LEARNING NOTATION GRAPH CONSTRUCTION FOR FULL-PIPELINE OPTICAL MUSIC RECOGNITION

Alexander Pacha
Institute of Information Systems Engineering, TU Wien, Austria
alexander.pacha@tuwien.ac.at

Jorge Calvo-Zaragoza
Pattern Recognition and Artificial Intelligence Group
University of Alicante, Spain
jcalvo@dlsi.ua.es

Jan Hajič jr.
Institute of Formal and Applied Linguistics,
Charles University, Prague
hajicj@ufal.mff.cuni.cz

ABSTRACT

Optical Music Recognition (OMR) promises great benefits to Music Information Retrieval by reducing the costs of making sheet music available in a symbolic format. Recent advances in deep learning have turned typical OMR obstacles into clearly solvable problems, especially the stages that visually process the input image, such as staff line removal or detection of music-notation objects. However, merely detecting objects is not enough for retrieving the actual content, as music notation is a configurational writing system where the semantic of a primitive is defined by its relationship to other primitives. Thus, OMR systems must employ a notation assembly stage to infer such relationships among the detected objects. So far, this stage has been addressed by devising a set of predefined rules or grammars, which hardly generalize well. In this work, we formulate the notation assembly stage from a set of detected primitives as a machine learning problem. Our notation assembly is modeled as a graph that stores syntactic relationships among primitives, which allows us to capture the configuration of symbols in a music-notation document. Our results over the handwritten sheet music corpus MUSCIMA++ show 95.2% precision, 96.0% recall, and an F-score of 95.6% in establishing the correct syntactic relationships. When inferring relationships on data from a music object detector, the model achieves 93.2% precision, 91.5% recall and an F-score of 92.3%.

1. INTRODUCTION

Optical Music Recognition is the field of research that investigates how to read music notation in documents computationally. This technology enables many computational tasks that, otherwise, could not be performed directly on the music sources themselves [17]. One interesting application of OMR is concerned with reconstructing the notes encoded in the music-notation document, also referred to as *replayability* [22]. In particular, the objective of the replayability application is to recover the pitches, onsets, durations, and velocities of notes from a document and export them into a symbolic representation. This symbolic representation—e.g., a MIDI file—is already a very useful abstraction of the music itself and allows for plugging in a wide range of music information retrieval tools. However, despite prolonged efforts, the replayability application is still under research [4, 7, 16, 36].

Given the wealth of information that is contained in a music score, the task of decoding its content is usually addressed by dividing the process into smaller stages that represent limited challenges. The general pipeline, proposed first by Bainbridge and Bell [3] and later refined by Rebelo et al. [29], is considered a de-facto standard, which organizes the process into four main blocks: i) preprocessing, which works over the input image to ease further steps and make the system more robust; ii) music object detection, which is in charge of retrieving and classifying all objects and glyphs of the image; iii) notation assembly, which must infer the relationships among the detected objects to reconstruct the music notation itself; and iv) encoding, which exports the symbolic reconstruction into the desired format, typically MIDI for replayability or an XML-based encoding such as MusicXML [15] or MEI [19] for further computational processing.

As our starting point towards completing the OMR pipeline, we assume that the music object detection stage can be solved reliably, which allows us to investigate how to deal with the later stages. In this paper, we want to focus in particular on the third stage, which is responsible for the notation assembly. Although previous work exists, most approaches are based on predefined rules that hardly generalize, and that only work for a limited set of scenarios. In contrast, we propose a well-principled machine learning approach, which addresses the problem in a generalizable way, provided there is convenient training data.

2. RELATED WORK

Most literature on OMR focuses on the first stages of the pipeline. This comes as no surprise because if one struggles with detecting music objects in an image reliably, it is understandable that subsequent stages that build on top of that are often neglected. With the appearance of deep learning in OMR, however, many steps that traditionally produced suboptimal results, such as the staff-line removal or symbol classification, have seen drastic improvements [14, 26] and are no longer considered obstacles for OMR development.

Deep learning also caused some steps to become obsolete or collapse into a single (bigger) stage. For instance, the music object detection stage, which was traditionally separated into segmentation plus classification stages, is currently addressed in a single step. Convolutional neural networks have been shown to be able to deal with the music object detection stage holistically, without having to remove staff lines at all [25]. A compelling advantage is the capability of these models to be trained in a single step by merely providing pairs of images and positions of the music objects to be found, eliminating the preprocessing step altogether [24, 35]. This issue has been the subject of intense recent research. A comparison of existing approaches to holistic music object detection is presented in the work of Pacha et al. [27].

Since the beginning of the OMR research, there have been attempts to complete the full pipeline, including the notation assembly stage. Below, we introduce some particular proposals to perform this stage that can be found in the existing literature. They can be broadly divided into grammar-based approaches, and approaches that rely on heuristics and pre-defined rules.

2.1 Grammar-based approaches

Formal grammars represent the most widely used description of music notation. This feels natural, given that music notation has syntactic rules and hierarchical structures that invite such a formalization. These grammars are manually built to describe the expected relationships among music-notation objects and then used to reconstruct the music notation from the detected primitives [1–3, 5, 6, 30, 33]. The 2D nature of music notation also inspired graph grammars, as in the work of Fahmy and Blostein [12]. A prominent example of this approach is the DMOS system, proposed by Coüasnon et al. [8,9], which uses a definite clause grammar for describing the relations between graphical objects on two levels: a graphical one that assists the recognition of symbols and a syntactic one, which introduces the musical semantics into the process.

2.2 Heuristical approaches

The other set of approaches relies on *ad hoc* rules for the music notation at hand. This includes assumptions about the configuration and position of the occurring primitives to reconstruct composite symbols and the notation graph [10, 23, 28, 34]. Rossant et al. [31] additionally considered fuzzy modeling, which allows for self-correction during the recognition [32]. Their system evaluated different hypotheses of recognized symbols to verify the compatibility between them.

3. NOTATION ASSEMBLY

The related works clearly show a lack of machine learning approaches. This work aims to fill that gap, by proposing a formulation of the notation assembly stage based on machine learning models. The advantage of such models is that they provide greater flexibility since they can vary their behavior by just changing the provided training set. This is especially interesting for OMR, where there is a great diversity of scenarios depending on the epoch or type of composition of the music scores.

The conventional OMR pipeline foresees that the notation assembly stage infers the relationships among previously detected music objects to retrieve the necessary information to infer the sequence of notes and rests.

Our approach understands that music notation can be modeled as a directed graph $G = (V, T)$, hereafter referred to as Music Notation Graph (MuNG). V represents the set of vertices, where $\zeta(v), v \in V$ is the label associated with a vertex. T represents the set of directed edges, such that $t_i = (v_1, v_2), t_i \in T, v_1, v_2 \in V$ denotes an edge from vertex v_1 to vertex v_2. The primitives that make up the music notation, such as noteheads or stems, are modeled as vertices of this graph, while the relationships between these symbols are modeled by the edges. In our MuNG, the edges are not labeled, but there are two types of relationships:

- Syntactic edges that relate elements syntactically. This includes relationships between primitives that make up a composite symbol, such as an eighth note, which consist of a notehead, a stem, and a flag or beam as well as general relationships, e.g., between an accidental and the notehead that is affected by it.
- Precedence edges that specify the temporal order between notes. In most cases, the position on the horizontal axis is sufficient to infer this kind of relationship; however, for polyphonic music, a more sophisticated mechanism is needed to handle ambiguous situations.

We can, therefore, define the set of edges as $T = S \cup P$, where S is the set of edges that define the syntactic relationships and P is the set of edges that define the precedence relationships. A graphical representation of MuNG is shown in Fig. 1. The primary goal of our work is to train a machine learning model to construct such a MuNG G from a music score image.

4. LEARNING MUSIC NOTATION GRAPH ASSEMBLY

There are existing algorithms that are capable of dealing with the input image and retrieving a set of detected music-notation primitives. In other words, these algorithms process the input and provide the set of vertices V, along with its associated labels and bounding-boxes. In order to complete the OMR pipeline for replayability, we also need to recover the set of edges T.

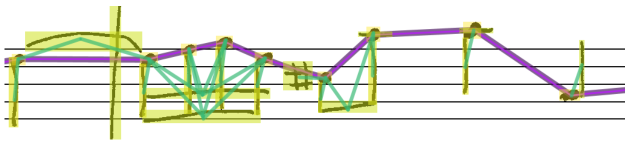


Figure 1: Graphical representation of a Music Notation Graph in a selected excerpt of music notation: vertices are highlighted with transparent yellow bounding boxes around the music-notation primitives, syntactic edges are shown as transparent cyan lines, and precedence edges are shown as transparent purple lines connecting the noteheads.

We propose a principled way of inferring T without resorting to a set of fixed rules but using machine learning. Our system learns to establish these relationships from a conveniently annotated training set so that the rules are implicitly modeled by the machine learning model.

The edges that relate vertices of the set T have an unlabeled binary nature; i.e. for each pair of vertices, a relationship either exists or not. Formally speaking, the inference of these relationships can be formulated as a function $f : V \times V \rightarrow \{0, 1\}$. However, given their different nature, the set of edges S and P are inferred by independent models. To learn the functions f_S and f_P, for the edges of S and P, respectively, we propose to train binary classifiers that receive two vertices and predict whether such relationship must be established or not. To do so, one would have to estimate the potential relationship between each pair of symbols, which entails high computational costs. However, it is obvious that most of these relationships are unfeasible. Since the music object detection stage also retrieves some associated information, such as the label $\zeta(v)$ associated to each vertex and the bounding box of that object in the input score image, we can use this information to filter edges by two criteria:

1. An edge is only feasible if the distance between the bounding boxes of their vertices falls below a certain threshold t. In other words, two vertices that are too far apart cannot be related.

2. An edge is only feasible if the labels of its associated vertices are "compatible", e.g., a notehead with a stem. This eliminates a large number of incompatible combinations, such as an edge between an accidental and a rest. The compatibility map is a fixed list of vertex pairs that, according to the syntax of modern music notation, can hold a relationship to each other.

Then, given two vertices v_1 and v_2, for which their edge is declared feasible, we train a deep convolutional neural network to predict whether there must be an edge from v_1 to v_2 or not. We generate a multi-channel image with a fixed size that serves as input features for the neural network, which consists of:

- Channel 1: the patch of the input score image that is centered at the objects represented by v_1 and v_2.
- Channel 2: the binary mask of the object v_1
- Channel 3: the binary mask of the object v_2

The required information to generate these multi-channel images can be obtained from the bounding boxes of v_1 and v_2, which are expected to be generated during the preceding music object detection stage. Note, that the masks for channel 2 and 3 are obtained from the bounding boxes and the underlying image, which means that the masks can (partially) include other objects as well unless the exact masks are provided via pixelwise segmentation [16, 35].

The network is then fed with this 3-channel image and trained to predict 1 if there should be a relationship between the vertices, and 0 otherwise. Visualizations of the input images are given in Fig. 2.

4.1 Dataset

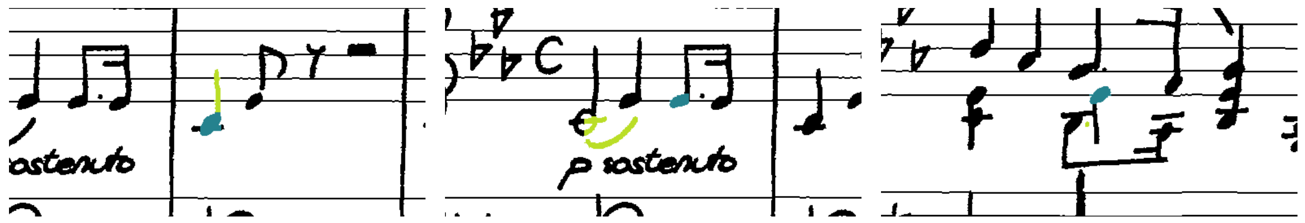
To carry out our experiments we need a corpus, which has annotations for both the individual symbols as well as their relationships. Currently, the only publicly available dataset which fulfills this requirement is the MUSCIMA++ dataset [18] of handwritten music notation. It provides symbol-level annotations as well as relationship annotations for 140 out of 1 000 images from the CVC-MUSCIMA dataset [13]. The MUSCIMA++ dataset contains 91 254 annotated symbols, consisting of both notation primitives and higher-level notation objects, such as key signatures or time signatures as well as 82 247 explicitly marked relationships between symbol pairs.

Unfortunately, the precedence relationships between notes are not included in the MUSCIMA++ dataset, so our experiments consider only the syntactic edges. However, the formulation and the proposed approach are very similar and should work for both kinds of edges.

4.2 Relationship Reconstruction

For learning the relationships, we train a convolutional neural network in PyTorch with five consecutive blocks, each consisting of a convolution, batch normalization, a non-linearity (ReLU), and max-pooling, before going into a fully connected layer with a single output neuron followed by a sigmoid activation function that produces the final estimation. The network has 28 865 parameters in total. We use the Binary Cross-Entropy loss and train with the Adam optimizer [20] until the validation performance has not improved for ten epochs, upon which we stop.

The data-loading routine presents the biggest challenge because it has to construct the multi-channel images as described in Sect. 4. To efficiently generate the set of vertex-pairs, we compute the pairwise distance between all objects in an image but filter them considerably afterward by the distance and compatibility criteria (see Sec. 4). The distance threshold was set to $t = 200$ pixels for including most valid edges from the MUSCIMA++ dataset. Valid relationships between objects that are further apart than 200 pixels are extremely rare and were neglected in favor of



(a) A positive example of two objects that are related.

(b) A negative example of two objects that are unrelated.

(c) A hard negative example of a dot that could be related to the notehead, but is not.

Figure 2: Three samples of images that are used during training. The mask given in channel 2 is shown as bright green overlay and the mask from channel 3 as cyan overlay.

computational efficiency. Our compatibility map contains 225 valid combinations of primitives. To improve the performance even further and simplify the classification task, the input image for the neural network is cropped to a sub-image of 512 $\times$ 256 pixels (width $\times$ height), containing the two objects of interest at its center. Both the distance threshold and the sub-image dimensions are hyperparameters that are dataset-dependent but can be obtained by running a statistical analysis on the used dataset.

We split the 140 images of the dataset into 60 % training data, 20 % validation data, and 20% test data. In each epoch, the network is trained on approximately 250 000 images of candidate pairs. Approximately 25 percent of the candidates contain positive examples. The best results were obtained after just 12 epochs before the network started to overfit and the validation performance declined. The source-code is publicly available on Github. [1]

4.3 Music Object Detection

Since the notation assembly stage begins after the music objects have been detected in the score image, we also wanted to evaluate, how well our approach works on actual detection results. For obtaining such results, we resort to a state-of-the-art music object detector as proposed by Pacha et al. [25] with a minor modification: While we do divide the full page into sub-images containing one stave each, we do not see the need for cutting the images any further. The model selection and training procedure remains unchanged. We split the dataset into 100 images for training, 20 images for validation and 20 images for testing, as proposed by the authors of the MUSCIMA++ dataset. The improved implementation is publicly available. [2]

We evaluate the trained model on the test set for the stave-wise individual images and report the Mean Average Precision (mAP) as defined for the COCO challenge [21] which is a unified metric, commonly used for object detection tasks. The trained model achieves 69.5 % mAP. For comparison, we also report a mAP of 93.3 % when using the mAP as defined for the PASCAL VOC challenge [11], which was used in the original paper. Finally, the images are merged into the full-page results upon we achieve: 34.5 % mAP / 45.2 % w-mAP [3] (COCO) and 53.8 % mAP / 80.9 % w-mAP (PASCAL). As our main focus is on learning relationships and not music object detection, we do not go into further details on these numbers. However, we want to point out that the COCO metric is very strict and probably underestimating the performance of the music object detector (see Fig. 3 for an example output).

4.4 Evaluation Protocol

Once the music objects have been detected, and their relationships established, the system can produce a complete MuNG that can be compared with the reference MuNG, provided as ground truth. However, it is necessary to first establish the correspondences between vertices from the prediction and the ground-truth. To do so, we assume that a detected object v_1 corresponds to a ground-truth object v_2 if they depict the same class $\zeta(v_1) = \zeta(v_2)$ and their Intersection over Union exceeds 50 %.

Once the vertices of the ground-truth are matched with the detected objects, it is possible to compute the statistics. If an established relationship is correct, it is considered a true positive (TP); if an established relationship is incorrect, it is considered a false positive (FP); and, if an expected relationship is not predicted, it is considered a false negative (FN). Then, we can compute precision (P), recall (R), and F-score (F_1) metrics:

$$P = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \quad F_1 = 2\frac{P \times R}{P+R}$$

P measures how reliable the established relationships are, whereas R measures the ability of the model to retrieve as many relationships as possible. F_1 summarizes both metrics with a single value.

Note that, although our evaluation is primarily focused on the relationships between objects, the used metrics are affected by the performance of the music object detector. Errors from earlier stages of the OMR process propagate to later stages. So if musical objects were missed, their relationships are counted as false negatives. To account for this, we evaluate our model in two ways:

[1] `https://github.com/OMR-Research/MungLinker`

[2] `https://github.com/apacha/MusicObjectDetector-TF`

[3] Weighted Mean Average Precision is the Mean Average Precision, weighted by the frequency of the occurring classes, which is higher because frequent classes yielded better results than rare ones.

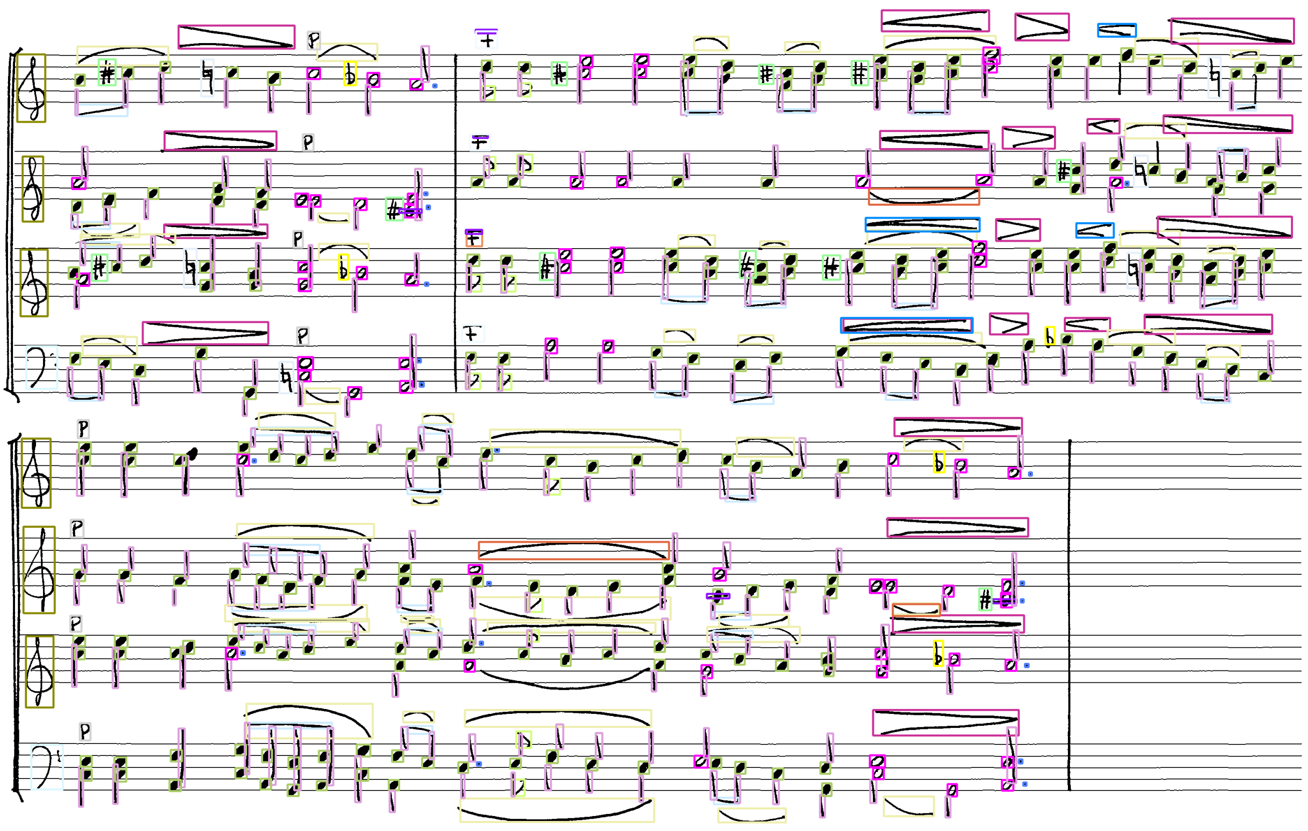


Figure 3: Sample output of the improved music object detector. Each detected object v has a box around it, with the color representing the class $\zeta(v)$ of the particular object, e.g., light green for full-noteheads.

- over a hypothetical set of perfect detections, which we can extract from the ground-truth of the corpus, and
- over the result of an actual music object detector, specifically using the state-of-the-art model, described in Sect. 4.3.

These settings allow us to answer the two questions: Does the proposed approach for reconstructing the MuNG with a machine learning model work at all? If yes, how well does the system perform in a real-world scenario, when confronted with (imperfect) object detector results instead of the perfect ground-truth bounding boxes?

4.5 Results

The main objective of our work is to demonstrate that the notation assembly stage can be formulated as a machine learning task. The main results of our experiments are given in Table 1. It can be observed that the proposed approach is highly effective: in all cases, values above 90 % are reported.

When starting from ground-truth music object detection, our model yields $P = 95.2\,\%$, $R = 96.0\,\%$, and $F_1 = 95.2\,\%$, which indicates a successful approach to completing the OMR pipeline. In case of starting from actual results of a state-of-the-art detector, performance decreases slightly to $P = 93.2\,\%$, $R = 91.5\,\%$, and $F_1 = 92.3\,\%$. We think this is because the location of the objects is not always exact (leading to a lower P) and missing symbols cause relationships to be irrecoverable (leading to a lower R).

	Graph Edges / Relationships		
	Precision	**Recall**	**F-Score**
Perfect Detection	95.2%	96.0%	95.6%
Real Detector	93.2%	91.5%	92.3%

Table 1: Overall performance of the proposed machine learning model to reconstruct syntactic edges of the Music Notation Graph (MuNG), given hypothetically perfect detection results (top row), and given results from a state-of-the-art detector (bottom row).

In order to provide more experimental insights, Table 2 reports 10 out of the 225 compatible combinations of relationships that are most common in the MUSCIMA++ dataset. As might be expected, the *notehead* primitives are involved in all of these frequent combinations. In this regard, our model obtains nearly optimal results for these over-represented cases. Note that these relationships are of particular relevance to be able to decode the notes that appear in the score. When comparing the individual results to the overall results in Table 1, the discrepancy can be explained by looking at the remaining 215 combinations that are not shown. Many of these have a much lower F_1, probably because they are under-represented in the dataset.

From	To	Number of candidate pairs in the dataset	F-Score on the test set
notehead-full	stem	158064	99.5%
notehead-full	beam	61253	98.7%
notehead-full	leger_line	47503	98.1%
notehead-full	slur	24738	96.4%
notehead-full	8th_flag	12877	97.7%
notehead-full	sharp	12563	97.5%
notehead-full	duration-dot	12305	96.7%
notehead-empty	stem	9488	100.0%
notehead-full	staccato-dot	8628	96.8%
notehead-full	natural	7160	98.7%

Table 2: Overview of the ten most common combinations of object-pairs, along with the number of generated candidate pairs in the dataset, as seen by the network. The last column contains the F-scores that were reported for the individual combinations when evaluating the trained model on the test set, containing the ground truth of music primitives v.

5. CONCLUSION AND OUTLOOK

In this work, we study how to complete the OMR pipeline from the previous efforts to detect the music objects within the input image. Our approach seeks the construction of a music notation graph that stores the information of the notation primitives as well as their syntactic and precedence relationships. We propose a machine learning model that can predict whether two primitives are related to each other or not.

Results over the set of syntactic relationships from the handwritten sheet music dataset MUSCIMA++ show that our approach is very effective. We obtain success rates close to the optimum when establishing the correct relationships from the ground-truth primitives ($F_1 = 95.6\%$). When re-evaluating the results starting from the primitives detected by a state-of-the-art music object detector, a slightly lower performance can be observed ($F_1 = 92.3\%$). These figures indicate that the notation assembly stage of the OMR pipeline can be solved reliably with a machine learning model, given a curated set of annotated scores. Comparing our approach to existing methods is extremely difficult, if not impossible, because:

- most existing solutions are black boxes with closed source-code, or there is no available implementation at all,
- only a few systems are capable of handling handwritten modern notation, and
- it is unclear how to compare the music notation assembly stage between two different systems, especially given that the notation graph is only an intermediate representation.

So, although the results are promising, we still see many interesting avenues for further research. For instance, by adding data augmentation during training to make the notation assembly model more robust against variations in the bounding box retrieval of the first stage. Also, we plan to look into providing other meaningful features to the network, such as the class labels $\zeta(v)$ of the involved primitives $v \in V$. Furthermore, we observed that the fixed-sized input patch given to the network is often covering a much larger area than required to contain the objects of interest, especially when they are very close (see Fig. 2c). This could be handled by using size-independent neural network layers such as *Global Pooling*, instead of flattening the features and feeding them into a fully-connected layer, allowing us to adjust the input patch for each sample.

We also believe that the notation assembly stage could benefit from having a broader set of hypotheses about the objects detected in the previous stage, instead of a fixed set of proposals. State-of-the-art music object detectors are based on statistical neural models that are able to provide a probability distribution over the whole set of possible detection hypotheses. When it comes to recognizing, we are typically interested in the most likely hypothesis—the one that is proposed as an answer—forgetting the other ones. However, it is certainly interesting to exploit this statistical modeling: the notation assembly algorithm could establish relationships that are more logical a priori, although the objects involved have a lower probability according to the object detector. These types of approaches have yet to be explored in the field of OMR.

And finally, for completing the OMR pipeline, the encoding stage is still missing. However, we see two benefits of the notation graph representation: the encoding can be implemented by experts in music encoding that are proficient in a particular format and given a complete graph representation, there is no restriction on the actual output format because the graph contains all the information that is present in the image.

6. REFERENCES

[1] Alfio Andronico and Alberto Ciampa. On automatic pattern recognition and acquisition of printed music. In *International Computer Music Conference*, Venice, Italy, 1982.

[2] David Bainbridge. *Extensible optical music recognition*. PhD thesis, University of Canterbury, 1997.

[3] David Bainbridge and Tim Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121, 2001.

[4] Arnau Baró, Pau Riba, Jorge Calvo-Zaragoza, and Alicia Fornés. From optical music recognition to handwritten music recognition: A baseline. *Pattern Recognition Letters*, 123:1–8, 2019.

[5] Pierfrancesco Bellini, Ivan Bruno, and Paolo Nesi. Optical music recognition: Architecture and algorithms. In *Interactive Multimedia Music Technologies*, pages 80–110. IGI Global, Hershey, PA, USA, 2008.

[6] Dorothea Blostein and Henry S. Baird. A critical survey of music image analysis. In *Structured Document Image Analysis*, pages 405–434. Springer Berlin Heidelberg, 1992.

[7] Jorge Calvo-Zaragoza and David Rizo. End-to-end neural optical music recognition of monophonic scores. *Applied Sciences*, 8(4), 2018.

[8] Bertrand Coüasnon, Pascal Brisset, and Igor Stéphan. Using logic programming languages for optical music recognition. In *3rd International Conference on the Practical Application of Prolog*, 1995.

[9] Bertrand Coüasnon and Jean Camillerapp. A way to separate knowledge from program in structured document analysis: Application to optical music recognition. In *3rd International Conference on Document Analysis and Recognition*, pages 1092–1097, 1995.

[10] Michael Droettboom, Ichiro Fujinaga, and Karl MacMillan. Optical music interpretation. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 378–387, Berlin, Heidelberg, 2002.

[11] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015.

[12] Hoda M. Fahmy and Dorothea Blostein. A graph grammar programming style for recognition of music notation. *Machine Vision and Applications*, 6(2):83–99, 1993.

[13] Alicia Fornés, Anjan Dutta, Albert Gordo, and Josep Lladós. CVC-MUSCIMA: A ground-truth of handwritten music score images for writer identification and staff removal. *International Journal on Document Analysis and Recognition*, 15(3):243–251, 2012.

[14] Antonio-Javier Gallego and Jorge Calvo-Zaragoza. Staff-line removal with selectional auto-encoders. *Expert Systems with Applications*, 89:138–148, 2017.

[15] Michael Good. MusicXML: An internet-friendly format for sheet music. Technical report, Recordare LLC, 2001.

[16] Jan Hajič jr., Matthias Dorfer, Gerhard Widmer, and Pavel Pecina. Towards full-pipeline handwritten OMR with musical symbol detection by u-nets. In *19th International Society for Music Information Retrieval Conference*, pages 225–232, Paris, France, 2018.

[17] Jan Hajič jr., Marta Kolárová, Alexander Pacha, and Jorge Calvo-Zaragoza. How current optical music recognition systems are becoming useful for digital libraries. In *5th International Conference on Digital Libraries for Musicology*, pages 57–61, Paris, France, 2018.

[18] Jan Hajič jr. and Pavel Pecina. The MUSCIMA++ dataset for handwritten optical music recognition. In *14th International Conference on Document Analysis and Recognition*, pages 39–46, Kyoto, Japan, 2017.

[19] Andrew Hankinson, Perry Roland, and Ichiro Fujinaga. The music encoding initiative as a document-encoding framework. In *12th International Society for Music Information Retrieval Conference*, pages 293–298, 2011.

[20] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *Computing Research Repository*, abs/1412.6980, 2014.

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014.

[22] Hidetoshi Miyao and Robert Martin Haralick. Format of ground truth data used in the evaluation of the results of an optical music recognition system. In *4th International Workshop on Document Analysis Systems*, pages 497–506, Brasil, 2000.

[23] Kia Ng. Music manuscript tracing. *Lecture Notes in Computer Science*, 2390:322–334, 2002.

[24] Alexander Pacha and Jorge Calvo-Zaragoza. Optical music recognition in mensural notation with region-based convolutional neural networks. In *19th International Society for Music Information Retrieval Conference*, pages 240–247, Paris, France, 2018.

[25] Alexander Pacha, Kwon-Young Choi, Bertrand Coüasnon, Yann Ricquebourg, Richard Zanibbi, and Horst Eidenberger. Handwritten music object detection: Open issues and baseline results. In *13th International Workshop on Document Analysis Systems*, pages 163–168, 2018.

[26] Alexander Pacha and Horst Eidenberger. Towards a universal music symbol classifier. In *14th International Conference on Document Analysis and Recognition*, pages 35–36, Kyoto, Japan, 2017.

[27] Alexander Pacha, Jan Hajič jr., and Jorge Calvo-Zaragoza. A baseline for general music object detection with deep learning. *Applied Sciences*, 8(9):1488–1508, 2018.

[28] Christopher Raphael and Jingya Wang. New approaches to optical music recognition. In *12th International Society for Music Information Retrieval Conference*, pages 305–310, Miami, Florida, 2011.

[29] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R.S. Marcal, Carlos Guedes, and Jamie dos Santos Cardoso. Optical music recognition: State-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012.

[30] K. Todd Reed and J. R. Parker. Automatic computer recognition of printed music. In *13th International Conference on Pattern Recognition*, pages 803–807, 1996.

[31] Florence Rossant and Isabelle Bloch. A fuzzy model for optical recognition of musical scores. *Fuzzy Sets and Systems*, 141(2):165–201, 2004.

[32] Florence Rossant and Isabelle Bloch. Robust and adaptive OMR system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP Journal on Advances in Signal Processing*, 2007(1):081541, 2006.

[33] Mariusz Szwoch. Guido: A musical score recognition system. In *9th International Conference on Document Analysis and Recognition*, pages 809–813, 2007.

[34] Lorenzo J. Tardón, Simone Sammartino, Isabel Barbancho, Verónica Gómez, and Antonio Oliver. Optical music recognition for scores written in white mensural notation. *EURASIP Journal on Image and Video Processing*, 2009(1):843401, 2009.

[35] Lukas Tuggener, Ismail Elezi, Jürgen Schmidhuber, and Thilo Stadelmann. Deep watershed detector for music object recognition. In *19th International Society for Music Information Retrieval Conference*, pages 271–278, Paris, France, 2018.

[36] Eelco van der Wel and Karen Ullrich. Optical music recognition with convolutional sequence-to-sequence models. In *18th International Society for Music Information Retrieval Conference*, Suzhou, China, 2017.