical elements to be distilled and their distributions to be modeled. The traditional approach attempted to craft a variety of audio features under this principle. A representative example is the seminal work by Tzanetakis and Cook [10]. They tackled the automatic music genre classification problem by using three groups of audio features: timbre, pitch, and rhythm. The timbre feature was formed by summarizing the zero-crossing rate, low-level spectral features, low-energy feature, and MFCCs within a texture window. The pitch feature was extracted by encapsulating the pitch content from a multipitch estimator into two types of histograms, one that contains harmony information and one that contains pitch-range information. The rhythmic feature was represented by a beat histogram that explains temporal regularity by counting intervals of periodic energy fluctuation via a subband analysis. Finally, they combined all features and applied them to classifiers, such as the *k*-nearest neighbors and Gaussian mixture models. Since this study laid a foundation for music classification and tagging, numerous research studies have developed new or better-tuned audio features and have followed the two-stage framework, where the hand-engineered features are used as input of a standard classifier.

This feature-engineering approach designs each computation step manually based on the domain knowledge. For example, Figure 1(a) shows the computation pipeline of MFCCs. The mel filter bank and discrete cosine transform are tailored based on psychoacoustics and signal processing knowledge, respectively. These hand-engineered features have advantages in that they are interpretable and usually expressed in a compact form. However, most hand-engineered audio features are based on short-time analysis and may not capture high-level information in music. In addition, the engineering process is separated from the data-driven optimization in the classifier. Currently, this two-stage approach seems to lead to an imperfect solution.

### *Feature learning*

The gist of deep learning is that the feature representations of input data can be learned by the algorithm via the deep neural networks. That is, learning is achieved layer by layer, with higher-level features learned in the deeper layers. This contrasts with the feature-engineering approach in that the domain knowledge is much less involved in finding the features and the input data are processed at a minimum level before they are fed into the algorithm. In the tide of deep learning, various feature-learning algorithms have been introduced and applied to music classification and tagging. We categorize them into the following three classes: low-level feature learning, convolutional neural networks (CNNs), and end-to-end learning models.

#### Low-level feature learning

Early studies focused on learning low-level audio features to replace the handcrafted features in the two-stage framework, as illustrated in Figure 1(b). One kind of research focused on learning a meaningful dictionary of spectrograms using unsupervised learning algorithms, such as the restricted Boltzmann machine, *K*-means, and sparse coding (e.g., [11]). These shallow feature-learning algorithms are usually trained to encode multiple frames of spectrograms into a high-dimensional sparse feature vector. They capture a variety of musically interpretable time–frequency patterns. The other kind of research focused on supervised feature learning that maps a short-term spectrum to genre or mood labels with a pretrained multilayer perceptron or deep belief networks (e.g., [12]). The hidden layer activations are used as learned features. While both groups deliver better performance than that using hand-engineered features in many music classification and tagging tasks, they are still limited to low-level feature learning, and the adopted framework still has two stages.

#### CNNs

Lately, CNNs have been the most widely used learning model in music classification and tagging tasks [13], [14]. Based on several seconds of audio as an input, CNNs can be improved in an end-to-end fashion to learn hierarchical features. However, as shown in Figure 1(c), most successful CNN models used the spectrogram (particularly, mel-spectrogram) as the input representation, indicating that domain knowledge is still helpful. Under different assumptions of locality and translation invariance on the time–frequency representation, several configurations of CNN models have been suggested [13], [14]. We describe more details about such models in the next section.

#### End-to-end learning models

More recently, a few attempts have been made to directly use raw waveforms as the input of CNNs [8], [13], [15]. As illustrated in Figure 1(d), no single step requires a hand-designed representation, thus realizing a complete end-to-end feature learning. Lee et al. proposed a successful model in music classification and tagging [15], [16]. They found that the model performs better when the bottom convolutional layer takes a small grain of samples (e.g., two or three samples) rather than a typical window size (e.g., 256 or 512 samples). However, as the filter size in the convolutional layer is smaller, the model becomes progressively deeper, and, as a result, it takes longer to train. More details about this type of model are described in the next section.

## Deep-learning models

In this section, we review three representative CNN models for music classification. The first two models are one-dimensional (1-D) [13] and 2-D CNNs [14], each of which has been applied in efforts to make networks more flexible. This trend toward

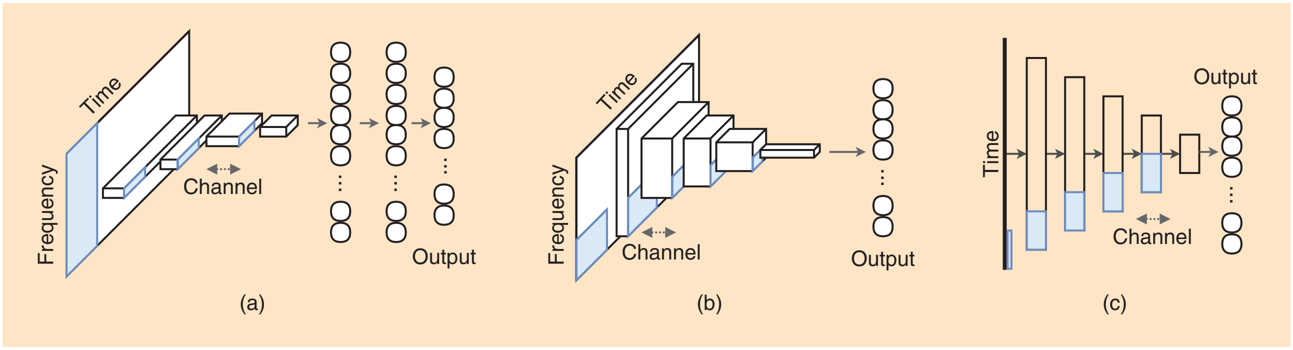

FIGURE 2. Block diagrams of (a) 1-D, (b) 2-D, and (c) sample-level CNNs. (a) and (b) are based on 2-D time–frequency representation inputs (e.g., mel-spectrograms or short-time Fourier transforms), and (c) is based on a time-series input.

greater flexibility continues with the most recent and most successful approach, the sample-level CNN [15], where a time-series audio signal is used as input. Additionally, we will introduce a few advanced methods that can improve performance. While there are many other kinds of architectures, we focus on CNN-based ones in this article, as they are more widely used.

For the sake of clarity, in this section we specify layers using Keras-style grammar (https://keras.io). A 2-D kernel is specified by its lengths in the frequency ($f$) and time ($t$) axes, e.g.,$(f, t)$. A convolutional layer with 2-D kernels measuring $(f, t)$, $N$ channels, $(s_1, s_2)$ strides, and "valid" (or "same") padding is denoted as Conv2D[filters = $N$, kernel_size = $(f, t)$, strides = $(s_1, s_2)$, padding = "valid"] with some of the parameters omitted if they follow the aforementioned default values. In addition, the parameter names can be omitted while keeping their order (i.e., like Python syntax). Conv1D is defined similarly, but the kernel size and stride are 1-D. Max-pooling layers are defined as MP1D(pool_size) and MP2D(pool_size). Finally, we specify the size of a feature map with $(F, T, N)$ for lengths of $F$ in the frequency axis, $T$ in the time axis, and $N$ channels.

## 1-D CNNs

Dieleman et al. [13] initiated some of the earliest advancements in the area of deep learning in music classification and tagging. Dieleman also made a significant early contribution with his blog post about his internship with Spotify (http://benanne.github.io/2014/08/05/spotify-cnns.html). The network structure is illustrated in Figure 2(a), and we call it *1-D CNN* in this article. Here, "1-D" refers to the dimensionality of the first layer's convolution operation and should not be confused with that of the kernel.

The assumed behavior of 1-D CNNs with respect to music signal input is straightforward. As mentioned previously, 1-D CNNs take a time–frequency representation, such as mel-spectrogram, as input. With the kernel height of $F$, the first convolutional layer "sees" the entire frequency range at once. That is to say, during training, the network finds some patterns that cover the entire frequency range. For example, the size of the first convolutional layer's kernel in [13] is (128, 4) with the number of output channels as 256, i.e., Conv2D[256, (128, 4), "valid"], resulting in (1, 599, 256)-sized feature maps. More convolutional layers and densely connected layers are shown as in Figure 2(a). This structure is musically plausible in some sense, as it puts a strong prior to the network design at the same time. To elaborate, we know that in images an object can appear in any location, making 2-D CNNs a popular design choice, as 2-D CNNs can deal with such spatial variants. However, this may not be the case for musical audio. In a time–frequency representation, a musical object or pattern can appear anytime, but not in any frequency band. This is because different musical components can exist in different frequency ranges with a minor shift. In other words, the invariance property we want to have may be mostly along the time axis. This characteristic enables us to see and interpret what is learned at the first convolutional layer. Because the learned kernels operate directly on the spectrogram input, we can visualize the kernels using the learned weights and see which genres of songs maximally activate them. For example, in Figure 3, we show the top four relevant tags for a few selected kernels. The tags are sorted (in descending order from top to bottom) based on the tag activation score of each kernel. We can see that the corresponding tags somehow explain each of the learned kernels.

One-dimensional CNNs are computationally efficient. Its first convolutional layer takes the entire frequency range, makes the feature maps of the subsequent layers much smaller (the length of the frequency axis becomes 1), and, accordingly, drastically reduces the total number of network parameters. However, this is actually a double-edged aspect of 1-D CNNs. A small number of parameters means it is easier to train the network with relatively small data sets. At the same time, it means that 1-D CNNs will not fully benefit from the development of hardware resources and large-scale data sets due to their limited representation power.

The aforementioned assumption, or the strong prior, of 1-D CNNs introduces a clear limitation: a complete lack of frequency-axis shift-invariance. In the first layer, the 128-dimensional frequency components are assumed to have their own meaning; therefore, a slight change along the frequency axis (i.e., pitch transposition) results in a significantly different activation. Using a slightly smaller kernel [e.g., (126, 4)] has been proposed as an alternative (while making it technically 2-D convolution), but it only provides a "global" shift-invariance. In other words, assuming a max-pooling of $(3, x)$ follows, this alternative approach is invariant to a global transposition by two semitones. However, it is not invariant to local changes, e.g., a combination of an $\epsilon_1$ frequency shift in the bass guitar component, an $\epsilon_2$ shift in the vocal component, and an $\epsilon_3$ shift in the piano component, where $\epsilon$s are different (unlike in the case of 2-D CNNs). As a result, the representations that 1-D CNNs learn in the first layer are limited to some common patterns of the entire frequency ranges.

### 2-D CNNs

With larger data sets and better hardware resources becoming available, a natural step is to increase network flexibility to improve representation learning, as in [14]. The network structure is illustrated in Figure 2(b). We call it *2-D CNNs* in contrast to 1-D CNNs, as they focus on the contiguous 2-D convolutional layers including the first one. The five-layer structure in [14], for example, gradually combines smaller time–frequency patterns to create larger ones with 2-D convolutional layers, e.g., Conv2D[32, (3, 3)], which allow for small shifts by the following max-pooling layers, e.g., MP2D[(2, 2)]. Since the kernel sizes are small, the padding strategy ("valid" or "same") is not of very much interest.

Two-dimensional CNNs assume that more flexibility will be helpful in finding the time–frequency patterns. The flexibility can have several aspects: the shift (or location) invariance along both axes, the size of the patterns, and small distortions within the patterns. They are realized by 2-D convolutional layers with small kernels (typically three-by-three) and 2-D max-pooling layers. Although this may contradict the different meanings of time and frequency axes mentioned in the previous section, 2-D CNNs have, in fact, performed better than 1-D CNNs. Thanks to their simple structure and good performance, 2-D CNNs may now be the most popular approach for music audio classification.

Two-dimensional CNNs usually demand better hardware than 1-D CNNs for two reasons. First, the parameters easily outnumber those of 1-D CNNs due to the use of contiguous 2-D kernels, which then require more memory. Second, the training and use of 2-D CNNs add a significant computation burden due to the large size of the feature maps, along which the kernel should be convolved. For example, with a 1-D CNN, all of the feature maps are of size $(1, x, N)$. The frequency axis is always of length 1, which makes the feature maps 1-D with channels. In contrast, with a 2-D CNN, the feature maps would be of size $(F, x, N)$, i.e., 2-D with channels. This significantly increases the computation in both the forward and backward passes of the model training process.

So far, we have reviewed the advantages and disadvantages of 2-D CNNs as compared with 1-D CNNs. In practice, 2-D CNNs offer some practical advantages. For example, improvements in hardware have enabled researchers and practitioners to use 2-D CNNs when they have sufficient data. Once the bottlenecks of the data size and hardware resource are resolved, the flexibility of 2-D CNNs may bring about better performance. Empirical evidence provided in [17] compares various CNN architectures according to number of parameters, computation use, and performance.

### Sample-level CNNs

As explained in the previous subsection, 2-D CNNs may lead to better results in music classification and tagging, as they provide more flexibility. Sample-level CNNs go further in the same direction by discarding the 2-D time–frequency input preprocessing stage and learning directly from the audio waveforms in an extremely granular way [15]. Although it was not the first approach that directly learns representations from the raw audio, it is the first architecture that has achieved a state-of-the-art performance with a significantly shorter kernel size than the regular window size in short-time analysis with a deep network.

Among the variations in [15], we explain the details of the $3^9$ model structure. As illustrated in Figure 2(c), the

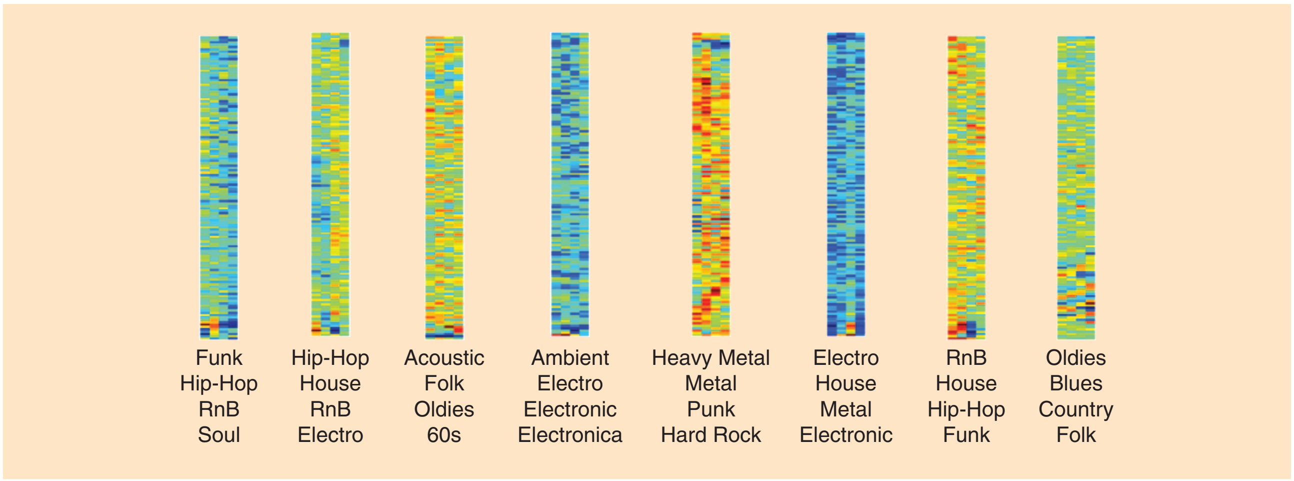

**FIGURE 3.** Visualization of the first convolution kernels in a trained 1-D CNN with relevant tags. For each kernel, an activation score for each tag is calculated by computing the average activation of the kernel for all songs with the tag. The four tags are those with the highest activation scores for the selected kernel. The kernels are of size 128 × 4, where 128 is the number of mel bins and 4 is the number of frames in time. The learned kernels can be interpreted as spectrotemporal patterns associated with acoustic characteristics of music with the tags. RnB: rhythm and blues.

model consists of one [Conv1D(filters = 128, kernel_size = 3, strides = 3)], 9 × [Conv1D(128, 3, 1) + MP1D(pool_size = 3)], and the output layer. The base (3) of the model name indicates the kernel size and stride of the layers while the exponent (9) means the number of Conv1D + MP1D modules. The first layer learns 128 1-D kernels, with which the layer can extract certain 1-D patterns at each time step. The activation of the first convolutional layer is based on size (time step, channels), and we can understand it as a 2-D time–frequency representation where each frequency component is not necessarily a pure sinusoid and the frequency axis is not sorted. Afterward, those basic nonsinusoid components are combined with convolutional layers. The effective operation in the subsequent convolutional layer is equivalent to that of 1-D CNNs.

The following three properties of sample-level CNNs, all of which are related to the extra flexibility of the model, may contribute to their strong performance. First, one of the motivations underlying sample-level CNNs is to learn "phase-invariant" representations. The time-domain kernels involve learning all the possible time shifts within the kernel window. Therefore, a large kernel may require even more filters to cover the variations. The deep stack of the short kernels and max-pooling layers in sample-level CNNs effectively takes care of the phase variation. Second, by learning kernels that are directly applied to the audio signal, sample-level CNNs improve the spectral bandwidth assigned for the input signal analysis. Finally, as previously mentioned, the kernels in the first convolutional layer of sample-level CNNs can be chosen to represent harmonic components rather than pure sinusoids, which form usual 2-D time–frequency representations, such as the spectrogram. This flexibility also improves the discriminative power of the learned features.

A downside of sample-level CNNs is their computation complexity. The authors of [15] informally reported that it took about three to seven times longer to train sample-level CNN models as compared with 1-D CNN models. A way to accelerate the training is to down-sample the waveform input [16], but researchers need to develop more efficient models.

## Advanced models

This section summarizes several advanced methods that have addressed various aspects of deep learning-based models. We note that these methods are designed to achieve different goals and that they are not mutually exclusive but can be combined in a model.

### Convolutional recurrent neural networks

A convolutional recurrent neural network (CRNN) is a variant of the CNN structure that uses recurrent layers to replace the final convolutional layers [17]. The CRNN model assumes that the long-term patterns are better encoded with recurrent layers than with convolutional layers. This is probably because the important patterns are shorter than the input duration. Therefore, the temporal dynamics of the patterns is a sequence of some short-term patterns rather than a whole, single pattern. The use of recurrent layers also makes the model flexible with respect to the input length, which can be useful for music classification. The network structure in [17] is based on 2-D CNNs, but we note that the recurrent layers can be added to other types of CNNs as well.

### Residual networks and squeeze-and-excitation networks

These network architectures have achieved state-of-the-art performance on ImageNet challenges in 2015 and 2017, respectively [18], [19]. Unlike the usual network structures, some layers in a residual network share skip connections, with which the layers are directly connected without any operation. Researchers have enthusiastically adopted this idea because it enables very deep networks (e.g., with more than 100 layers) to be trained. The squeeze-and-excitation network, by applying a trained channel-wise weighting, provides another way to enhance the representation of a layer. It was successfully applied for music autotagging in [20].

### Pairwise data

Finally, a more macroscopic modification of a network can be done with a different supervised learning scheme. When the label consists of pairwise similarities or ranking, it is possible to achieve metric learning by using a triplet loss function. A network using this function takes three data samples: an anchor, a positive item, and a negative item. The network learns respective representations, or embeddings, in a way that the embeddings of the anchor and the positive item are close to each other while those of the anchor and the negative item are not. In MIR, music content embeddings were used to predict music similarity in [21].

## Data sets and tasks

In this section, we describe four public data sets that have been widely used for music classification and tagging. One of the crucial elements in the success of deep learning is the availability of large-scale public data sets that are used not only for the training of deep-learning models but also for benchmark evaluation. The MIR community has organized an annual algorithm evaluation exchange called *Music Information Retrieval Evaluation eXchange* (*MIREX*) which includes several music classification and tagging tasks; see http://www.music-ir.org/mirex/wiki/MIREX_HOME for more information. However, the development of deep learning has not benefited much from this exchange chiefly because the MIREX data sets are not open to the public and both the training and testing are conducted by the MIREX committee. Also, the volumes of the hidden data sets are not sufficient to fully evaluate the deep models. Presumably, this may be attributed to the serious copyright issues related to music content because the commercial music is released through professional sound producers and the license is more restricted. The four public data sets presented below circumvent the issue by using trimmed or degraded audio clips, e.g., 30 s with 16- or 22.05-kHz sample rate, or copyright-free music tracks. We note that this is not a comprehensive list of available data sets but a selection

Table 1. Selected data sets for music classification.

| Data Sets | Number of Clips | Number of Artists | Main Task | Annotation | Audio | Year |
|---|---|---|---|---|---|---|
| GTZAN [10] | 1,000 | ~300 | Genre classification | Author's labeling | Yes | 2002 |
| MTAT [22] | 25,863 | 230 | Autotagging | Crowdsourced | Yes | 2009 |
| MSD [7] | 1 million | 44,745 | Autotagging | Crowdsourced | No | 2011 |
| FMA [23] | 106,574 | 16,341 | Genre classification | Artist's labeling | Yes | 2017 |

of those that have been used mainly to evaluate deep-learning models (Table 1).

### *GTZAN*
Despite its small size, GTZAN (its name derived from the name of George Tzanetakis, who assembled the data set) is one of the most widely used data sets for music genre classification [10]. It contains 1,000 pieces of 30-s audio clips (ten genres and 100 songs for each genre). The up-to-date version uses an artist-stratified split of 443, 197, and 290 audio clips for training, validation, and testing, respectively, with no repeated artists across these sets. The artist-stratified split is unique in the music domain because artists are likely to have similar styles of music across their own songs. We note that GTZAN has also been used for conducting a target task with a small volume of data in the context of transfer learning [16], [24].

### *MagnaTagATune*
MagnaTagATune (MTAT) is one of the most widely used benchmark data sets for music autotagging. It is a multilabel music classification task that annotates genre, mood, instruments, and other song descriptions heterogeneously [22]. The data set comes with tags and similarity annotations. The autotagging benchmark has been conducted using a different number of tags, including 188 tags (the original version), 160 tags (the MIREX 2009 version), and the most frequently used 50 tags. The 50-tag version is currently the most benchmarked. From the 16 predefined partitions of the data set, a common practice is to use the first 12 for training, the 13th for validation, and the remaining three for testing. This data set contains 25,863 30-s audio clips. Its midsize volume is appropriate for training a deep neural network. However, the data set has drawbacks. For example, some clips are cut from the same song, and the music styles are slightly different from popular chart music, as the music tracks are mainly obtained from independent musicians.

### *MSD*
The MSD is a cluster of complementary data sets created from contributions by the MIR community [7]. The original MSD contains artist-level metadata along with the Echo Nest (hand-engineered) audio features without access to the original audio. However, the MSD has been augmented by other metadata by matching the identification data (IDs), including the song-level tags, similarity, lyrics, cover songs, user listening history, and genre labels. To train deep neural networks that take spectrograms or waveforms, researchers have used 30-s preview audio clips downloaded from 7digital (https://www.7digital.com/). Also, the Last.fm tag annotations (https://www.last.fm/) have been widely used in benchmarking for music autotagging.

### *Free Music Archive*
The Free Music Archive (FMA) is the most recently published large-scale data set under the Creative Commons license [23] (http://freemusicarchive.org/). The data set provides the rich track-level, album-level, and artist-level metadata, including the genres, the number of listens, and tags. It is mainly oriented for genre classification, and there are four subsets for benchmarking: small, medium, large, and full. The small and medium subsets are for single-label genre classification, whereas the large and full subsets are for multilabel genre classification. Although the main task is genre classification, tag annotations are also included in the metadata.

### *Evaluation*
As mentioned previously, we set up the problems as either a multiclass (e.g., genre or mood classification) or a multilabel (e.g., autotagging) task. In the multiclass task, the models are primarily evaluated using the accuracy score. In the multilabel task, the predictions are regarded as independent binary outputs. Each of these outputs is evaluated in both annotation and retrieval (or ranking) contexts. The main metrics for annotation are precision, recall, and $F$ score. They are computed for each word label and are averaged. The metrics for retrieval include the area under the receiver–operator curve (AUC), the mean average precision, and the precision at (or up to) rank $K$ (P@$K$). Among them, AUC has been primarily used to compare different deep-learning models. Table 2 lists the performance comparison reported so far, showing how the AUC obtained for MTAT and MSD has improved over the years due to a cumulative effort from the MIR community.

We note that each of the metrics has slightly different characteristics. For example, the P@$K$ metric is important when we develop recommendation services because users tend to be interested in the top $K$-ranked results rather than all of them. Therefore, depending on the target application, one may choose different performance metrics when evaluating the results.

## Practical guide
This section describes several practical issues when applying a deep-learning model for music classification and tagging tasks.

**Table 2. A selection of results for music auto-tagging task. The AUC metric is used for the evaluation.**

| Models | Published Year | End to End | MTAT | MSD |
|---|---|---|---|---|
| 1-D CNN [13] | 2014 | No | 0.8815 | – |
| | | Yes | 0.8487 | – |
| Multiscale CNN [25] | 2016 | No | 0.8960 | – |
| 2-D CNN [14] | 2016 | No | 0.8940 | 0.8510 |
| Multi-D CNN [26] | 2017 | No | 0.8930 | – |
| CRNN [17] | 2017 | No | – | 0.8620 |
| Sample-level CNN [15] | 2017 | Yes | 0.9055 | 0.8812 |
| ReSE-2-multisample CNN [20] | 2018 | Yes | 0.9113 | 0.8847 |

### *Data preprocessing*
The first parameter to check is the sample rate. While 44.1 kHz is the standard for commercial music tracks, most data sets are down-sampled to 16 or 22.05 kHz. Researchers often reduce the sample rate even further (e.g., 8 or 12 kHz) as they observe that by reducing data, training is quicker without significantly affecting performance [14], [16]. The 1-D and 2-D CNNs take spectrograms as audio input. In particular, the mel-spectrogram or other log-frequency spectrograms are commonly used. This requires selecting short-term analysis parameters (e.g., window function, hop size), a mel-band size, and the log compression strength. Librosa (https://librosa.github.io/librosa/) is a widely used audio-processing library for this purpose. In contrast, sample-level CNNs do not require any preprocessing other than sample-rate conversion. The waveform input is already zero-centered, and the amplitude of commercial music clips is normalized well by postprocessing (e.g., audio mastering).

### *Data augmentation*
Data augmentation is a technique that regularizes the model by increasing the volume of data. For audio signals, the digital audio effects, such as pitch-shifting or time-stretching, are effective means to this end. However, this should be done within a range where the nature of music labels is not distorted. Data augmentation is not found in the music classification literature yet, but it may be useful when the data set of the target task is small. Musical Data Augmentation (http://muda.readthedocs.io/en/latest/) is a useful audio-processing library for this purpose.

### *Input length*
Semantic labels are usually annotated to each song at a track level, and the audio length is typically several minutes. Therefore, to use audio tracks to train the models, they need to be chopped into a fixed length of segments. This causes a tradeoff between model complexity and label noisiness. If the segment is shortened, a more compact model can be trained with greater input data. However, the labels inherited from the track level tend to be noisier due to the dynamic nature of music within a track, and the compact model can miss learning high-level musical features. On the other hand, if the segment is lengthened, the label noisiness will be mitigated and a more long-term structure can be learned. However, this requires having more complex models along with more data. The common practice is using segments between 3 and 6 s as input. Some complex models take up to 30 s [14], [17]. The assumption that the segments of a track share the same labels as the track has also been referred to as a *weakly supervised learning problem* [25], as when the segment-level supervision is noisy. An advanced method to deal with such an issue might be adding the so-called attention module to the neural network, as demonstrated by [27] for music mood classification.

## Applications
In this section, we explain how a neural network model pretrained on larger-scale labeled data for music classification and tagging can be applied to other tasks, such as classifying across data sets, making recommendations, thumbnailing music, and predicting hit songs, as illustrated in Figure 4.

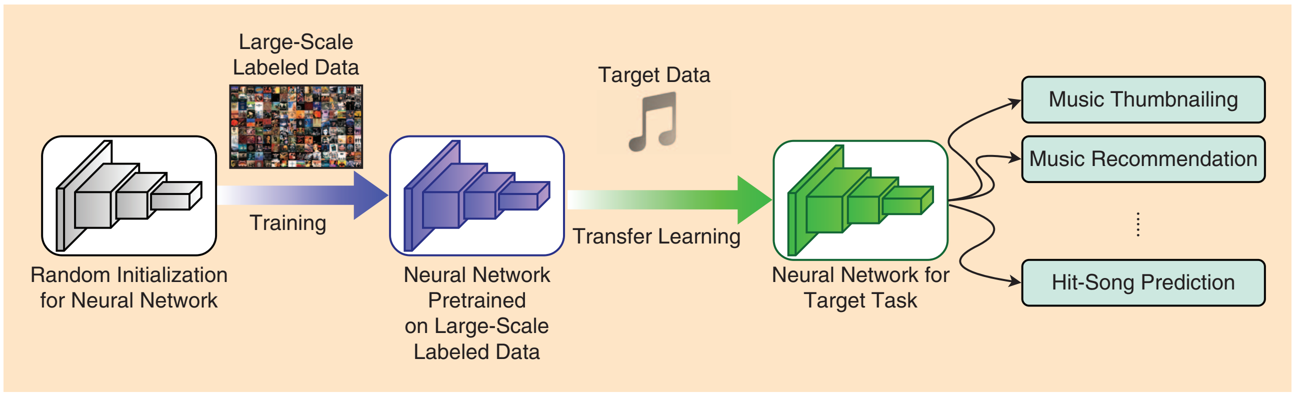
Large-Scale Labeled Data
Target Data
Training
Transfer Learning
Random Initialization for Neural Network
Neural Network Pretrained on Large-Scale Labeled Data
Neural Network for Target Task
Music Thumbnailing
Music Recommendation
Hit-Song Prediction

**FIGURE 4.** Transferring the knowledge of a neural network pretrained on larger-scale labeled data to other music applications. Refer to the “Applications” section for details.

Findings show that pretrained models using large-scale labeled data can provide a good estimate of the similarities among audio content (Figure 5). Therefore, with the so-called transfer-learning techniques, we can build classifiers for problems with sparsely labeled data on top of such pretrained models. For example, Choi et al. [24] used approximately 250,000 MSD preview clips to train a 2-D CNN to classify 50 music tags. They then showed that a concatenated feature vector using the activations of the feature maps of the CNN can serve as a nice general-purpose music representation. This is useful for a variety of other tasks, such as classifying ballroom dancing and other subgenres, predicting the emotions the music might stimulate, distinguishing between vocal and nonvocal sounds, and sorting various sound events, such as car horns and dog barks.

Pretrained models can also contribute to addressing the challenge of making content-based music recommendations. For instance, Pandora, powered by the Music Genome Project, can create various personalized playlists for each of its users by combining traditional collaborating filtering algorithms with the classified attributes of music [34]. Compared to the purely collaborating filtering methods, adding content filtering by means of pretrained models for music classification and tagging helps ensure acoustic consistency (e.g., similarities in genre/style, rhythmic patterns, vocal timbre, or expressed emotions) in the recommended list of music, which in turn improves the user experience. With content filtering, the (acoustic) diversity of the recommended music can also be controlled [2].

An interesting application of pretrained models is music thumbnailing, i.e., detecting the highlight of a song. Huang et al. [27] employed a pretrained model for music-mood classification and an attention module to learn to weigh the contribution of a song's different segments in deciding the overall mood of the song. Then, a moving window was used to aggregate the per-segment attention scores over time to pick the song's peak, assuming that the highlight is usually the most emotional part of a song. They achieved a promising result in highlight detection without using any labeled data related to music highlights.

Another interesting application is audio-based hit-song prediction. Yu et al. [28] used a pretrained 1-D CNN model for music classification as part of a bigger CNN model for predicting song popularity. The experiments showed that deep structures are indeed more accurate than shallow structures in predicting song popularity and that the use of the pretrained music classification model further improves the accuracy by a large margin. We believe that similarly pretrained models can also be applied to other problems.

## Limitations and future challenges

In this section, we discuss some major limitations of the existing methods for music classification and tagging, and we outline some directions for future research.

### Share of audio data

The problem of copyright infringement may limit widespread research on music classification and tagging. People cannot freely distribute the audio files from the data sets. Common approaches for getting around this issue include sharing precomputed features instead of the audio files, providing a list of IDs with which people may find the audio previews on the web, or using copyright-free music. The last approach provides more options as people can get the audio files for the entire songs. However, to work with popular music that people are familiar with (which are usually copyright protected), some other solutions are still needed. A possible approach is to automatically generate music that is similar to the popular music by using deep-generative models, such as generative adversarial networks [30].

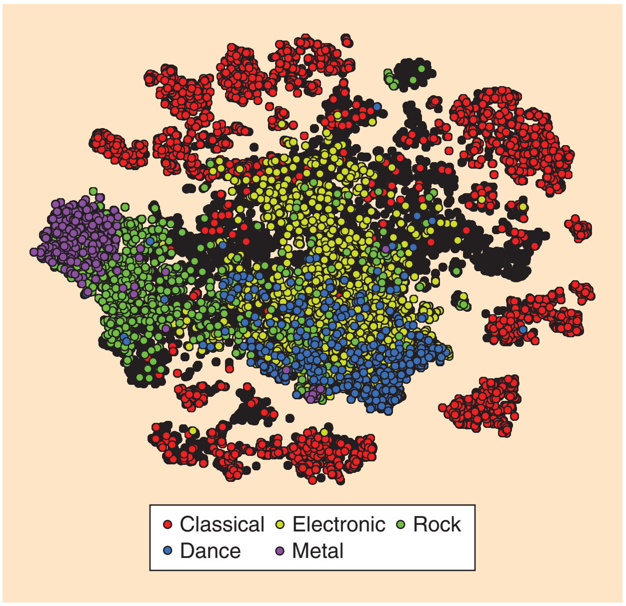

FIGURE 5. We generate T-distributed stochastic neighbor embedding visualization (best viewed in color) of the distribution of song embeddings by using a CNN model [29] trained on MTAT. The colors represent five different tags of music (provided by MTAT). The black dots denote songs that are not labeled by any of these tags. We see that songs with similar genres cluster together in the learned feature space.

### Musically meaningful network design

We also expect developers to make more use of peculiar characteristics of music in the design of deep neural networks. In the past few years, deep learning-based approaches to many MIR problems have established new state-of-the-art benchmarks. However, to explain the network and for better performance, future work is needed to bring back music-domain knowledge to the loop of network design. For instance, instead of expecting that the network can learn abstract representations of music in different hierarchies from the bottom up on its own, it might be better to inform the network (in a top-down fashion) of the midlevel features, such as the presence of syncopation, the extensive use of diminished chords, and the use of synthesizer. Then, classifiers could be built on top of these midlevel features. This requires a joint effort from the research community to put together resources and labeled data to model different layers of music knowledge and to conduct experiments to find out the best ways to use them in a neural network, with all the layers possibly trained in an end-to-end fashion.

Another way to incorporate music knowledge is to use not only the audio files but also the corresponding musical scores, if available. Musical scores contain rich information about the music piece, such as the melody line and the chord sequence. Score-informed approaches have been shown to greatly improve the performance of source separation [31]. By aligning a score with the audio recording of its actual performance, we can also extract performance-related features (e.g., stylistic changes in velocity, note duration, and the use of different playing techniques) that characterize how the performer interprets the piece of music. However, to date, little work has been done to use the audio and musical score jointly in a neural network. Future work can build, for example, a two-stream network that takes as input the audio file as well as its score or other symbolic representations.

We know that a music piece is usually composed of several elements, such as melody, chords, percussion, and baseline, and each of them is often played by different instruments [29]. However, most neural networks for music classification process audio inputs as a whole without distinguishing among the component sources of sound. While deep-learning approaches have led to the state-of-the-art results in sound-source separation and music classification, little work has been done to jointly tackle the two problems under a unified network. Requiring the neural network to learn to separate the musical sources that compose an audio mixture while performing feature learning can, therefore, be an important future direction.

### Vocabulary and personalization

The diversity and coverage of labels considered in classification and tagging models can also be increased. Ideally, it is better to have a granular vocabulary as fine as that of Pandora's Music Genome Project [1], which claims to have around 450 musical attributes. One possible solution is to leverage the abundant user-provided tags from social platforms, such as last.fm, Twitter, or SoundCloud. However, how to get rid of the social tags' noises and ambiguity while learning an effective music classifier remains an open issue. In imagining extreme possibilities, we foresee technologies that would enable end users to use arbitrary natural language as input (e.g., via voice commands) to query for music. For example, "Hey Google, I need music to make me feel better" and "Alexa, I cannot fall asleep. Maybe some music?" Such scenarios may be important given the ever-increasing popularity of artificial intelligence speakers. To support such retrieval applications, we need to collaborate with researchers from the speech community to better understand natural language. The vocabulary considered by our machines in describing music also has to be expanded and adapted to cope with the richness of natural language.

Moreover, the associations between music and some types of labels such as moods (e.g., "happy," "aggressive," "sad," "relaxing") and usages (e.g., "for exercising," "for reading") are known to be subjective. Therefore, it is more difficult to computationally model them. However, such labels are important, for example, if we want to automatically create playlists that fit a user's mood or activity. We surmise that the assignment of such labels has to be personalized, taking into account the listener's preference as well as the "personal definition" of those labels to the listener. Although much research has been done for music mood classification, it remains to date a challenge to effectively personalize such systems.

### Cross-modality approach

We see a lot of cross-modality research in the neighboring field of computer vision, which aims to combine the visual world with the textual world. Notable applications include image captioning, conditional image generation from visual attributes, and cross-modality retrieval. We expect similar attempts to flourish in the MIR community as well, not only for classification and tagging tasks (e.g., [33]) but also for generative tasks, such as tag-conditioned music generation, melody-conditioned lyrics generation, album cover generation, and music video generation. To facilitate research on these tasks, sharing pretrained models or knowledge (e.g., best practices in model training) can play an important role.

Music is important in our daily lives and there are many ways machine learning can improve or change the way we experience and create music. By summarizing what has been known thus far, we hope this article can encourage follow-up research to further enhance our modeling and understanding of music.

## Authors

***Juhan Nam*** (juhannam@kaist.ac.kr) received his B.S. degree in electrical engineering from Seoul National University, South Korea, in 1998 and his Ph.D. degree in music from Stanford University, California, in 2013 studying at the Center for Computer Research in Music and Acoustics. He is an assistant professor at the Graduate School of Culture Technology at the Korea Advanced Institute of Science and Technology (KAIST), South Korea. Before joining KAIST, he was a staff research engineer at Qualcomm, San Diego, California, from 2012 to 2014. He was also a software/digital signal processing engineer at Young Chang (Kurzweil), South Korea, from 2001 to 2006. He is interested in various topics at the intersection of music, audio signal processing, and machine learning. He is a Member of the IEEE.

***Keunwoo Choi*** (keunwoo.choi@qmul.ac.uk) received his B.S. degree in electric engineering and studied applied acoustics for his M.S. degree at Seoul National University, South Korea, in 2009 and 2011, respectively. He worked as a researcher at the Electronic and Telecommunications Research Institute, South Korea. He received his Ph.D. degree in 2018 from the Centre for Digital Music, School of Electronic Engineering and Computer Science, Queen Mary University of London, United Kingdom. In 2017, he received the Best Paper Award at the 18th International Society of Music Information Retrieval Conference. He is currently with Spotify Inc., New York. His research interests include music information retrieval and machine learning.

***Jongpil Lee*** (richter@kaist.ac.kr) received his B.S. degree in electrical engineering from Hanyang University, South Korea, in 2015. He received his M.S. degree in 2017 from the Graduate School of Culture Technology at the Korea Advanced Institute of Science and Technology, South Korea, and is currently pursuing his Ph.D. degree from the same institution. From July to September 2017, he was an intern at Naver Clova Artificial Intelligence Research. His current research interests include machine learning and signal processing applied to audio and music applications.

***Szu-Yu Chou*** (fearofchou@citi.sinica.edu.tw) received his bachelor's degree from National Formosa University, Huwei, Taiwan, in 2010. He is currently working toward his Ph.D. degree at National Taiwan University, Taipei. He is a research assistant in the Research Center for Information Technology Innovation at Academia Sinica. His research interests include music recommendation and music information retrieval. In 2015, he received the Best Paper Award at the IEEE International Conference on Multimedia and Expo.

***Yi-Hsuan Yang*** (yang@citi.sinica.edu.tw) received his bachelor's and Ph.D. degrees from National Taiwan University, Taipei, in 2006 and 2010, respectively. He is an associate research fellow at the Research Center for Information Technology Innovation, Academia Sinica. He is also a joint-appointment associate professor with National Cheng Kung University, Taiwan. His research interests include music information retrieval, affective computing, multimedia, and machine learning. He is an author of the book *Music Emotion Recognition*. He has been an associate editor of *IEEE Transactions on Affective Computing* and *IEEE Transactions on Multimedia* since 2016. He is a Senior Member of the IEEE.

## References

[1] M. Prockup, A. F. Ehmann, F. Gouyon, E. M. Schmidt, Ò. Celma, and Y. E. Kim, "Modeling genre with the Music Genome Project: Comparing human-labeled attributes and audio features," in *Proc. Int. Society for Music Information Retrieval Conf.*, Málaga, Spain, 2015, pp. 31–37.

[2] S.-Y. Chou, L.-C. Yang, Y.-H. Yang, and J.-S. Jang, "Conditional preference nets for user and item cold start problems in music recommendation," in *Proc. IEEE Int. Conf. Multimedia and Expo*, Hong Kong, 2017, pp. 1147–1152.

[3] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "A survey of audio-based music classification and annotation," *IEEE Trans. Multimedia*, vol. 13, no. 2, pp. 303–319, Apr. 2011.

[4] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems*, Lake Tahoe, NV, 2012, pp. 1097–1105.

[6] K. Choi, G. Fazekas, K. Cho, and M. Sandler. (2017). A tutorial on deep learning for music information retrieval. arXiv. [Online]. Available: https://arxiv.org/abs/1709.04396

[7] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," in *Proc. Int. Society Music Information Retrieval Conf.*, Miami, FL, 2011, pp. 591–596.

[8] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proc. Machine Learning Audio Signal Processing Workshop, Advances Neural Information Processing Systems*, Tokyo, Japan, 2017.

[9] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Feature learning and deep architectures: New directions for music informatics," *J. Intell. Inform. Syst.*, vol. 41, no. 3, pp. 461–481, 2013.

[10] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[11] J. Nam, J. Herrera, M. Slaney, and J. O. Smith, "Learning sparse feature representations for music annotation and retrieval," in *Proc. Int. Society for Music Information Retrieval Conf.*, Porto, Portugal, 2012, pp. 565–570.

[12] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks," in *Proc. Int. Society for Music Information Retrieval Conf.*, Utrecht, The Netherlands, 2010, pp. 339–344.

[13] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Florence, Italy, 2014, pp. 6964–6968.

[14] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proc. Int. Society for Music Information Retrieval Conf.*, New York, NY, 2016, pp. 805–811.

[15] J. Lee, J. Park, K. L. Kim, and J. Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," in *Proc. Sound and Music Computing Conf.*, Espoo, Finland, 2017, pp. 220–226.

[16] J. Lee, J. Park, K. L. Kim, and J. Nam, "SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification," *Appl. Sci.*, vol. 8, no. 1, p. 150, 2018.

[17] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, New Orleans, LA, 2017, pp. 2392–2396.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Honolulu, HI, 2016, pp. 770–778.

[19] J. Hu, L. Shen, and G. Sun. (2017) Squeeze-and-excitation networks. arXiv. [Online]. Available: https://arxiv.org/abs/1709.01507

[20] T. Kim, J. Lee, and J. Nam, "Sample-level CNN architectures for music auto-tagging using raw waveforms," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Aalborg, Denmark, 2018, pp. 366–370.

[21] R. Lu, K. Wu, Z. Duan, and C. Zhang, "Deep ranking: Triplet matchnet for music metric learning," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, New Orleans, LA, 2017, pp. 121–125.

[22] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *Proc. Int. Society for Music Information Retrieval Conf.*, Kobe, Japan, 2009, pp. 387–392.

[23] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *Proc. Int. Society for Music Information Retrieval Conf.*, Suzhou, China, 2017, pp. 316–323.

[24] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proc. Int. Society for Music Information Retrieval Conf.*, Suzhou, China, 2017, pp. 141–149.

[25] J.-Y. Liu and Y.-H. Yang, "Event localization in music auto-tagging," in *Proc. ACM on Multimedia Conf.*, Amsterdam, The Netherlands, 2016, pp. 1048–1057.

[26] J. Pons, O. Slizovskaia, R. Gong, E. Gómez, and X. Serra, "Timbre analysis of music audio signals with convolutional neural networks," in *Proc. European Signal Processing Conf.*, Kos, Greece, 2017, pp. 2744–2748.

[27] Y.-S. Huang, S.-Y. Chou, and Y.-H. Yang, "Music thumbnailing via neural attention modeling of music emotion," in *Proc. Asia Pacific Signal and Information Processing Assoc. Annu. Summit and Conf.*, Kuala Lumpur, Malaysia, 2017, pp. 347–350.

[28] L.-C. Yu, Y.-H. Yang, Y.-N. Hung, and Y.-A. Chen. (2017) Hit song prediction for pop music by Siamese CNN with ranking loss. arXiv. [Online]. Available: https://arxiv.org/abs/1710.10814

[29] S.-Y. Chou, J.-S. R. Jang, and Y.-H. Yang, "Learning to recognize transient sound events using attentional supervision," in *Proc. Int. Joint Conf. Artificial Intelligence*, Stockholm, Sweden, 2018, pp. 3336–3342.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[31] S. Ewert, B. Pardo, M. Mueller, and M. D. Plumbley, "Score-informed source separation for musical audio recordings: An overview," *IEEE Signal Process. Mag.*, vol. 31, no. 3, pp. 116–124, May 2014.

[32] J. Bosch, J. Janer, F. Fuhrmann, and P. Herrera, "A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals," in *Proc. Int. Society Music Information Retrieval Conf.*, Porto, Portugal, 2012, pp. 559–564.

[33] S. Oramas, O. Nieto, F. Barbieri, and X. Serra, "Multi-label music genre classification from audio, text and images using deep features," in *Proc. Int. Society for Music Information Retrieval Conf.*, Suzhou, China, 2017, pp. 23–30.

[34] S. Perez. (2018, Mar.). Pandora takes on Spotify with dozens of personalized playlists built using its Music Genome. TechCrunch. [Online]. Available: https://techcrunch.com/2018/03/28/pandora-takes-on-spotify-with-dozens-of-personalized-playlists-built-using-its-music-genome/

SP