MUSIC GENERATION WITH LOCAL CONNECTED CONVOLUTIONAL NEURAL NETWORK

Zhihao Ouyang[1]*, Yihang Yin*[2]*, Kun Yan*[3]*, Jian Wu*[1]*, Xiaolin Hu*[1]*, Shu-Tao Xia*[1]

[1]Department of Computer Science and Technology, Tsinghua University, China
[2]School of Computer Science and Engneering, Beihang University, China
[3]School of Software, Beihang University, China

ABSTRACT

Recent works demonstrate that the feedforward model like Convolutional Neural Network (CNN) can be applied to music generation task [1, 2]. However, we find the CNN model's result is limited because it is lack of temporal feature. To tackle this problem, we add Locally Connected Convolutional (**Local Conv**) layers to plain CNN. We put Local Conv layer to the different position of a small CNN model to find the appropriate way to add Local Conv architecture, then modify resNet with Local Conv layer. We conclude Local Conv provides the existing CNN architecture with the temporal characteristic which boosts CNN model to generate better music than both RNN model and naive CNN model. Besides, we analyze the generated music sheet and user study result to prove music generated by Local Conv model is better than RNN and naive CNN model. All the code and generated music samples are released at: `https://somedaywilldo.github.io/local_conv_music_generation/`

Index Terms— music generation, locally connected CNN, temporal feature

1. INTRODUCTION

In a short time scale, music can be regarded as a sequence of notes. In a long time scale, music consists of musical elements like chords, phrases. This complex structure makes music generation a challenging task.

As to music generation, there are two popular tasks, monophonic and polyphonic music generation. In general, monophonic music generation task has a difficulty to generate a long sequence due to the gradient vanish problem [2]. Several methods have been applied to model the music: [3] propose MuseGAN, which makes use of Transposed Convolutional Neural Networks (TCNN) and Generative Adversarial Networks(GANs) structure to generate symbolic multitrack music. [4] uses a graphical model aimed at modelling polyphonic music and specifically hymn-like pieces. Besides, there are lots of other models: Markov model [5], multi-layer LSTM(Long Short-Term Memory) [6], GANs [7, 8], and other LSTM based models [6, 4, 9, 10, 11, 12, 13, 14, 15].

For music generation task, recent research of feedforward models like CNN [1, 2, 16] gives us a new idea to overcome the music quality problems. They find CNN is more resistant to the gradient vanish. Besides, its complex structure let this model be an ideal choice to fit a large dataset. However, in practice, the music generated by plain CNN or ResNet like CNN [17] lacks overall consistency. CNN models generate lots of resting notes and super long notes.

In fact, unlike the LSTM, CNN using shared-weight filters is not suitable for extracting temporal features in music. Different regions of a music sequence have different local statistical characteristics. Therefore, the spatial stationary assumption of convolution cannot hold. To remedy this drawback of CNN, we introduce the Locally Connected Convolutional Neural Network [18, 19, 20] to sequence generation model. Because each timestep of the input sequence should have different characteristics, different parts in a specific sequence should not share the same kernel. For example, nearly every song follows specific rules of chord progression, which is a succession of musical chords typically in a unit of phrase. "C G Am E" is a common one of them, each chord corresponds to a bar. As Figure 1 shows, it is better for the first C major chord bar and the last E minor chord bar use different CNN kernels (Local Conv).

In our experiment, simple 3-layer CNN is used to explore the best strategy to add Local Conv layer; then we use this strategy to design resNet with Local Conv. We conclude the performance of all the CNN models in our paper with Local Conv layer outperform naive CNN models and LSTM model.

2. METHODS

2.1. Auto-regressive Model

In this paper, we leverage a simple auto-regressive model to let CNN generate the sequence. Define a piece of music is a sequence of music events, denote E as the set of music events, $E = \{E_1, E_2, \dots, E_m\}$, where m is determined by the range of notes. Define X_t as the random variable corresponding to the music event at time t. For CNN model, it predicts the next

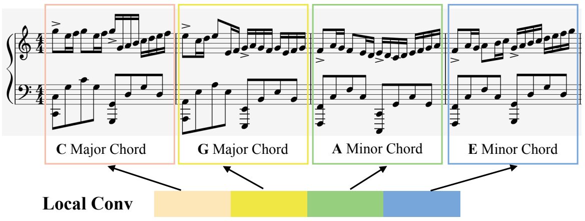


Fig. 1. Different parts of the music sequence have different characteristics. Thus, it is necessary to add different kernels into CNN(Local Conv) according to the time step.

notes by using k (time-step) previous music events. In a word, X_t is conditioned on $\{X_{t-k}, X_{t-k+1}, \ldots, X_{t-1}\}$. For a k-dimensional vector $\mathbf{a}$, for $S = \{1, 2, \ldots, m\}^k$, there should be:

$$P(X_t = E_i) = \sum_{\mathbf{a}\in S} P(E_i|(E_{a_{t-1}}, E_{a_{t-2}}, \ldots, E_{a_{t-k}})) \quad (1)$$

2.2. Locally Connected Convolutional Layer

CNN is the core of the aforementioned auto-regressive model, and Local Conv is the core of CNN. The Local Conv layer works similarly to the naive convolutional layer, except that weights are not shared. That is, a different set of filters is applied at each different patch of the input. Generally, for the sequence generation task, naive convolutional layer expands its kernel through channel dimension (Figure 2 (a)) while Local Conv layer expands its kernel from both channel and time step dimension (Figure 2 (b)).

Denote the input and output of CNN layer as F_{in} with N time step, and F_{out} with M time step respectively:

$$F_{in} = \{x_1, x_2, \ldots, x_N\}, \quad F_{out} = \{y_1, y_2, \ldots, y_M\} \quad (2)$$

Define the naive convolutional operation as f, CNN kernel size as K, padding as P, stride as S, so $i \in [1, N]$, $j \in [1, M - K]$, $j = (i - K + 2P)/S + 1$. The naive CNN mapping can be described as:

$$y_i = f(x_j, x_{j+1}, \ldots, x_{j+K-1}), \quad (3)$$

For a Local Conv operation, there will be M Local Convolutional operation f_i:

$$y_i = f_{\mathbf{i}}(x_j, x_{j+1}, \ldots, x_{j+K-1}) \quad (4)$$

2.3. Small Models

We use three 3-layer CNN models to decide where to put the Local Conv layer. For a Local Conv layer, let the previous layer's feature dimension as d, Local Conv layer's kernel size as k, channel number as n, stride as s, the number of weights W is decided by the following formula:

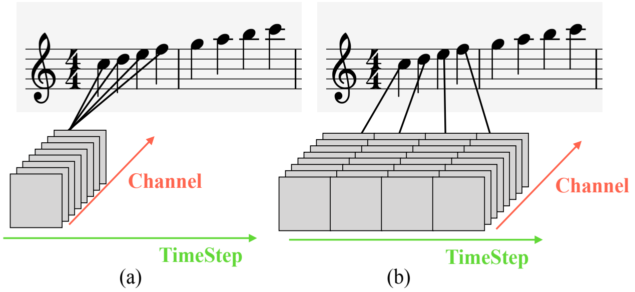


Fig. 2. Dimension for convolutional operation on music data. In (b), the Local Conv model adds more kernels through the time dimension which performs much better than traditional convolutional operation like (a).

$$W = \frac{d \times k \times n}{s} \quad (5)$$

From the Figure 3, we compare 3 main types of Locally Connected version CNN with naive CNN. All the general convolutional blocks can be replaced by Local Conv blocks like **AL-CNN** (All Local CNN). As Taigman mentioned in [20], Local Conv brings more weights to CNN model. It is necessary to mix Local Conv with naive CNN model to reduce this burden. So, we designed other two structure of CNN as **FL-CNN** (Front Local CNN) and **BL-CNN** (Back Local CNN). In general, FL-CNN model use much fewer weights than AL-CNN and BL-CNN because, from equation (5), the size of d tends to be smaller in the shallower layer.

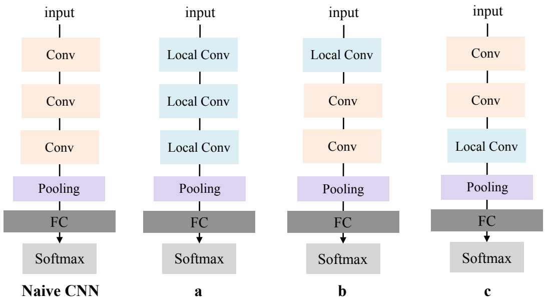


Fig. 3. Naive CNN and Different Local Conv CNNs.

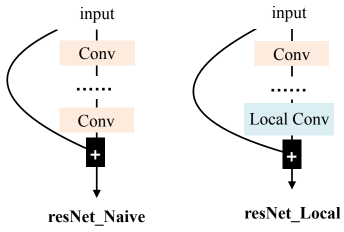


Fig. 4. Naive ResNet and Local Conv resNet.

2.4. ResNet Models

It is a natural idea putting the Local Conv to one of the most popular CNN model ResNet [17].

ResNet can be divided into a series of residual blocks. We list one block as an example to discuss our method to design resNet with Local Conv layer (Blue rectangle) in Figure 4. For deep ResNet, it is impractical to replace all the ordinary CNN layer with Local Conv layer. Therefore, Local Conv layers should be used as little as possible. Our later experiment shows the **BL-CNN** balance the weights cost and model's performance effectively, so Local Conv layer is put to the last layer of each basic ResNet block like ResNet_Local.

3. EXPERIMENTS

3.1. Dataset and Preprocessing

We choose Bach's polyphonic dataset [21] which contains 141 midi files and Wikifonia monophonic dataset [22] which contains 6,675 Lead Sheets as the datasets. We use tools from Google Magenta project to process music data. Each note event is extracted as a 38-dimension vector for monophonic melody generation, then all the notes' pitch is shifted to the range of C3 to C6. In our method, a 38-dimension vector comprises of 36 note-on events, 1 note-off event and 1 no-event [23].

As for polyphony music generation, we transform all the midi files to a python-list like object, utilizing Google Magenta's polyphony music encoder-decoder. Each number in the list refers to a combination of musical events. The detailed encode-decode algorithm is available Google Magenta's Polyphony RNN project [23]. For multiple notes in the same time-step, we will list them continuously in the descending order, and separate time-steps using a special event marked as an integral number.

Because our pipeline is effectively identical to the ubiquitous music generation models, we will omit an exhaustive background description of the detailed experiment setting and refer readers to [23].

3.2. Convergence Analysis

We compare the performance of 6 CNN models illustrated in Figure 2 and Figure 3 and the 1-layer LSTM model with 512 units:
(1) 3-layer Naive CNN model (Naive_CNN)
(2) 3-layer All Local CNN (AL_CNN)
(3) 3-layer Front Local CNN (FL_CNN)
(4) 3-layer Back Local CNN (BL_CNN)
(5) Naive 20-layer ResNet (Naive_ResNet20)
(6) Local Conv 20-layer ResNet (LocalConv_ResNet20)
(7) 1-layer LSTM with 512 units (LSTM).

From Table 1 and 2, comparing simple Local Conv models including AL_CNN, FL_CNN, BL_CNN and Naive_CNN, we find 3 Local Conv CNN models outperforms their naive version. Because $loss_{\ A} < loss_F < loss_B$ and we know $W_A > W_F > W_B$ from equation (5) as the d tends to be smaller in the shallower layer. When we design resNet_Local, BL_CNN is chosen as an effective structure (placing Local Conv to the last layer) for which balance the performance and weights cost.

Table 1. Monophonic music generation using Wikifonia dataset. We list the training loss; the convergence epoch (which epoch the training accuracy $\geq$ 0.85, otherwise, the convergence epoch = -1). The speed is tested by measuring the time cost for each model predict one music event on a GTX 1080Ti GPU. The best results are highlighted in **boldface**. ("ms/step" stands for the time cost of predicting one music event).

Model	Loss	Convergence Epochs	Time Cost (ms/step)
AL_CNN	0.4206	**60**	10.30
FL_CNN	0.7055	-1	8.99
BL_CNN	0.5791	-1	7.04
Naive_CNN	0.9023	-1	**5.58**
Naive_ResNet20	0.6179	-1	23.24
LocalConv_ResNet20	**0.0079**	61	48.66
LSTM	0.1890	64	62.26

Table 2. Polyphonic music generation using Bach dataset.

Model	Loss	Convergence Epochs	Time Cost (ms/step)
AL_CNN	0.1736	59	103.69
FL_CNN	1.1107	-1	88.50
BL_CNN	1.1088	-1	94.04
Naive_CNN	2.9557	-1	**74.10**
Naive_ResNet20	1.0802	-1	164.46
LocalConv_ResNet20	**0.1695**	**21**	191.57
LSTM	0.3138	89	313.96

LocalConv_resNet20 model shows a more obvious improvement (loss=0.0079, 0.1695) than its naive version (loss=0.6179, 1.0802).

From Table 1 and 2, no matter monophonic or polyphonic music generation task, the Local Conv models show three advantages:
(1) The loss of Local Conv Local Conv model is lower than LSTM and naive CNN models. For most of the music generation model like [12, 6, 9, 11, 13, 10, 24], a smaller training loss means better music the model will generate.
(2) Local Conv models converge faster than the LSTM and naive CNN model.
(3) The Local Conv CNN models run 2-3 times faster than the LSTM model.

In fact, better performance of Local Conv may be brought by its large number of weights through time-step dimension; therefore, we let all the CNN models have a close number of weights by adjusting the number of channels.

Table 3. Control the number of weights in each model

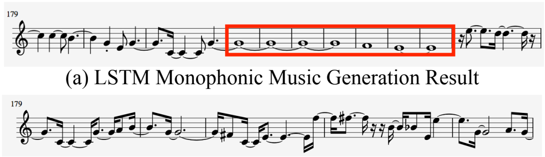
Model	Loss	Convergence Epochs	Weights (million)
AL_CNN_Large	**0.0092**	**6**	14.26
FL_CNN_Large	0.0126	7	12.25
BL_CNN_Large	0.0453	13	12.57
Naive_CNN_Large	1.2321	-1	10.86
Naive_ResNet20_Large	1.1524	-1	13.42
LocalConv_ResNet20	0.1695	21	12.64

Through the same polyphonic music generation experiment, from Table 3, all the LocalConv models supersede the naive CNN models with a similar number of weights. This result excludes the possibility that the improvement is brought by the increment of weights. For music generation task, models with more weights in the time step dimension (as Figure 2 (b) shows) perform better. In short, to improve the performance of CNN in music generation task, the Local Conv can be used in most layer of CNN architecture, and resNet_Local model in our paper is an ideal default choice.

3.3. Music Sheets Analysis and User Study

We notice the Local Conv model outperform the LSTM model and naive CNN model, especially when generating a long sequence. In the monophonic music generation task, we clip the generated music from the 179_{th} bar as Figure 5 shows. The Local Conv model's result is more diverse and meets music theory.

For polyphonic music generation task, the improvement of Local Conv is more obvious. Usually, it is hard for LSTM model to generate super long music sheet. The Local Conv model generates Bach-style result even when it generates long sequence (more than 100 time steps). However, the LSTM model generates some undesired stochastic result. Music sheets are showed in Figure 6. Each clip consists of three four-bar phrases generated by one of the proposed models and quantized by sixteenth notes.

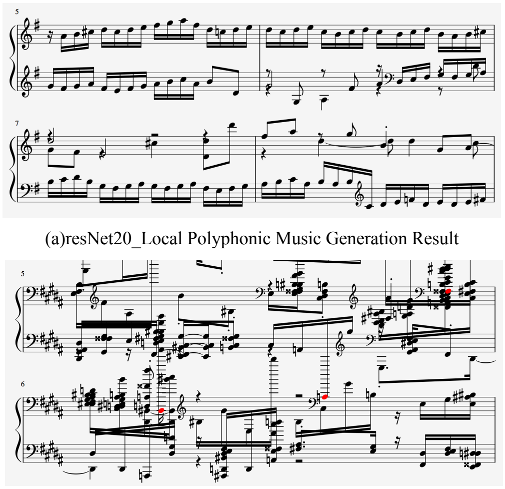


(a) LSTM Monophonic Music Generation Result

(b) resNet20_Local CNN Monophonic Music Generation Result

Fig. 5. Music Generated by single track music generation model trained on Wikifonia dataset.

As for user study, 10 music samples are selected from each model's generated result: our LSTM model, ResNet20_Naive model and ResNet20_LocalConv model, all trained from monophonic music generation task. Each music sample is clipped to 8 bar length. Via a survey website, 45 testers are required to judge the harmony and reality of all the music in a random order. From Figure 7, the ResNet20_LocalConv model outperforms ResNet20_Naive models and LSTM model (Pearson's chisquared test, for harmony score, $p = 1.95 \times 10^{-5}, 6.74 \times 10^{-3}$ respectively; for reality score, $p = 5.53 \times 10^{-3}, 5.50 \times 10^{-2}$ respectively).



(a)resNet20_Local Polyphonic Music Generation Result

(b) LSTM Polyphonic Music Generation Result

Fig. 6. Music generated by the polyphonic model trained on Bach dataset when the generate-length is more than 100 time steps.

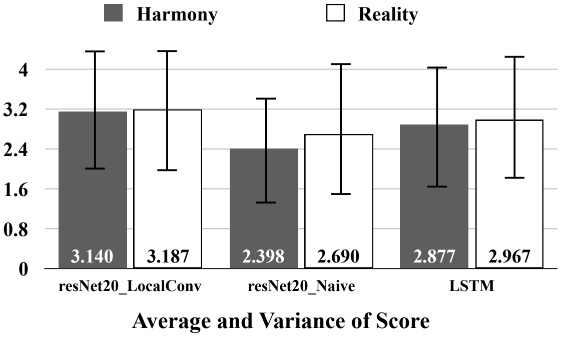


Fig. 7. User study result. The grey and white bars represent the harmony and reality score. All the scores range from 0-5.

4. SUMMARY AND FUTURE WORK

In this paper, we introduce the Local Conv structure which boosts the performance of CNN in music generation tasks. With temporal dependency, Local Conv CNN can generate relatively long sequences with pleasing structures. Different models are designed to utilize the feature of Local Conv. Next, we compare different model's result in two music generation tasks: monophonic and polyphonic music generation. The Local Conv version CNN models outperform Naive CNN models and LSTM model in many aspects such as lower loss, faster convergence, faster speed and better music quality. As music generation task is similar to many other NLP problems, we believe the improvement brought by Local Conv structure can be applied to other NLP tasks like language modeling.

5. REFERENCES

[1] Shaojie Bai, J Zico Kolter, and Vladlen Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[2] John Miller and Moritz Hardt, "When recurrent models don't need to be recurrent," *arXiv preprint arXiv:1805.10369*, 2018.

[3] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proc. AAAI Conf. Artificial Intelligence*, 2018.

[4] Gaëtan Hadjeres, François Pachet, and Frank Nielsen, "Deepbach: a steerable model for bach chorales generation," *arXiv preprint arXiv:1612.01010*, 2016.

[5] Andries Van Der Merwe and Walter Schulze, "Music generation with markov models," *IEEE MultiMedia*, vol. 18, no. 3, pp. 78–85, 2011.

[6] Hang Chu, Raquel Urtasun, and Sanja Fidler, "Song from pi: A musically plausible network for pop music generation," *arXiv preprint arXiv:1611.03477*, 2016.

[7] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.

[8] Hao-Wen Dong and Yi-Hsuan Yang, "Convolutional generative adversarial networks with binary neurons for polyphonic music generation," *arXiv preprint arXiv:1804.09399*, 2018.

[9] Natasha Jaques, Shixiang Gu, Richard E Turner, and Douglas Eck, "Tuning recurrent neural networks with reinforcement learning," 2017.

[10] Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Jonas Wiesendanger, "Jambot: Music theory aware chord based generation of polyphonic music with lstms," in *Tools with Artificial Intelligence (ICTAI), 2017 IEEE 29th International Conference on*. IEEE, 2017, pp. 519–526.

[11] Andrew Shin, Leopold Crestel, Hiroharu Kato, Kuniaki Saito, Katsunori Ohnishi, Masataka Yamaguchi, Masahiro Nakawaki, Yoshitaka Ushiku, and Tatsuya Harada, "Melody generation for pop music via word representation of musical properties," *arXiv preprint arXiv:1710.11549*, 2017.

[12] Huanru Henry Mao, Taylor Shin, and Garrison Cottrell, "Deepj: Style-specific music generation," in *Semantic Computing (ICSC), 2018 IEEE 12th International Conference on*. IEEE, 2018, pp. 377–382.

[13] Hongyuan Zhu, Qi Liu, Nicholas Jing Yuan, Chuan Qin, Jiawei Li, Kun Zhang, Guang Zhou, Furu Wei, Yuanchun Xu, and Enhong Chen, "Xiaoice band: A melody and arrangement generation framework for pop music," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2837–2846.

[14] Douglas Eck and Juergen Schmidhuber, "A first look at music composition using lstm recurrent neural networks," *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, vol. 103, 2002.

[15] Allen Huang and Raymond Wu, "Deep learning for music," *arXiv preprint arXiv:1606.04930*, 2016.

[16] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio.," in *SSW*, 2016, p. 125.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] Karo Gregor and Yann LeCun, "Emergence of complex-like cells in a temporal product network with local receptive fields," *arXiv preprint arXiv:1006.0448*, 2010.

[19] Gary B Huang, Honglak Lee, and Erik Learned-Miller, "Learning hierarchical representations for face verification with convolutional deep belief networks," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2518–2525.

[20] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[21] I Liu, Bhiksha Ramakrishnan, et al., "Bach in 2014: Music composition with recurrent neural network," *arXiv preprint arXiv:1412.3191*, 2014.

[22] Jian Wu, Changran Hu, Yulong Wang, Xiaolin Hu, and Jun Zhu, "A hierarchical recurrent neural network for symbolic melody generation," *arXiv preprint arXiv:1712.05274*, 2017.

[23] Contributors, "Google magenta," in *https://github.com/tensorflow/magenta*, 2018.

[24] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck, "A hierarchical latent vector model for learning long-term structure in music," *arXiv preprint arXiv:1803.05428*, 2018.