Music Notes Detection and Classification

Princeton Ferro pcf252@nyu.edu
Graduate School of Arts and Science
New York University
New York, NY 10003, USA

Bohan Zhang bz771@nyu.edu
Graduate School of Arts and Science
New York University
New York, NY 10003, USA

Yuteng Zhang yz7436@nyu.edu
Graduate School of Arts and Science
New York University
New York, NY 10003, USA

Editor: Princeton Ferro, Bohan Zhang and Yuteng Zhang

Abstract

Our project is sheet music recognition. We approached this in three steps: First, we trained a classifier on a data set of note head shapes. Then, we trained an object detection model on a data set of sheet music, labeling detected notes with their position on the staff line. Finally, we apply the note head classifier on the object bounding boxes to determine the note type. Our resulting pipeline takes an input image of sheet music and determines for each note its pitch and duration.

1. Introduction

Recognizing music requires determining the pitch and duration of each note on a printed score. Traditional methods have needed a preprocessing step where the staff lines are removed in order to ease detection of note head shapes. More recent methods use deep learning to overcome this obstacle.

This project contains two parts: a classifier for note heads and an object detector for notes on a staff. We use a CNN since it's one of the best models for image classification. The second part is object detection, which presented a number of challenges: The first challenge we faced was to find a proper dataset to use. There are many data sets for music recognition out there but few that come with bounding boxes for notes. At first we tried a dataset we found called DeepScore V2. However the result was not satisfying since it includes not only notes, but also many other

Figure 1: *Whole note*

Figure 2: *Eighth note*

musical elements that we don't want for this project. The large number of objects per image made training prohibitively slow. So we decided to generate our own dataset. First we tried writing a renderer in Python that could directly write shapes and lines to a pixel array. Because we controlled exactly how notes were rendered, we were able to generate pixel-perfect bounding boxes for each sample. However, the renders were of low quality and it would take too much effort to improve them to match professional music engraver software.

Later we found a better approach: using the software LilyPond, we were able to generate high-quality randomized sheet music, and we used OpenCV to generate the bounding boxes for the notes in each image.

2. Method

2.1 Note Head Classification

2.1.1 Dataset

We use a hand-crafted dataset of cropped notes on a staff line. The croppings are not usually centered and are of various sizes. The categories are: { `whole, half, quarter, eight, sixteenth` }. Figure 1 and Figure 2 are examples of the note images. Each image is resized to a canonical 64×64 pixels, first by padding the smaller dimension with white and then scaling the image.

2.1.2 Approach

We use a CNN with 18 layers consisting of Conv2d, MaxPool2d, Dropout2d, BatchNorm2d, and Linear layers. The details of the structure are shown in Figure 3. We then trained the network for 20 epochs. Our data set has 259 images.

2.2 Object Detection

2.2.1 Dataset

We used a music engraving app called LilyPond to generate sheet music. We randomly generate images of notes with various durations and pitches. To make staff line position detection easier, we limit the data to notes in the range C4 to G5 (inclusive), corresponding to staff line positions 0 - 11. There is a background class containing

Layer(type)	Output Shape	Param #
Conv2d-1	[-1, 64, 62, 62]	1792
MaxPool2d-2	[-1, 64, 31, 31]	0
Conv2d-3	[-1, 128, 29, 29]	73856
Dropout2d-4	[-1, 128, 29, 29]	0
MaxPool2d-5	[-1, 128, 14, 14]	0
Conv2d-6	[-1, 128, 13, 13]	65664
Conv2d-7	[-1, 256, 11, 11]	295168
Conv2d-8	[-1, 256, 9, 9]	590080
Conv2d-9	[-1, 256, 7, 7]	590080
Conv2d-10	[-1, 256, 6, 6]	262400
Conv2d-11	[-1, 256, 5, 5]	262400
Conv2d-12	[-1, 256, 4, 4]	262400
Dropout2d-13	[-1, 256, 4, 4]	0
MaxPool2d-14	[-1, 256, 2, 2]	0
BatchNorm2d-15	[-1, 256, 2, 2]	512
Linear-16	[-1, 256]	262400
Linear-17	[-1, 128]	32896
Linear-18	[-1, 6]	774

Figure 3: *CNN layers*

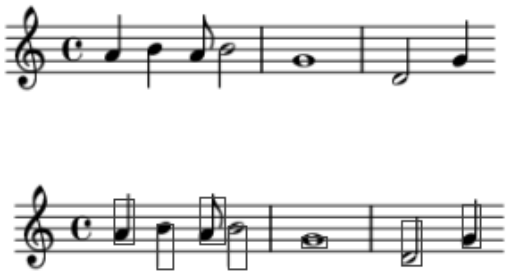

Figure 4: *Application of Openmsbb on Lilypond Image*

no notes. Thus, our object detector has 13 possible labels to assign to each bounding box. Our data set has 74 images.

2.2.2 Approach

After generating the dataset, the first thing we need to do is to find the bounding box for each of the notes in order to train the model. For finding accurate bounding boxes, we found a open source script on Github called openmsbb, it loads the image and returns the coordinates of bounding boxes for what it detects. We modified the code in order to handle some of the cases that it missed. Figure 4 showed the box that it generates for the above sample image. We saved the coordinates and the note information into a CSV file and put it along with the sheet music images.
For the training model, we used Pytorch's built-in Faster R-CNN framework with MobileNet V2 as the backbone. Faster R-CNN is a model that predicts both bounding boxes and class scores for potential objects in the image. We trained with 0.01 learning rate for 40 epochs and then lower the learning rate to 0.002 to train for 20 more epochs. The final loss was 0.03.

2.3 Wrap Things Together

Now we have finished building the classification model and object detection model. We can use the CNN model to classify the detected notes from the Object Detection model. The CNN model can now provide predictions over the cropped notes that were detected from the Object Detection model. For each note, our model can predict its location in the image, its staff line position, and note head, which corresponds to the note's pitch and duration.

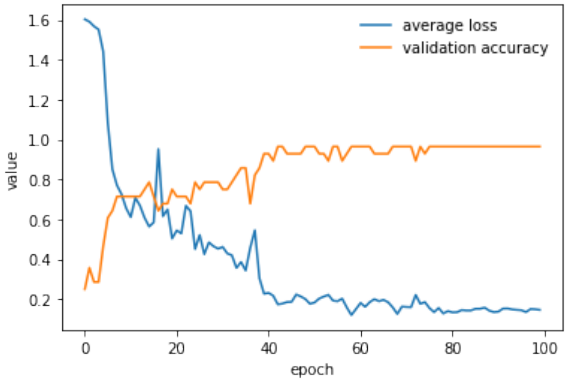

Figure 5: *CNN Loss and Accuracy per Epoch*

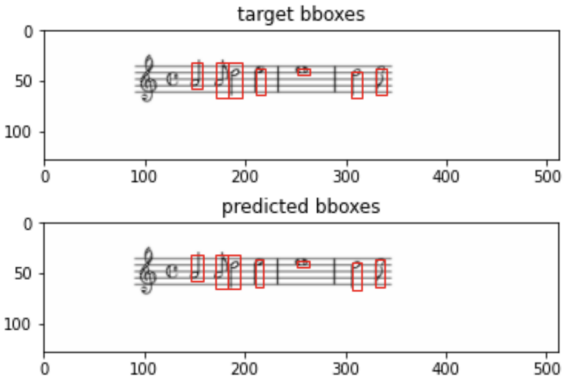

Figure 6: *Object Detection Model Predictions*

	IOU	Area	maxDets	Value
Average Precision	0.50:0.95	all	100	0.604
Average Precision	0.5	all	100	0.763
Average Precision	0.75	all	100	0.754
Average Precision	0.50:0.95	small	100	-1
Average Precision	0.50:0.95	medium	100	-1
Average Precision	0.50:0.95	large	100	0.656
Average Recall	0.50:0.95	all	1	0.529
Average Recall	0.50:0.95	all	10	0.655
Average Recall	0.50:0.95	all	100	0.655
Average Recall	0.50:0.95	small	100	-1
Average Recall	0.50:0.95	medium	100	-1
Average Recall	0.50:0.95	large	100	0.655

Figure 7: *Object Detection Model Average Precision and Recall*

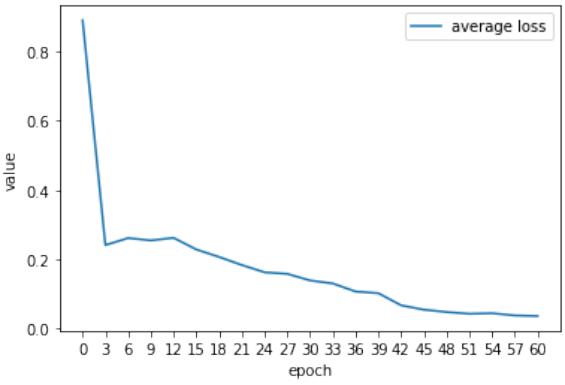

Figure 8: *Object Detection Model Avg. Loss*

3. Experiments

3.1 Performance of Classification Model

Figure 5 represents the Average loss and Validation accuracy per epoch of the note head classifier. The note head classifier achieves good performance on the testing data, converging to 100% after 40 epochs.

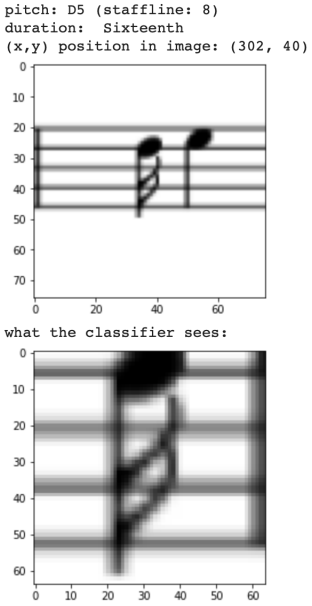

Figure 9: *Good Prediction*

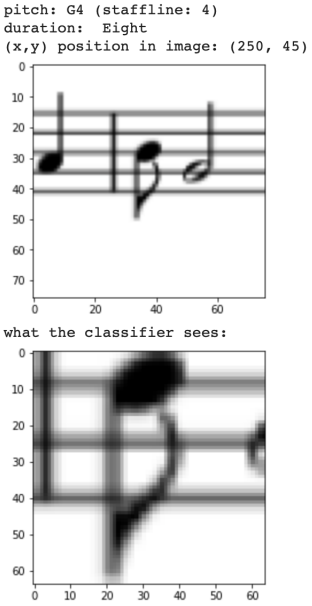

Figure 10: *Wrong Pitch*

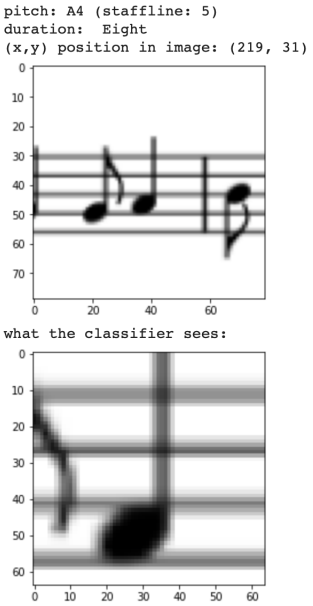

Figure 11: *Wrong Duration*

3.2 Performance of Object Detection Model

We applied the Object Detection model on the dataset generated from Lilypond and Openmsbb. Now the model can detect notes and predict the positions of each bounding boxes. Since there are many targets for each music sheet image, it is a little difficult for the Faster-RCNN model to train. So we trained the model with 80 epochs for better performance. Figure 7 shows the precision and recall with different IOU and area through the training process. The precision and recall values showed in the table are the area under the precision-recall curve. As we train more, the model has less improvements to reduce the average loss. So we stopped when the average loss was around 0.03. The average losses of 80 epochs are show in Figure 8.
An example of the Object Detection Model prediction is shown in Figure 6. The 'target bboxes' represent the ground truth from the dataset, and the 'predicted bboxes' shows the result that our model provides.

3.3 Final Model Performance

Initial performance of the model required a lower score threshold of 0.5 to detect most notes. Retraining with a higher learning rate for 80 more epochs improved this threshold to 0.7. mAP@IoU0.5:0.95 was 0.122 versus 0.604 after retraining.

During evaluation, we crop the detected notes by resizing the predicted bounding boxes and sending the image tensor into the note head classifier. We extend the predicted box along its smallest axis to get a square box before resizing. The problem

of this resizing technique is that it might include more details of the image than the original note, particularly if notes are spaced closely together. It may crop a portion of other notes, and some irrelevant details on the staff line to trick the classifier. Figure 11 is an example of this. We countered this issue by using noisy and partial crops in our training data for the note head classifier. So far, our music note recognizer can classify most of the note's duration correctly. But we found it is difficult to get the actual pitch. As we can see from Figure 9, sometimes the model can provide perfect prediction, while sometimes it gets the wrong pitch(see Figure 10).

4. Conclusion

Our project combines the classification network and object detection network, takes a sheet music with different notes, and detects the notes. For each detected notes, our model can classify what kind of note it is and what pitch it has. The result is relatively accurate and it could detect and classify most of the notes on the music sheet. However, the most important problem of our recognizer is that the pitch prediction is not accurate enough. We think this is because we use the Object Detection model to make prediction on the pitch. Our bounding box only select the body of the note, which does not tell the relationship between the note and the the staff line, so the model cannot learn too much about the position on the staff line to tell what pitch it is, even if we have the information of the pitch of the note in the dataset. Possible ways to make improvement could be: (1) Let the classifier classify the pitch of a note by both specify the pitch in the classification dataset and cropping the note with the entire staff line so that the classifier can know the relative position of the note on the staff line. (2) Use a separate linear classifier on the positions of the bounding boxes.

There are other things that we could do to improve our project. We could increase the size of the dataset for both models. For example, the dataset is still not tricky enough to let the CNN model tell some differences if the cropped image from the detector includes more irrelevant elements that can make distractions.

We could also improve our OpenCV code to make the ground truth bounding boxes more accurate. Furthermore, we have not investigated how to recognize stacks of notes in a chord.

5. GitHub Repository link

`https://github.com/cloudzhangyuteng/music-notes-detection-and-classification`

References

[1] *LilyPond*. https://lilypond.org

[2] *openmsbb*. https://github.com/Cjdcoy/openmsbb

[3] *Torchvision Object Detection Finetuning Tutorial Available electronically at.* https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

[4] *Staff line removal using line adjacency graph and staff line skeleton for camera-based printed music scores.*
2014 22nd International Conference on Pattern Recognition. https://projet.liris.cnrs.fr/imagine/pub/proceedings/ICPR-2014/data/5209c787.pdf