Music Classification and Recommendation using Neural Networks

Mounik Patel (1144127)
MSc (Computer Science) Project
Lakehead University
mpatel78@lakeheadu.ca

I. INTRODUCTION

Computer Science is expanding to reach beyond human thinking in the real world with limitless technologies like Machine Learning and Deep Learning. These technologies make crucial and complex daily-life tasks such as observation, prediction, classification, and recommendation easier by developing and improving the algorithms and models for such tasks. These technologies are currently being used in many projects. I will discuss about Music Classification and Recommendation using Neural Networks in this survey paper.

Over the last decade, music-streaming services have grown dramatically. While some music service provider companies such as Pandora and Spotify have pioneered and popularized music streaming, other giant technology companies such as Apple and Google have also been strengthening their music service platforms [1]. Although music-streaming services have made a vast amount of music available to consumers, the sheer scale of the service catalogs has made it difficult to find the right song for the users among so many options [1]. While a traditional approach to this problem has been introduced, which involves a conventional machine learning framework, the performance of such models has not been satisfactory. As a result, models based on deep neural networks are now used to classify and recommend music based on the various characteristics on which it can be categorized, such as genre, mood, instruments, and so on. In this survey article, I will go over many of these models in depth.

II. BASIC CONCEPTS

In comparison to previous eras, commercial music streaming services that can be accessed from mobile devices have increased the availability of digital music. Owing to the large dataset of songs available on the internet today, users are finding it difficult to discover new music available to them. Many music streaming services began by using collaborative filtering to suggest songs to their customers based on previous usage data, such as play history and song rating [1]. While collaborative filtering effectively retrieves songs and accommodates personalized recommendations, issues like popularity bias and the difficulty of recommending new music to users hampered its efficiency [2].

The content-based filtering approach is often viewed as a complement to the collaborative filtering approach's problems. Songs are retrieved using a content-based filtering technique that takes advantage of song descriptor similarities such as genre, mood, instruments, and so on. However, manual annotation has proven to be expensive and inefficient, indicating the need for better ways to automate the classification of music [1].

Humans use different characteristics derived from audio signals to identify or annotate music. Metal music, for example, is characterized by highly distorted electric guitar sounds and growling vocals [1]. Music tags are a set of descriptive keywords that convey high-level information about a music clip, such as emotion (sad, anger, happy), genre (jazz, classical), and instrumentation (guitar, strings, vocal, instrumental). Tags can be used for music discovery and recommendation because they provide high-level knowledge from the listener's viewpoint [3]. The basis of music classification and tagging is translating these acoustic and musical characteristics into numerical representations that computers can understand. This usually includes translating audio content into a time-frequency representation, extracting discriminative features, summarizing them over time, and repeating the feature extraction and summarization until the correct category for the music is decided. With advances in learning algorithms, the method of improving each feature extraction phase to achieve the best output has evolved from hand engineering based on domain knowledge to end-to-end learning [1].

While reviewing the evolution of approaches for feature representation [4], Nam et al. [1] separated them into two classes: feature engineering and feature learning.

A. Feature engineering

Given the generative nature of music production, an intuitive approach to music classification and tagging will necessitate the distillation of features on each axis of musical elements and the simulation of their distributions [1]. Under this theory, the conventional approach sought to build a range of audio features. Tzanetakis and Cook used this approach to solve the problem of automatic music genre classification by combining three types of audio features: timbre, pitch, and rhythm [5].

- The timbre feature was formed by summarizing the zero-crossing rate, low-level spectral features, low-energy

feature, and Mel-frequency cepstral coefficients (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 for harmony and the other for pitch range.
- The rhythmic feature was defined by a beat histogram, which uses a sub band analysis to count intervals of periodic energy fluctuation to illustrate temporal regularity.

Finally, they merged all the features and used classifiers including k-nearest neighbors and Gaussian mixture models to classify them. Numerous research studies have created new or enhanced audio features and followed the two-stage framework, where hand-engineered features are used as input to a standard classifier, since this study laid the foundation for music classification and tagging. Each computation stage is manually designed using domain knowledge in this feature-engineering approach. Currently, this two-stage strategy appears to yield an unsatisfactory result [1].

B. Feature learning

The basic idea behind deep learning is that deep neural networks can be used to learn feature representations of input data. That is, learning is achieved layer by layer, with higher-level features learned in the deeper layers. This contrasts with the feature-engineering approach as the domain knowledge is much less involved in finding the features, and the input data is processed to a minimal level before being fed into the algorithm [1].

Various feature learning algorithms have been developed and applied to music classification and tagging as part of the deep learning. They are categorized into the following three classes [1].

- Low-level feature learning: Early studies focused on learning low-level audio features to replace the handcrafted features in the two-stage framework. While one kind of research focused on learning meaningful dictionary of spectrograms using unsupervised learning algorithms [6], 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 [7]. Although both studies outperform hand-engineered features in a variety of music classification and tagging tasks, they are confined to low-level feature learning, and the adopted framework still has two stages.
- Convolutional Neural Networks (CNNs): CNNs have recently become the most common learning model for music classification and tagging tasks [3], [8]. The spectrogram (particularly the mel-spectrogram) was used as the input representation in many effective CNN models, suggesting that domain knowledge is still useful.
- End-to-end learning models: A few attempts have been made in recent years to use raw waveforms as the input to CNNs [8], [9]. No single step in the model necessitates the development of a hand-designed representation, resulting in complete end-to-end feature learning.

III. CNNs FOR MUSIC SIGNAL ANALYSIS

Choi et al. [3] reviewed the properties of CNNs with respect to music signals. The development of CNNs was motivated by biological vision systems where information of local regions is repeatedly captured by many sensory cells and used to capture higher-level information [10]. CNNs are thus designed to teach robust features that react to specific visual objects with local, translation, and distortion invariances. These benefits also apply to audio signals as well, even though the topology of audio signals (or their 2D representations) differs from that of a visual image [3].

CNNs have been used in a variety of audio processing tasks, with the assumption that visualizing time-frequency representations of auditory events may help identify or recognize them. While deep learning has the benefit of learning features, the architecture of the networks should be carefully designed, considering the degree to which the properties are desired [3].

There have been several approaches of applying CNNs to analyze audio signals. The type of input representations, convolution axes, sizes and numbers of convolutional kernels or sub samplings, and the number of hidden layers differ among CNN architectures [3].

A. TF-representation

Mel-spectrograms have been a popular feature for tagging [11] and learning latent features [12]. When the fundamental frequencies of notes must be precisely defined, such as in chord recognition [13] and transcription [14], the Constant-Q transform (CQT) is commonly used. When an inverse transformation is needed, it is preferable to use Short-time Fourier Transform (STFT) coefficients directly [15], [16]. STFT's frequency resolution in the low frequency range is insufficient to define the fundamental frequency as compared to CQT. On the contrary, STFT provides finer resolutions than mel-spectrograms in frequency bands > 2kHz given the same number of spectral bands which may be desirable for some tasks. However, it has not been the most common choice so far [3].

Recent research has concentrated on learning an optimized transformation from raw audio given a task. These are known as end-to-end models, and they can be used for both music [8] and speech [17]. The performance is comparable to the mel-spectrogram in speech recognition [17].

B. Convolution – kernel sizes and axes

Each convolution layer of size H x W x D learns D features of H x W, where D, H and W refer to the depth, height, and width of the learned kernels, respectively. The maximum size of a component that the kernel can precisely capture is determined by the kernel size. The layer would fail to learn a realistic representation of the data's share (or distribution) if the kernel size were too small [3]. As a result, in [13], relatively large kernels such

as 17x5 are suggested. However, there are two disadvantages of using large kernels. First, it is well understood that as the size of the kernel increases, the number of parameters per representation capacity increases as well. 5x5 convolution, for example, can be replaced with two stacks of 3x3 convolutions, resulting in fewer parameters. Second, invariance is not allowed within the range of large kernels [3].

Another critical feature of convolution layer is the convolution axes. Dieleman and Schrauwen uses 1D convolution along the time axis to learn the temporal distribution for tagging, if different spectral bands have different distributions and hence features should be learned for each frequency band [8]. The global harmonic relationship is considered at the end of the convolution layers in this case, and fully connected layers are used to capture it. 2D convolution, on the other hand, can learn both temporal and spectral structures [3].

C. Pooling – sizes and axes

Pooling uses an operation, normally a max function, to minimize the size of the feature map. The vast majority of works that depend on CNN systems have embraced it. Pooling, in essence, uses subsampling to minimize the size of the function map while retaining the information of an activation in the region rather than the entire input signal [3].

By discarding the original position of the selected values, this non-linear behavior of sub sampling often provides distortion and translation invariances. As a result, pooling size determines the location variance tolerance within each layer and reflects a trade-off between two variables that affect network performance. The network will not have enough distortion invariance if the pooling size is too small, and if it is too large, the location of features will be missed when they are needed. In general, the pooling axes correspond to the convolution axes, although this is not always the case [3].

IV. DEEP-LEARNING MODELS

Three representative CNN models for music classification are reviewed by Nam et al. [1]. One-dimensional (1-D) [8] and two-dimensional (2-D) CNNs [3] are the first two versions, both of which have been used to make networks more flexible. The most recent and most popular technique, the sample-level CNN [9], uses a time-series audio signal as input, continuing the trend toward greater versatility.

A. 1-Dimensional Convolutional Neural Network (1-D CNN)

Dieleman et al. [8] pioneered some of the first deep learning advances in the field of music classification and tagging. Figure 1(a) depicts the network structure, which is referred to as a 1-D CNN. The word '1-D' here refers to the dimensionality of the first layer's convolution operation, which is not to be confused with the kernel's dimensionality [1].

1-D CNNs take a time-frequency representation, like Mel spectrogram, as input. The primary convolutional layer scans the whole frequency range directly with the kernel height of F. The network will find some patterns that span the entire frequency range during training. Nam et al. [1] observed the kernels using the learned weights and mark which genres (tags) maximally activate them since the convolutional layer's kernels work directly on the spectrogram input. The tags are sorted (in descending order) based on the tag activation score of every kernel. Then, it will be checked if the corresponding tag explains each of the learned kernels in any way.

Computationally, 1-D CNNs are very effective. The primary convolutional layer takes the entire frequency range as an input and reduces the number of network parameters by making the feature maps of the subsequent layers much smaller. It is easier to train the network with relatively small datasets and a small number of parameters, but because of the limited representation capacity, it will not benefit large-scale datasets. As a result, the representation learned by 1-D CNNs in the first layer is limited to a few common patterns across the entire frequency range [1].

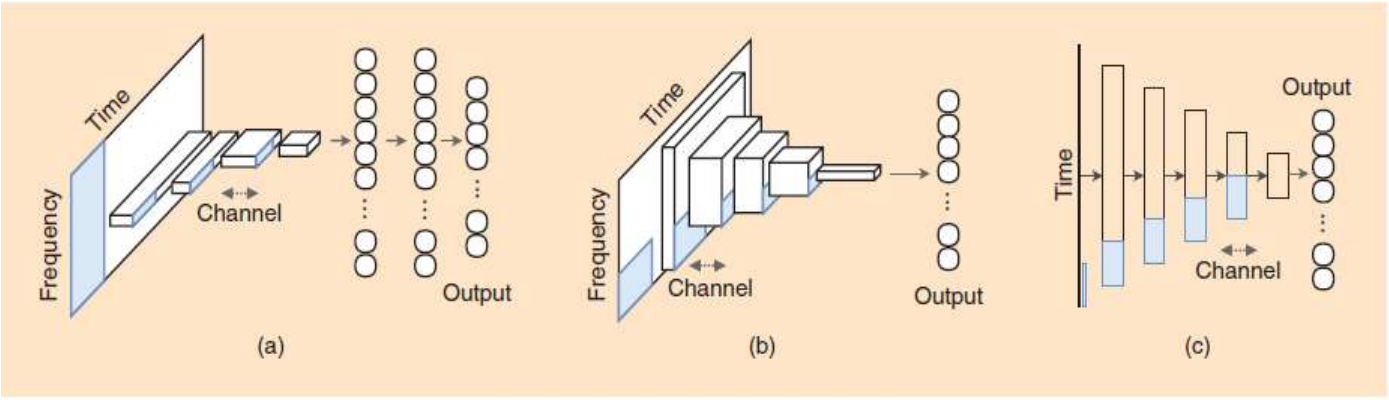

Figure 1. Block diagrams of (a) 1-D, (b) 2-D, and (c) sample-level CNNs. (a) and (b) are based on 2-D time-frequency representation inputs (e.g., mel-spectrograms or short-time Fourier transforms), and (c) is based on a time-series input [1].

B. 2-Dimensional Convolutional Neural Network (2-D CNN)

As larger datasets and better hardware resources become available, the network flexibility to enhance representation learning is a natural next step, as shown in [3]. Figure 1(b) depicts the network's structure. In comparison to 1-D CNNs, 2-D CNNs concentrate on the contiguous 2-D convolutional layers, including the first. 2-D CNNs allows more flexibility which may further help to find the time-frequency patterns. The flexibility can have various aspects: the shift-invariance along both axes (time-frequency), the dimensions of the patterns, and tiny distortions within the patterns. They are achieved by 2-D convolutional layers with small kernels and 2-D max-pooling layers [1].

Because of their simple structure and good performance compared to 1-D CNNs, 2-D CNNs may now be the foremost popular approach for music audio classification. 2-D CNNs usually demand better hardware than 1-D CNNs for two reasons: more parameters than 1-D CNNs because of the use of contiguous 2-D kernels (which requires more memory), and the training and application of 2-D CNNs add a significant computational burden because of the large size of the feature maps, along which the kernel should be convolved [1].

C. Sample-level Convolutional Neural Network

Since 2-D CNNs provide more flexibility, they can lead to better results in music classification and tagging, as discussed in the previous subsection. Sample-level CNNs (Figure 1(c)) take this a step further by discarding the 2-D time-frequency input preprocessing stage and learning directly from audio waveforms at a granular level. It effectively operates by utilizing the approach of 1-D CNNs within the subsequent convolutional layer, where each convolutional layer is known as a 2-D time-frequency representation [9].

The disadvantage of sample-level CNNs is their computation complexity. Training is accelerated by down sampling the waveform input [18], but more efficient and advanced models must be developed.

D. Advanced model - Convolutional Recurrent Neural Networks (CRNN)

The CRNN structure is a version of the CNN structure that replaces the final convolutional layers with recurrent layers [19]. The CRNN model assumes that recurrent layers are better at encoding long-term patterns than convolutional layers. This is most likely since the important patterns have a shorter length than the input duration. As a result, the patterns' temporal dynamics are a series of short-term patterns rather than a single long-term pattern. The use of recurrent layers also makes the model flexible with respect to the input length, which can be useful for music classification [1].

V. CRNNs FOR MUSIC CLASSIFICATION

Choi et al. [19] introduce a CRNN for music tagging. CRNN model is compared with two existing CNN models that have been used for music tagging while controlling the number of parameters with respect to their performance and training time per sample.

CNNs assume features that are in different levels of hierarchy and can be extracted by convolutional kernels. During supervised training, the hierarchical features are learned to complete a specific task. Recurrent neural networks (RNNs), which are commonly used to model temporal data such as audio signals or word sequences, have been combined with CNNs in this study. CRNN is the name given to this hybrid model. A CRNN is a modified CNN that uses an RNN to replace the last convolutional layers. CNNs and RNNs, respectively, play the roles of feature extractor and temporal summarizer in CRNNs. The use of an RNN for feature aggregation allows the networks to take the global structure into account while local features are extracted by the remaining convolutional layers [19].

A. Models

CRNN is compared with Conv1D and Conv2D, all of which are illustrated in Figure 2. Table 1 displays the specifications. The input size for all networks is 96x1366 (mel-frequency band x time frame). Since music tagging is a multi-label classification task, sigmoid functions are used as activation at output nodes [19].

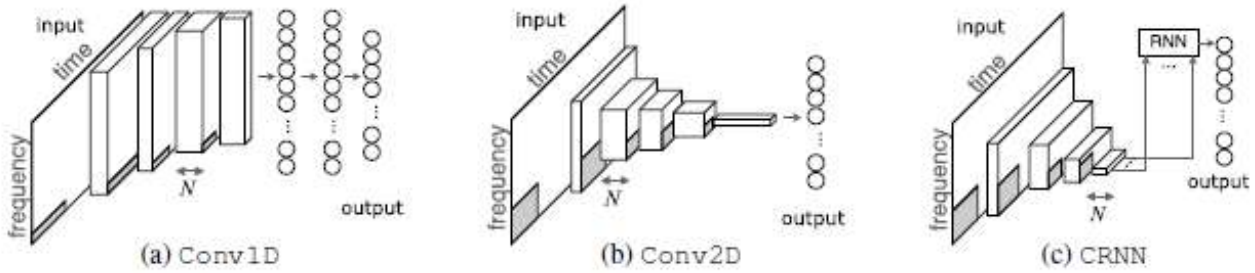

Figure 2. Block diagrams of Conv1D, Conv2D, and CRNN. The grey areas in (a), (b), and (c) illustrate 1D, 2D, and 2D convolution kernels, respectively. N refers to the number of feature maps of convolutional layers [19].

- Conv1D: The structures for music tagging [8] and genre classification [20] influenced Conv1D in Figure 2(a). The network is made up of 4 convolutional layers followed by 2 fully connected layers. 1-D convolutional layers (1x8 for all, i.e., convolution along time-axis) alternate with max-pooling layers ((1x4)-(1x5)-(1x8)-(1x8)) to produce narrow-and-tall feature maps. They are flattened and fed into a fully connected layer, which acts as the classifier [19].
- Conv2D: CNN structures with 2D convolution have been used in music tagging [3] and vocal/instrument classification [21]. As shown in Figure 2(b), Conv2D is made up of 5 convolutional layers with 3x3 kernels and max-pooling layers ((2x4)-(2x4)-(3x5)-(4x4)). At the final layer, the network reduces the size of feature maps to 1x1, with each feature covering the entire input rather than each frequency band. During the size reduction, temporal data is also progressively aggregated [19].
- CRNN: As shown in Figure 2(c), CRNN uses a 2-layer RNN with gated recurrent units (GRU) [22] to summarize temporal patterns on top of 2D 4-layer CNNs. The sizes of convolutional layers and max-pooling layers in its CNN sub-structure are 3x3 and ((2x2)-(3x3)-(4x4)-(4x4)), respectively. The size of the function map because of this sub-sampling is Nx1x15 (number of feature maps x frequency x time). They are then fed into a 2-layer RNN, with the last hidden state connected to the network's output [19].
- Scaling networks: The models are scaled by setting the number of parameters to 0.1 million, 0.25 million, 0.5 million, 1 million, and 3 million with a 2% tolerance. The descriptions of various structures, including the layer width (the number of feature maps or hidden units), are summarized in Table 1. The number of parameters in a network is regulated by changing the layer widths, while the depths and convolutional kernel shapes remain constant [19]. To optimize the representation capabilities of networks, the hierarchy of learned features is maintained while the numbers of features in each hierarchical layer are modified [23].

Conv1D						**Conv2D**						**CRNN**					
No. params ($\times 10^6$)	0.1	0.25	0.5	1.0	3.0		0.1	0.25	0.5	1.0	3.0		0.1	0.25	0.5	1.0	3.0
Layer type	Layer width					Type	Layer width					Type	Layer width				
conv1d	15	23	33	47	81	conv2d	20	33	47	67	118	conv2d	30	48	68	96	169
conv1d	15	23	33	47	81	conv2d	41	66	95	135	236	conv2d	60	96	137	195	339
conv1d	30	47	66	95	163	conv2d	41	66	95	135	236	conv2d	60	96	137	195	339
conv1d	30	47	66	95	163	conv2d	62	100	142	203	355	conv2d	60	96	137	195	339
fully-connected	30	47	66	95	163	conv2d	83	133	190	271	473	rnn	30	48	68	96	169
fully-connected	30	47	66	95	163							rnn	30	48	68	96	169
AUC score	.772	.781	.795	.806	.829		.799	.814	.821	828	.857		.823	.838	.842	.851	.855
Time/2.5k samples [s]	55	80	111	146	218		46	64	82	116	180		70	103	166	256	402

Table 1. Hyperparameters, results, and time consumptions of all structures. Number of parameters indicates the total number of trainable parameters in the structure. Layer width indicates either the number of feature maps of a convolutional layer or number of hidden units of fully connected / RNN layers. Max-pooling is applied after every row of convolutional layers [19].

B. Experiments

Choi et al. [19] used the Million Song Dataset [24] with last.fm tags. The networks were trained to predict the top-50 tag, which includes genres (e.g., rock, pop), moods (e.g., sad, happy), instruments (e.g., female vocalist, guitar), and eras (60s-00s). Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [3], [8]. Given that tagging is a multi-label classification, the recorded performance is measured on the test and by AUC-ROC (Area Under Receiver Operating Characteristic Curve).

- Memory-controlled experiment: The AUC score for each network is plotted against the number of parameters in Table 1. Except for 3M-parameter structures, AUC ranks CRNN > Conv2D > Conv1D for the same number of parameters. This suggests that CRNN should be used when memory usage is a bottleneck. CRNN outperforms Conv2D in all cases except when using 3M parameters. Because of its deeper RNN structure, CRNN has a slower convergence rate than Conv2D with 3M parameters [3]. CRNN has a lower AUC than Conv2D because of this. Conv2D shows higher AUCs than Conv1D in all cases [19].
- Computation-controlled comparison: The computational complexity is proportional to the time it takes to train and predict, and it varies depending on the number of parameters as well as the structure. CRNN has the best performance for training times under 150s with similar training time. Conv2D, on the other hand, outperforms CRNN with similar or longer training times (0.5M, 1M, and 3M parameters), with a training period of 180s (3M parameters). This means that depending on the target time budget, either Conv2D or CRNN may be used. The order of training speed is always Conv2D > Conv1D > CRNN when the number of parameters is the same. In other

words, network depth is negatively correlated with network speed (5, 6, and 20, respectively) [19].

- Performance per tag: Each tag is assigned to one of four categories: genres, moods, instruments, and eras, and is then sorted by AUC within each category. The music tagging task can be categorized as a multiple-task problem with these four categories, like four classification tasks. In every tag, the CRNN outperforms Conv2D and Conv1D, with Conv2D outperforming Conv1D in 47 of 50. From the perspective of multiple-task classification, this finding suggests that a structure that outperforms in one of the four tasks can also outperform in the others [19].

C. Conclusions

With many parameters, the above experiments showed that Conv2D and CRNN work similarly. Overall, it is discovered that CRNNs perform well in terms of parameter number and training time, suggesting the efficacy of their hybrid structure in music feature extraction and feature summarization [19].

VI. CRNNs FOR MUSIC RECOMMENDATION

Music recommendations are made by comparing one piece of music to another or by assigning a preference to one individual over another [12]. The aim of a music recommender system is to develop a system that can constantly find appealing new music while also understanding the users' musical tastes [25].

Adiyansjah et al. created a music recommender framework that can provide users with recommendations based on the similarity of features on audio signals. This system's approach can be defined as content-based music recommendation, in which the recommendations are based on perceptual similarity to what the user has previously heard [26]. This method necessitates the definition of a similarity metric, which is used to compare audio signals [12]. This study uses CRNNs for feature extraction and similarity distance to assess feature similarity.

A. Model

The research steps are briefly discussed in this section, with a focus on the architecture of CRNNs.

- Collecting music data: Data is gathered by downloading datasets that are publicly accessible. The music library of the Free Music Archive (FMA) provided the dataset [27]. This study uses a part of the FMA dataset by choosing the good quality records and maintaining the balance of data for each genre [26].
- Data cleaning and selection: First, the data were cleaned up by removing music with poor audio quality and music with the wrong genre label. Since music by the same creator has the same audio quality in the dataset, the cleaning process is made simple by sorting the music by creator. Music that combined two or more genres was also eliminated. Finally, when retaining data balance, seven genres were chosen: classical, electronic, folk, hip-hop, instrumental, jazz, and rock [26].
- Audio preprocessing: Data in the form of an audio signal is transformed into a spectrogram image because a suitable audio representation is needed as input for neural network architecture. To begin this conversion, the audio signal is first sampled at a specific rate. The frequency of the audio signal could then be measured at a given time using STFT at each sampling rate with a specific window length and hop length. A window function is used to smooth the frequency and prevent spectral leakage [26]. Finally, the STFT results are converted to Mel-scale because Mel-spectrograms are the most effective representation among other audio representations [3].
- Modeling neural networks: CNNs and CRNNs architecture are used to distinguish music genres. Since CRNNs can extract important features for prediction results as well as look at time sequence patterns, they are used. Finally, feature vectors produced prior to the classification layer can be used to make recommendations. The input dimensions of Mel-spectrogram image are 96x1366 for both CNNs and CRNNs architectures [26].
  - CNN architecture: With each layer's dimensional change, the CNN structure is retained. To preserve input dimensions, five layers of convolutional with kernel 3x3, feature maps (47-95-95-142-190), stride 1, and padding are used. Each convolutional is subjected to batch normalization and ReLU activation. With kernel ((2x4)-(2x4)-(3x5)-(4x4)) and same stride, max-pooling layers are used. As an output layer, the sigmoid function is used [26].
  - CRNN architecture: Two layers of RNN with GRU are used in the CRNNs architecture to summarize 2D temporal patterns from the effects of four CNN layers. CNNs was first performed on this model, which were used to extract local features. Sub-sampling produces feature maps with a large Nx1x15 (number of feature maps x frequency x time) that will be used for two RNN layers. The CRNN structure is preserved as the dimensions of each layer change. Four layers of convolutional with kernel 3x3, function maps (68-137-137-137), stride 1, and padding are used to preserve input dimensions. Batch normalization and ReLU activation are applied to each convolutional. Max-pooling layers are used for each convolutional layer with kernel ((2x2)-(3x3)-(4x4)-(4x4)) and same stride. Each convolutional layer employs a 0.1 dropout rate. Two layers of GRU with 68 feature maps are used, with the first layer having both input and output data in the form of sequences, and the second layer having input data in the form of a sequence and output data in the form of a single value. The sigmoid function is used as an output layer [26].

B. Implementation

To incorporate the recommender system, two approaches are used. The first approach solely relies on the cosine similarity value. Calculating the cosine similarity of extraction features (Equation 1) from one music to another is how the recommender system works. Since the extraction features are in vector form, their distance can be calculated. To begin, a piece of music is chosen for each genre to serve as the foundation for the recommender system. Following that, the neural networks are used to determine the basis music genre prediction. The feature vectors created prior to the classification layer are used as the foundation for recommendations. Cosine similarity calculations are conducted on other music features after the basis music features have been obtained. Here is the formula to measure cosine similarity between two pieces of music with several features equal to N, where the first piece of music has a feature vector x = [x1,x2,...xn] and the second piece of music has a feature vector y = [y1,y2,...,yn] [26]:

$$cos\,\theta = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

Equation 1. Formula for cosine similarity [26]

The second method makes use of both the cosine similarity value and music genre information. The user-selected music is used as the starting point for recommendations. The extraction vector for basis music is obtained from the best genre prediction model in the previous stage. The cosine similarity values are then sorted in descending order. Finally, the first five pieces of music with the highest value are recommended. The music recommender application is written in Python and incorporates several main libraries such as Tensorflow, Keras, Librosa, and Kivy. Users can switch to the 'Get relevant' button after clicking the corresponding three-dots button to the right of the music title to get recommendations from the currently playing music [26].

C. Experiment results

- Evaluation of Music Genre Classification Models:
  - Receiver Operator Characteristics (ROC) are commonly used to assess binary classification performance [26]. Precision-recall (PR) can provide more details about algorithm performance when dealing with imbalance datasets or skewed data [28].
  - The precision values indicate how accurately the model predicts the true music genre. The CRNNs outperforms the CNNs in classical and electronic music genres, while the CNNs outperforms the CRNNs in hip-hop music genres. The CNNs is slightly better than the CRNNs for other musical genres. When the true music genre is given, recall values define how often the model evokes the true conditions. The CRNNs outperforms in electronic, folk, and hip-hop genres. While the CNNs is superior to classical and instrumental music genres. . The CNNs outperforms the CRNNs for other music genres. In general, CRNNs perform better, suggesting that its hybrid structure is successful at extracting music features [26].
  - The true positive rate and false positive rate along with classification threshold of 0.5, are used to further explain the experiment results. The true positive rate is the same as the recall value, while the false positive rate describes how many negative music genres are mistakenly classified as positive. In classical, electronic, and instrumental styles, the CRNNs performs better. The CNNs are superior in the pop, hip-hop, and rock genres. The values of the two methods are identical in the jazz music genre. CRNNs outperform CNNs once more in this case [26].
  - ROC curves were used at all possible thresholds to summarize the classifier output. The true positive rate and false positive rate are used to create the ROC curve. F1 scores can also be used to evaluate the model's output by integrating the precision and recall values. Overall, the CRNNs model, which considers both frequency features and time sequence patterns, performs better. Its performances are like the CNNs in several genres [26].
- Evaluation of Music Recommender System:
  - Adiyansjah et al. [26] perform a listening experiment [29] with 30 participants to test the music recommender system. The experiments are focused on user reactions to the music recommendations that have been given. The recommender system used two methods in our experiments: the first method only uses the value of cosine similarity, while the second method uses both the value of cosine similarity and music genre information. The experiment's procedure is as follows: participants choose five of their preferred music, after which the music recommender system gives five recommendations. The participants then rate the music recommender system by indicating whether they like or dislike each recommendation. Finally, a statistical procedure is used to determine the significance of the differences in responses to the two methods among respondents [26].
  - The second method's average value is significantly better than the first method's. When looking at the user's response based on the sequence of music recommendations, the first rank recommendation is not favored by the user as a recommendation, even though the music in the other ranks is. As a result, it can be inferred that the five music recommendations are independent of one another. Furthermore, the second approach, which considers music genre information, is superior to the first [26].

D. Conclusions

- To improve the quality of music recommendations, music recommender systems should consider music genre information [26].
- Overall, CRNNs that consider both frequency features and time sequence patterns perform better. It shows how good its hybrid structure is at extracting music features [26].

VII. BENCHMARK DATASETS AND EVALUATION TASKS

This section describes four widely used public datasets for music classification and tagging (Table 2). The availability of large-scale public datasets, which are used not only for training deep-learning models but also for benchmark evaluation, is one of the most important factors in the success of deep learning [1].

A. GTZAN Dataset

Despite its limited size, the GTZAN dataset (named after George Tzanetakis, the dataset's creator) is one of the most used for music genre classification. It includes 1000 30-second audio clips (ten genres and 100 songs for each genre). For training, validation, and testing, the most recent version uses an artist-stratified split of 443, 197, and 290 audio clips, respectively, with no repeat artists in these sets [5].

B. MagnaTagATune Dataset

MagnaTagATune (MTAT) is one of the most used music auto tagging benchmark datasets. It is a multilabel music classification task that randomly annotates genre, mood, instruments, and other song details. Tags and similarity annotations are used in the dataset. The auto tagging benchmark has been done for various numbers of tags, with the 50-tag version currently being the most benchmarked. A standard practice is to use the first 12 partitions of the dataset for training, the 13th for validation, and the remaining three for testing. There are 25,863 30-second audio clips in this dataset [30].

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

Table 2. Selected benchmark datasets for music classification and recommendation [1].

C. Million Song Dataset

The Million Song Dataset (MSD) is a cluster of complementary datasets compiled by the Music Information Retrieval (MIR) community. Without access to the original audio, the original MSD includes artist-level metadata as well as the Echo Nest (hand-engineered) audio features. Other metadata, such as song-level tags, similarity, lyrics, cover songs, user listening history, and genre labels, have been added to the MSD by matching the identification data (IDs) [24].

D. Free Music Archive Dataset

The Free Music Archive (FMA) is the most recently published large-scale dataset under the Creative Commons license. The dataset includes track-level, album-level, and artist-level metadata, as well as genres, listen counts, and tags. It is primarily intended for genre classification, with four benchmarking subsets: small, medium, large, and full. The small and medium subsets are used to categorize single-label genres, while the large and full subsets are used to categorize multilabel genres [27].

E. Evaluation tasks

Nam et al. [1] divided the problems into two categories: multiclass (genre or mood classification) and multilabel (auto-tagging). The accuracy score in the multiclass task is used to evaluate the models. Within the multilabel task, the predictions are assumed to be independent binary outputs. Each of these outputs is assessed in terms of annotation as well as retrieval (or ranking). Precision, recall, and F score are important annotation metrics. They are computed for every word label and are averaged. The area under the receiver–operator curve (AUC), the mean average precision, and the precision at (or up to) rank K (P@K) are all retrieval metrics [1].

VIII. OTHER APPLICATIONS

As shown in Figure 3, a neural network model trained on larger scale labeled data for music classification and tagging can be applied to other tasks like classification across datasets, making recommendations, thumbnailing music, and predicting hit songs [1].

Pretrained models using large-scale labeled data may provide a reasonable approximation of audio content similarities, according to the findings. As a result, classifiers can be created for problems with sparsely labeled data on top of such pretrained models using so-called transfer-learning techniques. Pretrained models can also contribute to addressing the challenge of making content-based music recommendations [1].

Adding content filtering by pretrained models for music classification and tagging to the strictly collaborative filtering

methods helps ensure acoustic consistency in the recommended list of music, which improves the user experience. The (acoustic) diversity of the recommended music can also be controlled using content filtering [2].

Pretrained models can also be used for music thumbnailing, which is the detection of a song's highlight, and audio-based hit-song prediction [1].

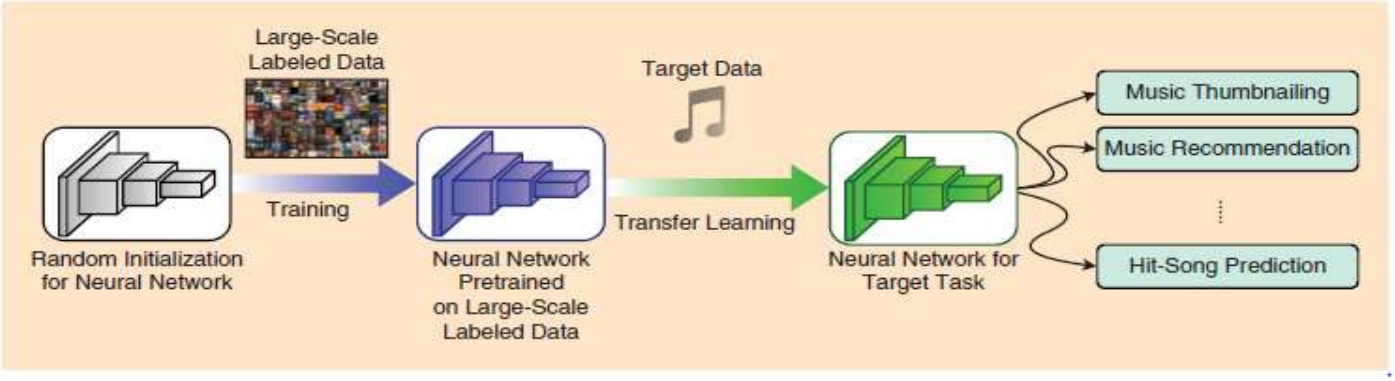

Figure 3. Transferring the knowledge of a neural network pretrained on larger scale labeled data to other music applications [1].

IX. LIMITATIONS AND FUTURE CHALLENGES

Nam et al. [1] has addressed some of the main shortcomings of current approaches for music classification and tagging, as well as some possible research directions in this section.

A. Share of audio data

The issue of copyright infringement could stifle widespread music classification and tagging research. The audio files from the datasets cannot be publicly distributed. Sharing precomputed features instead of audio files, providing a list of IDs from which people can find audio previews on the web, or using copyright-free music are all common ways to get around this issue [1].

B. Musically meaningful network design

Deep neural network developers are expected to incorporate more musical characteristics into their designs. A piece of music is normally made up of many components, such as melody, chords, percussion, and baseline, all of which are often played by various instruments [31]. Most neural networks for music classification, on the other hand, process audio inputs, without distinguishing between the various sound sources. Though deep-learning approaches have generated state-of-the-art results in sound-source separation and music classification, there has been little work done to tackle the two problems together in a unified network. While conducting feature learning, requiring the neural network to learn to isolate the music sources that compose an audio mixture may be an important future direction [1].

C. Vocabulary and personalization

It is also possible to expand the range of labels considered in classification and tagging models. Our machines' vocabulary for representing music must also be extended and modified to deal with the complexity of natural language. Furthermore, it is well recognized that the associations between music and such types of labels, such as moods (e.g., happy, sad, aggressive, relaxing) and usages (e.g., exercise, reading), are subjective. As a result, computational modelling them is more complicated. However, such labels are necessary if there is a need to build playlists that are tailored to a user's mood or behavior. The label assignment should be customized, considering the listener's preferences as well as the listener's 'personal understanding' of such labels [1].

D. Cross-modality approach

A lot of cross-modality research is seen in the neighboring field of computer vision, which aims to combine the visual world with the textual world. Similar efforts are expected to thrive in the MIR community, not only for classification and tagging tasks, but also for generative tasks like tag-conditioned lyrics generation, album cover generation, and music video generation. Sharing pre-trained models or expertise may be helpful in facilitating research on these tasks [1].

X. CONCLUSION

With the proliferation of music-streaming platforms, categorizing and recommending music to consumers has become a challenge. I learned how to engineer and learn features from music. I learned about CNNs and CRNNs, two essential deep learning architectures for music classification and recommendation. I learned about different types of deep-learning models that are used in music classification and recommendation. For classifying and recommending music, two experiments using CNNs and CRNNs were learned. I became acquainted with a variety of benchmark datasets for deep-learning tasks. Finally, I learned about various applications and limitations of deep-learning models used for music classification and recommendation.

XI. REFERENCES

[1] J. Nam, K. Choi, J. Lee, S. Chou, and Y. Yang, "Deep Learning for Audio-Based Music Classification and Tagging: Teaching Computers to Distinguish Rock from Bach," IEEE

Sig. Proc. Mag., vol. 36, no. 1, pp. 41-51, Jan. 2019.

[2] S. Chou, L. Yang, Y. Yang and J. R. Jang, "Conditional preference nets for user and item cold start problems in music recommendation," IEEE Int. Con. Multimedia and Expo (ICME), pp. 1147-1152, Jul. 2017.

[3] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," Int. Society for Music Info. Retrieval Conf. (ISMIR), pp. 805-811, Aug. 2016.

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

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

[6] J. Nam, J. Herrera, M. Slaney, and J. O. Smith, "Learning sparse feature representations for music annotation and retrieval," Proc. Int. Society for Music Info. Retrieval Conf. (ISMIR), pp. 565–570, Jan. 2012.

[7] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks," Proc. Int. Society for Music Info. Retrieval Conf., pp. 339–344, Jan. 2010.

[8] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," Proc. IEEE Int. Conf. Acoustics, Speech, and Sig. Proc. (ICASSP), pp. 6964–6968, May 2014.

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

[10] Y. LeCun, and Y. Bengio, "Convolutional networks for images, speech, and time series," The Handbook of Brain Theory and Neural Networks, vol. 10, pp. 3361, 1995.

[11] S. Dieleman, and B. Schrauwen, "Multiscale Approaches to Music Audio Feature Learning," Proc. Int. Society for Music Info. Retrieval Conf. (ISMIR), pp. 3-8, Nov. 2013.

[12] A. Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," Advances in Neural Information Processing Systems, Jan. 2013.

[13] E. J. Humphrey, and J. P. Bello, "Rethinking Automatic Chord Recognition with Convolutional Neural Networks," Int. Conf. on Machine Learning and App. (ICMLA), vol. 2, pp. 357-362, Dec. 2012.

[14] S. Sigtia, E. Benetos, and S. Dixon, "An End-to-End Neural Network for Polyphonic Music Transcription," IEEE/ACM Transactions on Audio, Speech, and Lang Proc., vol. 24, May 2016.

[15] K. Choi, G. Fazekas, M. Sandler, and J. Kim, "Auralisation of deep convolutional neural networks: Listening to learned features," Proc. Int. Society for Music Info. Retrieval Conf. (ISMIR), Oct. 2015.

[16] A. Simpson, G. Roma, and M. Plumbley, "Deep Karaoke: Extracting Vocals from Musical Mixtures Using a Convolutional Deep Neural Network," Aug. 2015.

[17] T. Sainath, R.J. Weiss, A. Senior, K. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform CLDNNs," INTERSPEECH, 2015.

[18] J. Lee, J. Park, K.L. Kim, J. Nam, "SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification," Appl. Sci. vol. 8, pgs. 150, Jan. 2018.

[19] K. Choi, G. Fazekas, M. Sandler and K. Cho, "Convolutional recurrent neural networks for music classification," IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP), pp. 2392-2396, Mar. 2017.

[20] J. Wülfing, and A.R. Martin, "Unsupervised Learning of Local Features for Music Classification," Proc. Int. Society for Music Info. Retrieval Conf. (ISMIR), Jan. 2012.

[21] J. Schlüter, "Learning to Pinpoint Singing Voice from Weakly Labeled Examples," Proc. Int. Society for Music Info. Retrieval Conf. (ISMIR), pp. 44-50, 2016.

[22] K. Cho, B. Merriënboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," Sep. 2014.

[23] R. Eldan, and O. Shamir, "The Power of Depth for Feedforward Neural Networks," Conference on Learning Theory, Dec. 2015.

[24] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset,". Proc. Int. Society for Music Info. Retrieval Conf. (ISMIR), pp. 591-596, Jan. 2011.

[25] J. O'Bryant, "A survey of music recommendation and possible improvements," 2017.

[26] Adiyansjah, A. Gunawan and D. Suhartono, "Music Recommender System Based on Genre using Convolutional Recurrent Neural Networks," Procedia Comp. Science, vol. 157, pp. 99-109, Sep. 2019.

[27] M. Defferrard, K. Benz, P. Vandergheynst, and X. Bresson, "FMA: A Dataset for Music Analysis," Proc. Int. Society for Music Info. Retrieval Conf. (ISMIR), Dec. 2016.

[28] J. Davis, and M. Goadrich, "The Relationship Between Precision-Recall and ROC Curves," Proc. Int. Conf. on Machine Learning (ICML), vol. 6, pp. 233-240, Jun. 2016.

[29] D. Bogdanov, and P. Herrera, "How Much Metadata Do We Need in Music Recommendation? A Subjective Evaluation Using Preference Sets," Proc. Int. Society for Music Info. Retrieval Conf. (ISMIR), pp. 97-102, Jan. 2011.

[30] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," Proc. Int. Society for Music Info. Retrieval Conf. (ISMIR), pp. 387–392, Jan. 2009.

[31] S.Y. Chou, J.S.R. Jang, and Y.H. Yang, "Learning to recognize transient sound events using attentional supervision," Proc. Int. Joint Conf. Artificial Intelligence (IJCAI), pp. 3336–3342, Jul. 2018.