Vol. 11 No. 2 Tahun 2025, pp 153-164
Jurnal Teknologi dan Manajemen Informatika
http:// http://jurnal.unmer.ac.id/index.php/jtmi
P-ISSN: 1693-6604 E-ISSN: 2580-8044

Rule-Based Pitch Inference in Optical Music Recognition on Polyphonic Scores using YOLOv12

Derend Marvel Hanson Prionggo [1], Evan Tanuwijaya [2]

Universitas Ciputra, Indonesia

Article Info

Article History
Received : 13-10-2025
Revised : 20-11-2025
Accepted : 26-11-2025

Keywords
Optical Music Recognition;
YOLOv12;
Convolutional Neural Network;
Partitur polifonik;
Rule-based Pitch Inference

✉ Corresponding Author
Derend Marvel Hanson Prionggo,
Universitas Ciputra Surabaya,
derend101@gmail.com

ABSTRACT

Optical Music Recognition (OMR) faces significant challenges when applied to polyphonic music scores, due to the high symbol density and the overlapping of notes. This study proposes a hybrid method of combining the detection of noteheads using YOLOv12 with rule-based pitch inference, which converts the spatial position of the detected noteheads into accurate pitch information. The dataset used in this study is DeepScoresV2-Dense, which is processed through annotation conversion, image normalization, and staff extraction as a reference to infer the pitch of a note. The YOLOv12 model was trained for 30 epochs using a transfer learning approach, resulting in an mAP50 value of 0.75, a precision of 0.85, and a recall of 0.58 on the validation data. The implementation of rule-based pitch inference successfully achieved a pitch accuracy of 0.87 with an F1 score of 0.87, demonstrating a balance between accuracy and completeness of prediction. This result shows that the integration of YOLOv12 and rule-based pitch inference can be an effective solution for pitch extraction in polyphonic music scores, with potential applications in music information retrieval, digital music score conversion, and an artificial intelligence-based music learning system.

INTRODUCTION

Music is one of the main forms of cultural expression and communication, which requires detailed notation of elements such as pitch, rhythm, and dynamics [1]. However, reading musical notation on music scores often poses a challenge, especially for those without a formal background in music [2]. This challenge has led to the development of computer-based solutions that convert music sheets into formats that are easier to access and understand [3]. One field that emerged from this problem is Optical Music Recognition, which is a technique for detecting and recognizing musical notation from images of music sheets so that they can be converted into digital format [4].

Research on Optical Musical Recognition (OMR) continues to develop, especially with advances in deep learning methods, particularly Convolutional Neural Networks (CNN), which have significantly improved the accuracy of musical symbol recognition [5]. Convolutional Neural Networks (CNNs) are widely used for image-based pattern recognition because they can automatically perform feature extraction and dimensionality reduction within the network, reducing the need for manual preprocessing [6], and also enabling effective recognition of complex patterns within images [7]. This characteristic makes CNN-based architectures, such

This is an open access article under the CC BY-SA license.
DOI: https://doi.org/10.26905/jtmi.v11i2.16291

as the YOLO family, suitable for detecting fine-grained musical elements like noteheads and staff lines in OMR. However, the majority of studies still focus on monophonic music scores, which are relatively simple, while research on polyphonic scores, which generally contain a more complex musical structure with multiple notes on a single staff, are still rarely conducted [8]. The visual complexity of polyphonic scores, such as the overlapping notations and variations in position due to different clefs or key signatures, poses major challenges in notehead detection [9].

This research proposes the use of the latest CNN architecture, YOLOv12, which offers enhanced local feature extraction capabilities compared to previous generations [10]. Each progression in YOLO variants has shown consistent improvements in small-object detection performance. In particular, YOLOv8 demonstrated significant gains across all model sizes, achieving up to a 33.21% increase in detection accuracy compared to YOLO-NAS for small objects [11]. These advancements suggest that newer YOLO architectures are increasingly capable of handling dense and fine-grained visual structures, providing strong motivation for exploring YOLOv12 in polyphonic OMR. Using an object detection approach, YOLOv12 can recognize hierarchical patterns such as noteheads, stems, and staff lines, while rule-based pitch inference is used to map the vertical positions of the detected objects into pitch names based on the clef and other musical signs [12]. This process includes training YOLOv12 to detect noteheads, extracting staff lines, calculating the distance between the lines as a pitch reference, and evaluating performance using mean Average Precision (mAP) and pitch accuracy.

Previous studies have shown that CNN models such as YOLOv8 can achieve mAP50–95 above 85% on the PrIMuS dataset for detecting monophonic musical symbols [13], even outperforming Mask R-CNN in precision, recall, and inference speed [14]. However, no research has applied YOLOv12 in OMR. In other domains, such as small-object detection in marine imagery, YOLOv12 has been shown to increase precision up to 0.93 and recall to 0.91 while providing shorter inference time compared to YOLOv8 and YOLOv10 [15], making it an ideal candidate for handling the overlapping challenges found in polyphonic OMR. The novelty of this research lies in the application of YOLOv12 for notehead detection in polyphonic sheet music and its integration with rule-based pitch inference, enabling the vertical coordinates of detections to be directly converted into pitch information without requiring a sequential symbol decoder. This approach is expected to improve processing efficiency while maintaining accuracy, making it relevant for Music Information Retrieval (MIR) applications and AI-based music learning systems.

METHODS

Research Stages

This study adopts a mixed methodology that combines deep learning techniques with a rule-based approach to address the complexities of detection and pitch inference in polyphonic musical scores. The research framework consists of five main stages: data collection and preprocessing, training the YOLOv12 model for notehead detection, staff-line detection, implementing rule-based pitch inference, and comprehensive evaluation. This approach is designed to overcome the limitations of conventional OMR systems, which generally focus on monophonic scores and struggle with overlapping notations in polyphonic music.

The system architecture integrates the real-time object detection capabilities of YOLOv12 with the deterministic nature of rule-based pitch inference. YOLOv12 was selected because it is the latest architecture in the YOLO family, incorporating an attention-centric design that allows for high-accuracy detection of small objects while maintaining real-time speed. Integration with rule-based pitch inference ensures that the spatial coordinates of detected noteheads can be converted into accurate pitch information based on musical context such as clef, key signature, and their relative positions to the staff lines.

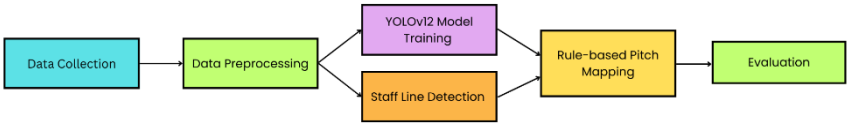


Figure 1. The Architecture of Rule-Based Pitch Inference

Data Collection

The dataset used in this study is DeepScoresV2, one of the most comprehensive datasets for computer vision–based music symbol recognition [16]. This dataset contains more than 255,385 images of digital polyphonic music scores with a total of 151 million symbol instances covering 135 classes of notation elements, such as noteheads, rests, clefs, staff lines, accidentals, time signatures, and ligatures. Each image, in .png format, represents a full page of a musical score and is equipped with multi-level annotations, including axis-aligned bounding boxes, oriented bounding boxes, semantic segmentation, and instance segmentation, allowing detailed modeling of visual structures.

This study utilizes a subset of DeepScoresV2, namely DeepScoresV2-Dense, which consists of 1,714 images with high symbol density and complex notation variations. This subset was chosen because it is more representative of the challenges involved in detecting symbols in densely packed and overlapping polyphonic sheet music [16]. In addition to visual data, DeepScoresV2 also provides musical metadata such as clef type, key signature, and time signature, which are essential for the rule-based pitch inference stage. This information ensures accurate conversion of the vertical coordinates of noteheads into pitch values, as the same spatial position may represent different notes depending on the clef or time signature. The DeepScoresV2-Dense subset has also been divided into a training set and validation set to train and evaluate the performance of the YOLOv12 model in detecting noteheads and performing precise pitch inference.
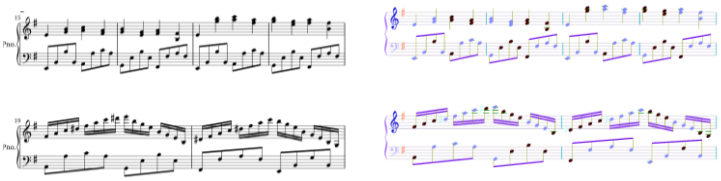


Figure 2. Example of Images from the DeepScoresV2-Dense Dataset

Data Preprocessing

The data preprocessing stage is conducted to ensure that the dataset can be optimally processed by the YOLOv12 architecture. This process involves a series of systematic steps, ranging from annotation conversion and image size normalization to preparing the data in a format suitable for model training. The first step is converting XML annotations to the YOLO format. The DeepScoresV2-Dense dataset provides original annotations in XML files containing positional information for musical symbols.

Each symbol is represented by a pixel-based bounding box along with a class label indicating the type of symbol (e.g., notehead, clef, time signature, etc.). To be compatible with YOLOv12, these annotations must be converted into the YOLO labeling format consisting of class_id, x_center, y_center, width, and height [17]. This conversion process includes recalculating coordinates so that they are normalized relative to image dimensions, ensuring each value lies within the range [0, 1].

The next step is image size normalization. All images in the dataset are resized to a standard resolution of 640 × 640 pixels. This normalization ensures consistent image dimensions across the dataset, facilitating feature extraction by YOLOv12 and accelerating the training process. In addition, staff line extraction is applied as a spatial reference for the pitch inference stage. This step is performed using horizontal line detection techniques via morphological operations and the Hough transform [18].

This method is robust to minor distortions and variations in line thickness or gaps, allowing precise localization of the five staff lines even in the presence of overlapping musical symbols. The resulting staff line position information is then used in the subsequent rule-based pitch inference stage to map the vertical position of noteheads to their correct pitch values. The final output of the data preprocessing stage is a collection of normalized .png sheet music images, along with YOLO-formatted annotation files containing symbol positions and class labels. All of this data, which has been separated into training and validation sets, will be used to train and evaluate the performance of the YOLOv12 model in detecting noteheads.

YOLOv12 Model Training

The YOLOv12 model training is conducted to detect notehead elements in polyphonic sheet music, where the detected notehead positions will be used to calculate the distance between the nearest staff lines and the notehead itself to determine the pitch of the note. The model used is the small variant (YOLOv12s) with a single target class (notehead). The YOLOv12 architecture is chosen because of its advantages in efficiently detecting small and overlapping objects, making it highly suitable for detecting noteheads in complex polyphonic scores where symbols are small and often overlap [19]. This training process is designed to ensure that the model can recognize noteheads in polyphonic sheet music with high accuracy, even in the presence of visual complexities such as staff lines, clefs, time signatures, and other overlapping musical symbols.

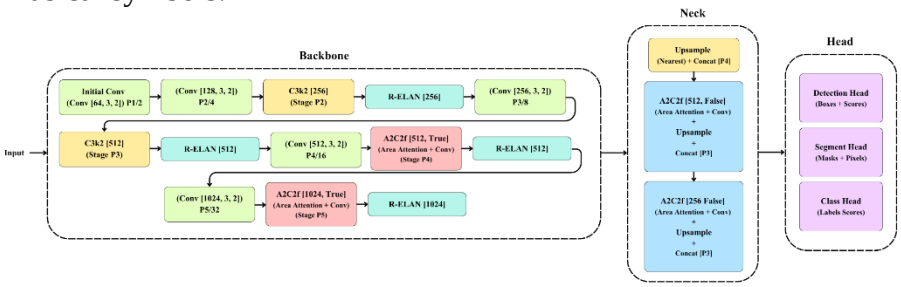


Figure 3. YOLOv12 Architecture Diagram [20]

Training was conducted using the Ultralytics YOLOv12 framework (version 8.2.0), implemented in a Python 3.10 environment with support from an NVIDIA Quadro RTX 5000 GPU to accelerate computation. The programming environment was developed in Jupyter Notebook, enabling code documentation and visualization results within a single platform. The initial pretrained model used was yolov12s.pt, previously trained on the COCO dataset for general object detection. This transfer learning approach allows the model to leverage fundamental feature representations such as lines, curves, and textures, thereby accelerating convergence when adapted to the OMR domain. The model training uses these pretrained weights to speed up convergence and maximize its ability to extract essential features such as lines and circular shapes that are relevant to musical notation.

The dataset was trained for 30 epochs with a batch size of 8, an image resolution of 640×640 pixels, and the Adam optimizer combined with a CosineAnnealing learning rate scheduler to dynamically adjust the learning rate. This strategy helps prevent overfitting while maintaining stable convergence throughout training. The initial learning rate was set to 0.001, while the momentum and weight decay followed the YOLOv12 default configuration—momentum = 0.937 and weight decay = 0.0005—which has been optimized for performance.

During training, several key metrics were used as evaluation references. The mean Average Precision (mAP) at an IoU threshold of 0.5 (mAP50) and across the 0.5–0.95 range (mAP50–95) served as primary indicators for measuring notehead detection performance across varying sizes and positions. Additionally, precision and recall were used to assess the balance between false positives and false negatives. Visualizations of the loss curve and mAP curve for each epoch were monitored to track convergence and ensure that the model did not experience underfitting or overfitting. After 30 epochs of training, the best-performing model

was selected based on the highest mAP score on the validation set. The trained model was then saved in .pt format for use in the rule-based pitch inference stage.

Staff Line Detection

Staff line detection begins by converting the sheet music image into a black-and-white binary format. The pixel intensity values are inverted so that the staff lines, originally dark, become high-intensity pixels (255). Next, a horizontal projection is computed by summing the pixel intensity values for each row of the image using the equation:

$$projection(y) = \Sigma x[255 - I(x, y)]$$

where I(x, y) is the pixel intensity value at coordinate (x, y). The projection result produces a graph showing the intensity distribution along the vertical axis. Peaks in this graph indicate rows with high-intensity concentration, which correspond to the positions of the staff lines. To detect these peaks, the find_peaks function from the SciPy library is used, with parameters for minimum peak distance and prominence adjusted accordingly.

The vertical center y_c of each detected notepad bounding box is normalized relative to the bottom staff line position y_b using the following formula:

$$n = \frac{y_b - y_c}{d_s}$$

where n represents the relative vertical position in staff-step units (lines or spaces). A positive value of n means the notehead lies above the bottom line; negative values indicate positions below the staff. To account for minor detection inaccuracies or staff-line deviations, a tolerance threshold τ of approximately ± 0.3 staff steps is used when determining whether a notehead is positioned "on a line" or "in a space." This spatial tolerance provides robustness while preserving pitch resolution, ensuring that small vertical fluctuations do not cause incorrect pitch assignments.

The vertical positions of these detected staff lines serve as crucial reference points for mapping notehead vertical coordinates to musical pitches in the subsequent rule-based pitch inference. Accurate detection of staff lines is essential because small errors in their positioning can lead to incorrect note pitch assignments, thereby directly compromising the final OMR accuracy.

Rule-Based Pitch Inference

The next stage of this research method is rule-based pitch inference, which is the process of determining the pitch of each notehead based on its spatial position relative to the staff lines. A rule-based approach was chosen because it provides transparent, accurate interpretation aligned with music notation theory without requiring additional model training.

a. Extraction of Structural Information – Before inference is performed, the system utilizes the detected noteheads produced by the YOLOv12 model and the detected staff lines obtained through the horizontal projection profile method. Each detected staff line provides the vertical reference needed to map the y-coordinates of notehead bounding boxes. Musical metadata from the dataset annotations (such as clef type, key signature, and time signature) is also used as additional context, since each clef has a different relationship between staff line positions and pitch names.
b. Determining the Relative Position of Noteheads – The vertical center (y-center) of each YOLOv12-detected bounding box is compared with the positions of the five nearest staff lines. The distance between lines is calculated to obtain the standard spatial interval (e.g., staff spacing in pixels). The relative position of the notehead—whether directly on a line, between two lines, or several steps above or below the staff—is the key factor in determining the pitch. For example, in a treble clef image, the bottom line represents E4,

and each line spacing corresponds to the next pitch step (F4, G4, A4, B4, etc.). A similar set of rules is applied to the bass clef, though with a different pitch pattern (e.g., G2, B2, D3, F3, A3).

c. Applying Pitch Mapping – Based on the computed relative positions, the system applies the following rules to convert coordinates into pitch:
   1. Determine the reference line (the lowest staff line) according to the clef type.
   2. Calculate the number of staff steps (up or down) from the reference line to the notehead's y-coordinate.
   3. Convert the number of steps into a pitch name following the diatonic sequence (A–G), and adjust for the key signature to apply accidentals (♯/♭) when necessary.

   The mapping between distance and note pitch can be described in the following table:

Table 1. Position–Pitch Mapping for Treble Clef

Relative Position to Staff Lines	Vertical Distance (in "staff steps"*)	Note Name (Pitch)
2 steps above the 5th line	+2	C6
1 step above the 5th line	+1	A5
5th line	0	G5
Space between 4th and 5th lines	-1	F5
4th line	-2	E5
Space between 3rd and 4th lines	-3	D5
3rd line	-4	C5
Space between 2nd and 3rd lines	-5	B4
2nd line	-6	A4
Space between 1st and 2nd lines	-7	G4
1st line	-8	E4
1 step below the 1st line	-9	D4
2 steps below the 1st line	-10	C4

Table 2. Position-Pitch Mapping for Bass Clef

Relative Position to Staff Lines	Vertical Distance (in "staff steps"*)	Note Name (Pitch)
2 steps above the 5th line	+2	E4
1 step above the 5th line	+1	D4
5th line	0	C4
Space between 4th and 5th lines	-1	B3
4th line	-2	A3
Space between 3rd and 4th lines	-3	G3
3rd line	-4	F3
Space between 2nd and 3rd lines	-5	E3
2nd line	-6	D3
Space between 1st and 2nd lines	-7	C3
1st line	-8	B2
1 step below the 1st line	-9	A2
2 steps below the 1st line	-10	G2

d. Pitch Accuracy Evaluation – The pitch inference results are compared with expert-annotated ground truth from the validation images. The evaluation uses the Pitch Accuracy (PA) metric, calculated as the ratio of correct pitch predictions to the total predictions. Pitch accuracy is defined per notehead, meaning that each detected notehead is evaluated individually against its ground-truth pitch label. A prediction is counted as correct only when the system both detects the notehead and assigns the correct pitch corresponding to its staff position and clef context. This per-notehead evaluation is consistent with symbol-level OMR measurement practices, and it ensures that polyphonic textures, where multiple

notes may appear simultaneously in a chord, are assessed based on the correctness of each individual note rather than aggregated per beat or harmonic event. This metric assesses the success of the system not only in detecting visual symbols but also in accurately interpreting their musical meaning.

RESULTS AND DISCUSSION

YOLOv12 Model Training Results

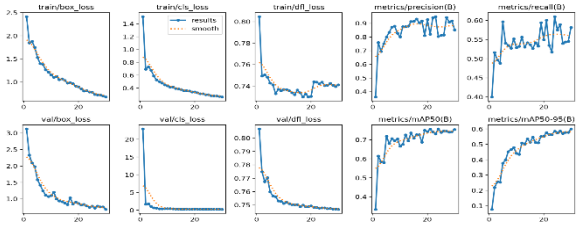


Figure 4. Training Loss Curve, Validation Loss Curve, and Mean Average Precision (Map) During The Training of The Yolov12 Model

In the first 10 epochs, the training loss dropped sharply from an initial value of around 2.5 to 1.0. This decrease indicates that the model quickly learned basic visual patterns such as staff lines and notehead shapes. Between epochs 10 and 20, the loss continued to decrease more slowly until it stabilized around 0.5, reflecting the refinement of the convolutional network parameters. After epoch 30, the loss entered a plateau phase with small fluctuations, indicating that the model had reached a point of convergence.

On the other hand, the mAP50 value (IoU = 0.5) increased from about 0.35 in the early epochs to 0.754 at epoch 30. The mAP50–95 value, which is stricter because it measures average accuracy across various IoU thresholds, also increased steadily, reaching 0.600 at the end of training. The upward trend in mAP, aligned with the decline in loss, shows that the model was able to learn effectively without experiencing overfitting.

Performance evaluation was conducted on a validation set comprising 20% of the images from the DeepScoresV2-Dense subset.

.

Table 3. Evaluation Results of Notehead Detection on the Validation Data

Metric	Result	Interpretation
mAP50	0.754	Detection accuracy is quite high at the IoU 0.5 threshold, indicating the model's ability to capture the target objects.
mAP50-95	0.6	A value of 0.6 indicates that the model maintains stable accuracy even when detecting small, dense, or partially overlapping noteheads.
Precision	0.852	The proportion of correct predictions to total predictions is high, indicating relatively low false positives.
Recall	0.582	The ratio of correctly detected objects is showing a moderate level of model sensitivity; the model is able to capture most noteheads, but some remain undetected, particularly in visually challenging regions.
F1-Score	0.69	A value of 0.69 for the harmonic mean of Precision and Recall, which reflects the balance of the model's overall detection performance, indicates that the model achieves a good trade-off between correctly identifying noteheads and capturing as many ground-truth noteheads as possible.
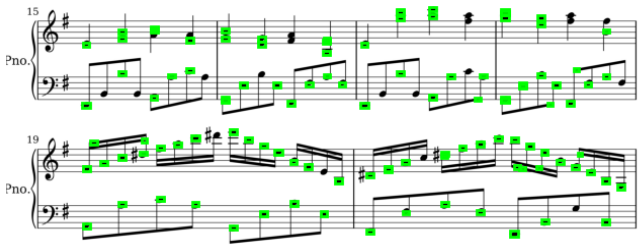


Figure 5. Notehead Detection Results Using YOLOv12

Figure 5 shows an example of the detection visualization on the validation score. The green bounding boxes indicate the noteheads successfully detected by the model. The model is capable of detecting a large portion of noteheads that overlap with stems or barlines, as well as small noteheads located in areas with dense staff lines. This success demonstrates the strength of YOLOv12 in handling small object detection, which has previously been a challenge in OMR research based on YOLOv8 [13].

However, several types of errors were observed, categorized as follows:

a. Missed Detection – Some very small noteheads or those obscured by image noise were not detected.
b. False Positive – Certain graphic elements such as augmentation dots or accidental signs were occasionally detected as noteheads due to similar shapes.

The recall value of 0.58 indicates that a substantial number of noteheads (approximately 42%) remain undetected, particularly in areas with very high symbol density. This imbalance highlights a key trade-off in the model's detection strategy. The low recall is primarily caused by the extreme visual complexity of the DeepScoresV2-Dense dataset, where overlapping noteheads, stems, beams, accidentals, and staff lines create cluttered regions that are difficult for YOLOv12 to separate. Although YOLO architectures excel in small-object detection, distinguishing multiple objects in close proximity remains a challenge.

Another contributing factor is the relatively short training duration. With only 30 epochs and a small batch size of 8, the model may not have fully converged on the nuanced features needed to detect faint, small, or partially occluded noteheads. Complex polyphonic OMR tasks often require longer training cycles for deeper feature representation learning. Dataset imperfections, such as slightly blurred or unevenly rendered noteheads, further reduce recall.

This low detection rate impacts the final Optical Music Recognition (OMR) results primarily by reducing the completeness of the pitch extraction. The incomplete detection leads to missing pitch information for the undetected notes, which lowers the overall musical fidelity and completeness of the OMR output. While the detected noteheads can be mapped to pitch with reasonable accuracy using rule-based pitch inference, the incomplete detection decreases the reliability of the overall score transcription.

Nevertheless, the mAP50 value of above 0.75 indicates that the model still achieves a reasonable level of detection accuracy for supporting the pitch inference stage. The correctly detected noteheads provide sufficiently reliable vertical coordinates for rule-based pitch mapping. With this performance, YOLOv12 shows promise not only for digitally engraved polyphonic scores but also for potential applications to handwritten or older printed music scores with varied symbol densities.

Staff Line Detection Results

After the YOLOv12 training process is completed and the noteheads have successfully been detected, the next step is to detect the staff lines, which serve as an important reference in the rule-based pitch inference process. Staff line detection is performed separately from the

YOLOv12 model by utilizing the horizontal projection profile and a peak detection algorithm. This step aims to obtain the vertical coordinates of each staff line in the score image so that the positions of the noteheads can be accurately mapped to their corresponding musical pitches. Based on experimental results, using parameters *distance = 20 pixels* and *prominence = 300000* produced the most stable detection on images with varying staff line thicknesses.

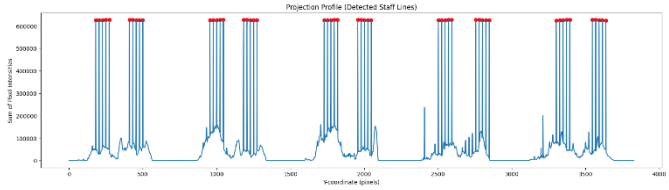


Figure 6. Projection Profile of the detected staff lines

Figure 6 shows an example of the projection profile graph from one of the validation score images. The red points on the graph indicate the projection peaks that were successfully identified as staff lines. The periodically occurring peak patterns represent the five standard lines in each staff system. This visualization not only facilitates visual verification of the detection results but also provides insight into the intensity distribution, which assists in tuning the *distance* and *prominence* parameters. Stable detection is evident from the relatively uniform spacing between peaks, indicating the effectiveness of this method on scores with high symbol density.

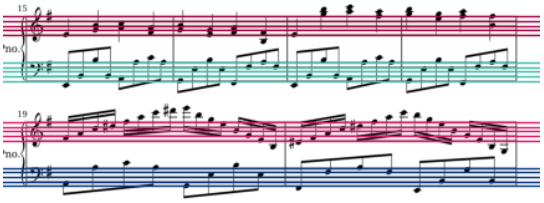


Figure 7. Staff line detection results

Accurate vertical positioning of the staff lines is a key factor in the rule-based pitch inference process. The computed spacing between the lines is used to normalize the notehead center (y-center) coordinates before mapping them to pitch names. Precise line detection ensures that the system can distinguish between notes separated by only a half step (e.g., E4 vs. F4 in the treble clef). With these results, the horizontal projection profile method is proven to be effective for detecting staff lines in polyphonic music scores with high accuracy. Successful line detection serves as a crucial foundation for the next stage, namely mapping the notehead positions into pitch information through rule-based pitch inference.

Rule-based Pitch Inference Implementation Results

The process of converting coordinates into pitch is carried out through the following steps:

a. Vertical Coordinate Normalization – The vertical center (y-center) of each notehead bounding box is normalized based on the average spacing between staff lines. This normalization converts pixel coordinates into staff-step units, allowing direct comparison with musical notation interval patterns.
b. Determining Relative Position – For each notehead, the system calculates its relative distance to a reference line (e.g., the bottom line of the staff). This position determines whether the notehead is located on a line, between two lines, or below/above the staff.
c. Mapping to Pitch Names – Based on the clef type (treble/bass) and key signature obtained from metadata, the system applies interval calculations to convert the relative position into the corresponding pitch name (e.g., E4, G4, A3).

The evaluation was performed on the same DeepScoresV2-Dense validation subset used for testing YOLOv12. For each image, the pitch inference results were validated by a music notation expert. Table 4 presents a summary of the key metrics.

Table 4. Rule-Based Pitch Inference Evaluation Results

Evaluation Metric	Average Value
Pitch Accuracy (PA)	0.87
Precision	0.89
Recall	0.85
F1-Score	0.87

A pitch accuracy score of 0.87 indicates that more than 87% of the detected noteheads were successfully mapped to the correct pitch. The precision value, which is slightly higher than the recall, suggests that the system is more conservative in producing predictions, thereby reducing the number of false positives (incorrect pitch predictions).
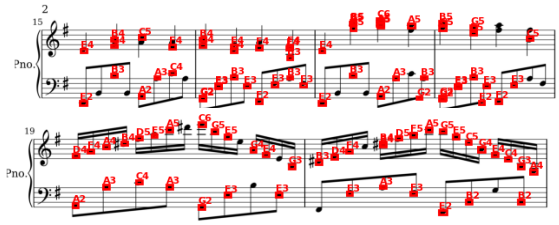


Figure 8. Rule-based Pitch Inference Results

Figure 8 shows an example of the inference results on one of the score images. The detected noteheads are shown with red bounding boxes, while the text above each box represents the predicted pitch (e.g., "E4" or "G5"). This visualization demonstrates the system's ability to:

a. Identify noteheads that overlap with stems,
b. Correctly determine pitch in dense areas with tightly spaced staff lines, and
c. Handle extreme vertical variations, including ledger lines above or below the staff.

Although the overall performance is quite high, several sources of error were still observed, including:

a. Staff Line Detection Errors – Print distortions or image noise cause projection peaks to become less clear, resulting in staff lines being detected with a deviation of 2–3 pixels. Even small deviations like this can shift the pitch by one staff step.
b. Key Signature Mismatch – In scores with frequent key signature changes, the system sometimes fails to adjust notes according to accidentals (♯/♭), leading to incorrect pitch predictions.
c. Incomplete Notehead Detection – If YOLOv12 misses small or highly overlapping noteheads, pitch cannot be generated because the vertical coordinate is unavailable.

Further analysis shows that the majority of errors (approximately 60% of all pitch mistakes) are caused by small deviations in staff line detection. This emphasizes the importance of precise staff line detection as the foundational step for pitch inference.

CONCLUSIONS AND RECOMMENDATIONS

This study successfully developed a deep learning-based Optical Music Recognition (OMR) system capable of identifying noteheads in polyphonic musical scores and transforming their spatial coordinates into pitch data through a rule-based pitch inference approach. Utilizing the YOLOv12 architecture optimized for small object detection, the system achieves accurate and efficient notehead identification in score images with high symbol density. Training on the DeepScoresV2-Dense subset for 30 epochs via transfer learning yielded mAP50 of 0.75, mAP50–95 of 0.6, precision of 0.85, and recall of 0.58, evidencing the model's capability to reliably identify target objects despite symbol overlaps that challenge YOLOv12 detection.

The relatively modest recall value suggests approximately 42% of noteheads remain unidentified, primarily attributed to extreme symbol density, overlapping symbols in polyphonic scores, and constraints in model training duration. Nevertheless, the rule-based

pitch inference implementation successfully transformed identified noteheads into pitch data, achieving a pitch accuracy of 0.87 and an F1-score of 0.87, indicating balanced precision and completeness for the detected note subset. However, missed noteheads considerably affect overall completeness and musical fidelity of OMR outputs, especially in densely notated sections where multiple notes overlap or obscure one another.

REFERENCES

[1] P. Yu and H. Chen, "Deep Multilevel Cascade Residual Recurrent Framework (MCRR) For Sheet Music Recognition," *IEEE Access*, vol. 12, pp. 6941–6960, 2024, doi: https://doi.org/10.1109/access.2024.3350880

[2] J. Calvo-Zaragoza, J. Hajic, and A. Pacha, "Understanding Optical Music Recognition," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, 2020, doi: https://doi.org/10.1145/3397499

[3] A. Ríos-Vila, D. Rizo, J. M. Iñesta, and J. Calvo-Zaragoza, "End-To-End Optical Music Recognition For Pianoform Sheet Music," *International Journal on Document Analysis and Recognition*, vol. 26, no. 3, pp. 347–362, 2023, doi: https://doi.org/10.1007/s10032-023-00432-z

[4] F. Simonetta, R. Mondal, L. A. Ludovico, and S. Ntalampiras, "Optical Music Recognition In Manuscripts From The Ricordi Archive," *AM '24: Proceedings of the 19th International Audio Mostly Conference: Explorations in Sonic Cultures*, pp. 260–269, 2024, doi: https://doi.org/10.1145/3678299.3678324

[5] E. Shatri and G. Fazekas, "Optical Music Recognition: State Of The Art And Major Challenges," 2020, doi: https://doi.org/10.48550/arXiv.2006.07885

[6] S. N. Budiman, S. Lestanti, H. Yuana, and B. N. Awwalin, "SIBI (Sistem Bahasa Isyarat Indonesia) Berbasis Machine Learning Dan Computer Vision Untuk Membantu Komunikasi Tuna Rungu Dan Tuna Wicara," *Jurnal Teknologi dan Manajemen Informatika*, vol. 9, no. 2, pp. 119–128, 2023, doi: https://doi.org/10.26905/jtmi.v9i2.10993

[7] R. Nahak *et al.*, "Klasifikasi Jenis Rumah Adat Malaka Menggunakan Metode Convulational Neural Network (CNN)," *Jurnal Teknologi dan Manajemen Informatika*, vol. 9, no. 2, pp. 91–98, 2023, doi: https://doi.org/10.26905/jtmi.v9i2.10352

[8] Y. Li, H. Liu, Q. Jin, M. Cai, and P. Li, "Tromr:Transformer-Based Polyphonic Optical Music Recognition," *ICASSP, IEEE International Conference On Acoustics, Speech And Signal Processing*, 2023, doi: https://doi.org/10.1109/icassp49357.2023.10096055

[9] E. H. Kheng, C. P. Liew, T. Lan, and K. G. Tan, "Advancing Handwritten Musical Notation Recognition Using Deep Learning: A Convolutional Neural Network-Based Approach With Improved Accuracy," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 38, no. 3, 2024, doi: https://doi.org/10.1142/s0218001424520074

[10] R. H. Rafliansyah, B. Rahmat, and C. A. Putra, "Klasifikasi Suara Instrumen Musik Tiup Menggunakan Metode Convolutional Neural Network," *Merkurius : Jurnal Riset Sistem Informasi dan Teknik Informatika*, vol. 2, no. 4, pp. 01–09, 2024, doi: https://doi.org/10.61132/merkurius.v2i4.119

[11] N. D. Hendrawan and R. Kolandaisamy, "A Comparative Study Of Yolov8 And YOLO-NAS Performance In Human Detection Image Article Info ABSTRACT," *Jurnal Teknologi dan Manajemen Informatika*, vol. 9, no. 2, pp. 191–201, 2023, doi: https://doi.org/10.26905/jtmi.v9i2.12192

[12] C. R. Wairata, E. R. Swedia, and M. Cahyanti, "Pengklasifikasian Genre Musik Indonesia Menggunakan Convolutional Neural Network," *Sebatik*, vol. 25, no. 1, pp. 255–261, 2021, doi: https://doi.org/10.46984/sebatik.v25i1.1286

[13] G. H. Romão, H. S. Lara, and J. N. Brito, "Testing Yolov8's Efficacy As A Pitch And Duration Detector Across Digitally Written Monophonic Music Scores," *OBSERVATÓRIO DE LA ECONOMÍA LATINOAMERICANA*, vol. 22, no. 9, pp. 01-16, 2024, doi: https://doi.org/10.55905/oelv22n9-133

[14] R. Sapkota, D. Ahmed, and M. Karkee, "Comparing Yolov8 And Mask R-CNN For Instance Segmentation In Complex Orchard Environments," *Artificial Intelligence in Agriculture*, vol. 13, pp. 84–99, 2024, doi: https://doi.org/10.1016/j.aiia.2024.07.001

[15] J. Ma, Y. Zhou, Z. Zhou, Y. Zhang, and L. He, "Toward Smart Ocean Monitoring: Real-Time Detection Of Marine Litter Using Yolov12 In Support Of Pollution Mitigation," *Marine Pollution Bulletin*, vol. 217, 2025, doi: https://doi.org/10.1016/j.marpolbul.2025.118136

[16] L. Tuggener, Y. P. Satyawan, A. Pacha, J. Schmidhuber, and T. Stadelmann, "The Deepscoresv2 Dataset And Benchmark For Music Object Detection," *in 2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 9188–9195, 2021, doi: https://doi.org/10.1109/icpr48806.2021.9412290

[17] E. Tanuwijaya and C. Fatichah, "Penandaan Otomatis Tempat Parkir Menggunakan YOLO Untuk Mendeteksi Ketersediaan Tempat Parkir Mobil Pada Video CCTV," *BRILIANT: Jurnal Riset Dan Konseptual*, vol. 5, no. 1, pp. 189-198, 2020, doi: https://doi.org/10.28926/briliant.v5i1.434

[18] G. Chen, L. Zhang, W. Zhang, and Q. Wang, "Detecting The Staff-Lines Of Musical Score With Hough Transform And Mathematical Morphology," *2010 International Conference On Multimedia Technology*, 2010, doi: https://doi.org/10.1109/ICMULT.2010.5631269

[19] F. F. De Vega, J. Alvarado, and J. V. Cortez, "Optical Music Recognition And Deep Learning: An Application To 4-Part Harmony," *2022 IEEE Congress On Evolutionary Computation (CEC) 2022*, 2022, doi: https://doi.org/10.1109/CEC55065.2022.9870357

[20] X. Yin, Z. Zhao, and L. Weng, "MAS-YOLO: A Lightweight Detection Algorithm For PCB Defect Detection Based On Improved Yolov12," *Applied Sciences (Switzerland)*, vol. 15, no. 11, 2025, doi: https://doi.org/10.3390/app15116238

ACKNOWLEDGEMENT

Funded by:
Ciputra University Surabaya as part of the Implementation of the Internal Research Grant Program under the DIP Student Scheme
Fiscal Year 2025/2026
between the LPPM of Ciputra University Surabaya and the Researcher
Number 006/UC-LPPM/DIP-M/KP3/IX/2025, 19 September 2025