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 a 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*.

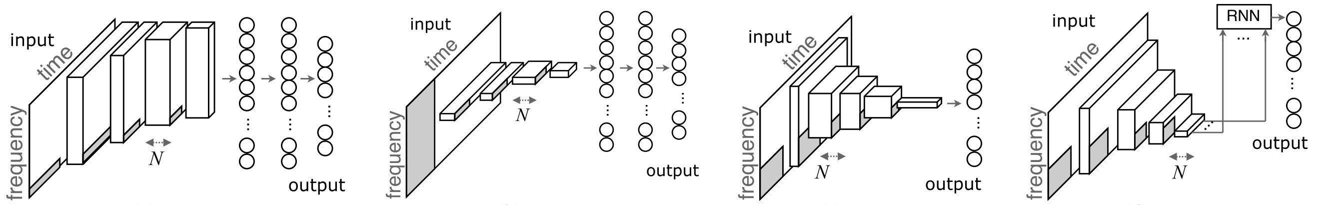
2. MODELS

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 96×1366 (mel-frequency band$\times$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 normalization [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].

*This work has been part funded by FAST IMPACt EPSRC Grant EP/L019981/1 and the European Commission H2020 research and innovation grant AudioCommons (688382). Mark Sandler acknowledges the support of the Royal Society as a recipient of a Wolfson Research Merit Award.

†Kyunghyun Cho thanks the support by Facebook, Google (Google Faculty Award 2016) and NVidia (GPU Center of Excellence 2015-2016)



(a) `k1c2` (b) `k2c1` (c) `k2c2` (d) `CRNN`

Fig. 1: Block diagrams of `k1c2`, `k2c1`, `k2c2`, and `CRNN`. The grey areas illustrate the convolution kernels. N refers to the number of feature maps of convolutional layers.

2.1. CNN - `k1c2`

`k1c2` in Figure 1a is motivated by structures for genre classification [13]. The network consists of 4 convolutional layers that are followed by 2 fully-connected layers. One-dimensional convolutional layers (1×4 for all, i.e., convolution 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.

2.2. CNN - `k2c1`

`k2c1` in Figure 1b is motivated by structures for music tagging [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 kernels that are applied to the whole frequency band. After then, one-dimensional convolutional layers (1×4 for all, i.e., convolution 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 compress the information of whole frequency range into one band in the first convolutional layer and this helps reducing the computation complexity vastly.

2.3. CNN - `k2c2`

CNN structures with 2D convolution have been used in music 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 1b. 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 different scale by gradual 2D sub-samplings. Also, using 2D subsampling enables the network to be *fully-convolutional*, which ultimately results in fewer parameters.

2.4. 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 pattern can be aggregated better with RNNs then CNNs, while relying on CNNs on input side for local feature extraction.

In `CRNN`, RNNs are used to aggregate the temporal patterns 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 convolutional 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 $N\times1\times15$ (number of feature maps$\times$frequency$\times$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 parameters 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 provide an approximate upper bound of the structure complexity. Table 1 summarises the details of different structures including 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 importance of the numbers of the feature maps of convolutional layers over the number of hidden units in RNNs.

Layer widths are changed to control the number of parameters 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, considering 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*), instruments (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-

k1c2						**k2c1**						**k2c2**						**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		0.1	0.25	0.5	1.0	3.0
Layer type	Layer width					Type	Layer width					Type	Layer width										
`conv2d`	15	23	33	47	81	`conv1d`	43	72	106	152	265	`conv2d`	20	33	47	67	118	`conv2d`	30	48	68	96	169
`conv2d`	15	23	33	47	81	`conv1d`	43	72	106	152	265	`conv2d`	41	66	95	135	236	`conv2d`	60	96	137	195	339
`conv2d`	30	47	66	95	163	`conv1d`	43	72	106	152	265	`conv2d`	41	66	95	135	236	`conv2d`	60	96	137	195	339
`conv2d`	30	47	66	95	163	`conv1d`	87	145	212	304	535	`conv2d`	62	100	142	203	355	`conv2d`	60	96	137	195	339
`FC`	30	47	66	95	163	`conv1d`	87	145	212	304	535	`conv2d`	83	133	190	271	473	`rnn`	30	48	68	96	169
`FC`	30	47	66	95	163	`FC`	87	145	212	304	535							`rnn`	30	48	68	96	169
						`FC`	87	145	212	304	535												

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.

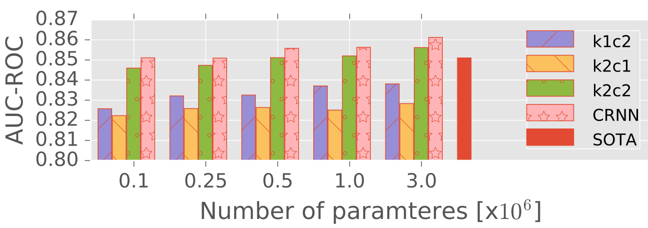


Fig. 2: AUCs for the three structures with $\{0.1, 0.25, 0.5, 1.0, 3.0\} \times 10^6$ parameters. The AUC of SOTA is .851 [2].

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 downsample 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 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-label classification. Models and split sets are shared online[1].

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.

[1] `https://github.com/keunwoochoi/icassp_2017`

`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 `k2x2` 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.

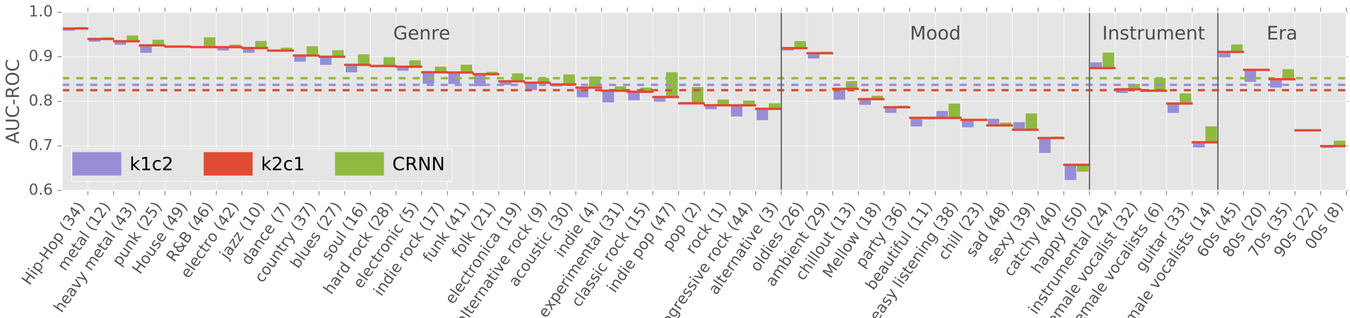


Fig. 3: AUCs of 1M-parameter structures. *i*) The average AUCs over all samples are plotted with dashed lines. *ii*) AUC of each tag is plotted using a bar chart and line. For each tag, red line indicates the score of `k2c1` which is used as a baseline of bar charts for `k1c2` (blue) and `CRNN` (green). In other words, blue and green bar heights represent the performance gaps, `k2c1`-`k1c2` and `CRNN`-`k2c1`, respectively. *iii*) Tags are grouped by categories (genre/mood/instrument/era) and sorted by the score of `k2c1`. *iv*) The number in parentheses after each tag indicates that tag's popularity ranking in the dataset.

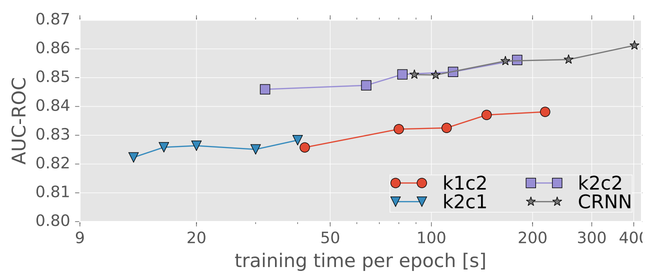


Fig. 4: AUCs of the structures in training time - AUC plane. Each plot represents four different parameters, $\{0.1, 0.25, 0.5, 1.0, 3.0\} \times 10^6$, from left to right.

3.2. Computation-controlled comparison

We further investigate the computational complexity of each structure. The computational complexity is directly related to the training and prediction time and varies depending not only on the number of parameters but also on the structure. The wall-clock training times for 2500 samples are summarised in Table 1 and plotted in Figure 2.

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 3 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 task 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

[1] Sander Dieleman and Benjamin Schrauwen, "End-to-end learning for music audio," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6964–6968.

[2] Keunwoo Choi, George Fazekas, and Mark Sandler, "Automatic tagging using deep convolutional neural networks," in *International Society of Music Information Retrieval Conference*. ISMIR, 2016.

[3] Siddharth Sigtia and Simon Dixon, "Improved music feature learning with deep neural networks," in *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2014.

[4] Paulo Chiliguano and Gyorgy Fazekas, "Hybrid music recommender using content-based and social information," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2618–2622.

[5] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen, "Deep content-based music recommendation," in *Advances in Neural Information Processing Systems*, 2013, pp. 2643–2651.

[6] Keunwoo Choi, George Fazekas, and Mark Sandler, "Explaining deep convolutional neural networks on music classification," *arXiv preprint arXiv:1607.02444*, 2016.

[7] Duyu Tang, Bing Qin, and Ting Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1422–1432.

[8] Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen, "Convolutional recurrent neural networks: Learning spatial dependencies for image representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 18–26.

[9] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, 2016.

[10] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[11] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[13] Tom LH Li, Antoni B Chan, and A Chun, "Automatic musical pattern feature extraction using convolutional neural network," in *Proc. Int. Conf. Data Mining and Applications*, 2010.

[14] Jan Wülfing and Martin Riedmiller, "Unsupervised learning of local features for music classification.," in *International Society of Music Information Retrieval Conference*. ISMIR, 2012, pp. 139–144.

[15] Jan Schlüter, "Learning to pinpoint singing voice from weakly labeled examples," in *International Society of Music Information Retrieval Conference*. ISMIR, 2016.

[16] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[17] Ronen Eldan and Ohad Shamir, "The power of depth for feedforward neural networks," *arXiv preprint arXiv:1512.03965*, 2015.

[18] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere, "The million song dataset," in *Proceedings of the 12th International Society for Music Information Retrieval Conference, Miami, Florida, USA, October 24-28, 2011*, 2011.

[19] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference*, 2015.

[20] François Chollet, "Keras," *GitHub repository: https://github. com/fchollet/keras*, 2015.

[21] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al., "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.

[22] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.