Music Classification using an Improved CRNN with Multi-Directional Spatial Dependencies in Both Time and Frequency Dimensions

Zhen Wang*✉, Suresh Muknahallipatna†, Maohong Fan‡, Austin Okray*, Chao Lan*

*Department of Computer Science, University of Wyoming
†Department of Electrical and Computer Engineering, University of Wyoming
‡Department of Chemical Engineering, University of Wyoming
Emails: zwang10@uwyo.edu, sureshm@uwyo.edu, mfan@uwyo.edu, aokray@uwyo.edu, clan@uwyo.edu

***Abstract*—In music classification tasks, Convolutional Recurrent Neural Network (CRNN) has achieved state-of-the-art performance on several data sets. However, the current CRNN technique only uses RNN to extract spatial dependency of music signal in its time dimension but not its frequency dimension. We hypothesize the latter can be additionally exploited to improve classification performance. In this paper, we propose an improved technique called CRNN in Time and Frequency dimensions (CRNN-TF), which captures spatial dependencies of music signal in both time and frequency dimensions in multiple directions. Experimental studies on three real-world music data sets show that CRNN-TF consistently outperforms CRNN and several other state-of-the-art deep learning-based music classifiers. Our results also suggest CRNN-TF is transferable on small music data sets via the fine-tuning technique.**

***Index Terms*—Music Auto-Tagging, Genre Classification, Deep Learning, Convolutional Recurrent Neural Network, Multi-Directional Spatial Dependency**

I. Introduction

Music classification is a fundamental and important task in music data analysis. Traditional methods first extract hand-crafted music signal features and develop classifiers based on them [1]. In recent years, there has been an increasing interest in applying deep learning to extract deeper signal features for developing classifiers, and numerous studies have shown this outperforms traditional classification methods [2], [9], [10]. A successfully applied deep learning technique is the Convolutional Recurrent Neural Network (CRNN) [9]. It first applies a Convolutional Neural Network (CNN) to extract local features from the music signal spectrum, and then applies a Recurrent Network Network (RNN) to extract the temporal dependency of the music signal.

Although CRNN has achieved state-of-the-art performance on several music data sets, there remains space for improvement. In particular, the current technique only applies RNN to capture spatial dependency of the music signal in its time dimension, while ignoring the spatial dependency in its audio frequency dimension. For better clarity, we show an example output of the CNN component in CRNN in Figure 1. In the standard technique (left), CNN outputs an activation map of size (1, 15, 32), where the first dimension is audio signal frequency and the second is time; then RNN is applied on this map to extract temporal dependency of the signal. It is clear this RNN cannot extract dependency in the frequency dimension because the map only contains one frequency.

We hypothesize that using RNN to additionally capture spatial dependency in the music frequency dimension can improve performance of the current CRNN technique. To this end, we propose CRNN-TF, an improved technique that extracts spatial dependencies in both Time and Frequency dimensions of the music signal in multiple directions. Specifically, we first propose a modified CNN component with different kernel matrices and pool strides so it can output an activation map of mid-level time-frequency representation shown in Figure 1 (right). Then, we propose a novel multi-directional scanning strategy to convert the activation map into eight sequences, and feed each sequence into one grid LSTM block in the RNN layer; the outputs of all LSTM blocks are concatenated to form a high-level feature vector of the music signal for developing a classifier at the fully connected layer. Experimental studies on three real-world data sets demonstrate that CRNN-TF consistently outperforms several state-of-the-art deep learning-based music classifiers (including CRNN) on two music classification tasks: music genre classification and music tagging. Our results also suggest the CRNN-TF parameters can be transferred from a large music data set to a small music data set via the fine-tuning technique.

The remainder of this paper is structured as follows: Section 2 reviews the related works; Section 3 presents our proposed CRNN-TF technique; Section 4 presents experimental results, and conclusions are drawn in Section 5.

II. Related Works

A. Deep Learning for Music Data Analysis

Deep learning techniques are being extensively applied in music classification tasks (e.g., [12], [23], [24]). Hamel and Eck [24] applies Deep Belief Network to music auto-tagging and genre classification; Lee et al. [23] applies Convolutional Deep Belief Network to music genre and artist classification; Dieleman and Schrauwen [11] and Choi et al. [10] applies Convolutional Neural Network (CNN) for music auto-tagging;

978-1-7281-2009-6/$31.00 paper N-20443.pdf

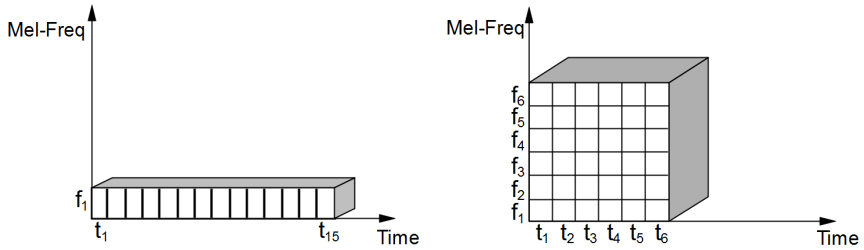

Fig. 1: Activation maps of CRNN (left) and CRNN-TF (right).

Lee et al. [12] develops a novel CNN to directly learn features from raw music audio signal.

CRNN is a promising deep learning technique that has achieved state-of-the-art performance on several music data analysis tasks including music classification [9], polyphonic sound event detection [4] and music transcription [22]. However there remains space for improvement. In this paper, we propose to improve CRNN by using additional RNNs to extract spatial dependency in the audio frequency dimension, and we show it improves classification performance.

B. Multi-Directional Spatial Dependency

In [5], Zuo et al. designs a CRNN where CNN outputs activation maps containing middle-level feature information extracted from raw pixel images. They propose a quad-directional scanning technique which converts the map into four sequences and feeds each sequence into an RNN. Zuo's work verifies the effectiveness of the multi-directional scanning strategy, but it focuses on image data and only considers four scanning orders. However, it remained unclear whether the strategy would work on music data, which is fundamentally different from image data [3]. In this paper, we verify the effectiveness of multi-directional scanning on music data; we also propose to scan music audio spectrum in eight orders and show it results in improved performance compared with the four scanning orders used in Zuo's work.

C. Deep Transfer Learning

Deep transfer learning has been applied to music classification. For example, Shtern et al. [6] apply a CNN trained on image data to extract features of music data for genre classification, and show this achieves high classification accuracy. Kim et al. [7] transfer information across different artist group factors to improve genre classification. Following the transfer scheme in [20], we propose to transfer parameters of our proposed CRNN-TF from a source task to a target task, and show it improves the latter's classification performance.

D. Spectrogram Representation of Music Audio Signal

There are many ways to represent music audio signal, depending on the application. In this paper, for fair comparison with [9]–[11], [14], we choose the log-amplitude mel-spectrogram (LAMS) representation. LAMS is one type of spectral representation; the latter is widely used in music data analysis such as automatic tagging [9], onset detection [25] and feature learning [26], because it has lower dimension than raw audio signal and can retain more information than traditional hand-crafted features (e.g. features constructed from audio signals like beat frequency, pitch, etc.). More specifically, LAMS is a scaled mel-spectrogram, which represents the short-term power spectrum of an audio signal and captures low level details along several dimensions such as pitch, amplitude and timbre; LAMS is obtained by taking logarithm (with base 10) on the magnitude of mel-spectrogram. Finally, LAMS is a matrix with one dimension representing time and another dimension representing mel-frequency; many studies have treated this as an efficient and perceptually relevant representation of the original signal and feed it into CNN for various learning tasks [4], [7], [9]–[11], [14], [16], [27].

III. The Proposed CRNN-TF Network

In this section, we present the proposed CRNN-TF network. Its architecture is shown in Figure 2. It has four components: preprocessor, CNN, multi-directional RNN and fully-connected layer. Each component is elaborated below.

A. The Preprocessor

The preprocessor transforms a raw music signal into a log-amplitude mel-spectrogram and feeds it into CNN. We construct it in the same way as the one used in [9]. Specifically, the transformation is computed as follows. Considering different songs have different lengths, we first trim the audio signal of each song into 29 seconds. The parameters of the transformation are set as follows: down-sampling rate is set to $12kHz$, hop size is 256, 512-point FFT is chosen, and the number of mel-bins is 96. These setups result in a 96-by-1360 matrix of mel-spectrogram, where each row corresponds to one mel-frequency scale and each column corresponds to one time frame. [1].

B. The CNN Network

The CNN network learns multiple spectro-temporal descriptors from the mel-spectrograms and outputs their activation maps. Compared with the CNN used in [9], our CNN component also has 4 layers but uses different convolutional and pooling operators in order to obtain multiple mel-frequencies in the activation map. In our CNN, each layer contains a set

[1]In experiment we implement the mel-spectrogram transformation using the Librosa library with ffmpeg in Python [21]

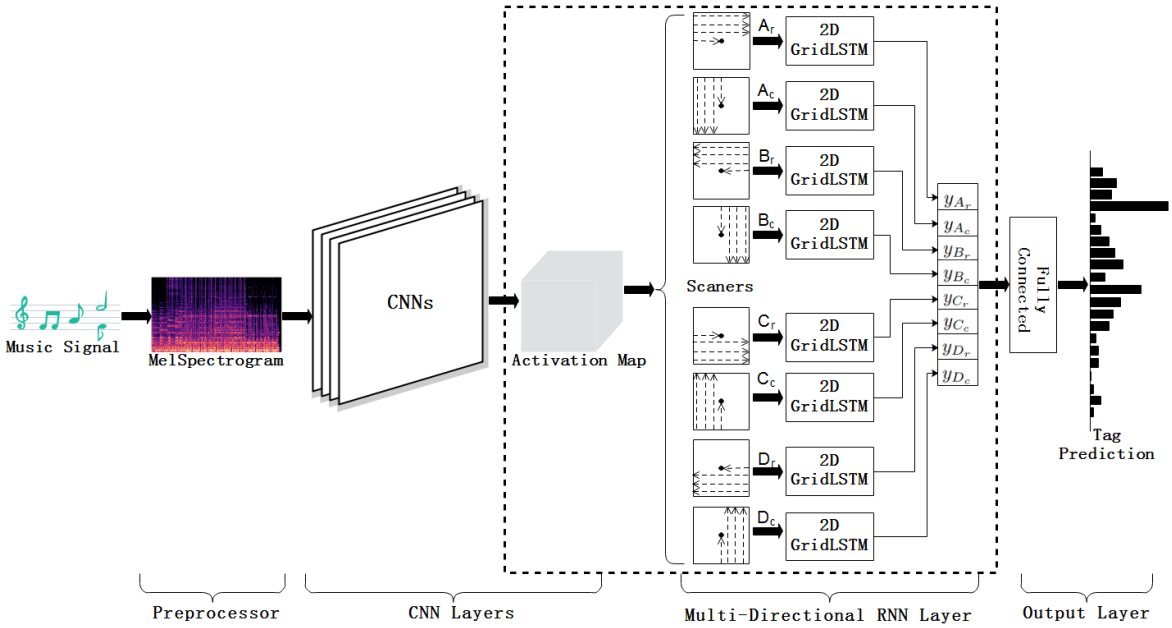

Fig. 2: Architecture of the Proposed CRNN-TF for Music Classification. The dashed frame is proposed to capture the multi-directional spatial dependencies in time and frequency dimensions.

of 3-by-3 filters and 2-by-4 max-pooling strides; the first layer contains 64 filters, and each of the rest three layers contains 128 filters respectively. This allows us to obtain an activation map of size (6, 6, 128), which is different from the size (1, 15, 32) activation map obtained in [9]. Note the new map contains six mel-frequencies, and now we can extract their spatial dependency by feeding them into a multi-directional RNN.

C. The Multi-Directional RNN Network

The multi-directional RNN network first converts an activation map into eight sequences using a new multi-directional scanning strategy, and then feeds each sequence into a Grid LSTM block to extract spatial dependency of its elements.

To convert a map into eight sequences, we propose a multi-directional scanning strategy. Each sequence is generated from one scanning approach (E, F, G), where E specifies if the scanning order is top-down or bottom-up, F specifies if the order is left-to-right or right-to-left, and G specifies if scanning is row-wise or column-wise. In total, there are eight combinations of the three variables which gives rise to eight scanned sequences. For better understanding the strategy, we show an example in Figure 3. The leftmost matrix is a two-dimensional feature map output from CNN, and its four vertexes are denoted by A, B, C and D, respectively. On the right side, we show the eight scanners and their results. For example, result A_r (Vertex A, row-wise scanner) is obtained by the scanner (E=top-down, F=left-to-right, G=row-wise), and result C_c (Vertex C, column-wise scanner) is obtained by the scanner (E=bottom-up, F=left-to-right, G=column-wise).

After multi-directional scanning, we obtain eight sequences and feed each into a standard Grid LSTM network [13] (which is a powerful instance of RNN that can alleviate many practical problems such as vanishing gradient or exploding gradient). Each Grid LSTM block contains 32 grid cells and each grid cell consists of two LSTM cells that respectively model spatial dependencies in the time dimension and depth dimension.

At the end of the multi-directional RNN layer, each block will output a 32-dimensional feature vector and vectors of all blocks will be passed to the fully-connected layer for constructing a classifier.

D. The Fully-Connected Layer

At the fully-connected layer, we construct a classifier that takes the feature vector output from the multi-directional RNN layer as input and outputs a label of the music.

First, the eight feature vectors output from the multi-directional RNN layer are concatenated to form a 256-dimensional feature vector y

$$y = [y_{A_r}, y_{A_c}, y_{B_r}, y_{B_c}, y_{C_r}, y_{C_c}, y_{D_r}, y_{D_c}]^T. \quad (1)$$

Then y is fed into a standard fully-connected network, which consists of a linear transformation layer and a softmax layer, and each output neuron corresponds to the conditional probability of one music label (e.g., a music genre or a music tag).

IV. Learning CRNN-TF

Let X be an input music signal and ω be a music label. Let θ_* be the set of CRNN-TF parameters, including (1) weights $W_{k^{(L)}}$ and bias terms b_k^L of the CNN network, (2) weights $W_\xi^{(\psi,l,\varphi)}$, $U_\xi^{(\psi,l,\varphi)}$ and bias terms $b_\xi^{(\psi,l,\varphi)}$ of the RNN network, (3) weights W_o and bias b_o of the fully-connected layer.[2] The

[2]Superscripts index network layers and subscripts index parameters in a layer. Readers interested in the details of these parameters are referred to [10] for CNN and [13] for RNN.

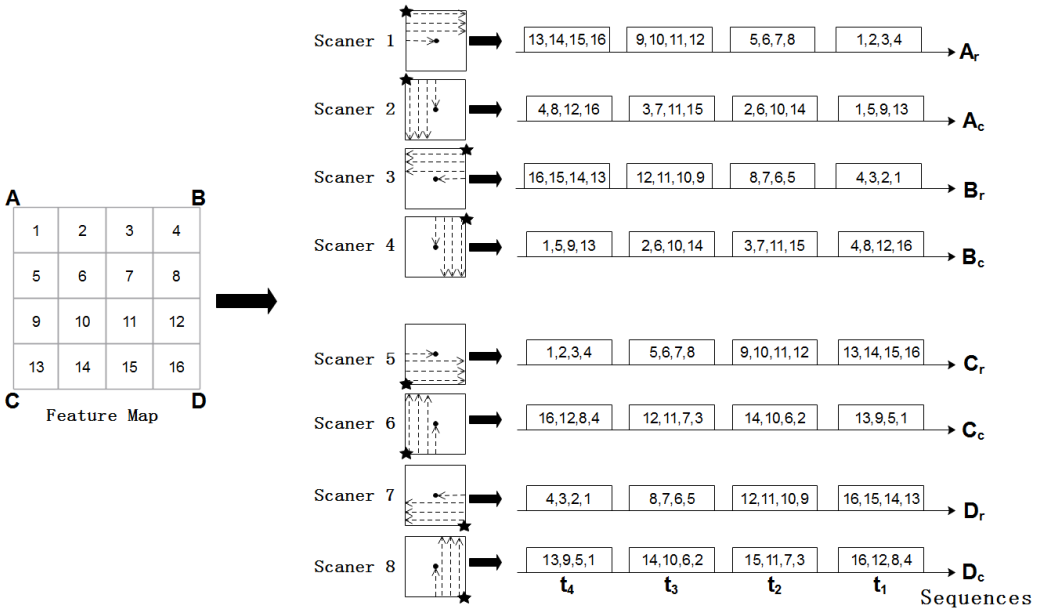

Fig. 3: An example of multidirectional scanning strategy. In each direction, the asterisk is the starting position of the scanning.

objective function of CRNN-TF is

$$\begin{aligned}
&\mathscr{L}(\boldsymbol{\theta}_*) \\
&= -\frac{1}{|\mathscr{A}|}\sum_{X,\omega} P(\omega|\boldsymbol{X};\boldsymbol{\theta}_*)\log(P(\omega|\boldsymbol{X};\boldsymbol{\theta}_*)) \\
&\quad + \lambda_1 \sum_{L,k}(||W_k^{(L)}||^2 + ||b_k^{(L)}||^2) \\
&\quad + \lambda_2 \left(||W_o||^2 + ||b_o||^2\right) \\
&\quad + \lambda_3 \sum_{\xi,\psi,\varphi}(||W_\xi^{(\psi,l,\varphi)}||^2 + ||U_\xi^{(\psi,l,\varphi)}||^2 + ||b_\xi^{(\psi,l,\varphi)}||^2),
\end{aligned} \tag{2}$$

where λ_1, λ_2 and λ_3 are hyper-parameters to balance the weight decay of different components in the network.

We adopt two different loss functions for two music classification tasks, namely, genre classification and music tagging. The former is a multi-class classification problem, and we use the softmax activation plus the categorical cross entropy as the loss function. The latter is a multi-label classification problem, and we use sigmoid activation plus binary cross entropy as the loss function.

To further reduce overfitting, we employ the dropout technique at the fully-connected layer, with a masking vector q applied on the concatenated feature vector y as follows:

$$g = W_o(y \odot q) + b_o, \tag{3}$$

where $\odot$ is the Hadamard product, W_o is the linear transform function and b_o is the bias term.

V. Experiment

A. Data Preparation

In this project, we experimented on three real-world music data sets, namely, the MagnaTagATune [8] data set, the Free Music Archive [17] data set and the Gtzan [18] data set.

The MagnaTagATune (MTAT) data set contains 25,860 ~30 second $16kHz$ sampled music audio clips annotated with 188 tags. Our task is to predict tags of clips based on their audio signals. The tags are unbalanced, e.g., 'guitar' is associated with 4,851 clips, while 'choral' is associated with 490 clips. In the experiment, we focused on predicting the most frequent 50 tags (e.g., pop, rock, ambient, soft, guitar, piano), and removed the clips associated with none of these tags from the entire data set. The original data set comes in 16 parts. For fair comparison, we followed [9] and used the first 12 parts for training, the 13$^{\text{th}}$ part for validation and the remaining 3 parts for testing. Eventually, we obtained 15,244 training clips, 1,529 validation clips, and 4,332 testing clips. The MTAT data set is used for evaluating the music tagging task.

The Free Music Archive (FMA) data set released in 2017 is a large collection of audio tracks categorized into 16 root genres (rock, electronic, experimental, hip-hop, folk, instrumental, pop, international, classical, historic, jazz, country, soul-rnb, spoken, blues and easy listening). All tracks are mp3-encoded, with sampling rate of 44,100Hz, bit rate 320$kbit/s$, and in stereo. It comes with 4 subsets, and we chose the medium-size subset of 25,000 tracks. Our task is to classify tracks into 16 genres based on their audio signals. Also, the genres of medium-size FMA data set have a highly unbalanced distribution. For example, the number of tracks with genre "rock" is 7,103 while the number of tracks with genre "Easy Listening" is only 21. We adopted the suggested data set splitting method in [17], and obtained 19,922 training tracks, 2,505 validation tracks and 2,573 testing tracks. The FMA data set is used for evaluating the music genre classification task.

The GTZAN data set contains 1,000 30-second audio tracks that are categorized into 10 genres. Each genre contains 100

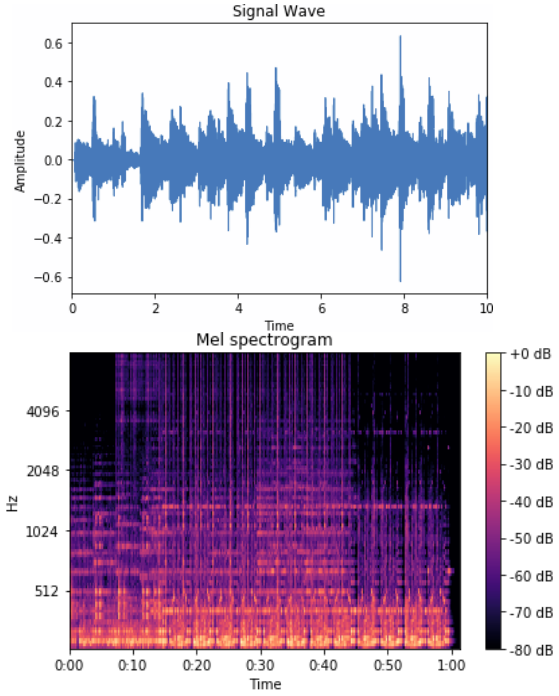

Fig. 4: An Example of the Music Audio Signal Wave and its Preprocessed Mel-Spectrogram

tracks. The tracks are all 22,050Hz Mono 16-bit audio files in .wav format. Our task is to classify tracks into different genres based on their audio signals. We experimented on this data set by 10-fold cross validation, using 9 folds for training and 1 fold for testing at each trial. The GTZAN data set is used for evaluating the transferability of CRNN-TF.

On each of the experimented data sets, we first applied the preprocessor to transform its raw music audio signals into their mel-spectrograms, and then input the spectrum features of music audios into CRNN-TF. In Figure 4, we show an example of the music audio signal wave and its preprocessed mel-spectrogram.

B. Competing Methods

We compare CRNN-TF with the following state-of-the-art deep learning-based music classification techniques.

1. Fully Convolutional Neural Networks (FCN) [10] is a very popular deep learning technique for music auto-tagging. It has a fully convolutional network architecture which consists of 3 to 7 convolutional layers in conjunction with subsampling layers (of different pooling sizes and strides).

2. CRNN [9] is another popular deep learning technique for music auto-tagging.

3. Timbre CNN [14] is a popular deep learning technique to learn timbre representations of music audio signals. It has an efficient implementation of the timbre descriptors which uses different filter shapes in different layers, and their filter sizes are selected from {25, 50, 5, 100 } for dependency in the frequency dimension and from {1, 3, 5, 7} for dependency in the time dimension.

4. End-to-end [11] is a first end-to-end deep learning technique for music auto-tagging. Unlike the other techniques which input preprocessed spectrum of music signals into CNN, the End-to-end technique directly inputs music audio signals into CNN for feature learning.

We implemented all competing methods using Keras with tensorflow. Our implementations followed the exact configurations specified in the references, and the unspecified ones were automatically optimized by Keras.

C. Network Training Protocol and Implementation

To train CRNN-TF using back-propagation, we used a batch sample size of 32 and set the initial learning rate to 0.001 with a 0.9 decay rate after each epoch. We used ELU activation function with $\alpha = 1.0$. We trained the network using the ADAM optimizer (with default setting), and applied batch normalization after every convolutional layer in CNN. While training the fully-connected layer, we apply dropout technique with a dropout rate of 0.5. The hyper-parameters of regularizers are set as $\lambda_1, \lambda_2, \lambda_3 = 10^{-6}$ by experience.

We implemented the network using Keras and TensorFlow[3]. All experiments were conducted on a GPU machine with an NVIDIA TESLA-K40.

D. Evaluation Metrics

For comprehensive evaluation, we used multiple evaluation metrics for the two music classification tasks: music tagging and genre classification.

1) Metrics for Music Tagging: Music tagging is a multi-label classification task, and we chose the metrics of AUC score, Recall@k (k=1,3,5,10), Normalized Discounted Cumulative Gain (NDCG) and Jaccard Similarity Coefficient (JSC). We briefly introduce these metrics in the following.

Recall@k measures the fraction of the relevant tags that are successfully retrieved in the top k predictions. For music audio, given the set of k true tags $G_{1:k}$ and a set of top k predicted tags $\hat{R}_{1:k}$, Recall@k is computed as:

$$Recall@k = \frac{|G_{1:k} \cap \hat{R}_{1:k}|}{|G_{1:k}|}. \tag{4}$$

Discounted Cumulative Gain (DCG) measures the ranking quality of the predicted tags, and it increases when more relevant tags are placed higher in the predicted list. Normalized DCG (NDCG) is obtained by dividing DCG by the "ideal DCG" obtained when the predicted tags are most relevant and perfectly ranked. Specifically, we have

$$DCG = \sum_{i=1}^{|\hat{R}_{1:k}|} \frac{rel_i}{\log_2(i+1)}, \tag{5}$$

Where $rel_i \in \{0, 1\}$ is the relevance of the i_{th} ranked result and higher rel_i means this result is more relevant. The ideal DCG has $rel_i = 1$ for every ranked result and thus is

$$Ideal.DCG = \sum_{i=1}^{|G_{1:k}|} \frac{1}{\log_2(i+1)}. \tag{6}$$

[3]https://github.com/FishInMedi/CRNN-TF

TABLE I: Testing Performance of Music Tagging on the MTAT Data Set

Method	AUC	Recall@1	Recall@3	Recall@5	Recall@10	NDCG@50	JSC
FCN [10]	0.8940	0.3168	**0.5811**	0.7121	0.8713	0.8187	0.3501
Timbre CNN [14]	0.8930	0.2849	0.5338	0.6645	0.8290	0.7815	0.3242
End-to-end [11]	0.8815	0.2905	0.5345	0.6669	0.8360	0.7842	0.3256
CRNN [9]	0.8926	0.3040	0.5605	0.6919	0.8537	0.8014	0.3335
CRNN-TF	**0.9078**	**0.3200**	0.5786	**0.7151**	**0.8727**	**0.8201**	**0.3607**

Then NDCG is

$$NDCG = \frac{DCG}{Ideal.DCG}. \tag{7}$$

Jaccard Similarity Coefficient (JSC) measures the fraction of correctly predicted tags. Given a set of true tags G and a set of predicted tags $\hat{R}$, JSC is the cardinality of their intersected set divided by the cardinality of their union set, that is,

$$JSC(G, \hat{R}) = \frac{|G \cap \hat{R}|}{|G \cup \hat{R}|}. \tag{8}$$

For all metrics, we reported experimental results averaged over all tags. For AUC, we also reported a score for each tag.

2) Metrics for Genre Classification: Genre classification is a multi-class classification task, and the experimented data sets have imbalanced class distribution. To give more robust measurement, we first treated each genre as a binary label and evaluated its prediction performance; then we reported the average performance over all genres. The metrics we chose are AUC score, recall, precision, f1-score and accuracy (i.e., the fraction of misclassified genres).

E. Results of Music Tagging on the MTAT Data Set

In this section, we present experimental results of the music tagging task on the MTAT data set.

First, we show the overall testing performance in Table I. We observe that CRNN-TF outperforms the state-of-the-art deep learning based classifiers across almost all metrics, suggesting the superiority of this technique. In particular, the fact that CRNN-TF outperforms CRNN supports our hypothesis that incorporating spatial dependency in the audio frequency dimension can improve music classification performance. To be specific, we see that CRNN-TF improves the state-of-the-art AUC, NDCG@50 and JSC's by 1%∼3% points. Improvements of recall appear to be less significant compared to FCN, and this may be that, in FCN model, dropout of 0.5 is added after every max-pooling layer which preventing the network from overfitting even at the earlier stage/layers.

Then, in Figure 6 we show AUC scores on 24 tags; these tags are categorized into three classes: genre, instrument and mood. We see that CRNN-TF achieves the best performance on all tags of genre and mood, and on 5 out of 9 tags of instrument. This may imply that audio frequency correlation is more important for categorizing music genre and mood, but less so for categorizing instrument. We also see that FCN achieves better AUC scores on tags of instrument. This implies that designing different filter shapes and stride lengths for capturing pitch, loudness, duration and timbre information at different levels of granularity (which is exactly the motivation of FCN [10]) is important for instrument recognition.

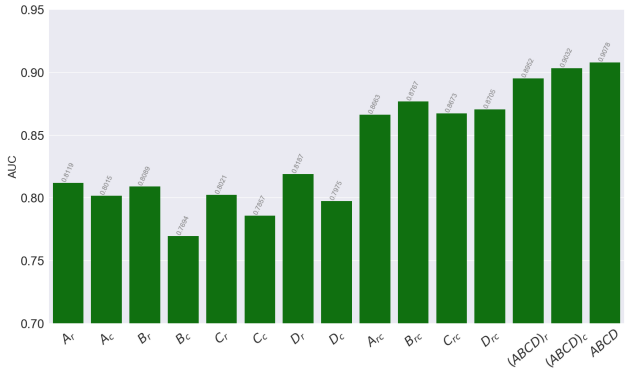

Fig. 5: Testing Performance on MTAT by Feeding Different Subsets of Sequences into RNNs

Finally, we show the effectiveness of the proposed multi-directional scanning strategy. We proposed the strategy to scan an activation map from eight directions and feed all directional scanned sequences into RNNs; our hypothesis is this is better than feeding any subset of the sequences. To verify this hypothesis, in Figure 5 we show different classification performances obtained by feeding different subsets of sequences into RNNs. Result by $X_{r/c}$ is obtained by feeding a single sequence $X_{r/c}$ into RNN (and using the output 32-dimensional feature vector to construct classifier at the fully-connected layer). Similarly, result X_{rc} is obtained by feeding two sequences X_r and X_c into RNNs, and result $ABCD_{r/c}$ is obtained by feeding four sequences $A_{r/c}, B_{r/c}, C_{r/c}$ and $D_{r/c}$ into RNNs. From the figure, we see AUC score increases when more sequences are fed to RNNs, which verifies the benefit of applying our proposed multi-directional scanning strategy.

F. Results of Genre Classification on the FMA Data Set

In this section, we present experimental results of the music genre classification task on the FMA data set.

First, we show the overall testing performance in Table II. We see that CRNN-TF outperforms state-of-the-art classifiers in most metrics. And it consistently beats CRNN which

TABLE II: Testing Performance of Music Genre Classification on the FMA Data Set

Method	AUC	Recall	F1-Score	Accuracy
FCN	0.907	0.430	0.403	0.639
TimbreCNN	0.891	0.364	0.350	0.617
End-to-end	0.891	0.384	0.345	0.614
CRNN	0.903	0.407	0.402	0.634
CRNN-TF	**0.910**	**0.435**	**0.423**	**0.647**

supports our hypothesis of using audio frequency correlation to improve performance. FCN achieves the highest precision among all methods. A possible reason is that FCN adopts more a sophisticated CNN architecture with more layers, more kernels and a multiscale strategy. We believe that, by replacing our current CNN architecture with theirs, CRNN-TF can attain further performance improvement.

To get more insights on why CRNN-TF is achieving higher classification accuracy, in Figure 7 we visualize the distribution of testing data in three feature spaces based on (1) raw features, (2) features extracted by CRNN and (3) features extracted by CRNN-TF.[4] We have two observations. First, genres are better clustered in the CRNNs feature space than in the raw feature space; this explains why CRNN models can achieve significant accuracy. Second, several genres (e.g., 'Classic') are better clustered in the CRNN-TF feature space than in the CRNN feature space; this explains why CRNN-TF can achieve higher accuracy than CRNN.

G. Transferability of CRNN-TF on the GTZAN Data Set

In this section, we briefly evaluated the transferability of CRNN-TF. We first trained CRNN-TF on a source data set MTAT and then transferred its pre-trained parameters to assist learning another CRNN-TF on a target data set GTZAN. Both data sets use the same CRNN-TF architecture, except their fully-connected layers have different number of output neurons – On the source data set we used 50 neurons corresponding to 50 tags, and on the target data set we used 10 neurons corresponding to 10 genres. We examined three transfer strategies:

1. Freeze CRNN-TF: Freeze weights of the pre-trained CRNN-TF, and retrain the FC layer.

2. Freeze CNN, fine-tune RNN: Freeze weights of the pre-trained CNN, and retrain weights of RNN and FC layer.

3. Fine-tune CRNN-TF: Use pre-trained weights of CRNN-TF to initialize the training of another CRNN-TF.

We compared the above strategies with two baseline methods: (1) CRNN-TF without any transfer; (2) MFCC+SVM [16], a representative non-deep learning music classifier.

We performed 10-fold cross validation on the target data set, with 9 folds used for training and 1 fold used for testing, and reported the average classification accuracy in Table III. We observe that transfer CRNN-TFs generally outperform their non-transfer counterpart, with fine-tune CRNN-TF achieving 2.3% performance improvement. Freeze CRNN-TF performs poorly, which may imply the two data sets do have distinct distributions. This is further supported by the fact that Freeze CNN + Fine-tune RNN improves CRNN-TF, and Fine-tune CRNN-TF attains the highest improvement. In summary, the results suggest that CRNN-TF is transferable but it must be carefully done between heterogeneous tasks.

[4]Features extracted by CRNN/CRNN-TF are those output by RNNs and used to construct classifier at the fully-connected layer. And we use the tSNE [19] technique to visualize data distribution.

TABLE III: Testing Performance on the Gtzan Data Set

Method	Accuracy
MFCC + SVM	0.660
CRNN-TF	0.701
Freeze CRNN-TF	0.592
Freeze CNN + Fine-tune RNN	0.711
Fine-tune CRNN-TF	**0.724**

VI. Conclusion

In this paper, we propose an improved CRNN technique called CRNN-TF for music classification. Compared with CRNN that only uses RNN to extract spatial dependency in the music signal time dimension, CRNN-TF uses RNN to extract dependencies in both the time and audio frequency dimensions. In experiments, we show CRNN-TF consistently outperforms CRNN and several other state-of-the-art deep learning-based music classifiers across two music classification tasks on three real-world music data sets under a variety of evaluation metrics.

References

[1] Casey MA, Veltkamp R, Goto M, Leman M, Rhodes C, Slaney M. Content-based music information retrieval: Current directions and future challenges. Proceedings of the IEEE. 2008; 96(4):668-96.
[2] Hamel P, Wood S, Eck D. Automatic Identification of Instrument Classes in Polyphonic and Poly-Instrument Audio. In ISMIR 2009.
[3] Wyse L. Audio spectrogram representations for processing with convolutional neural networks. CoRR 2017.
[4] Cakir E, Parascandolo G, Heittola T, Huttunen H, Virtanen T, Cakir E, Parascandolo G, Heittola T, Huttunen H, Virtanen T. Convolutional recurrent neural networks for polyphonic sound event detection. IEEE/ACM Trans. Audio, Speech and Language Processing, 2017.
[5] Zuo Z, Shuai B, Wang G, Liu X, Wang X, Wang B, Chen Y. Convolutional recurrent neural networks: Learning spatial dependencies for image representation. In CVPR Workshop 2015.
[6] Shtern M, Ejaz R, Tzerpos V. Transfer learning in neural networks: an experience report. In CASCON 2017; pp. 201-210.
[7] Kim J, Won M, Serra X, Liem C. Transfer Learning of Artist Group Factors to Musical Genre Classification. In Companion of the The Web Conference 2018 on The Web Conference 2018; pp. 1929-1934. International World Wide Web Conferences Steering Committee.
[8] Law E, West K, Mandel MI, Bay M, Downie JS. Evaluation of Algorithms Using Games: The Case of Music Tagging. In ISMIR 2009.
[9] Choi K, Fazekas G, Sandler M, Cho K. Convolutional recurrent neural networks for music classification. In ICASSP, 2017.
[10] Choi K, Fazekas G, Sandler M. Automatic tagging using deep convolutional neural networks. CoRR 2016.
[11] Dieleman S, Schrauwen B. End-to-end learning for music audio. In Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), 2014.

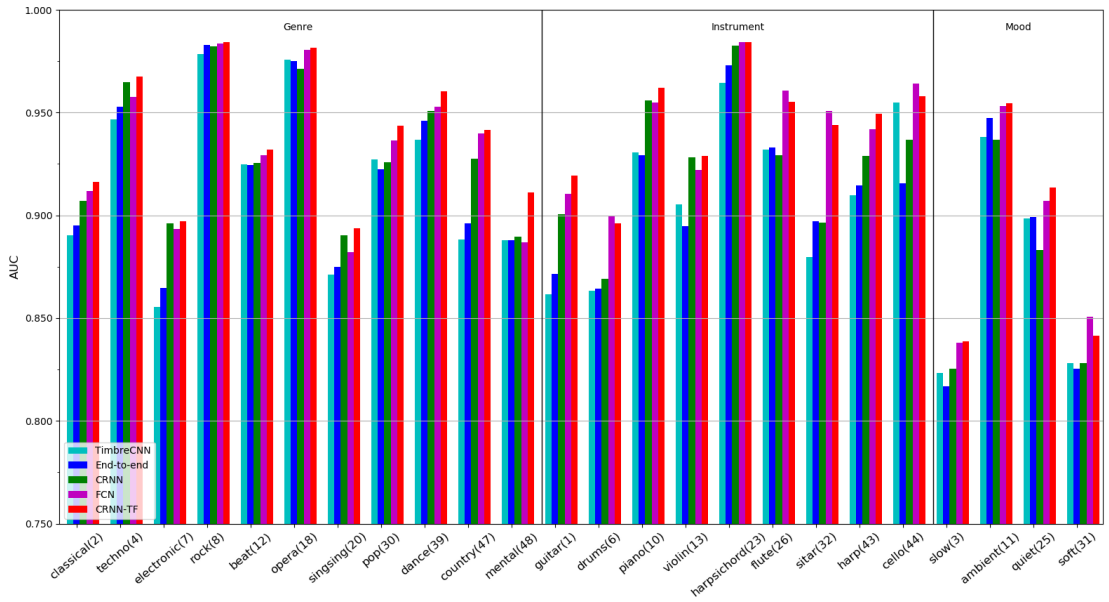

Fig. 6: AUC Scores of Popular Tags on the MTAT Data Set. X axis is the tag index and Y axis is the AUC score. For each tag, the number in the parenthesis is the popularity rank of the tag.

(a) Raw Features (b) CRNN Features (c) CRNN-TF Features

Fig. 7: Distribution of FMA Testing Data under in Different Feature Spaces.

[12] Lee J, Park J, Kim KL, Nam J. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. CoRR 2017.
[13] Kalchbrenner N, Danihelka I, Graves A. Grid long short-term memory. CoRR 2015.
[14] Pons J, Slizovskaia O, Gong R, Gmez E, Serra X. Timbre analysis of music audio signals with convolutional neural networks. In Signal Processing Conference, 2017 25th European; pp. 2744-2748.
[15] Gl U, Thielen J, Hanke M, Van Gerven M. Brains on beats. In NIPS 2016.
[16] Choi K, Fazekas G, Sandler M, Cho K. Transfer learning for music classification and regression tasks. CoRR 2017.
[17] Defferrard M, Benzi K, Vandergheynst P, Bresson X. Fma: a dataset for music analysis. CoRR 2016.
[18] Sturm BL. The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. CoRR 2013.
[19] Maaten LV, Hinton G. Visualizing data using t-SNE. Journal of machine learning research. 2008; 9(Nov):2579-605.
[20] Oquab M, Bottou L, Laptev I, Sivic J. Learning and transferring mid-level image representations using convolutional neural networks. In CVPR 2014.
[21] McFee B, Raffel C, Liang D, Ellis DP, McVicar M, Battenberg E, Nieto O. librosa: Audio and music signal analysis in python. In Proceedings of the 14th python in science conference 2015; pp. 18-25.
[22] Sigtia S, Benetos E, Dixon S. An end-to-end neural network for polyphonic piano music transcription. IEEE/ACM Transactions on Audio, Speech, and Language Processing. 2016; 24(5):927-39.
[23] Hamel P, Eck D. Learning Features from Music Audio with Deep Belief Networks. In ISMIR 2010; (Vol. 10, pp. 339-344).
[24] Lee H, Pham P, Largman Y, Ng AY. Unsupervised feature learning for audio classification using convolutional deep belief networks. In Advances in neural information processing systems 2009; pp.1096-1104.
[25] Schlter J, Bck S. Improved musical onset detection with Convolutional Neural Networks. In ICASSP 2014; pp. 6979-6983.
[26] Van den Oord A, Dieleman S, Schrauwen B. Deep content-based music recommendation. In NIPS, 2013.
[27] Lee J, Nam J. Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. IEEE signal processing letters. 2017; 24(8):1208-12.