Pitch Recognition and Note Detection from Sheet Music

Janger Wangers: Jeffrey Liang (Jliang73), Austin Liu (Afliu), Seyoung Jang (sjang31), Eric Wang (ewang182)
Brown University

Abstract

Our project detects notes and identifies each note's pitch from sheet music using image segmentation and Convolutional Neural Networks. Our pipeline takes a PDF of a piano music sheet, extracts individual notes using Computer Vision operations, calculates their pitch by comparing their y-axis position with the staff lines, and predicts the note type using our pre-trained CNN.

1. Introduction

All our group members share a common passion for music, ranging from guitar and piano to drums and music composition. As beginners, we all experienced the challenges of reading sheet music and translating it into sound. To make music more accessible, especially for novice musicians, we were inspired to build a deep learning model that converts PDF sheet music into audio, allowing users to hear how a musical piece is supposed to sound. Our ultimate goal is to enable the automatic conversion of visual sheet music into audio playback.

When looking into existing literature, we noticed a lot of people experiencing difficulties in accessing existing Optical Music Recognition (OMR) tools. A majority of the OMR systems are either proprietary or not publicly available, limiting accessibility for researchers and musicians. In addition, they often fail in complex or noisy sheet music. In response, we decided to build our own modular pipeline that focuses specifically on reliable pitch identification and note type classification from piano scores.

The core problem we are addressing is a combination of supervised learning, image segmentation, and CNN-based detection to extract relevant features from sheet music. Our approach integrates both computer vision and neural networks to convert visual music notation into structured digital data.

2. Related Work

Oemer is an open-source end-to-end OMR system that served as a core reference for our project [4]. It demonstrated a full pipeline for converting scanned sheet music into symbolic representations using image processing and machine learning techniques. We adopted and simplified Oemer's staff removal method and CNN architecture to build a lightweight system tailored specifically for classifying common note types and estimating note pitches. Oemer's structure helped us better understand how OMR processing is done and gave us a guideline on how to pre-process our data.

3. Data

From DoReMi [3] we used their scanned PDFs of songs and their corresponding OMR (Optical Music Recognition) XML files.

The dataset contains 5,218 images representing 44 songs, where each song is divided into multiple pages. Each page has an associated OMR XML file. Preprocessing was necessary to clean the labels, filter irrelevant classes, and adjust the bounding boxes to match cropped images.

4. Methodology

Our pipeline extracts individual notes from scanned piano sheet music and determines their pitch by analyzing the vertical positioning of each notehead relative to the staff lines. In parallel, the system classifies the note type as a quarter, half, or whole note using a CNN. Given the timeline and scope of the project, we chose to focus on these three note types, as they are the most commonly encountered in beginner-level piano sheet music.

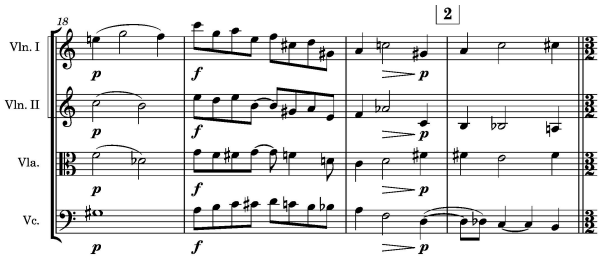

Figure 1. Original Image

4.1. Staff line Extraction

Using binary inversion and horizontal projection (summing across each row), we highlight staff lines as peaks and

group every five lines into musical staves. We extract the upper bound, lower bound, and unit distance between each staff line.

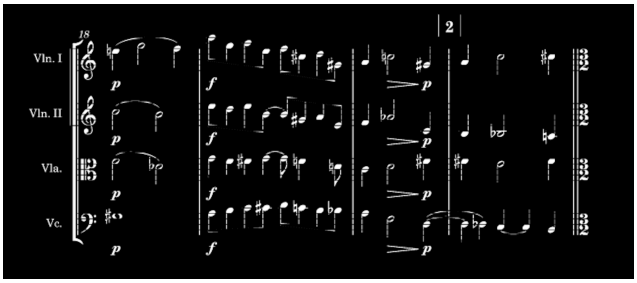

Figure 2. Staff lines extracted

4.2. Morphological Operation

We applied morphological dilation and erosion operations to prepare the images for notehead extraction. Dilation expands the foreground regions by replacing each pixel with the maximum value covered by the kernel. Erosion shrinks the foreground regions by replacing each pixel with the minimum value covered by the kernel.

We used these two operations hand in hand. Closing is the process of dilation, then erosion, filling gaps, and smoothing our noteheads. Opening is the process of erosion and then dilation, removing small specks and thin artifacts such as stems or slurs.

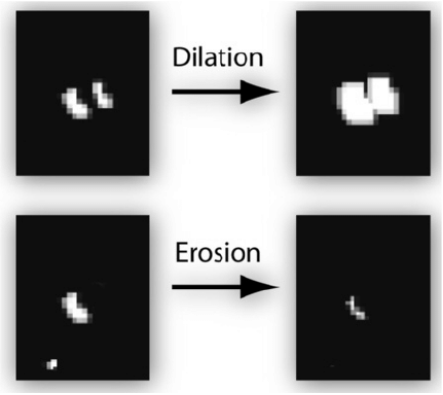

Figure 3. Dilation and erosion used for the closing and opening morphological operation [1]

Here are the results of the morphological operations on our test image. The first closing fills in the gaps within notes created by the staff line removal. The opening operation removes large artifacts, dynamic markings, and barlines while preserving the notehead structure. The second closing uses a tall kernel to merge adjacent vertical fragments (i.e., chunks of the time signature), making the removal process easier.

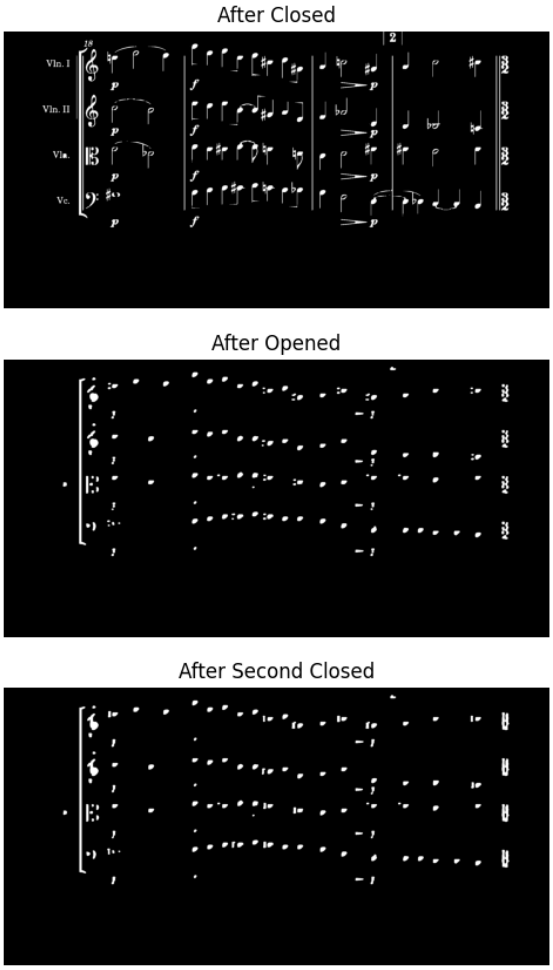

Figure 4. After first closing operation

4.3. Watershed Algorithm

Despite morphological cleaning, closely spaced noteheads often appeared as merged blobs. To address this, we used the watershed algorithm [2]. Figure 5 displays a sample run of the watershed algorithm where we separate three notehead clusters to draw a boundary between one and the other two noteheads.

We first generated a heatmap to visualize pixel distances from object boundaries. To isolate distinct notehead regions, we removed foreground pixels located more than 40% away from the edge. The center of each blob is labeled as a *peak*, representing a local maximum in the distance map. The watershed algorithm then simulates water flooding outward from each peak until flows from different peaks meet, forming boundaries. These intersections appear as dark blue outlines in the Watershed Markers diagram. The final segmentation is shown in Figure 5, correctly identifying a single notehead on the left.

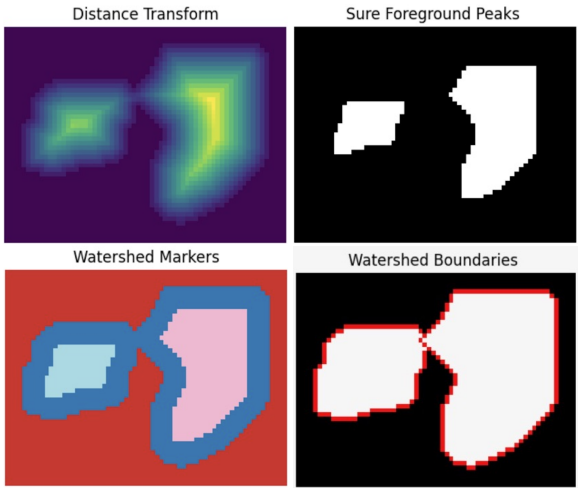

Figure 5. Process of applying the watershed algorithm to split a cluster of three note heads (distance transform → sure foreground peaks → watershed markers → watershed boundaries)

4.4. CNN Classification

We originally wanted to do a multi-class classification CNN that would recognize notes as quarter, half, and whole. However, we eventually found it easier to split it up into 2 binary classification CNNs, one for differentiating between hollow and non-hollow notes and one for differentiating between whole vs half notes.

4.4.1 Hollow vs Non-Hollow Note Classification

For this task, we designed a simple three-layer CNN to capture the distinguishing features of filled versus unfilled noteheads. Since it is a binary classification problem, we employed binary cross-entropy as our loss function and Adam as the optimizer. We used an 80/20 split for training and validation, and found that training for 25 epochs was the most optimal number.

4.4.2 Whole vs Half Note Classification

Classifying whole and half notes was perhaps the most challenging part of our project since there are only minor differences between whole and half notes. Upon observation, we noticed that the key feature our CNN model should focus on is whole notes having a straight oval hollow compared to half notes that are slightly tilted at an angle.

With a basic three-layer CNN structure, the model was over-regularizing and was struggling to learn the subtle difference between whole and half notes. To give the network more freedom to learn, we slightly increased the dropout rates in every block. Higher dropout rates prevented the network from relying too heavily on a singular feature and learning a specific combination of filters. As a result, by reducing the model's sensitivity to noise in the training set, the model was forced to be more robust.

After training the model over multiple epochs, we observed that the loss converges to 0.0 from around 20-25 epochs as seen in Figure 6. As a result, our model is trained on 25 epochs to prevent overfitting. In addition, we implemented early stop regularization to stop the training process once the model's performance on the validation data starts to degrade.

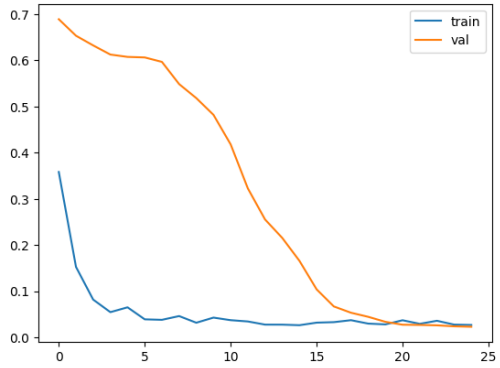

Figure 6. Loss function for whole and half note classification CNN across epochs

Furthermore, we added a L_2 weight to penalize large weights and smooth the decision boundary. By constraining particular connections from growing too big, we prevent overfitting and enhance the model's ability to generalize to unseen data.

The classification result of our CNN model is displayed in Figure 7.

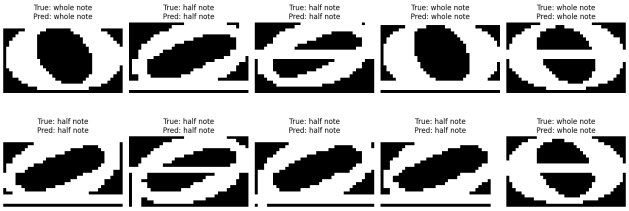

Figure 7. Classification output results from our whole vs. half note CNN model

4.5. Pitch Estimation Algorithm

To determine the musical pitch of each notehead, we compared each notehead's y-axis position to the staff line. Using standard musical notation rules (e.g., notes sit either on lines or spaces), the vertical distance was mapped to a specific pitch. For simplicity, we assumed all staff lines are in treble clef since clef detection is not within the scope of our project.

This was straightforward to implement because our preprocessing already produced the staff lines metadata (y_min, y_max, and line spacing height).

5. Results

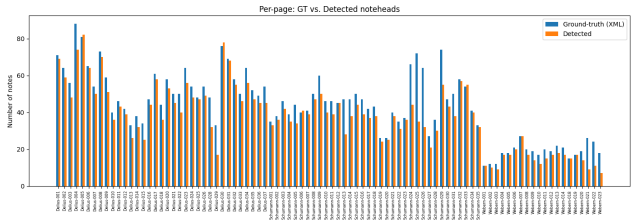

Figure 8. Detection accuracy per image. Number of identified noteheads divided by the total number of noteheads.

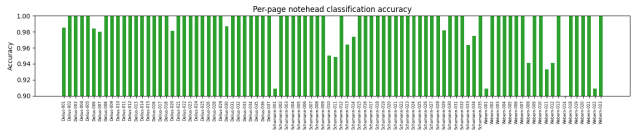

Figure 9. Accuracy of CNN hollow vs non-hollow notehead classification

5.1. Notehead Classification

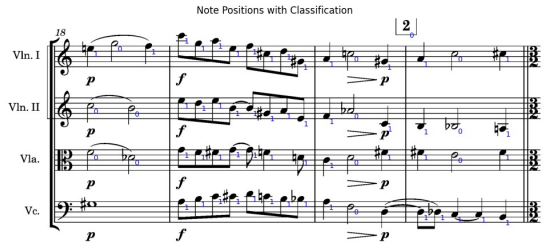

Figure 10. Notehead classification results labeled on the original sheet music

5.2. Pitch Identification

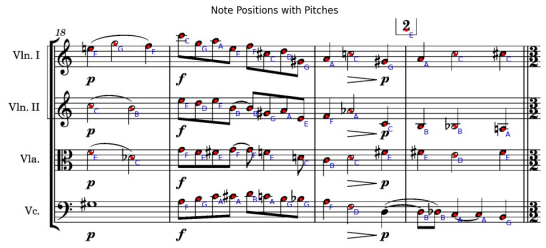

Figure 11. Pitch idenfitication results labeled on the original sheet music

6. Challenges

Removing staff lines required extensive experimentation with morphological operations. At the same time, aggressive filtering risked removing parts of noteheads that overlapped with staff lines. We had to carefully balance kernel size and iteration count to eliminate lines while retaining musical symbols.

It was challenging to cleanly separate whole notes from surrounding noise/blobs. Our approach involved reconnecting broken outlines of hollow noteheads to make them more detectable. However, the same process sometimes unintentionally merged nearby notes or distorted the shape of filled notes. Finding the right balance between restoring hollow noteheads and avoiding unintended merging took significant trial and error.

Last but not least, distinguishing whole vs. half notes was especially challenging, as the only differences are subtle variations in size and ellipse orientation—features that are hard for a CNN to reliably learn and something we are still working on.

7. Reflection

We learned that the model's performance is majorly dependent on the quality of the input data. We had to build our own pipeline from scratch – removing staff lines, extracting notehead regions, and tuning morphological operations. A major challenge was balancing removal of unwanted noise without erasing noteheads. This involved carefully designing dilation, erosion, and closing steps, followed by rigorous contour and geometric filtering to retain only valid black, half, and whole notes.

Adding on, our initial goal was to train a multi-class classifier to distinguish black, half, and whole notes. But the shape similarities between hollow note types made it difficult to separate half and whole notes. As a solution, we switched to a two-stage binary classification pipeline: first detecting whether a note was filled (quarter/eighth/sixteenth) or hollow (half/whole), then distinguishing between half and whole notes using separate geometric heuristics and classifiers. This decomposition significantly improved our accuracy for detecting if its hollow or filled and we are still working on whole note classification.

References

[1] Victor Mateevitsi et al. Dilation and erosion: A series of dilation operations cause the shadow to merge with the insect's body. https://www.researchgate.net/figure/Dilation-and-erosion-A-series-of-dilation-operations-fig2_274454691, 2015. Accessed: 2025-04-30. 2

[2] OpenCV Team. Image segmentation with watershed algorithm. https://docs.opencv.org/4.x/d3/db4/

tutorial_py_watershed.html. Accessed: 2025-04-30. 2

[3] Steinberg Media Technologies and Queen Mary University of London (Elona Shatri and George Fazekas). Doremi. https://github.com/steinbergmedia/DoReMi, 2025. Accessed: 2025-04-30. 1

[4] Yoyo, Christian Liebhardt, and Sayooj Samuel. Breezewhite/oemer: End-to-end optical music recognition (omr) system, 2023. 1