CS229 Project Report
Automatic Music Transcription for Polyphonic Piano Music

Chen Cen
ccen@stanford.edu

An Jiang
jianga@stanford.edu

November 21, 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 [11] To transform the music note detection problem into a image recognition problem using CNN network. We will generate our training data from the MIDI library [7] which contains hundreds of classical piano music and use existing CNN image recognition algorithm with Tensorflow to build and train our model.

II. Related work

Some existing approaches for AMT include directly analyzing music notes by SVM [10] or HMM [9]. Our approach is to use constant Q transform to convert notes into slices of spectrograms and train a CNN model to correctly categorize the note to one of 128 notes from the music track. Alex Krizhevsky proposed a multi-layer CNN architecture consisting of alternating convolutions and nonlinearities [2] which is already implemented in Tensorflow [1]. It is able to achieve the CIFAR-10 [5] classification problem with 11% error in 75 minutes of training.

III. Data and processing

For training set, we started with MIDI files which contain the note information of the music (ground truth) and can be transformed to sound tracks in mp3 format (input data). The MIDI files are processed with mido python library [3]. Since the format of MIDI files is not in time series, we utilized the *play* function in the library which plays the MIDI messages in the real time manner. This enabled us to record the start and stop time of each note in the file. Assuming a step size of 0.1 second, we can construct an array with 10 times of the music length in seconds and each note in the track can be filled in the array according to the time it is on. After processing the MIDI file, we will have an array of integers that indicates the music note present in the current time step (in MIDI note numbers, from 0 to 128). The mp3 files are analyzed with librosa python library [8]. The library is capable of audio processing and spectral transformations. To map the frequencies of the music track better, we used constant-Q transform to generate the spectrogram.

Then for each time step, we generate a 32x32

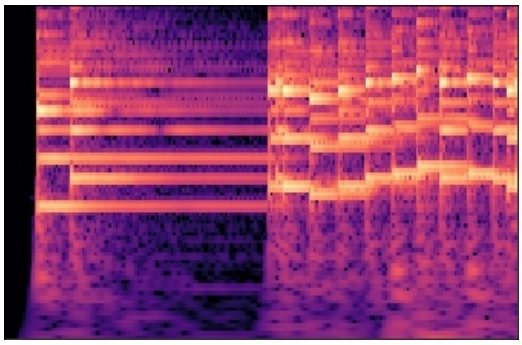

Figure 1: *Example of Constant-Q Spectrogram*

thumbnail of the spectrogram within the step 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

We first generate our training data from section III, then feed the image - label ground truth (an snippet of training data can be find in out Github website [4]) into a ImageNet classification CNN model introduced in [6]. This net contains eight layers with weights. The first five are convolutional neural networks that is composed of 60 million parameters and 650,000 neurons, some of which are followed by max-pooling layers, while the last three are fully-connected layers. The last layer is a final 1000-way softmax function that outputs the distribution over 1000 labels. For our project, we changed the softmax layer to 128 classes to match the number of representable notes from MIDI file format.

This network maximizes multinomial logistic regression or log-likelihood of the correct label under the prediction distribution. 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 non-linearity ($f(x) = max(0, x)$) as the activation function for faster convergence. The following figure from the paper compared the algorithm performance between ReLU and tanh activation functions on the CIFAR-10 dataset.

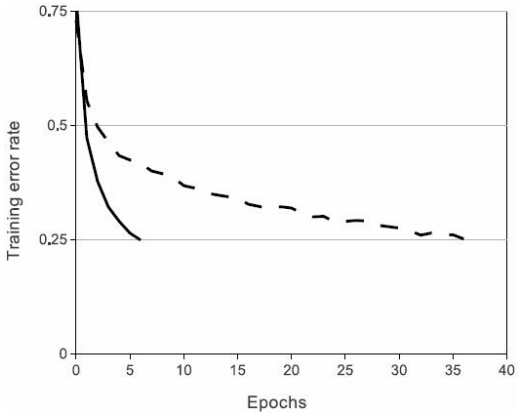

Figure 2: *Performance comparison between ReLU(dashed) and tanh(solid) as activation function*

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.

V. Example

For the development of the python and Tensorflow program, we first used a manually created MIDI file which is a track of a tonal scale from C3 to C7 with 0.5s length for each note. By using the step size of 0.1 second, we have a 245x1 array: [48,48, 48, 48, 48, 49, 49, 49, 49, 49, 50...96, 96, 96, 96, 96] which represents the playing note for each time step. The 245 thumbnail spectrograms for every 0.1 second of the track are also generated from the preprocessing python script.

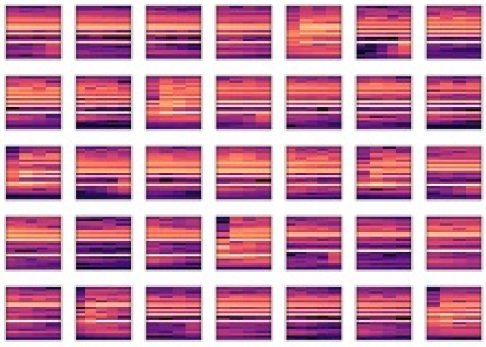

Figure 3: *Part of the Generated Thumbnails from the Example Track*

Unfortunately, we are currently stuck at feeding the packaged data for the training of the model. The Tensorflow program requires a very strict stucture of input data, therefore we need to calibrate the preprocessed data correctly and feed to the neural network to train it appropriately.

VI. Next Step

After training the model using the preprocessed data sets, we will evaluate the performance and debug any bias or variance problem of the model. We plan to follow the instructions mentioned in the course to improve the performance, which includes but not limited to methods like using sigmoid or tanh as the activation function, using different batch sizes or weight decay, using different music sample rates or image size of the thumbnail. Furthermore, currently our algorithm only works for monotonic music pieces. We will look into ways to generalize this model to recognize polyphonic music pieces after the current model is able to produce good results.

VII. Contribution

Chen Cen: Worked on model development with Tensorflow.
An Jiang: Programmed the preprocessing Python script (preprocessing.py), will participate in model debugging.
We both worked on the preliminary Music Information Retrival research and model decision process for the project.

References

[1] Convolutional neural networks.

[2] cuda-convnet.

[3] Ole Martin Bjorndalen. Mido - midi objects for python.

[4] Cen Chen and An Jiang. Cs229 project github page.

[5] Alex Krizhevsky. The cifar-10 dataset, 2009.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[7] Bernd Krueger. Classical piano midi page.

[8] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, Dan Ellis, Fabian-Robert Stoter, Douglas Repetto, Simon Waloschek, CJ Carr, Seth Kranzler, Keunwoo Choi, Petr Viktorin, Joao Felipe Santos, Adrian Holovaty, Waldir Pimenta, and Hojin Lee. librosa 0.5.0, February 2017.

[9] Eita Nakamura, Kazuyoshi Yoshii, and Shigeki Sagayama. Rhythm transcription of polyphonic piano music based on merged-output hmm for multiple voices. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(4):794–806, 2017.

[10] Graham E. Poliner. Classification-based music transcription, 2008.

[11] Daylin Troxel. *17th International Society for Music Information Retrieval Conference*.