# Implementation of Optical Music Recognition for a Web-Based Common Music Notation Image Reader

Arleen Chrysantha Gunardi
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
chrysantharleen@gmail.com

Rinaldi Munir
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
rinaldi@staff.stei.itb.ac.id

***Abstract*—Music notation serves as a standardized visual language for preserving musical works across generations. Despite advances in digital technology, a significant number of music scores remain in printed form, highlighting the need for effective digitization methods to support documentation, education, and distribution.**

**This study presents the implementation of Optical Music Recognition (OMR) for a web-based application to convert monophonic printed music notation images into digital formats: MusicXML and MIDI. The system integrates image preprocessing to enhance input quality and comprises image segmentation, OMR model prediction, conversion and correction to MusicXML, and subsequent conversion to MIDI. An existing pretrained OMR model from Calvo-Zaragoza & Rizo (2018), trained on the PrIMuS dataset, was adopted as the recognition backbone. This model achieved a Character Error Rate (CER) of 3.88% and Sequence Error Rate (SER) of 58.22%. To further improve performance, a voting mechanism adapted from ROVER method was applied, generating five prediction variants with a 0.1 dropout rate during inference, resulting in a reduced CER of 2.99% and SER of 41.86%.**

**System evaluation shows that the voting mechanism effectively corrects prediction errors, particularly in pitch determination, but is less effective for images with thick, closely spaced staff lines. With improved input image quality, the proposed system shows strong potential as a reliable solution for printed music score digitization.**

***Keywords*—*Optical Music Recognition (OMR), music score digitization, MusicXML, MIDI***

## I. Introduction

Music notation serves as a universal visual representation used to preserve and communicate musical ideas across different cultures and historical periods. It is a medium between composers and performers, conveying musical interpretation and reproduction of musical works [1]. By presenting musical elements into a standardized symbolic language, music notation allows continuous appreciation, study, and performance of musical compositions while retaining their artistic and expressive intents envisioned by the composers.

Historically, musical notation has been written on paper. However, the advancement of digital technology has introduced new methods for representing music in machine-readable formats, which support further editing, sharing, playback, and analysis. Among these, MusicXML became a widely adopted standard due to its structured, flexible, and interoperable design [2]. The digital representation of music offers considerable advantages for modern musicians, educators, and archivists, increasing the demand for efficient methods to convert printed scores into digital formats.

Despite the growing need for digital music archives, numerous existing musical works remain archived as printed music sheets. The manual transcription of these scores into digital form is often time-consuming and requires specialized knowledge in music notation. To address this challenge, Optical Music Recognition (OMR) technology has been developed to automate the conversion process. OMR enables computers to recognize and interpret musical symbols from scanned images using pattern recognition, transforming them into structured formats such as MusicXML or MIDI [3].

While OMR systems have shown considerable progress in recent years, issues related to recognition accuracy and usability persist. Many existing models struggle with noise, dense notation, and low-quality scans, which can result in misinterpretations. For example, Acedo and Josep [4] report that their OMR system has difficulty processing scores with high levels of visual noise. Similarly, Wel and Ullrich [5] observed that their model's accuracy declines with dense or overlapping notations, and the system developed by Alfaro-Contreras et al. [6] demonstrates limited effectiveness on distorted or degraded images.

To address these limitations, this research proposes a web-based application that integrates an improved OMR model tailored for printed monophonic music notation. The system employs image preprocessing techniques to enhance input quality and reduce detection errors. A voting-based inference mechanism adapted from ROVER method is implemented to increase prediction accuracy by aggregating multiple outputs. The web-based application supports the generation of corrected MusicXML and MIDI formats. In addition to its practical functionality, the system is designed with accessibility in mind, making it a useful educational tool for a broader audience. This work contributes to the advancement of OMR systems by offering a solution that is not only accurate and robust but also user-friendly.

## II. Literature Review

### A. Common Music Notation

Common Music Notation (CMN) is a standardized visual system for representing music, enabling communication of musical ideas across cultures and eras [7] [8]. Originating around 650 AD to record musical works [9], CMN has developed into the modern notation system used worldwide. It consists of interrelated graphical symbols whose meaning depends on spatial placement and contextual relationships, including notes—defined by pitch, duration, intensity, and timbre [10]—with visual features such as noteheads, stems, and flags indicating duration. Pitch is determined by vertical placement on the staff lines, the use of clefs (treble, bass, alto, tenor), and ledger lines for extended ranges. Accidentals (sharp, flat, natural) alter pitch by semitone intervals, while key signatures apply these alterations consistently throughout a piece. Rhythm is conveyed through note values, rests, and duration extensions such as dots or ties, with measures grouping beats according to time signatures [10]. The interpretation of each symbol depends on its relationship with other symbols and the notational context, making CMN a complex system compared to alternative notation systems.

### B. Digital Music Representations

Music notation can be represented digitally in several formats, among which MusicXML and Musical Instrument Digital Interface (MIDI) are widely used. MusicXML, introduced in 2003, is a universal standard for representing common music notation across various music software applications [11]. Based on XML, it encodes semantic information about how music is written and performed, enabling both human and machine readability [11] [12]. MusicXML is primarily designed for the complete preservation and exchange of digital sheet music.

MIDI, in contrast, is a communication protocol established in 1983 for recording and controlling synthesizers and other electronic instruments [13]. Rather than storing audio or full notation, MIDI encodes performance instructions—such as pitch, velocity, and duration—through discrete digital messages [14]. These messages facilitate real-time interaction between instruments and software, making MIDI particularly suited for live performance and instrument control. Whereas MusicXML serves as a comprehensive visual and semantic representation of music for score distribution, MIDI focuses on performative representation without written visual notation.

### C. Digital Image Processing

Digital image processing refers to the use of digital computers to process digital images, defined as continuous functions over a two-dimensional field with intensity or gray-level values [15]. It encompasses image acquisition, sampling, quantization, transformation, and enhancement in both spatial and frequency domains to improve image quality and facilitate accurate interpretation. Specific techniques relevant to music notation recognition include perspective transformation to correct geometric distortion and align the viewpoint with the target plane for improved object recognition and data augmentation [16], denoising to remove visual artifacts (e.g., noises and uneven lighting) using filters such as mean, median, or alpha-trimmed mean, and binarization to convert grayscale or color images into binary form via global, local, or adaptive thresholding methods, such as Otsu's algorithm for bimodal histograms [17].

### D. Computer Vision

Computer vision, a subfield of artificial intelligence, aims to extract meaningful information from visual components by emulating the human brain's perception and interpretation of visual stimuli [18]. It integrates concepts from digital image processing, pattern recognition, artificial intelligence, and computer graphics, with specific techniques tailored to the application domain and data type. Core processes in computer vision include image acquisition, segmentation, understanding, and feature extraction. Pattern recognition, a key branch of computer vision, focuses on identifying objects by transforming images to improve interpretability and quality, often in conjunction with segmentation techniques that divide images into homogeneous pixel regions based on attributes such as color, intensity, and texture [19].

Deep learning, a subset of machine learning, models high-level data abstractions using multiple layers of artificial neurons [20]. Its effectiveness in large-scale visual object recognition has been demonstrated through various architectures, most notably Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Convolutional Recurrent Neural Networks (CRNNs). CNNs are optimized for image-based tasks through convolutional layers for local feature extraction, ReLU activation for non-linearity, pooling for spatial downsampling, fully connected layers for feature integration, and softmax for probabilistic classification [21]. RNNs, in contrast, are suited for sequential data by incorporating temporal dependencies through recurrent connections and hidden layers [22]. CRNNs combine the spatial feature extraction of CNNs with the temporal modeling of RNNs, making them highly effective for tasks requiring both spatial and sequential analysis, such as speech recognition, text recognition, and Optical Music Recognition (OMR) [23]. While CRNNs offer superior representational power, they demand greater computational resources and careful training strategies.

### E. Optical Music Recognition

Optical Music Recognition (OMR) is a research domain focused on the reading of music notation computationally [10] Definitions of OMR vary, ranging from task-specific interpretations, such as converting musical scores into MIDI, to broader views positioning OMR as a specialized form of Optical Character Recognition (OCR) for music scores. Traditional OMR pipelines involve preprocessing of sheet music, staff line detection and removal, musical object identification, classification, and reconstruction of the music sheet [24]. Modern approaches utilize deep learning architectures, including CNNs, RNNs and Sequence-to-

Sequence models, to address the translation of music notation within its contextual framework [5].

The performance of OMR systems can be assessed using various metrics, depending on the approach. For end-to-end or string-based OMR systems, Character Error Rate (CER), defined in (1), and Sequence Error Rate (SER), are common metrics. CER measures the average proportion of symbol edits—substitutions ($S_i$), deletions ($D_i$), and insertions ($I_i$)—transform a predicted sequence into the ground truth, normalized by the total number of symbols ($N_i$). Alternatively, SER evaluates the proportion of predicted sequences that contain at least one error (*E*) relative to the total number of sequences (*N*) [25].

$$CER = \frac{1}{n}\sum_{i=0}^{n}\frac{S_i + D_i + I_i}{N_i} \quad (1)$$

$$SER = \frac{E}{N} \quad (2)$$

SER is considered more stringent, as it only counts sequences as correct when they match the reference perfectly, whereas CER evaluates symbol-level accuracy. This distinction makes SER particularly suitable for semantic reconstruction tasks, while CER remains valuable for low-level symbol recognition.

### F. *Error Reduction Methods in Recognition System*

Error reduction in recognition systems has been approached through both output-level and model-level techniques. One prominent output-level method is Recognizer Output Voting Error Reduction (ROVER), introduced by NIST in 1997, which combines transcriptions from multiple recognizers and applies a voting mechanism to select the most frequent word choice, thereby reducing the Word Error Rate (WER) compared to individual systems [26]. At the model level, dropout regularization has been employed in neural architectures such as Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNN) to mitigate overfitting during training. Past studies presented its effectiveness for lowering error rates in speech recognition systems trained with Connectionist Temporal Classification (CTC) [27].

## III. Solution Design

This study proposes a web-based application for converting printed monophonic common music notation image into MusicXML, MIDI, and PDF formats. The system integrates image preprocessing, a fine-tuned OMR model, post-processing corrections, and a user-friendly web interface. Fig. 1 illustrates the overall workflow, begins with an input music sheet image and proceeds through perspective transformation, denoising, automatic rotation, and binarization. The processed image is segmented into individual staff lines and passed through a fine-tuned OMR model for symbol prediction. The prediction output encoding is converted into MusicXML and refined through a correction algorithm to comply with standard common music notation rules and subsequently converted into MIDI. The system outputs MusicXML, MIDI, and PDF files that can be downloaded by the user.

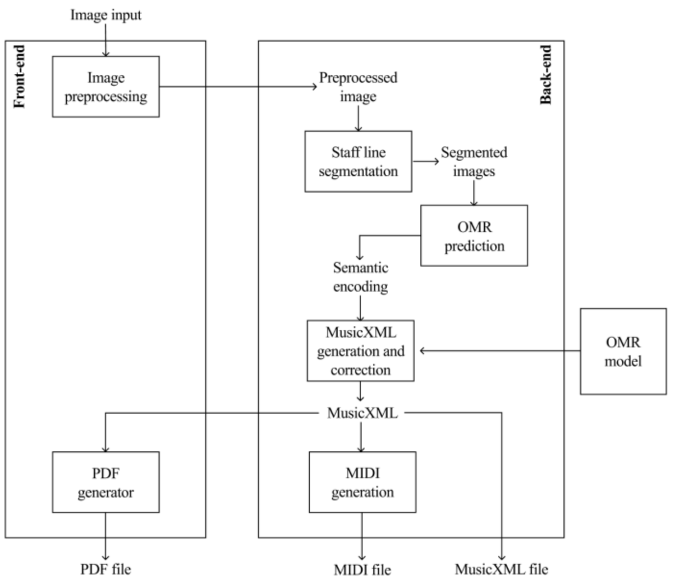


Fig. 1. Gerenal architecture and workflow

To address the problem of reduced recognition accuracy in challenging image conditions, the proposed solution integrates an image preprocessing module directly into the web application. This module enables users to enhance the captured score image quality prior to OMR processing, including correcting skews by perspective transformation and automatic rotation, denoising with Gaussian filter, and binarization using Otsu method or thresholding by value chosen by the user. Automatic rotation is done after perspective transformation by choosing the best angle, chosen by analyzing the pixel distribution histogram to see which angle the staff lines aligned horizontally the most. This step ensures that the input to the recognition model is both visually clear and structurally consistent.

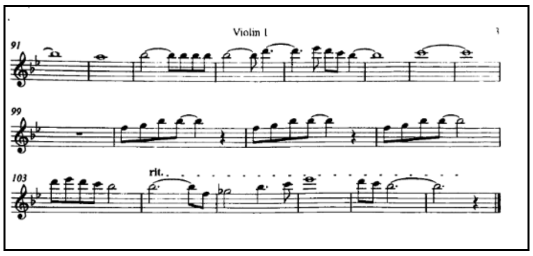
To mitigate reduced recognition accuracy under challenging input image conditions, the proposed solution incorporates an image preprocessing module within the web application. The module enhances the quality of the captured score image before OMR processing by applying several operations:

- perspective transformation to correct skewed images;
- automatic rotation to align staff lines horizontally, performed after perspective correction by selecting the orientation that maximizes horizontal alignment of staff lines, as determined from the pixel distribution histogram;
- denoising with Gaussian filtering; and

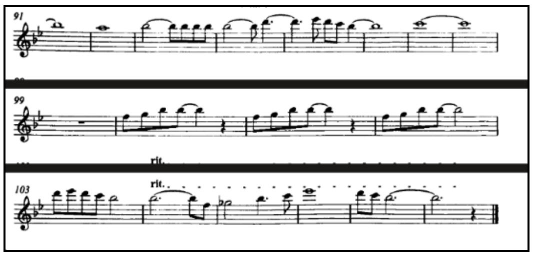
- binarization using either Otsu's method or a fixed threshold specified by the user.

These preprocessing steps ensure that the input to the recognition model is visually clear and structurally consistent, thereby improving subsequent symbol recognition.

Furthermore, staff line segmentation was performed to divide the musical score image into individual lines images, as the OMR model is limited to processing images containing a single staff at a time. A standard staff consists of five horizontal lines, with additional ledger lines appearing optionally above or below them. The segmentation process begins by projecting the pixel values of the binary image onto the vertical axis to generate a distribution histogram. Peaks in this histogram indicate the locations of staff lines, which are used to determine segmentation boundaries. For potential ledger lines, margins are added above and below each identified region. The resulting segments contain isolated staves suitable for OMR processing. An example of this segmentation procedure is illustrated in Fig. 2.



(a)



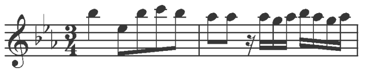
(b)

Fig. 2. PrIMuS dataset example. (a) ideal image. (b) distorted image. (c) semantic encoding. (d) agnostic encoding.

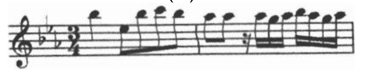
## A. Dataset

The OMR model in this study was trained using the Printed Images of Music Staves (PrIMuS) dataset [28], which contains 87,678 music score fragments. Each fragment consists of two measures with clef, key signature, and time signature information, and is provided in multiple forms: ideal images, distorted images, Music Encoding Initiative (MEI) format, Plaine and Easie (PAE) code, as well as semantic and agnostic encoding. For training and evaluation, the dataset offers two types of ground truth encodings: semantic and agnostic. Semantic encoding represents the score as a sequence of music symbols with musical meaning, while agnostic encoding treats each notation element as an isolated graphical symbol without contextual interpretation.

Fig. 3 visualizes an example of both encodings for a score fragment containing three flats, corresponding to an E♭ major key signature. In the semantic representation, this is encoded as a single token (keySignature-EbM), while the agnostic representation encodes each flat individually (accidental.flat-L3, accidental.flat-S4, accidental.flat-S2). Although the agnostic representation provides a finer level of granularity and is useful for handling non-standard notations, it is generally more verbose. For end-to-end OMR systems aimed at producing structured digital music formats such as MusicXML or MIDI, semantic encoding is more suitable, as it preserves the contextual and sequential relationships necessary for musical interpretation.



(a)



(b)

```
clef-G2, keySignature-EbM, timeSignature-3/4, note-
Bb5_quarter, note-Eb5_eighth, note-Bb5_eighth, note-
C6_eighth, note-Bb5_eighth, barline, note-Ab5_eighth,
note-Ab5_eighth, rest-sixteenth, note-Ab5_sixteenth,
note-G5_sixteenth, note-Ab5_sixteenth, note-
Bb5_sixteenth, note-Ab5_sixteenth, note-G5_sixteenth,
note-Ab5_sixteenth, barline
```

(c)

```
clef.G-L2, accidental.flat-L3, accidental.flat-S4,
accidental.flat-S2, digit.3-L4, digit.4-L2, note.quarter-
S6, note.beamedRight1-S4, note.beamedBoth1-S6,
note.beamedBoth1-L7, note.beamedLeft1-S6, barline-L1,
note.beamedRight1-L6, note.beamedLeft1-L6,
rest.sixteenth-L3, note.beamedRight2-L6,
note.beamedBoth2-S5, note.beamedLeft2-L6,
note.beamedRight2-S6, note.beamedBoth2-L6,
note.beamedBoth2-S5, note.beamedLeft2-L6, barline-L1
```

(d)

Fig. 3. PrIMuS dataset example. (a) ideal image. (b) distorted image. (c) semantic encoding. (d) agnostic encoding.

## B. OMR Model Experiment Design

The OMR model is based on a selected pretrained CRNN architecture, chosen after comparative evaluation against CNNs and RNNs individually. While CNNs effectively extract spatial features and RNNs capture sequential dependencies, the CRNN combines these advantages, enabling an end-to-end recognition that can capture both the spatial and temporal aspects of music notation. The CNN layers extract visual features from individual notation symbols, then the RNN layers process the sequential structure to reconstruct the musical content.

The experiment began with dataset preparation, which involved splitting the data into training and validation subsets. Each image was normalized and resized to match the model's input requirements, followed by preprocessing to ensure the data quality. Data augmentation was also applied, introducing transformations such as blurring, contrast and brightness adjustments, sharpening, and noise addition. This process aimed to enhance the model's robustness by increasing data variability. After the dataset preparation, the pretrained OMR model was loaded and fine-tuned to adapt to variations in notation style and symbol representation. This transfer learning strategy allowed the model to leverage previously learned representations while significantly reducing training time and computational cost. Upon completing training, the model was evaluated using transcription quality metrics against the ground truth: Character Error Rate (CER), defined in (1), and Sequence Error Rate (SER), defined in (2). Finally, the trained model weights and training checkpoints were stored to support reproducibility and further experimentation.

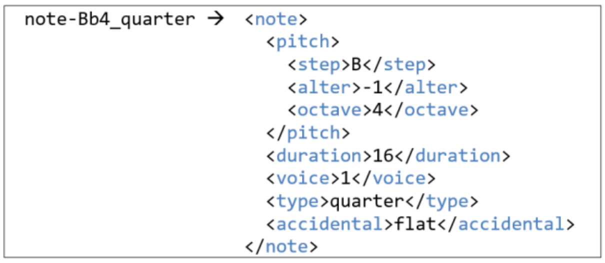
### *C. Post-Processing with Voting Mechanism Design*

To reduce error rates and enhance model performance, postprocessing was applied using a voting-based mechanism inspired by the Recognizer Output Voting Error Reduction (ROVER) method [26]. This approach combines multiple prediction outputs and selects the token with the highest occurrence frequency as the final result. However, since the underlying model is deterministic and produces identical outputs for the same input image, prediction variability was generated by enabling dropout during inference. This technique, commonly referred to as Monte Carlo dropout, allows stochastic variations in predictions and has been shown to improve ensemble robustness [27].

Experiments were conducted by varying the number of aggregated predictions $N$ and the dropout rate $r_d$. Following prior work [27], a baseline configuration of $N = 4$ and $r_d = 0.2$ was adopted, while additional combinations were also tested with $N \in \{4, 5\}$ and $r_d \in \{0.2, 0.1, 0.05\}$. Each input image was processed $N$ times according to the chosen parameters, and the final output sequence was obtained by majority voting at the token level.

### *D. MusicXML Conversion and Correction Algorithm Design*

The semantic encoding predictions produced by the OMR model were converted into MusicXML through a parsing process that mapped each semantic token to its corresponding XML representation, as illustrated in Fig. 4. This step enabled the output of the recognition model to be transformed into a widely supported digital notation format, consistent with existing standards.



```
note-Bb4_quarter →  <note>
                      <pitch>
                        <step>B</step>
                        <alter>-1</alter>
                        <octave>4</octave>
                      </pitch>
                      <duration>16</duration>
                      <voice>1</voice>
                      <type>quarter</type>
                      <accidental>flat</accidental>
                    </note>
```

Fig. 4. Semantic encoding to MusicXML mapping example.

To ensure the generated MusicXML adhered to common music notation rules, a correction algorithm was implemented:

- clef adjustment by matching the predicted clef with typical instrument ranges (e.g., G2 for violin or flute, F4 for cello or tuba), preventing octave misalignments caused by incorrect clef predictions;
- measure duration validation to enforce rhythmic consistency by computing note and rest durations relative to the time signature; overfilled measures were automatically split using barlines with tied notes applied when durations exceeded the remaining capacity; and
- key signature-based pitch alteration to ensure that sharps and flats were consistently applied across the score according to the predicted key signature (e.g., F♯ and C♯ in D major), unless explicitly modified by accidentals.

These corrections preserved both the structural and tonal integrity of the transcribed score, serving more accurate interpretation in downstream applications.

### *E. Web Application Design*

The proposed web-based application consists of the frontend and the backend, as illustrated in Fig. 1. The frontend provides user interaction through four modules: image upload, image preprocessing, MusicXML viewer, and PDF generator from MusicXML. The front end communicates with the backend via APIs for computational processing. The backend is composed of four modules: image (staff line) segmentation, music notation prediction using the OMR model, conversion of semantic encoding into MusicXML with an integrated correction function, and generation of MIDI files from MusicXML.

The system does not require a permanent database, since both the input image and OMR results are processed within a single session and discarded afterward. All data are stored temporarily in memory during processing and are deleted once the output has been displayed or downloaded. This design minimizes storage requirements while ensuring efficient, session-based processing.

## IV. Implementation and Results

The implementation is divided into two main stages: model development and web application development. The model development stage includes dataset preparation, model training and evaluation, as well as post-processing. The experiments were conducted in a Kaggle notebook environment with a P100 GPU, using Python and several supporting libraries, including Tensorflow, OpenCV Python, Supervision, and NumPy. The implemented OMR model was limited to predicting only a subset of musical symbols, including clefs, time signatures, key signatures, barlines, notes, rests, rhythmical symbols, and accidentals. Other musical notations, such as ornaments, dynamics, and textual annotations, were not included in the scope of this work.

### *A. Dataset Preparation*

The dataset preparation stage ensured that the data were suitable for training the model. This experiment uses PrIMuS dataset, consisting of 87,678 image snippets of music scores with ground-truth semantic labels. Semantic labels were first validated against a predefined vocabulary, which mapped each symbol into a numerical index and vice versa. The vocabulary was restricted to clefs, time signatures, key signatures, barlines, notes, rhythmic symbols, and accidentals, excluding ornaments, articulations, text, and other musical elements.

The dataset was then randomly split into 90% for training and 10% for validation. Each score image was converted to grayscale, normalized, and resized before preprocessing, which included brightness and contrast adjustment, sharpening, blurring, and noise injection to enhance variability. To improve efficiency, data were loaded in batches and folds, enabling staged training without exceeding memory capacity. Additionally, multithreading was applied to accelerate the loading of images and labels.

### *B. Model Training and Evaluation*

In this study, we adopted the OMR model proposed by Calvo-Zaragoza & Rizo [25], which achieved strong recognition performance of 3.88% CER and 58.22% SER on the evaluation dataset. Preliminary experiments with training from scratch using EfficientNetB0 and fine-tuning approaches were conducted, but these models exhibited significantly higher error rates compared to the pretrained model. Consequently, the pretrained model was selected and integrated into the proposed system.

### *C. Model Post-Processing*

As described in Section III-C, a voting mechanism with stochastic prediction generation was designed to enhance recognition performance. Table I summarizes the experimental results obtained by varying the number of aggregated predictions ($N$) and dropout rates ($r_d$) on the inference. The configuration with $N = 5$ and $r_d = 0.1$ achieved the best performance, reducing the Sequence Error Rate (SER) from 58.22% to 41.86% and the Character Error Rate (CER) from 3.88% to 2.99%. This demonstrates that the proposed voting strategy substantially improves recognition robustness by reducing both sequence-level and symbol-level errors by finding consistency across multiple prediction variants.

TABLE I. POST-PROCESSING EXPERIMENT RESULT

| **Model Configuration** | | **Evaluation** | |
|---|---|---|---|
| *N* | $r_d$ | **CER** | **SER** |
| Baseline (without voting) | | 3.8843% | 58.2183% |
| 4 | 0.2 | 3.2243% | 43.8120% |
| 4 | 0.1 | 3.0655% | 42.5915% |
| 4 | 0.05 | 3.0698% | 42.7512% |
| 5 | 0.2 | 3.1459% | 42.4204% |
| **5** | **0.1** | **2.9928%** | **41.8615%** |
| 5 | 0.05 | 2.9990% | 42.3178% |

It can be observed that the error rates decrease as the dropout rate is reduced from $r_d = 0.2$ to $r_d = 0.1$, indicating that a moderate level of stochasticity provides sufficient variability to generate complementary prediction variants while maintaining prediction reliability. However, when the dropout rate is further reduced to $r_d = 0.05$, the performance slightly degrades compared to $r_d = 0.1$. This behavior indicates the reduced diversity of predictions: with too little dropout, the prediction variants become overly similar, limiting the effectiveness of the voting mechanism. In contrast, a higher dropout rate such as 0.2 introduces excessive noise, which negatively affects the quality of individual predictions. Hence, $r_d = 0.1$ offers the best trade-off between diversity and reliability of predictions, resulting in the most effective voting mechanism configuration.

### *D. Web Application*

The web application described in Section II-E was implemented using a client-server architecture. The frontend was developed with Next.js, providing modules for image upload, preprocessing preview, MusicXML visualization, and PDF generation, along with download functionality for MusicXML, MIDI, and PDF outputs. The frontend was implemented in TypeScript, supported by libraries including Tailwind CSS and Shadcn UI for user interface styling, OpenCV.js for client-side image processing, OpenSheetMusicDisplay for MusicXML rendering, and jsPDF with svg2pdf.js for score export.

The backend was implemented with FastAPI, which uses RESTful APIs to handle computational tasks such as staff-line segmentation, OMR model inference, semantic encoding conversion to MusicXML with correction, and MIDI generation. The backend was developed in Python with supporting libraries, including TensorFlow and Keras for model inference, and Music21 for MIDI file generation.

To ensure efficient processing, the system employs session-based data management, where all input and output data are handled in memory during execution and discarded upon completion. This eliminates the need for persistent storage and minimizes computation overhead.

The application was tested using a set of monophonic music notation images (with details presented in Section V). The implementation demonstrates that the system is capable of generating MusicXML, MIDI, and PDF outputs in real time, confirming its effectiveness as an accessible OMR tool through a web interface.

## V. TESTING AND EVALUATION

The proposed OMR system was evaluated using the CER and SER as metrics, computed according to (1) and (2). The evaluation employed four test images consisting of short musical notation sequence captured with a mobile phone camera and uploaded via the web application. Each input was processed through preprocessing, staff-line segmentation, OMR prediction, and conversion to MusicXML, followed by application of the correction algorithm. Reference labels were defined in semantic encoding form, enabling a direct comparison with both the raw model predictions and the

TABLE II. TESTING RESULTS

| Test Case ID | Evaluation | | | | | Image Characteristics |
|---|---|---|---|---|---|---|
| | CER (No Voting) | CER (Voting) | ΔCER | SER (No Voting) | SER (Voting) | |
| T01 | 82.5% | 60% | -22.5% | 100% | 100% | Quarter notes, wide spacing; original image straight with uneven lighting (right side darker), good print; preprocessed image straight with enhanced contrast, though staff-line thickness remains inconsistent. |
| T02 | 61.25% | 45.15% | -16.39% | 100% | 100% | Quarter and eighth notes, tighter spacing, with ignored text; original image slightly skewed, uneven lighting, inconsistent staff printing; preprocessed image straight with improved contrast, staff lines partly broken or uneven but better than T01. |
| T03 | 65% | 52.5% | -12.5% | 100% | 100% | Quarter and eighth notes, dense first sequence and sparse second with multirest; original image slightly skewed, uneven lighting, good print; preprocessed image straight with enhanced contrast, staff thickness more consistent. |
| T04 | 63.49% | 61.9% | -1.59% | 100% | 100% | Predominantly eighth notes, very tight spacing, thicker and denser staff lines; original image slightly skewed, uneven lighting, good print; preprocessed image straight with consistent but thicker staff lines. |
| **Average** | **68.13%** | **54.89%** | **-13.24%** | **100%** | **100%** | - |

corrected outputs. The test cases were selected to cover diverse conditions in terms of staff density, note types, and image quality (lighting, noise, and staff-line thickness). The summary of the characteristics of the test images can be seen in Table II.

The system performance on each test case is summarized in Table II, presenting average CER and SER with and without the voting mechanism. Overall, voting improved performance substantially, reducing mean CER from 68.13% to 54.89% across the four cases. This indicates that dropout-based prediction ensembles successfully mitigate inconsistent character recognition.

For further qualitative analysis, the comparison between results with and without voting highlights distinct error patterns, the corrective impact of voting, and the corresponding changes in CER. In the first case (T01), errors were dominated by incorrect key signatures, occasional pitch misclassifications (e.g., C5 recognized as C#5), and clef misplacements. Voting substantially improved pitch consistency by aligning notes with the correct key signature, reducing the CER by 22.5%. However, mismatches in note duration and beat values remained unresolved. In the second case (T02), dense note spacing caused frequent pitch inaccuracies, accidental misinterpretations, and inconsistent durations. Voting improved pitch accuracy, particularly with accidentals, and reduced CER by 16.39%, though errors in rhythm persisted due to difficulties in symbol separation under dense notation.

In the third case (T03), the system frequently misclassified multirests as whole rests and struggled with dense note sequences. Voting led to slight improvements in pitch recognition, lowering CER by 12.5%, but multirest detection remained problematic, as their elongated shapes with embedded numbers closely resemble staff elements. Finally, in the fourth case (T04), most pitches were initially recognized correctly, but rhythm errors were common, such as misclassifying eighth notes as sixteenth notes, compounded by thick staff lines that obscured flags and beams. Voting corrected accidentals and reinforced pitch accuracy but failed to address rhythm errors, resulting in only marginal CER improvement of 1.59%. Overall, these findings indicate that voting significantly enhances pitch recognition by mitigating key-related ambiguities, while rhythm and duration recognition remain vulnerable to errors stemming from dense notation and degraded staff-line quality.

Despite improvements, error rates remain high. SER reached 100% in all cases, since any single-character error causes a sequence to be counted as incorrect. However, qualitative analysis indicates most errors are minor—often pitch deviations within one staff line—arising from noise or staff thickness inconsistencies. Rhythm errors, in contrast, caused by visual similarity of note stems and flags. Thus, although quantitative error rates are high, most mistakes fall within ranges that could be corrected by improved preprocessing, noise handling, or user's intervention by manual correction. Future work should therefore emphasize robustness against noise and staff-line variability.

## VI. CONCLUSION

This study presented the development of a web-based application integrating Optical Music Recognition (OMR) to convert printed common music notation into digital formats such as MusicXML, MIDI, and PDF. The system employed a pretrained CRNN-based OMR model by Calvo-Zaragoza & Rizo (2018) [25], enhanced with a voting mechanism adapted from ROVER by aggregating multiple dropout-based predictions. This approach improved overall recognition performance, reducing the Character Error Rate (CER) from 3.88% to 2.99% and the Sequence Error Rate (SER) from 58.22% to 41.86%.

The system was implemented end-to-end, including preprocessing, staff-line segmentation, symbol prediction, MusicXML generation, post-correction, and MIDI rendering. Experimental evaluation demonstrated that the application successfully supports all processing stages and provides an accessible interface for users to digitize and utilize music scores. Qualitative analysis further revealed that most recognition errors were minor, primarily involving pitch or rhythm misclassification due to staff-line inconsistencies, symbol similarities, or image noise. The voting mechanism proved effective in mitigating pitch-related errors, though rhythm recognition remains a challenge.

Future work should focus on enhancing robustness against noise and staff variability by employing more diverse training datasets and experimenting with alternative architectures. Additional development can be done by providing interactive manual correction tools, incorporating user feedback into retraining, and advancing the MusicXML correction algorithms to cover a broader set of notation rules. Integrating other features such as MIDI playback and automated score cropping would further improve usability.

## ACKNOWLEDGMENT

The authors would like to express their gratitude to God for His blessings. The authors would also like to acknowledge Mr. Dr. Ir. Rinaldi Munir, M.T., for his valuable guidance and feedback throughout the research. The authors also wish to recognize the contributors of prior works in Optical Music Recognition, which provided the foundation and inspiration for this study. Finally, sincere thanks are given to family, friends, and colleagues for their continuous support and encouragement during the research.

## REFERENCES

[1] S. T. Pope, "Music Notations and the Representation of Musical Structure and Knowledge," *Perspectives of New Music*, vol. 24, no. 2, p. 156, 1986, doi: https://doi.org/10.2307/833219.

[2] M. Good and G. Actor, "Using MusicXML for file interchange," *Proceedings Third International Conference on WEB Delivering of Music*, p. 153, doi: https://doi.org/10.1109/wdm.2003.1233890.

[3] J. Calvo-Zaragoza, J. H. Jr., and A. Pacha, "Understanding Optical Music Recognition," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–35, Sep. 2020, doi: https://doi.org/10.1145/3397499.

[4] A. Acedo and A. Josep, "From image to MIDI: Implementing a complete OMR system for sheet music," *Dipòsit Digital de Documents de la UAB*, 2023. https://ddd.uab.cat/record/272800 (accessed Aug. 04, 2025).

[5] E. van der Wel and K. Ullrich, "Optical Music Recognition with Convolutional Sequence-to-Sequence Models.," *Proceedings of the 18th ISMIR Conference* (Cornell University), pp. 731–737, Oct. 2017, doi: https://doi.org/10.5072/zenodo.243774.

[6] M. Alfaro-Contreras, J. Calvo-Zaragoza, and J. M. Iñesta, "Approaching End-to-End Optical Music Recognition for Homophonic Scores," *Pattern Recognition and Image Analysis*, pp. 147–158, Jan. 2019, doi: https://doi.org/10.1007/978-3-030-31321-0_13.

[7] Prince George's Community College. "PGCC Open Music Theory-Fundamentals," March 2024. Humanities LibreTexts, https://human.libretexts.org/Courses/Prince_Georges_Community_College/PGCC_Open_Music_Theory-Fundamentals

[8] K. Lassfolk. "Music Notation as Objects: An Object-Oriented Analysis of the Common Western Music Notation System," *University of Helsinki*, Nov. 2004, https://helda.helsinki.fi/bitstream/10138/19386/2/musicnot.pdf.

[9] Classic FM, "How did music notation begin?," *Classic FM*, Aug. 07, 2024. https://www.classicfm.com/discover-music/music-theory/origins-music-notation/

[10] J. Calvo-Zaragoza, J. H. Jr., and A. Pacha, "Understanding Optical Music Recognition," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–35, Sep. 2020, doi: https://doi.org/10.1145/3397499.

[11] M. Good and G. Actor, "Using MusicXML for file interchange," *Proceedings Third International Conference on WEB Delivering of Music*, p. 153, doi: https://doi.org/10.1109/wdm.2003.1233890.

[12] M. Good and MakeMusic, "MusicXML 4.0," *W3C Community Group*, 2021. https://www.w3.org/2021/06/musicxml40/ (accessed Aug. 15, 2025).

[13] G. Loy, "A critical overview," *The Journal of the Acoustical Society of America*, vol. 77, no. S1, pp. S74–S74, Apr. 1985, doi: https://doi.org/10.1121/1.2022489.

[14] MIDI Manufacturers Association, "MIDI 2.0 Core Specification Collection," *MIDI.org*, Jan. 17, 2024. https://midi.org/midi-2-0-core-specification-collection

[15] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. New York, Ny: Pearson, 2018.

[16] K. Wang, B. Fang, J. Qian, S. Yang, X. Zhou, and J. Zhou, "Perspective Transformation Data Augmentation for Object Detection," *IEEE Access*, vol. 8, pp. 4935–4943, 2020, doi: https://doi.org/10.1109/access.2019.2962572.

[17] X. Xu, S. Xu, L. Jin, and E. Song, "Characteristic analysis of Otsu threshold and its applications," *Pattern Recognition Letters*, vol. 32, no. 7, pp. 956–961, May 2011, doi: https://doi.org/10.1016/j.patrec.2011.01.021.

[18] [1]F. Alsakka, I. El-Chami, H. Yu, and M. Al-Hussein, "Computer vision-based process time data acquisition for offsite construction," *Automation in Construction*, vol. 149, p. 104803, May 2023, doi: https://doi.org/10.1016/j.autcon.2023.104803.

[19] A. Gharipour and A. W.-C. Liew, "Segmentation of cell nuclei in fluorescence microscopy images: An integrated framework using level set segmentation and touching-cell splitting," *Pattern Recognition*, vol. 58, pp. 1–11, Oct. 2016, doi: https://doi.org/10.1016/j.patcog.2016.03.030.

[20] X. Hao, G. Zhang, and S. Ma, "Deep Learning," *International Journal of Semantic Computing*, vol. 10, no. 03, pp. 417–439, Sep. 2016, doi: https://doi.org/10.1142/s1793351x16500045.

[21] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a Convolutional Neural Network," *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, Aug. 2017, doi: https://doi.org/10.1109/icengtechnol.2017.8308186.

[22] A. Subasi, "Machine learning techniques," *Practical Machine Learning for Data Analysis Using Python*, pp. 91–202, 2020, doi: https://doi.org/10.1016/b978-0-12-821379-7.00003-5.

[23] X. Li and X. Wu, "Long Short-Term Memory based Convolutional Recurrent Neural Networks for Large Vocabulary Speech Recognition," *arXiv (Cornell University)*, Jan. 2016, doi: https://doi.org/10.48550/arxiv.1610.03165.

[24] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. S. Marcal, C. Guedes, and J. S. Cardoso, "Optical music recognition: state-of-the-art and open issues," *International Journal of Multimedia Information Retrieval*, vol. 1, no. 3, pp. 173–190, Mar. 2012, doi: https://doi.org/10.1007/s13735-012-0004-6.

[25] J. Calvo-Zaragoza and D. Rizo, "End-to-End Neural Optical Music Recognition of Monophonic Scores," *Applied Sciences*, vol. 8, no. 4, p. 606, Apr. 2018, doi: https://doi.org/10.3390/app8040606.

[26] H. Schwenk and J. Gauvain, "Improved ROVER using Language Model Information," 2017, *Automatic Speech Recognition: Challenges for the New Millenium*, 47-52, 2000, https://api.semanticscholar.org/CorpusID:18390942.

[27] A. Vyas, P. Dighe, S. Tong, and H. Bourlard, "Analyzing Uncertainties in Speech Recognition Using Dropout," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6730–6734, May 2019, doi: https://doi.org/10.1109/icassp.2019.8683086.

[28] "PrIMuS dataset," *Dlsi.ua.es*, 2018. https://grfia.dlsi.ua.es/primus/ (accessed Aug. 15, 2025).