CS229 Final Project Report Automatic Music Transcription for Monophonic Piano Music via Image Recognition

Chen Cen ccen@stanford.edu

An Jiang jianga@stanford.edu

December 15, 2017

I. Introduction

Music transcription has been a long-time challenging task even for human. It takes a significant amount of time and effort for an experienced musician to listen to a song or music and transcribe it into music sheets. Automatic Music Transcription (AMT) automates the process of transcribing musics and plays an important role in music information retrieval(MIR). Even though the research for AMT is still in infancy, the results so far have been proved to be very educational to both the areas of Machine Learning and Music Composition. In this project, we tried to tackle the problem of music transcription using a new approach proposed by [1] that transforms the music note detection problem into an image recognition problem using Convolutional Neural Network (CNN). We gathered our training and testing data from the MAPS database [2] which contains recordings and MIDI files of isolated piano notes and built upon the existing CNN image recognition algorithm with Tensorflow for the music transcription problem.

II. Related work

Some existing approaches for AMT include directly analyzing music notes by SVM [3] or HMM [4]. They tried to seperate the transcription problem into note and rhythem recgnition problem. Our approach is to use train a CNN model that is able to correctly label notes in the track from the spectrogram of it. To solve an image classification problem, Alex Krizhevsky proposed a multi-layer CNN architecture consisting of alternating convolutions and nonlinearities [5] which is already implemented in Tensorflow [6]. It is able to achieve 11% error with 75 minutes of training for the CIFAR-10 [7] classification problem. Our model architecture is developed on top of the structure used in the article.

III. Data and processing

We first started with MIDI and text files which contain the label, on and off time of each note (ground truth) and the actual sound recordings in wav format (input data). The MIDI files are processed with mido python library [8] to obtain the length of the track. The text files are transformed into note maps containing the label information for each existing note. From the note map, we constructed an array of integers that indicate the music note in MIDI note numbers which is present in the current time step. The wav files are processed with librosa python library [9] which is capable of audio analysis and music visualization. To map the frequencies of the music track better,

we used constant-Q transform instead of Short-Time Fourier Transform (STFT) to generate the spectrogram. From the diagrams below, we can see that constant-Q transform streches the low frequency domain which makes the notes more distinguishable and the difference in test accuracy using these two methods confirms our theory.

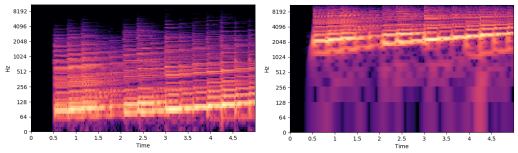

Figure 1: *Example of STFT (left) and Constant-Q (right) Spectrogram*

For each time step, we generate a 32x32 thumbnail of the spectrogram of the track and save it as a JPEG image. The images and the array of note labels for each time step are combined into a cPickle data package which will be fed into the training model directly. The data package will be a dictionary with *train*, *valid* and *test* keys to indicate the usage of each data set.

IV. Methodology

The preprocessed daset are fed into a ImageNet classification CNN model introduced in [10]. It is a multi-layer architecture consisting of alternating convolutions and nonlinearities. Specifically, it consists of two CNN layers with two fully connected layer with ReLU activation function, followed by a 128-way softmax output. The convolutional layer processses data only for its receptive field and reduce the number of parameters to learn. The following pooling layer combines the outputs of neuron clusters at one layer into a single neuron for the next layer. The norm layer is a special normalization layer introduced in the paper to aid generalization and to reduce overfitting.

Figure 2: *Original CNN structure*

The algorithm maximizes the multinomial logistic regression objective, which is equivalent to maximizing the average across training cases of the log-probability of the correct label under the prediction distribution. Softmax regression calculates the cross-entropy between the normalized predictions and a 1-hot encoding of the label abd applies a softmax nonlinearity to the output of the network. For regularization, the algorithm also applies the usual weight decay losses for all learned variables. The objective function for the model is the sum of the cross entropy loss and all these weight decay terms. The model is trained using stochastic gradient descent with a batch size of 128 examples, momentum of 0.9 and a weight decay of 0.0005. Instead of using sigmoid or tanh as the activation function, this model used ReLU nonlinearity ($f(x) = max(0, x)$) as the activation function for faster convergence.

Due to the large amount of parameters to learn by this net, some data augmentation is introduced to reduce overfitting. Specifically, random images are selected to be distorted in the following ways: reflecting the image horizontally, randomizing the images brightness and contrast, linearly scaling image to have zero mean and unit norm. Finally, 32x32 images are resized to 24x24. With these data augmentations, the average error decreased from 26% to 11% on the CIFAR-10 datasets.

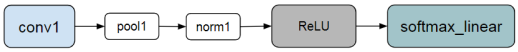

Figure 3: *Simplified CNN structure*

Initially we have conducted most of the experiments with the original CNN structure, but after few changes we are unable to raise the performance further. Comparing the image classification problem and note labeling problem, we decided to try to simplify the neural network since the spectrograms are more straightforward and more distinctive on the difference between different notes. So we reduced the CNN structure with only one set of convolutional, pooling, normalization and ReLU layers and trained it with the same dataset. To

our suprise, the performance of the simplified model does not suffer much loss compared to the original model.

V. Results

We have conducted several experiments on the image processing procedure, learning parameters and CNN structure to improve the performance of the model. All the data below are obtained by training the neural network with the same dataset with more than 8000 images after 5000 steps while the data are split for training and testing with the ration of 4:1.

Model	CE (L2)	Total Loss	Test Accuracy
Original CNN and parameters	0.356	0.52	0.782
STFT	0.581	0.916	0.516
Learning rate fast decay	0.228	1.23	0.76
No image distortion	0.192	0.356	0.773
Larger image size	0.14	0.23	0.823
Simple CNN	0.325	0.553	0.773
Simple CNN (10k steps)	0.151	0.297	0.78

Table 1: *Performance results from experiments*

We can see that the original network used for image classification is not performing so badly though still not as good as the accuracy achieved by the original article (86%). Thus, we have tested several methods with the hope to improve the result: to make the cross entropy converges faster, we tried to modify the decay rate of the learning rate; to reduce the noise of the input, we skipped the random distortion of the images before feeding to the model; to enhance the quality of the dataset, we increased the resolution of the images. As a result, improving the quality of the images have increased the test accuracy the most, to an extend of 82.3%, though the training time cost is much heavier since the dataset size have increased significantly.

From the results above we also noticed that the simple version of CNN does not underperform much from the original CNN model, especially after 10,000 steps of training. At the same time, the L2 loss and total loss for simple CNN converges to the same value and with the similar speed as the original CNN. As shown in the plots below, the simplified CNN even performs better in terms of the stability of the convergence for the total loss. Thus we concluded that the simple CNN model can achieve the same performance in music note classification problem, with the benefit of less time cost for training.

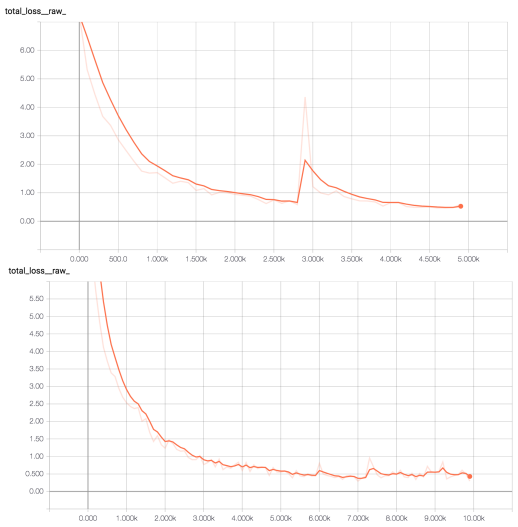

Figure 4: *Total loss vs. steps for original CNN (top) and simplified CNN (bottom)*

After training the simple CNN model, we utilized it to generate a MIDI track from a pinao recording of tonal scales as a demo example. The graph below shows the comparison of the original track and the generated track labeled by the CNN model. We found that although the accuracy is only around 80%, most of the mis-labeled notes do not deviate much from the corret labels, therefore maintaining the upward and downward trend of the tonal scale.

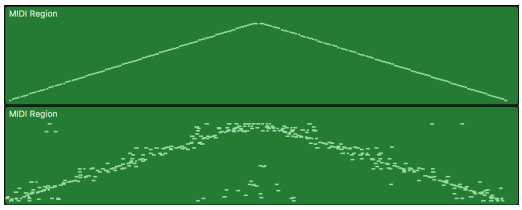

Figure 5: *Original (top) and generated (bottom) track*

VI. Conclusion

From the initial experiments on the original CNN mode, we have discovered that after we removed the image compressing process from 32x32 into 24x24 pixles, the overall training accuracy increased by 5 percent, from 78% to 83%. We believe the reason behind this is that the images processed by the original model are real world objects such as cats, dogs, ships etc. Given the complexity of these images, compressing or pooling process will not lose too much feature information and could even help the images in one category to be more distinguishable from other categories. However, our inputs or spectrograms are relatively simple images with limited color contrasts which have high similarities with each other. Therefore, further compressing them would lose some of the labeling information, making the machine learning model less accurate. Based on this observation, we later on simplified our model architecture to only 1 layer of CNN and 1 layer of local ReLU and the new model achieves similar results of the original model and takes less time to train in general.

VII. Future Steps

We believe there are still room for improvements on dataset processing and the CNN model to further increase the accuracy. We can apply more contrasts on the spectrogram to make the different notes more distinguishable. The neural network can be modified with more convolutional layers and less pooling layers to solve the note classification problem better. Furthermore, we can explore the polyphonic music transcription problem using the same methodology. The polyphonic music tracks can be split into monophonic tracks using Independent Component Analysis (ICA) or other machine learning algorithms and the same CNN method can be applied.

VIII. Contribution

Chen Cen: Worked on model training with Tensorflow.
An Jiang: Programmed the preprocessing and postprocessing Python script, edited most of the presentation and reports.
We both worked on the preliminary Music Information Retrival research and all the experiments on CNN model for the project. The project code base can be accessed at [11].

References

[1] D. Troxel, *17th International Society for Music Information Retrieval Conference*.

[2] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.

[3] G. E. Poliner, "Classification-based music transcription," 2008.

[4] E. Nakamura, K. Yoshii, and S. Sagayama, "Rhythm transcription of polyphonic piano music based on merged-output hmm for multiple voices," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 4, pp. 794–806, 2017.

[5] "cuda-convnet." `https://code.google.com/archive/p/cuda-convnet/`.

[6] "Convolutional neural networks." `https://www.tensorflow.org/tutorials/deep_cnn`.

[7] A. Krizhevsky, "The cifar-10 dataset." `http://www.cs.toronto.edu/~kriz/cifar.html`, 2009.

[8] O. M. Bjorndalen, "Mido - midi objects for python." https://github.com/olemb/mido.

[9] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, D. Ellis, F.-R. Stoter, D. Repetto, S. Waloschek, C. Carr, S. Kranzler, K. Choi, P. Viktorin, J. F. Santos, A. Holovaty, W. Pimenta, and H. Lee, "librosa 0.5.0," Feb. 2017.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[11] C. Chen and A. Jiang, "Cs229 project github page." https://github.com/flamearrow/229Project.