CSCI 1430 Final Project Report: Optical Music Recognition

The Daily Midi: Sorin Cho, Sohum Gupta, Jusung Lee, Ivan Zhao
Brown University
8th May 2020

Abstract

For our final project, we wanted to implement a high-level method for Optical Music Recognition, in order to produce MIDI files from images of sheet music. We decided to use an approach that combined traditional computer vision algorithms with deep learning techniques in order to detect and classify features, and use them to construct a MIDI file. In the end, while our program was able to correctly locate features, our convolutional neural network did not accurately classify features, so we were only able to produce MIDI files that could play songs with fixed-length notes. However, we believe that, with a working CNN model, we would theoretically be able to detect and classify many musical features, and create a robust program for Optical Music Recognition.

1. Introduction

Over the past decade, many researchers have tried to solve the problem of Optical Music Recognition. Our project was to try to create a program that, given an image of sheet music, would produce a MIDI file that plays that music displayed in the image. This task has been difficult in the past due to the semantics and linguistics of sheet music, specifically revolving around features such as key signature, change in clefs, accidentals, and others. Thus, there is a lot of information that needs to be extracted from the sheet music to produce this audio file, and even the most basic form of this task needs at least note lengths and pitches. The approach we decided to take was to use traditional image recognition techniques to identify pitches of notes and to use deep learning techniques to identify all other symbols related to rhythm, key, etc, and then compile these described features and their locations in order to create a MIDI file. If this problem were to be solved, it could be used as an teaching aid in music education, or even be useful in digitizing pieces of old sheet music in order to create some sort of virtual archive.

2. Related Work

Before starting the project, we found lots of similar work online, and we pulled from multiple papers in order to decide the method we used for this project. One of the most helpful papers that we read through was Andy Zeng's paper on Optical Music Recognition. [3] We used the ideas of staff detection, Hough Circle Transform, and bounding boxes from Andy's paper, and implemented them in our model as well. Then, for our convolutional neural network model, we used the Deepscores Dataset, and pulled many techniques from a paper by L. Tuggener et al. in order to determine how to use the dataset. [2] Finally, we used the AlexNet architecture as a model for our CNN model. [1]

3. Method

We decided to use a combination of traditional and deep-learning computer vision techniques to extract and classify features. In terms of preprocessing, we used staff detection and staff removal to clean up the image. Then, we used the Hough Circle Transform to detect noteheads, and we used bounding boxes and a convolutional neural network to classify features. Finally, we used feature locations to match features to pitches, and created a MIDI file using the PrettyMidi library.

3.1. Staff Detection

Staff detection was a relatively simple process, mostly due to the fact that staffs in sheet music are relatively structured. The basic process for detecting staff lines is to sum up number of black pixels on each row of the image - then, for rows on the image where there are staff lines, this sum will be extremely high. Thus, in order to isolate these rows, we first threshold the sums to some parameter (represented as a fraction of the width of the image). This was in order to make sure that we only found the rows in which there were high intensity values. After this, we used non-maximal suppression from the Skimage library in order to find single row values corresponding to the center pixels of each staff line.

3.2. Staff Removal

The easiest way to approach this task was to removal all horizontal lines in the image. This was done by utilizing erosion and dilation from the OpenCV library. Erosion and dilation are similar to convolution. However, when the kernel is applied to a section of the image, only the lowest/highest pixel value is retained. As a result, utilizing kernels shaped like a vertical and horizontal line helps isolate vertical and horizontal features.

3.3. Hough Circle Transform

We used Hough Circle Transform to detect noteheads in both the image and in our prospective bounding box approach. Hough Circle functions by overlaying circles of a known radius around edges of an image. If enough circles overlap at a maximum point, there exists a circle centered at that point. We used staff distance to approximate the notehead height and used that as an input to an OpenCV Hough Circle function to locate all noteheads in a preprocessed image. We then outputted and later used this notehead positional information for pitch reference. We also planned to use Hough Circle to locate noteheads in our bounding boxes, as is explained in the next section.

3.4. Bounding Box

For the feature classification, we needed to extract features isolated in an image, say a whole note or dynamic symbol, to input into our CNN. We used OpenCV's contour function which draws curves joining points along a boundary of equal color or intensity. These drew around all relevant symbols in our preprocessesed image, and rectangular cutouts around said symbols formed the bounded features, ready for classification. After classification, we would perform HCT on these bounding boxes to find noteheads, again to locate pitches.

3.5. Convolutional Neural Net

We attempted at utilizing a CNN to classify musical features. We utilized various architectures due to difficulties that will be discussed in the Results section. Tweaks were made to the number and size of convolutional layers and dense layers. Whenever we had convolutional layers, we utilized max pooling. For the dense layers, we used ReLu activations and a dropout rate of 50%. The final denser layer had a softmax activation.

3.6. Pitch Conversion

After finding the locations for each of the notes in the image and the rows corresponding to staff lines, we had to match the notes to pitches and put them in order. The first thing we did was to match each pitch to a staff (i.e. a set of five staff lines). Then, after that, we calculated the number of steps above the bottom staff line for each note and its corresponding staff, where a step is calculated as half of the vertical distance between two staff lines. Then, once we knew this, we could match each note to its corresponding pitch, based off of the key signature, which we assumed to be C-Major, as we were unable to extract key signature information. Then, finally, we used the PrettyMidi library to construct the final MIDI file, adding notes staff by staff, in order of increasing x-values for the notes on each staff.

4. Results

Staff detection turned out to be very reliable and outputted accurate results for the staff distances.

This helped in staff removal and Hough Circle Transform. In regards to staff removal, this was also effective, however, since all horizontal lines were removed, open circle notes (such as half notes and whole notes) would have the horizontal edges of the circles removed as well. This was also the case for any horizontal features on symbols such as eighth notes, sharps, clefs, etc. As a result, we focused on the accuracy of quarter note detection since these features were not largely distorted by the staff removal.

For Hough Circle Transform, since a notehead's diameter is about the same length as the staff distance, we utilized staff distance to find noteheads. Through this method, we were able to match against almost all quarter notes. However, it would also detect other features such as time signature, key signature, clefs, song title, and song lyrics. This led us to remove these features before attempting to detect the notes.

For Bounding Boxes, this proved to be pretty successful. This is because given a page of notes, once we remove the staff on the sheet music, essentially everything on the page was a feature (as seen in the figure below).

After many attempts made with different variations in the architecture, parameters, and data, we were unable to successfully classify musical features using a neural network. We made tweaks to the learning rate, number of classes, size of the data set, number of epochs, and the size of batches. Regardless of the tweaks we made, we would achieve consistently higher testing accuracy and also reach very high training and testing accuracies (around 98%¡). However, when we tried the network on input not from the training or testing data, it's classification performance would be very poor. We believe the higher testing accuracy than training accuracy was a result of the simplicity of the class and the fact that many images of a single feature were close to identical, and thus the training data and testing data had little variance. We thought some potential issues were incorrect indexing of the classes, improper shuffling of data, and overfitting the network to the data set. However, we verified that these were not the issues, and we also attempted testing with a much simplified network training on fewer epochs. We also attempted adding noise to the data and adjusting our

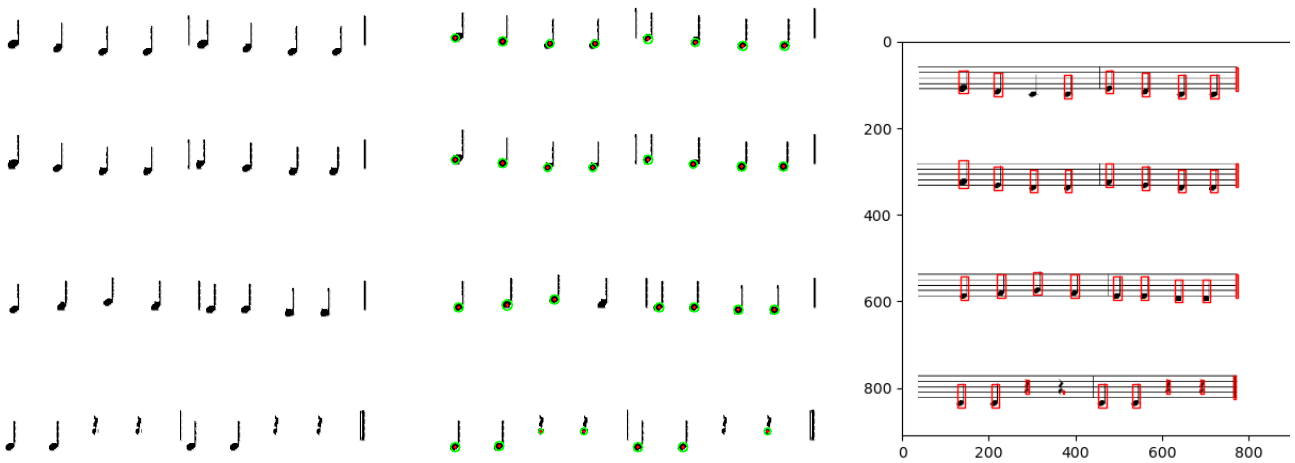

Figure 1. Processing and Transforms *Left:* Staff-removed image. *Center:* Hough Circle Transform on processed image. *Right:* Contoured features bounded.

input images to resemble the training/testing data as close as possible (through scaling and proper centering of features). These attempts did not result in any better outputs from the network. We also checked that the network at least classified the training data correctly (which is verified by the results given by Tensorflow, but just in case), and the network did classify the features correctly. Eventually, we could not identify the cause of this weird behavior, and as a result, stuck to the traditional techniques preceding this section.

With all other parts working other than the CNN, we were still able to produce reasonably good MIDI files. However, beyond fixed length quarter notes, we could not venture beyond to other musical features due to the lack of a working CNN. Although the MIDI file generation worked well, it was very dependent on the reliability of the Hough Circle Transform. Variations and incorrect circle detections would result in wrong pitches in the music. Regardless of this dependency, we were able to produce mostly correct music from an image of simple sheet music.

4.1. Discussion

The methods in themselves is pretty similar to what current research in OMR is currently doing. In order to get the data, the sheet music itself must be decently preprocessed to get all the relevant information. The part of our method that has the most potential is the CNN model, which should theoretically be able to output correctly classified features. Also, the CNN model is the most generalizable part of my model, since it should theoretically work on any type of sheet music, provided that the model has been trained on a large enough dataset. The rest of our method, while proving fairly successful on the images we tested, might not necessarily extend well to handwritten sheet music. Thus, we decided to stick to these traditional computer vision methods, and narrow the scope of our project to non-handwritten sheet music.

However, we still had lots of issues with our method. One of the issues we had in preprocessing was tweaking parameters to remove all horizontal lines. We found that different images often responded better to different parameters, and we had to find parameters that would work fairly well for all images we tested. Also, we found that it was hard to identify certain features, both using Hough Circle Transform and bounding boxes, especially those features which were distorted after staff removal, such as half notes. On top of this, we found that both the Hough Circle Transform and bounding box algorithms often removed features that we didn't want to remove, and detected features that we didn't want to detect. We also had issues with our CNN which we were unable to resolve, which we have discussed before. Overall, if we were able to work out the kinks with our CNN model and were able to optimize the Hough Circle Transform, we believe that our method would be a fairly robust model for Optical Music Recognition.

5. Conclusion

In this project, we implemented a start to finish program that takes in an an image of sheet music and outputs the corresponding MIDI file that plays the song. This was done through various preprocessing techniques, deep learning models, and then outputting the MIDI file. This project matters because digitizing the way people use sheet music is super helpful. For example, if musicians want to convert sheet music and use that on their computer, they could use this software. Additionally, if we want to digitize older music, this would be one potential way of doing so. In going forward, the biggest thing would be to us an end-to-end neural network that does all of the classifcation, detection, and

corresponding output without any of the other algorithms needed for feature detection. In addition, the other way that this project could be improved upon going forward would be to increase the size of the dataset (we used only a subset of the DeepScores dataset to improve classification accuracy) and to use all the possible classes. Thus, it would be able to recreate the entire sheet music.

References

[1] G. H. A. Krizhevsky, I. Sutskever. Imagenet classification with deep convolutional neural networks. page 9, 2012. 1

[2] J. S. M. P. T. S. L. Tuggener, I. Elezi. Deepscores – a dataset for segmentation, detection and classification of tiny objects. pages 1–10, 2017. 1

[3] A. Zeng. Optical music recognition. pages 1–5, 2014. 1

Appendix

Team contributions

Sorin Cho I worked on staff removal and the CNN. For staff removal, I looked up image processing techniques, and primary utilized the OpenCV library. For the CNN, I worked mainly on getting the model to run. This included small tweaks the input and a lot of tweaks to hyperparameters and architecture of the CNN. In addition to this, I also helped with training the CNN on GCP and helped with testing the saved model on raw image data.

Sohum Gupta I worked on two parts of the project, namely staff detection and pitch conversion. For staff detection, I wrote functions that were able to detect rows in the image corresponding to staff lines and find the average distance between two notes on a staff line. In terms of pitch conversion, I wrote functions that matched feature locations to notes and pitches based on their distance from the detected staff lines, and also constructed the final MIDI file. Apart from this, I wrote various image visualization functions, and helped out with training the CNN model.

Jusung Lee I worked mainly on the Hough Circle Transform, narrowing down the parameters and image preprocessing that allowed for a functional circle detection and writing functions for applications to post-classification bounding boxes. I also helped with contouring images features and converting that to bounded boxes for CNN input.

Ivan Zhao I worked on the preprocessing and CNN part of the project, specifically doing some work around staff removal, bounding box detection and merging (and tuning those paramaters). For the CNN, I built out most of the functionality for architecture, preprocessing the dataset, and also figuring out the dataset. I also worked on hooking up all the pieces together at the end in our main file.