AUTO-SUSPED: LEARNING PIANO PEDALING IN STYLE

Wonjun Yi
KAIST School of Electrical Engineering
lasscap@kaist.ac.kr

Pilsun Eu
KAIST School of Electrical Engineering
pilz@kaist.ac.kr

ABSTRACT

Research in automatic piano music generation has only recently started to involve piano pedals as a part of the task. In this work, we train various neural network architectures with piano sustain pedal control-change (CC) data using different categories within the MAESTRO classical piano music dataset to study the performances of basic models and test the suitability of neural networks in an automatic piano pedal styling task. By changing the ***temporal scanning range*** of convolution kernels and the depth of the network structure, we show that both factors are relevant in the accuracy of pedaling style prediction. Currently, our best CNN-based Auto-SusPed model predicts a specific composer's pedaling style and a specific musical era's style with accuracies of around 90%.

1. INTRODUCTION

Pedaling techniques have been continuously developed by pianists over the centuries for additional expressive dimensions that would take performances above and beyond what is possible with just presses of keys. The modern piano now typically comes with three types of pedals, used to emphasize, express and ease different parts of the performance; the *soft pedal*, the *sostenuto pedal* and the damper pedal also known as the *sustain pedal*. In particular, the sustain pedal is the most frequently used pedal, and has the simple function of allowing the sound to freely vibrate even if the note is released, as long as the pedal is depressed. Famously described by the pianist Anton Rubinstein as "the soul of the piano"[1], the use of the sustain pedal contributes significantly in defining a performer's style, capturing the characteristics of a musical era, or setting specific moods. It is evidently an attractive idea to be able to harness this "soul" and apply it to recordings or tracks yet void of style.

Despite its principal role in piano music, there is surprisingly little work in the field related to the information retrieval and handling of piano pedaling. Even among the impressive feats achieved within the last 30-or-so years with automatic music transcription (AMT), pedal transcription has only been receiving attention relatively recently with notable benchmark work appearing as late as 2018[2]. Most importantly - to the best of our knowledge - the *generation* of piano music with appropriate pedaling using neural networks is an application of deep learning that is still at its inception.

Of course, the subtlety in the use of pedals for any piano performance is a heavily subjective matter with no single correct answer. This means attempts of training models to learn *good* pedaling patterns may be difficult or downright meaningless. But if the task can be narrowed down to a matter of recreating the *style* of another performance, then there may be a more reachable goal at hand. This has natural applications where style imitation may be desired, such as producing musical scores with pedal indications that reflect on the early-Romantic techniques. Such a job would otherwise require years of musical experience and knowledge, if done by hand.

In this work, we aim to extract the characteristic aspects that differentiate piano performances by using their pedaling styles, and propose an automatic pedal styling model by comparing the performance between various types of network architectures when applied to the task. While all three types of piano pedals can be used to play in certain styles, we only handle data and results related to the sustain pedal in this study, in order to focus on the feasibility of the defined problem on the plate; no disrespect to the art involved with the use of the other two pedals.

2. RELATED WORKS

In the field of AMT, there are recent works that have proposed models that detect sustain pedal onset times using machines learning approaches[2] (Liang et al., 2018), as well as neural network approaches[3] (Liang et al., 2019), including a work which released a neural network-based open source program that handles both note and pedal transcription simultaneously[4] (Kong et al., 2021).

The last AMT model was used to process data samples in a research[5] that brought to attention the rarely discussed subject of generative models for piano pedals. The authors this work trained a Transformer model with the AILabs1k7 pop music MIDI dataset, and showed that acceptable results could be achieved with using neural networks for automatic piano pedaling.

Our work attempts to extend this work to classical music, and more specifically to extracting different styles of pedaling rather than generation of generally agreeable rendering of undecorated tracks. We expect this task to face more challenges compared to pop music pedal styling due to the complexity of pedaling involved with classical performances, compared to pop music. To this end, we explore various types of models, to tap into the relationship between pedaling and performance style, and how models could exploit them.

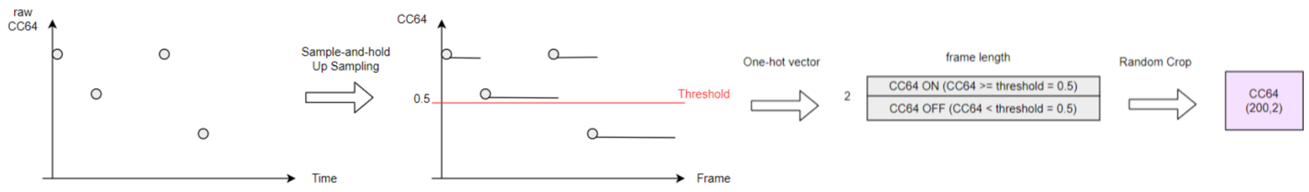

raw CC64
Time
Sample-and-hold Up Sampling
CC64
0.5
Threshold
Frame
One-hot vector
frame length
2
CC64 ON (CC64 >= threshold = 0.5)
CC64 OFF (CC64 < threshold = 0.5)
Random Crop
CC64 (200,2)

Figure 1. Preprocessing MIDI CC training data.

3. DATA PROCESSING

3.1 Dataset

We use the MAESTRO V3.0.0 dataset provided by Magenta, composed of ~200 hours of professional piano performance recordings with corresponding MIDI transcriptions.[6] The MIDI data includes all three types of piano pedal signals generated with pedal detection and aligned with transcribed notes. The agreement between the recording and MIDI note labels are accurate to within 3 ms. The MIDI note data is loaded in the form of piano rolls by utilizing the pretty_midi python library with each file's dimension size determined by time sequence length and 88 piano keys.

3.2 MIDI CC

The pedal depression level data over time is stored within the MIDI file as control change (CC) messages. In particular, the sustain pedal information stored in CC64, with the pedal depression represented as integer values between 0 and 127. For simplicity of training we threshold the sustain pedal values, treating depression levels higher than 64 as '0' or 'on', and those lower than 64 as '1' or 'off'.

It is noted that the MIDI CC messages are not provided as a series of values over regular time intervals, but as flags only at time indices where there are change in the pedal depression level. To align the raw CC messages with note transcription, the data is up-sampled to fill the time gap before thresholding by sample-and-hold method, where each sample is repeated over the unspecified time indices until the next CC message. This process is shown in Figure 1. A random crop of the CC messages within the entire time sequence is used as test data. In Figure 1, CC64 (200, 2) refers to 200 time frames, corresponding to 5 seconds at a sampling rate of 16 kHz, and the two states 'on' and 'off', respectively.

3.3 Labeling by style

In order to organize the dataset more appropriately to our aim of training models to extract certain styles of pedaling, we selected two categories that the MAESTRO data could be labeled with: composer and era. The composer of each piano music was labeled using the tags readily available with the dataset. The era of each piano music was labeled with one of five broad musical eras (baroque, classical, pre-1900 romantic, post-1900 romantic, modernist) based on the general musical consensus of which era the piece's composer is most identified with. This was mostly related to the period of time during which the given composer was most active in their musical work.

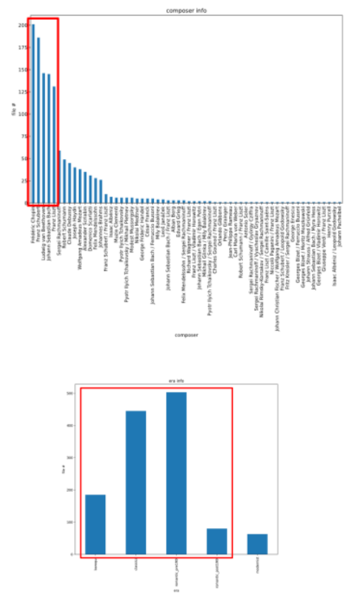
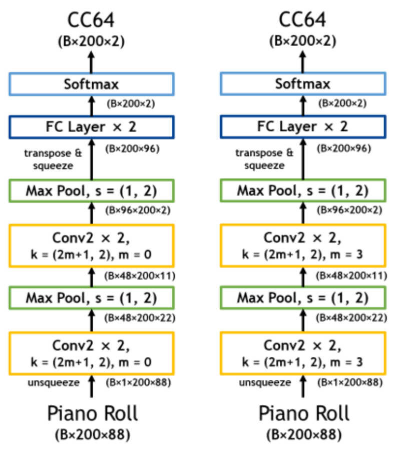


Figure 2. Labeling MAESTRO dataset by composer (up) and by era (down)

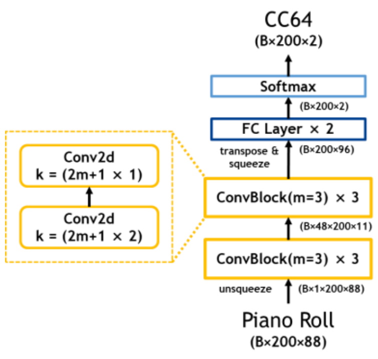
As shown in Figure 2, the MAESTRO data is unbalanced in terms of the categories for which piano music is available, for both composer and musical era. To make sure we work with large enough dataset for each label, we chose the top 5 composers (Chopin, Schubert, Beethoven, Bach, Liszt) and top 4 eras (baroque, classical, pre-1900 romantic, post-1900 romantic) which are labels with more than 100 data. Finally, we partition each of the composer and era dataset into training, validation and test dataset in an 8:1:1 ratio.

4. EXPERIMENT

4.1 Model Architecture



(a) (b)



(c)

Figure 3. MIDI CC generator architecture.
(a) CNN model, m = 0; (b) CNN model, m = 3;
(c) Deep CNN model, m = 3.

To approach the task, we tested three different convolutional neural network (CNN) based architectures, whose diagrams are shown in Figure 3. For each model, the ***temporal scanning range*** denoted by ***m*** refers to the width of the kernel towards either side from the center, in the direction of time. This is effectively how widely the model can see in time when learning from the MIDI data, in terms of number of frames.

The baseline model, Figure 3(a), is a simple CNN model with 4 convolution layers and 2 max pooling layers followed by 2 fully-connected layers (FC). Each convolution layer is followed by batch normalization and ReLU activation, every max pooling layer and FC layer is followed by a dropout with p = 0.5, except for the last FC layer. Starting with an input of (200, 88) piano roll, the 88 piano key dimension is reduced into a size of 2 – the on and off states for the pedal.

From the baseline model, we modify the architecture by changing the scanning range m, Figure 3(b), and the network depth, Figure 3(c), to test how they affect the results.

4.2 Training

Each proposed model is trained over 10000 iterations using batches of size 16, using the Adam optimizer with a learning rate of 6×10^{-4} and a scheduler of step size 1000, decay factor of 0.98, with the softmax loss used as the loss function. During training, we save the trained model every 1000 iterations and select the model with the highest validation accuracy. The models were trained using the NVIDIA GeForce RTX 3080 GPU.

4.3 Evaluation

The *accuracy* of the predicted CC sequence compared to the test data was used as the metric, defined by the ratio of frames with correct pedal on/off state to the total number of time frames in the piano music.

5. RESULTS

5.1 On/Off pedal detection results

Table 1 and 2 show the performance of each model in generating MIDI CC with styles of different composers and musical eras. The result is visualized using error maps that highlight the frames where the model has made incorrect predictions compared to the ground truth, as in Figure 4.

Model	Liszt	Schubert	Chopin	Bach	Beethoven
CNN(m=0)	0.8309	0.7560	0.8120	0.8826	0.7691
CNN(m=3)	0.9119	0.8763	0.9008	0.9145	0.8762
DeepCNN	**0.9120**	**0.8887**	**0.9073**	**0.9267**	**0.8860**

Table 1. Average accuracy; by composer

Model	Baroque	Classical	Romantic (pre-1900)	Romantic (Post-1900)
CNN(m=0)	0.8805	0.7369	0.8184	0.8690
CNN(m=3)	0.9100	0.8655	0.9059	0.9221
DeepCNN	**0.9204**	**0.8892**	**0.9154**	**0.9245**

Table 2. Average accuracy; by era

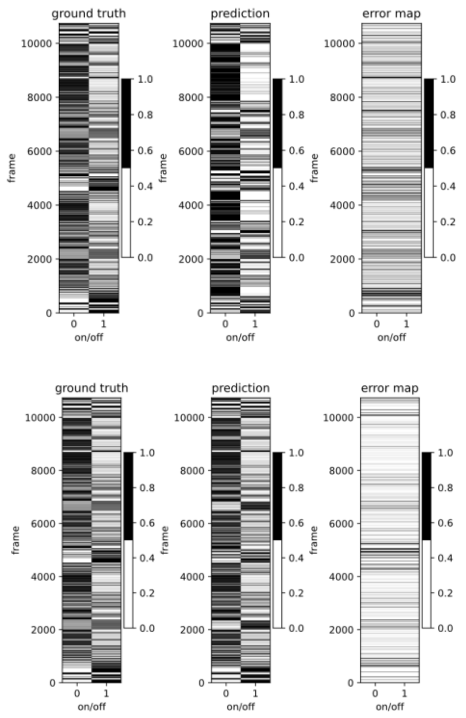


Figure 4. Example error map; baseline CNN model (up), deep CNN model (down).

As an example, Figure 4 shows that the error map for the baseline CNN model is darker than the deep CNN model, indicating a higher rate of prediction error.

5.2 Multi-level threshold pedal detection results

In classical piano music, it is common for performers to employ different levels of sustain pedal depressions, putting the pedal dampers in a spectrum of states, beyond simply up or down. The four most commonly known pedal techniques include quarter, half, three-quarters and full pedal[7] – this is an intuitive lead for a more flexible way of training MIDI CC generation models.

To this end, we test the same models, under the same environment with only minute modifications, such that they can handle MIDI CC data with 4 pedal level intervals instead of just two states, on and off. The four intervals correspond to CC64 messages values between (0, 31), (32, 63), (64, 95) and (96, 127). The results are shown in Table 3 and 4.

Model	Liszt	Schubert	Chopin	Bach	Beethoven
CNN(m=0)	0.4733	0.4058	0.4149	0.4245	0.4415
CNN(m=3)	0.5343	0.5242	0.5231	0.4603	0.5260
DeepCNN	**0.5523**	**0.5448**	**0.5284**	**0.4677**	**0.5468**

Table 3. Average accuracy; by composer, multi-level.

Model	Baroque	Classical	Romantic (pre-1900)	Romantic (Post-1900)
CNN(m=0)	0.4434	0.3753	0.4187	0.4861
CNN(m=3)	0.4670	**0.5142**	0.5093	**0.5496**
DeepCNN	**0.4774**	*OOM*	**0.5263**	0.5368

Table 4. Average accuracy; by era, multi-level. (*OOM: out of memory)*

The predictions in comparison to the ground truth in four different levels are clearly visualized in Figure 5.

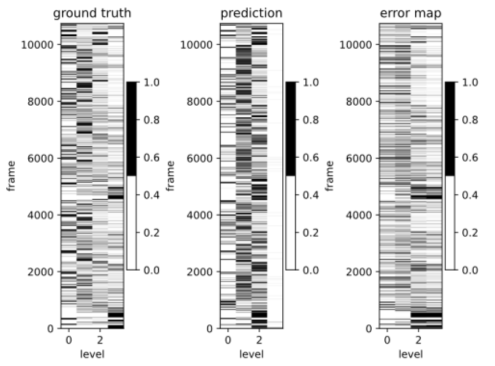


Figure 5. Prediction and error map, multi-level; deep CNN model. Levels: 0 = full pedal; 3 = no pedal.

5.3 Unseen data test

To observe the performance with an unseen label, a piece of music by Rachmaninoff was tested in composer-trained deep CNN models. The results are shown in Table 5.

Model	Liszt	Schubert	Chopin	Bach	Beethoven
Accuracy	**0.9469**	0.9431	0.9438	0.8960	0.9382

Table 5. Composer-trained model performance on unseen Rachmaninoff piano music

6. DISCUSSION

Even with a basic CNN architecture, the MIDI CC generator shows reasonable accuracy of more than 73%. Table 1 and 2 show that the performance increases significantly with wider scanning range *m* and deeper architecture for both tasks involving composer and era. It is evident that the appropriate pedaling requires some finite range of instantaneous temporal context, and that the relationship between pedaling patterns and notes are based on complex interpretations. The models perform better on musical eras than on composers, perhaps due to the larger era dataset per label, and the high specificity of a composer's style compared to a general era. In the case of testing an unseen label, the predictions had high accuracy, especially for Liszt, who interestingly happens to have been more musically active during the time closest to Rachmaninoff than any of the other composers.

Despite reasonable performance, there is a common issue of fast convergence, as shown in Figure 6, meaning that the models stop learning very early on. Moreover, as seen in error maps such as in Figure 4, there tends to be inaccuracies within sections where the ground truth pedal is pressed for particularly long periods of time; the models often prefer to pedal much more actively within any given interval of time. These observations point out that the models are not quite able to capture the meaning of pedaling at certain musical phrases, or rather not pedaling too much, and have not been provided with the means of learning it.

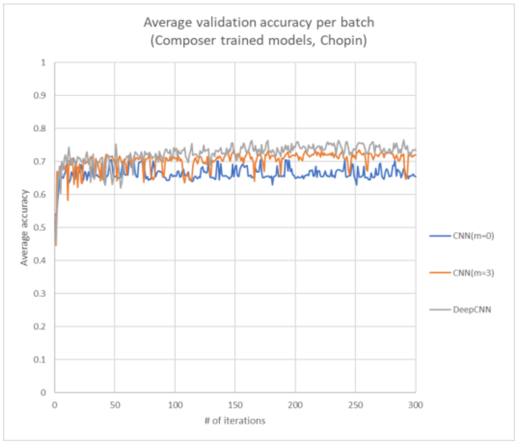


Figure 6. Validation accuracy change over iterations

When the MIDI CC data is divided into multiple levels, our approach proved to be lacking. The error map in Figure 5 reveals poor prediction between full and three-quarters pedaling. It is likely that this occurs because the models are not provided with much demerit when they fail to make subtle decisions between different pedaling techniques.

There is certainly motivation to think that musically-aware changes will improve model performance. Unlike a series musical notes, which are independent musical elements, pedaling is completely meaningless without the concurrent musical notes. Semantically, it is necessary to provide direct links between pedaling and the note information, such as rhythm. Also, as the primary purpose of sustain pedals is to control the extensions and breakings of note lengths, it may be advisory to factor in whether the pedal is pressed during or in between notes - or even the number of notes played at the time, as pedals are also often used to layer many notes.

Aside from the lack of semantic aid for the model to gain more 'musical insight', there were also some minor issues, involving corner cases in terms of data preprocessing and heavy model parameters (ref. Table 4)

We are optimistic in that further research into this seldom explored field could lead to a useful tool for both composers and players, as well as provide a puzzle piece for a more complete picture of what defines a piano performance style, an insight that could also be used in reverse in the field of automatic music generation.

7. AUTHOR CONTRIBUTIONS

"Wonjun Yi" processed the MIDI dataset, designed and constructed the MIDI CC dressing models, conducted experiments and edited the presentation video. "Pilsun Eu" studied the task context from previous works, provided general directions related to musical aspects and MIDI handling, drafted the final report and presentation.

8. REFERENCES

[1] Rosenblum, Sandra P. "Pedaling the piano: A brief survey from the eighteenth century to the present." *Performance Practice Review 6*, no. 2 (1993): 8.

[2] Liang, Beici, György Fazekas, and Mark Sandler. "Piano legato-pedal onset detection based on a sympathetic resonance measure." In *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 2484-2488. IEEE, 2018.

[3] Liang, Beici, György Fazekas, and Mark Sandler. "Piano sustain-pedal detection using convolutional neural networks." In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 241-245. IEEE, 2019.

[4] Kong, Qiuqiang, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. "High-resolution Piano Transcription with Pedals by Regressing Onset and Offset Times." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2021).

[5] Ching, Joann, and Yi-Hsuan Yang. "Learning To Generate Piano Music With Sustain Pedals." *arXiv preprint arXiv:2111.01216* (2021).

[6] Magenta, *The MAESTRO Dataset*. Accessed 2021.12.17. https://magenta.tensorflow.org/da-tasets/maestro

[7] Liang, Beici, György Fazekas, and Mark Sandler. "Measurement, recognition, and visualization of piano pedaling gestures and techniques." *Journal of the Audio Engineering Society 66*, no. 6 (2018): 448-456.