CONVOLUTIONAL RECURRENT NEURAL NETWORKS FOR MUSIC CLASSIFICATION

Keunwoo Choi, György Fazekas, Mark Sandler*  
Queen Mary University of London, London, UK  
Centre for Digital Music, EECS  
E1 4FZ, London, UK  
keunwoo.choi@qmul.ac.uk  

Kyunghyun Cho†  
New York University  
Computer Science & Data Science  
New York, NY, USA  
kyunghyun.cho@nyu.edu  

ABSTRACT

We introduce a convolutional recurrent neural network (CRNN) for music tagging. CRNNs take advantage of convolutional neural networks (CNNs) for local feature extraction and recurrent neural networks for temporal summarisation of the extracted features. We compare CRNN with three CNN structures that have been used for music tagging while controlling the number of parameters with respect to their performance and training time per sample. Overall, we found that CRNNs show a strong performance with respect to the number of parameter and training time, indicating the effectiveness of its hybrid structure in music feature extraction and feature summarisation.

Index Terms— convolutional neural networks, recurrent neural networks, music classification

1. INTRODUCTION

Convolutional neural networks (CNNs) have been actively used for various music classification tasks such as music tagging [1, 2], genre classification [3, 4], and user-item latent feature prediction for recommendation [5].

CNNs assume features that are in different levels of hierarchy and can be extracted by convolutional kernels. The hierarchical features are learned to achieve a given task during supervised training. For example, learned features from a CNN that is trained for genre classification exhibit low-level features (e.g., onset) to high-level features (e.g., percussive instrument patterns) [6].

Recently, CNNs have been combined with recurrent neural networks (RNNs) which are often used to model sequential data such as audio signals or word sequences. This hybrid model is called a convolutional recurrent neural network (CRNN). A CRNN can be described as a modified CNN by replacing the last convolutional layers with an RNN. In CRNNs, CNNs and RNNs play the roles of feature extractor and temporal summariser, respectively. Adopting an RNN for aggregating the features enables the networks to take the global structure into account while local features are extracted by the remaining convolutional layers. This structure was first proposed in [7] for document classification and later applied to image classification [8] and music transcription [9].

CRNNs fit the music tagging task well. RNNs are more flexible in selecting how to summarise the local features than CNNs which are rather static by using weighted average (convolution) and subsampling. This flexibility can be helpful because some of the tags (e.g., mood tags) may be affected by the global structure while other tags such as instruments can be affected by local and short-segment information.

In this paper, we introduce CRNNs for music tagging and compare them with three existing CNNs. For correct comparisons, we carefully control the hardware, data, and optimisation techniques, while varying two attributes of the structure: i) the number of parameters and ii) computation time.

2. MODELS

We compare CRNN with $k1c2$, $k2c1$, and $k2c2$, which are illustrated in Figure 1. The three convolutional networks are named to specify their kernel shape (e.g., $k1$ for 1D kernels) and convolution dimension (e.g., $c2$ for 2D convolutions). The specifications are shown in Table 1. For all networks, the input is assumed to be of size $96 \times 1366$ (mel-frequency band x time frame) and single channel. Sigmoid functions are used as activation at output nodes because music tagging is a multi-label classification task.

In this paper, all the convolutional and fully-connected layers are equipped with identical optimisation techniques and activation functions — batch normalisation [10] and ELU activation function [11]. This is for a correct comparison since optimisation techniques greatly improve the performances of networks that are having essentially the same structure. Exceptionally, CRNN has weak dropout (0.1) between convolutional layers to prevent overfitting of the RNN layers [12].
as illustrated in Figure 1b. The network reduces the size of
information along time-axis) and max-pooling layers ((1×4)-(1×5)-
(1×8)-(1×8)) alternate. Each element of the last feature
map (the output of the 4-th sub-sampling layer) encodes a
feature for each band. They are flattened and fed into a
fully-connected layer, which acts as the classifier.

This model compress the information of whole frequency
range into one band in the first convolutional layer and this
helps reducing the computation complexity vastly.

CNN - k2c1

k2c1 in Figure 1b is motivated by structures for music tag-
ing [1] and genre classification [14]. The network consists of
5 convolutional layers that are followed by 2 fully-connected
layers. The first convolutional layer (96 × 4) learns 2D ker-
nels that are applied to the whole frequency band. After then,
one-dimensional convolutional layers (1×4 for all, i.e., con-
volution along time-axis) and max-pooling layers ((1×4) or
(1×5)) alternate. The results are flattened and fed into a fully-
connected layer.

This model allows time and frequency invariances in dif-
erent scale by gradual 2D sub-samplings. Also, using 2D
subsampling enables the network to be fully-convolutional,
which ultimately results in fewer parameters.

CNN - k2c2

CNN structures with 2D convolution have been used in mu-
sic tagging [2] and vocal/instrumental classification [15].
k2c2 consists of five convolutional layers of 3×3 kernels
and max-pooling layers ((2×4)-(2×4)-(2×4)-(3×5)-(4×4))
as illustrated in Figure 1c. The network reduces the size of
feature maps to 1×1 at the final layer, where each feature
covers the whole input rather than each frequency band as in
k1c1 and k2c1.

This model allows time and frequency invariances in dif-
ferent scale by gradual 2D sub-samplings. Also, using 2D
subsampling enables the network to be fully-convolutional,
which ultimately results in fewer parameters.

CRNN

CRNN uses a 2-layer RNN with gated recurrent units (GRU)
[16] to summarise temporal patterns on the top of two-
dimensional 4-layer CNNs as shown in Figure 1c. The
assumption underlying this model is that the temporal pat-
tern can be aggregated better with RNNs than CNNs, while
relying on CNNs on input side for local feature extraction.

In CRNN, RNNs are used to aggregate the temporal pat-
terns instead of, for instance, averaging the results from
shorter segments as in [1] or convolution and sub-sampling
as in other CNN’s. In its CNN sub-structure, the sizes of con-
volutional layers and max-pooling layers are 3×3 and (2×2)-
(3×3)-(4×4)-(4×4). This sub-sampling results in a feature
map size of N × 1 × 15 (number of feature maps × frequency × time).
They are then fed into a 2-layer RNN, of which the last hidden
state is connected to the output of the network.

2.5. Scaling networks

The models are scaled by controlling the number of par-
ameters to be 100,000, 250,000, 0.5 million, 1M, 3M with 2%
tolerance. Considering the limitation of current hardware and
the dataset size, 3M-parameter networks are presumed to pro-
vide an approximate upper bound of the structure complexity.
Table 1 summarises the details of different structures includ-
ing the layer width (the number of feature maps or hidden
units).

The widths of layers are based on [1] for k1c2 and k2c1,
and [2] for k2c2. For CRNN, the widths are determined based
on preliminary experiments which showed the relative impor-
tance of the numbers of the feature maps of convolutional lay-
ers over the number of hidden units in RNNs.

Layer widths are changed to control the number of pa-
rameters of a network while the depths and the convolutional
kernel shapes are kept constant. Therefore, the hierarchy of
learned features is preserved while the numbers of the features
in each hierarchical level (i.e., each layer) are changed. This is
to maximise the representation capabilities of networks, con-
sidering the relative importance of depth over width [17].

3. EXPERIMENTS

We use the Million Song Dataset [18] with last.fm tags. We
train the networks to predict the top-50 tag, which includes
genres (e.g., rock, pop), moods (e.g., sad, happy), instru-
ments (e.g., female vocalist, guitar), and eras (60s – 00s).
214,284 (201,680 for training and 12,605 for validation) and
25,940 clips are selected by using the originally provided
training/test splitting and filtering out items without any top-
The model is built with Keras [20] and Theano [21]. We use ADAM for learning rate control [22] and binary cross-entropy as a loss function. The reported performance is measured on test set and by AUC-ROC (Area Under Receiver Operating Characteristic Curve) given that tagging is a multi-class classification problem. Models and split sets are shared online. We use early-stopping for the all structures – the training is stopped if there is no improvement of AUC on the validation set while iterating the whole training data once.

### 3.1. Memory-controlled experiment

Figure 2 shows the AUCs for each network against the number of parameters. With the same number of parameters, the ranking of AUC is CRNN > k2c2 > k1c2 > k2c1. This indicates that CRNN can be preferred when the bottleneck is memory usage.

CRNN outperforms k2c2 in all cases. Because they share the same 2D-convolutional layers, this difference is probably a consequence of the difference in RNNs and CNNs the ability of summarising the features over time. This may indicate that learning a global structure is more important than focusing on local structures for summarisation. One may focus on the different layer widths of two structures – because recurrent layers use less parameters than convolutional layers, CRNN has wider convolutional layers than k2c2 with same number of parameters. However, even CRNN with narrower layer widths (0.1M parameters) shows better performance than k2c2 with wider widths (0.25M parameters).

k2c2 shows higher AUCs than k2c1 and k1c2 in all cases. This shows that the model of k2c2, which encodes local invariance and captures local time-frequency relationships, is more effective than the others, which ignores local frequency relationships. k2c2 also uses parameters in a more flexible way with its fully-convolutional structure, while k2c1 and k1c2 allocate only a small proportion of the parameters to the feature extraction stage. For example, in k1c2 with 0.5M parameters, only 13% of the parameters are used by convolutional layers while the rest, 87%, are used by the fully-connected layers.

k2c2 structures (>0.5M parameters) shows better performances than a similar but vastly larger structure in [2], which is shown as state of the art in Figure 2. This is because the reduction in the number of feature maps removes redundancy.

The flexibility of k1c2 may contribute the performance improvement over k2c1. In k2c1, the tall 2-dimensional kernels in the first layer of k2c1 compress the information of the whole frequency-axis pattern into each feature map. The following kernels then deal with this compressed representation with temporal convolutional and pooling. On the other hands, in k1c2, 1-dimensional kernels are shared over time and frequency axis until the end of convolutional layers. In other words, it gradually compress the information in time axis first, while preserving the frequency-axis pattern.

---

**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.

<table>
<thead>
<tr>
<th>Layer type</th>
<th>Layer width</th>
<th>Type</th>
<th>Layer width</th>
<th>Type</th>
<th>Layer width</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv2d</td>
<td>15 23 34 47 81</td>
<td>conv1d</td>
<td>43 72 106 152 265</td>
<td>conv2d</td>
<td>20 33 47 67 118</td>
<td>conv2d</td>
</tr>
<tr>
<td>conv2d</td>
<td>15 23 34 47 81</td>
<td>conv1d</td>
<td>43 72 106 152 265</td>
<td>conv2d</td>
<td>41 66 95 135 236</td>
<td>conv2d</td>
</tr>
<tr>
<td>conv2d</td>
<td>30 47 66 95 163</td>
<td>conv1d</td>
<td>43 72 106 152 265</td>
<td>conv2d</td>
<td>41 66 95 135 236</td>
<td>conv2d</td>
</tr>
<tr>
<td>conv2d</td>
<td>30 47 66 95 163</td>
<td>conv1d</td>
<td>87 145 212 304 535</td>
<td>conv2d</td>
<td>62 100 142 203 355</td>
<td>conv2d</td>
</tr>
<tr>
<td>FC</td>
<td>30 47 66 95 163</td>
<td>conv1d</td>
<td>87 145 212 304 535</td>
<td>conv2d</td>
<td>83 133 190 271 473</td>
<td>rnn</td>
</tr>
<tr>
<td>FC</td>
<td>30 47 66 95 163</td>
<td>conv1d</td>
<td>87 145 212 304 535</td>
<td>conv2d</td>
<td>30 48 68 96 169</td>
<td>CRNN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FC</td>
<td>87 145 212 304 535</td>
<td>FC</td>
<td>30 48 68 96 169</td>
<td>CRNN</td>
</tr>
</tbody>
</table>

**Fig. 2**: AUCs for each network against the number of parameters [×10^6]. The AUC of SOTA is 0.851 [2].

50 tags. The occurrences of tags range from 52,944 (rock) to 1,257 (happy).

We use 30-60s preview clips which are provided after trimming to represent the highlight of the song. We trim audio signals to 29 seconds at the centre of preview clips and down-sample them from 22.05 kHz to 12 kHz using Librosa [19]. Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [2, 1]. The number of mel-bins is 96 and the hop-size is 256 samples, resulting in an input shape of 96×1366.

The model is built with Keras [20] and Theano [21]. We use 30-60s preview clips which are provided after trimming to represent the highlight of the song. We trim audio signals to 29 seconds at the centre of preview clips and down-sample them from 22.05 kHz to 12 kHz using Librosa [19]. Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [2, 1]. The number of mel-bins is 96 and the hop-size is 256 samples, resulting in an input shape of 96×1366.

The model is built with Keras [20] and Theano [21]. We use 30-60s preview clips which are provided after trimming to represent the highlight of the song. We trim audio signals to 29 seconds at the centre of preview clips and down-sample them from 22.05 kHz to 12 kHz using Librosa [19]. Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [2, 1]. The number of mel-bins is 96 and the hop-size is 256 samples, resulting in an input shape of 96×1366.

The model is built with Keras [20] and Theano [21]. We use 30-60s preview clips which are provided after trimming to represent the highlight of the song. We trim audio signals to 29 seconds at the centre of preview clips and down-sample them from 22.05 kHz to 12 kHz using Librosa [19]. Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [2, 1]. The number of mel-bins is 96 and the hop-size is 256 samples, resulting in an input shape of 96×1366.

The model is built with Keras [20] and Theano [21]. We use 30-60s preview clips which are provided after trimming to represent the highlight of the song. We trim audio signals to 29 seconds at the centre of preview clips and down-sample them from 22.05 kHz to 12 kHz using Librosa [19]. Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [2, 1]. The number of mel-bins is 96 and the hop-size is 256 samples, resulting in an input shape of 96×1366.

The model is built with Keras [20] and Theano [21]. We use 30-60s preview clips which are provided after trimming to represent the highlight of the song. We trim audio signals to 29 seconds at the centre of preview clips and down-sample them from 22.05 kHz to 12 kHz using Librosa [19]. Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [2, 1]. The number of mel-bins is 96 and the hop-size is 256 samples, resulting in an input shape of 96×1366.

The model is built with Keras [20] and Theano [21]. We use 30-60s preview clips which are provided after trimming to represent the highlight of the song. We trim audio signals to 29 seconds at the centre of preview clips and down-sample them from 22.05 kHz to 12 kHz using Librosa [19]. Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [2, 1]. The number of mel-bins is 96 and the hop-size is 256 samples, resulting in an input shape of 96×1366.

The model is built with Keras [20] and Theano [21]. We use 30-60s preview clips which are provided after trimming to represent the highlight of the song. We trim audio signals to 29 seconds at the centre of preview clips and down-sample them from 22.05 kHz to 12 kHz using Librosa [19]. Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [2, 1]. The number of mel-bins is 96 and the hop-size is 256 samples, resulting in an input shape of 96×1366.

The model is built with Keras [20] and Theano [21]. We use 30-60s preview clips which are provided after trimming to represent the highlight of the song. We trim audio signals to 29 seconds at the centre of preview clips and down-sample them from 22.05 kHz to 12 kHz using Librosa [19]. Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [2, 1]. The number of mel-bins is 96 and the hop-size is 256 samples, resulting in an input shape of 96×1366.

The model is built with Keras [20] and Theano [21]. We use 30-60s preview clips which are provided after trimming to represent the highlight of the song. We trim audio signals to 29 seconds at the centre of preview clips and down-sample them from 22.05 kHz to 12 kHz using Librosa [19]. Log-amplitude mel-spectrograms are used as input since they have outperformed STFT and MFCCs, and linear-amplitude mel-spectrograms in earlier research [2, 1]. The number of mel-bins is 96 and the hop-size is 256 samples, resulting in an input shape of 96×1366.
The input compression in k2c1 results in a fast computation, making it merely overlaps in time with other structures. The time consumptions of the other structures range in a overlapping region.

Overall, with similar training time, k2c2 and CRNN show the best performance. This result indicates that either k2c2 or CRNN can be used depending on the target time budget.

With the same number of parameters, the ranking of training speed is always k2c1 > k2c2 > k1c2 > CRNN. There seems two factors that affect this ranking. First, among CNN structures, the sizes of feature maps are the most critical since the number of convolution operations is in proportion to the sizes. k2c1 reduces the size of feature map in the first convolutional layer, where the whole frequency bins are compressed into one. k2c2 reduces the sizes of feature maps in both axes and is faster than k1c2 which reduces the sizes only in temporal axis. Second, the difference between CRNN and CNN structures arises from the negative correlation of speed and the depth of networks. The depth of CRNN structure is up to 20 (15 time steps in RNN and 5 convolutional layers), introducing heavier computation than the other CNN structures.

3.3. Performance per tag
Figure 4 visualises the AUC score of each tag of 1M-parameter structures. Each tag is categorised as one of genres, moods, instruments and eras, and sorted by AUC within its category. Under this categorisation, music tagging can be considered as a multiple-task problem equivalent to four classification tasks with these four categories.

The CRNN outperforms k2c1 for 44 tags, and k2c1 outperforms k1c2 for 48 out of 50 tags. From the multiple-task classification perspective, this result indicates that a structure that outperforms in one of the four tasks may perform best in the other tasks as well.

Although the dataset is imbalanced, the tag popularity (number of occurrence of each tag) is not correlated to the performance. Spearman rank correlation between tag popularity and the ranking of AUC scores of all tags is 0.077. It means that the networks effectively learn features that can be shared to predict different tags.

4. CONCLUSIONS
We proposed a convolutional recurrent neural network (CRNN) for music tagging. In the experiment, we controlled the size of the networks by varying the numbers of parameters to for memory-controlled and computation-controlled comparison. Our experiments revealed that 2D convolution with 2d kernels (k2c2) and CRNN perform comparably to each other with a modest number of parameters. With a very small or large number of parameters, we observed a trade-off between speed and memory. The computation of k2c2 is faster than that of CRNN across all parameter settings, while the CRNN tends to outperform it with the same number of parameters.
5. REFERENCES


