Optical Music Recognition Literature Review - CSES ORCA

Michelle Dong, Luke Wang

1 Introduction

Optical Music Recognition (OMR) is the task of automatically converting scanned images of printed or handwritten music scores into machine-readable notation, usually in the form of MusicXML files. Although OMR has been studied for decades due to the increasing demand for sheet music digitization, recent models still struggle to produce accurate interpretations of scanned/written music scores. These issues stem from the complex, high visual density, nature of music scores, and its variability of music notation, including overlapping symbols and notes. In this paper, we review traditional OMR pipelines, examine the integration of deep-learning to the field, and discuss new methodologies based on the limitations of past models, as well as looking at a case study on a widely used commercial OMR application.

2 Traditional OMR pipeline

The traditional OMR pipeline as illustrated by Rebelo et al [2] consists of four main stages: image preprocessing, music symbol recognition, musical information reconstruction, and the construction of a musical notation model [1], [2]. In the preprocessing stage, the input image is enhanced using filters and adjusted for optimal brightness and contrast, before being deskewed, deblurred, denoised, and analyzed to estimate staff line positions and spacing [1]. Next, the extraction of musical symbols like staves, noteheads, stems, accidentals and rests is performed before classifying them using methods ranging from k-nearest neighbors (kNNs) and support vector machines (SVM) to neural networks [1]. In the musical information reconstruction stage, noteheads are combined with beams, assigned duration and pitch, and linked to accidentals, basically combining previously extracted and classified features to prepare for the next step. Finally, all the reconstructed information is translated into a usable format such as MIDI or MusicXML files.

Despite the traditional pipeline being theoretically robust due to its modularity, it is still prone to early-stage errors, such as incorrect time-signature readings, which can propagate downward and affect the rest of the interpretation. Additionally, its overreliance on manual tuning makes it more susceptible to errors when faced with variation in handwritten music scores. These issues have led to a greater push towards deep learning approaches.

3 Deep Learning and Transformer Integration

The growing popularity of deep-learning, and transformer architectures in particular, has led to its adoption in OMR. This typically comes in the form of using an end-to-end image-to-sequence model with a visual encoder (often a CNN or vision transformer) to extract features from the score image [3], and a transformer decoder that autoregressively generates a sequence of musical tokens, typically in symbolic notation format like those used in MusicXML files. Many recent

models cite the Sheet Music Transformer (SMT) [4] as the baseline to their usage of transformer decoders, since it was one of the first to handle full-page, polyphonic music scores.

Because the deep learning pipeline does not rely on modularity like traditional approaches do, identifying and debugging errors within the model is a lot more difficult. Additionally, this approach does not allow the model to know or understand music theory unless it is enforced in postprocessing. Furthermore, benchmarks and metrics are still somewhat messy due to the recency of this pipeline [5].

4 Examination of Existing OMR Implementations: HOMR and LEGATO

Examining the architectural differences between existing state-of-the-art implementations provides valuable insights for design decisions for ORCA's implementation of OMR. LEGATO (Large-Scale End-to-End Generalizable Approach to Typeset OMR) and HOMR (Homer's Optical Music Recognition) represent two fundamentally different approaches to optical music recognition, each with distinct architectural philosophies and processing workflows that may offer complementary strengths for ORCA's implementation strategy.

LEGATO [3] adopts an end-to-end multimodal transformer architecture built upon the Llama-3.2-11B-Vision model, treating OMR as a direct image-to-sequence translation task that outputs ABC notation. The LEGATO system processes full-page and multi-page music score images through a pipeline. First the score images are divided into overlapping segments with aspect ratios of 1:4. Then those images are divided into four patches, which are encoded by the 836M-parameter pretrained vision encoder of the Llama 3.2-11B-Vision model into latent embeddings, which then serve as cross-attention keys and values in the 101M-parameter transformer decoder which autoregressively generates ABC tokens.

In contrast, HOMR [7] follows a more traditional approach using multiple stages of processing, including image pre-procressing, segmentation using U-Net, symbol detection to create bounding ellipses and boxes around note heads and staff fragments, anchor-based staff detection using clef and bar lines as reference points, vision transformer-based rhythm parsing, and generating the MusicXML file. The following is a more detailed summary of HOMR's approach including the relevant script files:

1. The first step, image pre-processing, involves autocropping (in autocrop.py), block-based background removal and CLAHE (Contrast Limited Adaptive Histogram Equalization) illumination normalization (both in color_adjust.py), and morphological staff line enhancement (in staff_detection.py).
2. Then, U-Net semantic segmentation is performed on patches using a sliding window approach. This produces six semantic output channels, staff lines, note heads, symbols, clefs, stems/rests, and a composite all-symbols mask. This is all in inference_segnet.py.

3. The symbol detection step involves creating geometric objects from the pixel-level segmentation masks. Through contour detection, HOMR creates rotated bounding ellipses for note heads, rotated bounding boxes for staff line fragments and axis-aligned bounding boxes for clefs and key-signatures. This is all in bounding_boxes.py
4. HOMR detects complete musical staffs using an anchor-based algorithm that identifies clefs and bar lines as reliable reference points, constructs five-line staffs using the anchors, calculates the distance between staff lines, merges grand staffs, and assigns detected notes to parent staffs with calculated vertical positions. This is all in staff_detection.py. Furthermore, HOMR also corrects warped staffed lines in staff_dewarping.py after figuring out where the staffs are.
5. Notably, HOMR's transformer-based rhythm parsing creates five parallel streams of tokens defined in vocabulary.py: rhythm, pitch, lift, articulation, and position. There is also a separate "states" that keeps track of the current clef (for upper and lower staff) and the current key signature. The inference itself is in decoder_inference.py. The transformer that HOMR uses is based on Polyphonic-TrOMR [9].
6. The final step of HOMR is to generate the MusicXML output file from the transformer's output token streams. This is done by grouping symbols into chords and detecting tuplets and time signatures to calculate proper divisions. The process of building the MusicXML file is done in music_xml_generator.py.

The model architectures and recognition techniques used by LEGATO and HOMR reflect their divergent design philosophies. While LEGATO's [3] end-to-end methodology involves using a general-purpose vision encoder model with a decoder to generate music notation, HOMR's [7] approach relies on more specialized models and makes more use of music domain knowledge. HOMR's architecture explicitly encodes musical structure through its multi-stream token generation, whereas LEGATO implicitly learns these constraints with its ABC notation-generating decoder. These differences in technique between LEGATO and HOMR are valuable to analyze and test for ORCA's implementation of OMR.

5 Current Issues with Our Implementation: Usage of HOMR

While reviewing usages of deep learning OMR models [6], [8], we found that a large majority were trained and tested on Beethoven's 32 piano sonatas. Although this could be chalked up to the wide availability of scanned PDFs and audio files of these sonatas, based on our own piano-playing experience, we theorize that Beethoven's sonatas were chosen because they are among some of the least visually dense pieces of music. Most of Beethoven's sonatas do not stray far from the five staff lines and are not accidental heavy, instead focusing more on the performer's interpretation of the piece. However, this would suggest that much of the reported "success" in these papers are likely skewed by the relative simplicity of Beethoven's sonatas.

Keeping this in mind, we decided to test the boundaries of HOMR [7], a model that we ultimately chose to use in our implementation. To better pinpoint HOMR's specific issues, we used already-scanned images of existing sheet music, avoiding blurred, tilted or warped real life images that could interfere with identifying the true sources of error.

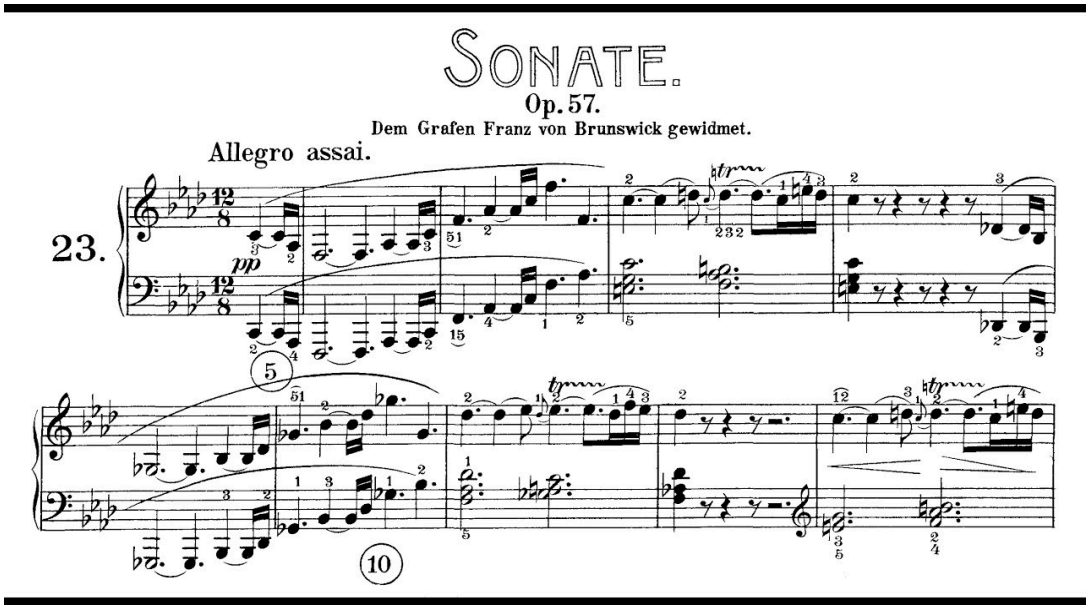

Figure 1

The Beethoven Appassionata (Piano Sonata No. 23 in F minor, Op. 57) (see Figure 1) unsurprisingly had the best accuracy when it came to translating the piece. It had:

- 2 pitch / accidental / missing-or-extra-note issues
- 1 ornament / special-notation issue
- 1 tie-handling / sustain-notation issue

Comparatively, the model performed poorly to Chopin and Debussy, as they tend to have more complex music theory usage and higher visual density in their pieces.

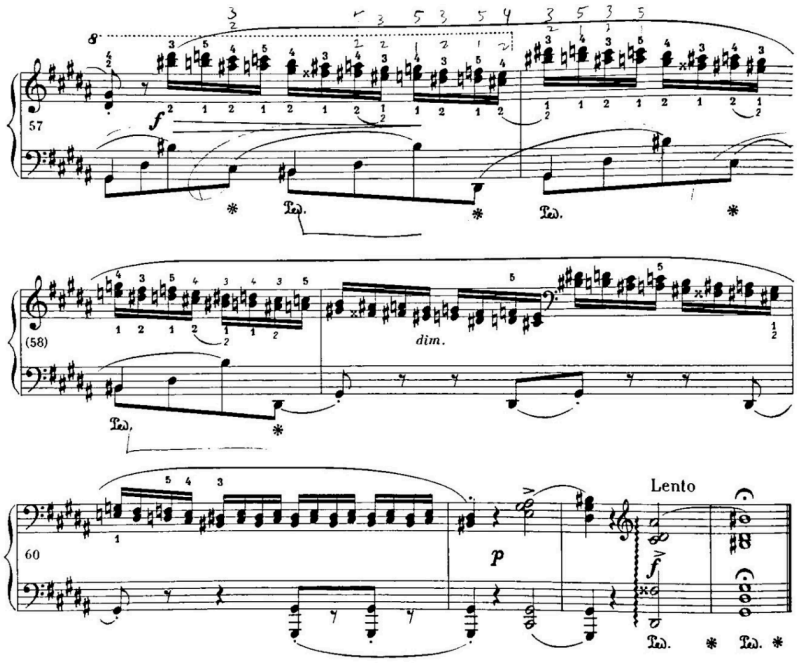

Figure 2

For Chopin Etude Op 25 No. 6 (see Figure 2), more traditional/simple errors appear, such as missing notes and missing accidentals. However, there was a **peculiar** issue where half an entire measure was cut off and combined with the next measure. This will be further tested in the future. Overall the model made the following errors:

- 5 rhythm / meter / duration / partitioning issues
- 7 pitch / accidental / missing-or-extra-note issues
- 2 ornament / special-notation issues
- 1 clef / voice-assignment issue

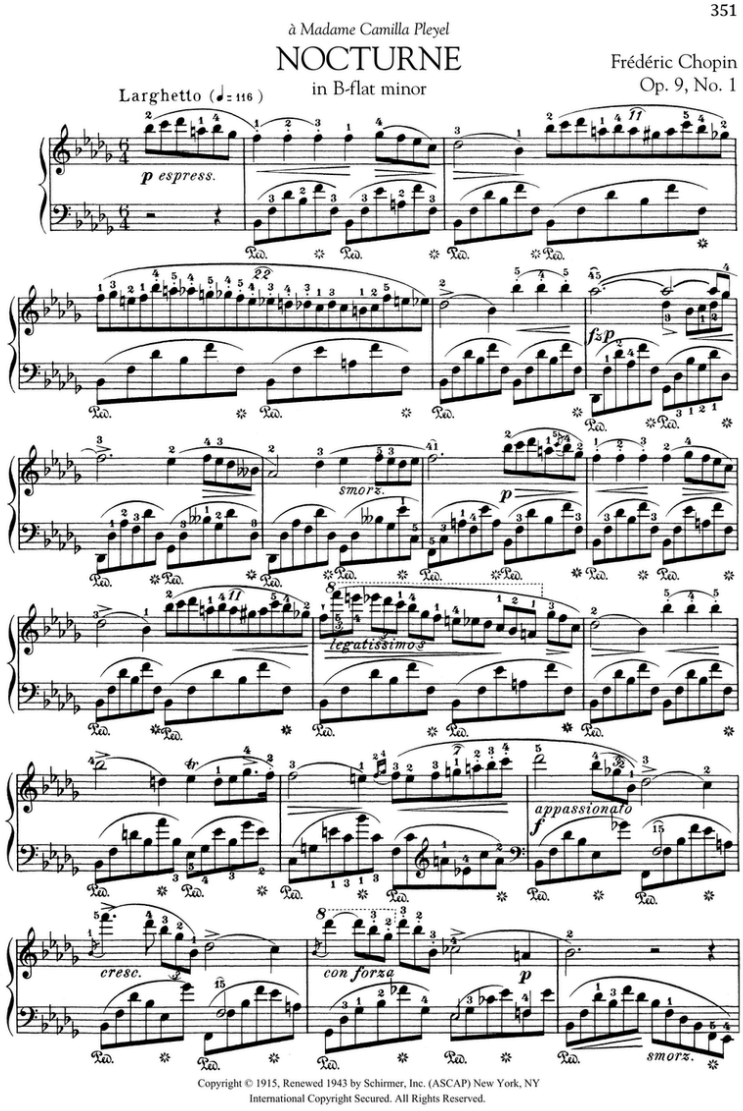

Figure 3

While we will only show the Chopin Nocturne results, Debussy Arabesque No. 1 exhibited very similar conceptual errors. In general, the model made the following errors:

- 8 rhythm / meter / duration / partitioning issues
- 12 pitch / accidental / extra-or-missing-note issues
- 5 ornament / special-notation recognition issues
- 1 conceptual / interpretation-standard issue (grace-note timing convention)

More specifically, we observed that the model completely ignored ties, trills (see Figure 1 for examples), and most grace notes. Additionally, it appears that the model has likely never seen the “up an octave” symbol or the 2 against 3 rhythm. A 2 against 3 polyrhythm is usually when you play 2 notes in the left hand (LH) and 3 notes in the right hand (RH) and is commonly seen in

Romantic era piano pieces, a big contrast to the classical Beethoven-style pieces most models use. This often leads to the model either misclassifying the LH rhythm or the RH rhythm.

6 Next Steps

The first step would be to increase the variety of the training and testing dataset to ensure the model learns not to ignore ties, trills, grace notes, and the “up an octave” symbol before retraining. If those issues can be sufficiently addressed, we would then add more real-life images and use data augmentation to synthetically generate additional data and simulate blur and warping in the dataset.

With regards to postprocessing, we would like to add measure segmentation for better duration and rhythm accuracy. This measure-by-measure double-checking would ensure that the RH and LH play note duration makes sense and there are no issues such as the RH and LH notes adding up to different numbers of beats. Furthermore, being able to read the key signature and noting the main key and corresponding scale of the piece would help in correcting any notes that the model is unsure of and is estimating. This could look like flagging any pitches whose confidence level is only slightly above the threshold and then reevaluating the top three choices while also taking the scale into account (i.e., notes within the scale weighted more heavily than notes outside the scale). This would at least ensure that the "estimated" notes make musical sense and do not sound dissonant.

For ORCA’s implementation of OMR, it is worthwhile to investigate the effectiveness of a methodology that strikes a balance between using pretrained transformer models as seen in LEGATO and the more explicit domain knowledge-based approach seen in HOMR. Figuring out the strengths and weaknesses of these approaches, while also adding and testing our own unique processing methods, would be a useful step to take to implementing a better OMR system.

7 Case Study

PlayScore[2] [10] is a widely-used commercial OMR application that enables users to scan sheet music using their phone camera and play the music back. Its accessibility makes it popular among performers who need a quick way to hear a playback of their sheet music. As a popular consumer-facing music-recognition solution on the market, it provides valuable insights into the current state of commercially available OMR software.

We stress-tested the app using Chopin Nocturne Op. 9 No. 1 (see Figure 1) and the app produced the following errors:

- 1 rhythm / meter / duration / partitioning issue
- 3 pitch / accidental / missing-or-extra-note issues
- 1 ornament / special-notation issue
- 1 texture / chord-vs-single-note parsing issue

- 1 conceptual / systemic issue

The biggest difference between the app results and HOMR is that it recognized the polyrhythms, showing that the network(s) PlayScore[2] uses have seen these types of rhythms before in their training data. Strangely, the model starts off strong and starts making mistake after mistake. This could be due to potential error propagation commonly seen in models that follow the traditional pipeline (see section 2).

8 References

[1] E. Shatri and G. Fazekas, "Optical music recognition: State of the art and major challenges," in *Proc. Int. Conf. Technologies for Music Notation and Representation (TENOR'20/21)*, Hamburg, Germany, 2020, pp. 175–184.

[2] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. S. Marçal, C. Guedes, and J. S. Cardoso, "Optical music recognition: State-of-the-art and open issues," *Int. J. Multimed. Inf. Retr.*, vol. 1, no. 3, pp. 173–190, 2012.

[3] G. Yang *et al*., "LEGATO: Large-scale end-to-end generalizable approach to typeset OMR," *arXiv preprint* arXiv:2506.19065, 2025.

[4] A. Ríos-Vila, J. Calvo-Zaragoza, and T. Paquet, "Sheet Music Transformer: End-to-End Optical Music Recognition Beyond Monophonic Transcription," *arXiv preprint* arXiv:2402.07596, 2024.

[5] D. Byrd and M. Schindele, "Prospects for improving OMR with multiple recognizers," in *Proc. 7th Int. Conf. Music Information Retrieval (ISMIR)*, Victoria, Canada, 2006, pp. 41–46.

[6] F. Kurth, M. Müller, C. Fremerey, Y.-h. Chang, and M. Clausen, "Automated synchronization of scanned sheet music with audio recordings," in *Proc. 8th Int. Conf. Music Information Retrieval (ISMIR)*, Vienna, Austria, 2007, pp. 261–266.

[7] C. Liebhardt, *homr: Optical Music Recognition software*, GitHub repository, Available: https://github.com/liebharc/homr.

[8] C. Fremerey, M. Müller, F. Kurth, and M. Clausen, "Automatic mapping of scanned sheet music to audio recordings," in *Proc. 9th Int. Conf. Music Information Retrieval (ISMIR)*, Philadelphia, PA, USA, 2008, pp. 413–418.

[9] Y. Li *et al*., *Polyphonic-TrOMR: Transformer-based polyphonic optical music recognition* (TrOMR), GitHub repository, NetEase, Available: https://github.com/NetEase/Polyphonic-TrOMR.

[10] "What is optical music recognition?," *PlayScore* blog, Nov. 8, 2025. [Online]