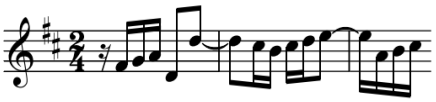
Current State of the Project:

Classification model:

Previously, we wanted to measure the coordinates of every single note and staff-line on the image and find the "bounding boxes" of each note detected and then use these coordinates to depict in what order the notes should be played. We found the idea above was complicated and unnecessary after reading into the research paper "End-to-End Neural Optical Music Recognition of Monophonic Scores."

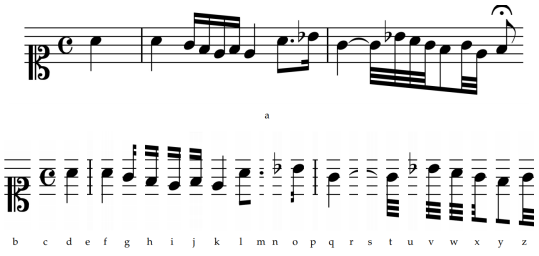
By utilizing a Convolutional Recurrent Neural Network (CRNN) and a Connectionist Temporal Classification (CTC) loss function, this model addresses the score as a single unit rather than as a sequence of isolated elements that have to be classified separately. As shown below, the output of the classification model is done using an "image-to-text" framework. This is highly advantageous when we need to translate all the notes into a media file to be played-back.



```
clef-G2, keySignature-DM, timeSignature-2/4, rest-sixteenth, note-F#4_sixteenth,
note-G4_sixteenth, note-A4_sixteenth, note-D4_eighth, note-D5_eighth, tie, barline,
note-D5_eighth, note-C#5_sixteenth, note-B4_sixteenth, note-C#5_sixteenth,
note-D5_sixteenth, note-E5_eighth, tie, barline, note-E5_sixteenth,
note-A4_sixteenth, note-B4_sixteenth, note-C#5_sixteenth
```

Dataset/Training & Mobile App :

For training, we are using the PrIMUS dataset for our classification model. Our "MusicClassificationObject" class in the model file translates all of the dataset's images into tensors and converts its musical notated text into numbers to be used as labels. We are also utilizing a CRNN model provided by a github source that is compatible with pytorch. To modify the dataset, we are currently working on separating each image frame-by-frame for greater accuracy of the model. This helps the CNN network process the input image and extract its adequate features. Currently, our Android app has the functionality to take photos, which will later be used to scan a musical score and play it back.



What tasks were done:

- **John**: Developed the classification model that contains the dataset and labels of each note. Built along with Houlin in coordinating the dataset/classification tasks.
- **Jefferson**: Learned about basic Android development, created and configured the skeleton code for the application, and got it to work correctly with github's workflow, assisted Ying Qi with the camera functionality.
- **Ying Qi**: Learned about basic Android development, built along Jefferson on the skeleton model created by Jefferson. Got the camera function of the app correct with Jefferson's help.
- **Houlin He**: Split portions of music score into sequence of music notes.

Proposal Changes

We have completely reworked the milestones since the original ones were unproductive. Firstly, the distribution of tasks between members was ineffective. The early stages of the ML task do not require all four people to work on the

dataset/classification model. Instead, we revamped it so that half of the members work on the ML aspect and the other half work on the front-end/back-end of the interface simultaneously. Secondly, the previous milestones revolved around our initial understanding of the classification model (described above in 'Current State of the Project'). This was also unnecessary given our new classification model, so we shifted our milestones to focus on the model's accuracy and intake of dataset variety.

Updated Milestone 1:

- We have implemented some basic but important functionality for the android application as described above
- Prepare all the datasets correctly for the model. This includes splitting the musical score as shown above and developing a Music Score Classifier model.

Updated Milestone 2:

- Configure the CRNN and train it with the dataset. Print the notes on the staff-line with their respective durations and in sequential order with decent accuracy.
- Learn and create a viable Django server in which the ML component can be hosted on
- Maybe add an option on the application to select for existing photos of musical scores in the device in addition to the already implemented camera function (since milestone 2 is very short, this can be moved to milestone 3 if there is not enough time)

Updated Milestone 3:

- Establish a working connection between the server and the application; send the classification output (the stream of text) back to the server.
- Host the machine learning model onto the server
- Mix the current dataset with new datasets that contain chords and multiple staff-lines (A piano score consisting of two staff-lines) and train our model on this.

Updated Milestone 4:

- More polished UI/UX for the application
- Translate the classification output into a MIDI file, so it can correctly play-back all the notes correctly. Ensure to send this MIDI file back to the server.
- Overall testing of the final project and finishing any other unfinished tasks that we were not able to complete in other milestones

Current Challenges

Classification model:

- The splitting of notes is still inaccurate on some music scores (basically some of those have line 6 being drawn across the notes)
- The input of the classification model only takes a single staff line and the chords are not considered in the dataset either. For future milestones, we will create additional datasets with these features.

Django Server:

- We do not yet have a good grasp about the difficulty of this component since none of the group members have experience with server-side work, so the milestones set for this remain uncertain.
- We also need to find a place to host the server too.

Android App and User Interface:

- The main challenge is to find a way to connect the local data to the server as well as pulling data from the server. This part is likely to be done after the server has been set up.