Music Symbol Detection with Faster R-CNN Using Synthetic Annotations

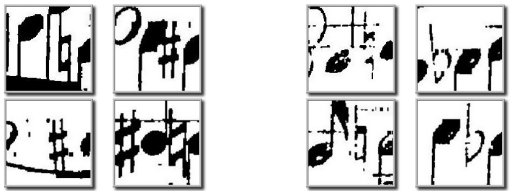
Kwon-Young Choi, Bertrand Coasnon, Yann Ricquebourg
Univ Rennes, CNRS, IRISA, F-35000
Rennes, France
{kwon-young.choi, bertrand.couasnon, yann.ricquebourg}@irisa.fr

Richard Zanibbi
Rochester Institute of Technology
Rochester, USA
rlaz@cs.rit.edu

***Abstract*—Accurately detecting music symbols in images of historical, complex, dense orchestral or piano printed scores can be challenging due to old printing techniques or time degradations. Because segmentation problems can vary widely, a data driven approach like the use of of deep learning detectors is needed. However, the production of detection annotations (symbol bounding boxes + classes) for such systems is costly and time consuming. We propose to train such model with synthetic data and annotations produced by a music program. We analyze which classes are relevant to the detection task and present a first selection of music score typesetting files that will be used for training. To evaluate our model, we plan to compute quantitative results on a synthetic test set and provide qualitative results on a few manually annotated historical music scores.**

***Index Terms*—Optical Music Recognition, Deep Learning, Symbol Detection**

Introduction: Music symbol detection in historical printed orchestral and piano scores is a challenging problems because of the complexity, density and degradations often present in those scores. Segmentation problems caused by lack of space and time degradations can vary widely as shown in Figure 1. Instead of resolving the segmentation task in a manual fashion with expert knowledge, a better solution is to use trainable models and recent in advance in the field of deep learning detectors makes them suitable for this task.



(a) Touching symbols (b) Broken symbols

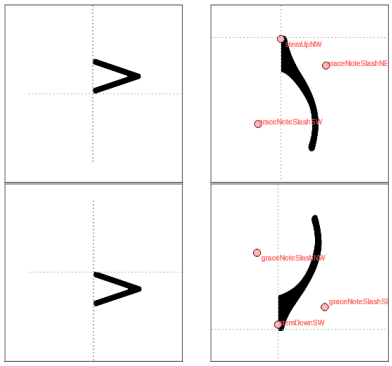
Fig. 1: Common segmentation challenges occurring in real historical piano and orchestral music scores because of engraving or degradation problems.

Previous work [1] has shown that deep learning detectors such as the Faster R-CNN can accurately detect handwritten music symbols. However, the production of annotations for such systems is costly and time consuming. We propose to train such model with synthetic data and annotations produced by a music typesetting software such as MuseScore. Using synthetic data for training machine learning models is a well known subject, as shown by the work of [2] which is able to produce historical synthetic document. However, the use of synthetic data and annotation for the detection of music symbols has still not been applied to the field of Optical Music Recognition. Our end goal is to apply a detector trained only on synthetic data on real historical music scores.

MuseScore Synthetic Data Generation: MuseScore [3] is an open-source music typesetting program that has recently developed a branch for generating data annotations suitable for training classification and detection models used in Optical Music Recognition (OMR) pipeline. Each page of a music score is transformed into a pair of files: an image and an XML files containing a list of symbols present in the image. All symbols are annotated with their class and bounding box in the form of the top-left coordinate and width/height of the symbol. Symbols can be nested, meaning that they are a composite of other symbols or primitives. For example, a nested symbol like the grace note contains elements like a flag, stem, notehead and possible slash, all of which are annotated in the produced XML file with their respective bounding boxes. This opens the possibility of training classifiers and detectors on synthetic data while having lots of flexibility on the class set and composition used. The only limitation of the class set used is imposed by the Standard Music Font Layout (SMuFL) [4] which defines glyphs used in music typesetting software.

Class Set Selection: We first limit the number of classes to the minimal amount possible and group visually similar symbols in the same class. We use these principles to ease the task difficulty for the detector. SMuFL defines all glyphs used to typeset music scores. This standard is diverse and contains around 2600 glyphs. The organization of glyphs are mainly based on their semantic and contextual use. This means that this standard can contains multiple glyphs with the same visual appearance but with only a minor transformation: translation, symmetry or scale as shown in Figure 2.

In music notation, some symbols are built from a number of simple primitives, e.g. flags, rests or dynamics. Using the SMuFL standard, these symbols cannot be decomposed into primitives as they are defined as a single glyphs. Therefore, we choose to use a different class for each different flag, rest and dynamic symbol up to the *flag64th*, *rest64th* and third level of dynamic like *dynamicFFF*. While dynamic symbols cannot be decomposed into their letter primitives, constructs like time



(a) Translation (b) Symmetry

Fig. 2: Visually similar glyphs with different classes in SMuFL. 2a shows symbols *articAccentAbove/Below* which are the same > symbol translated. 2b shows *flag8th* up and down produced with a symmetry.

signature can be decomposed into single digits. We choose to define nine classes corresponding to the different digits from 0 to 9. These digits can be used for the time signature, but also for finger annotation in piano scores.

Very small and complex constructs like grace notes are left for future work as their small size and complexity could be too difficult to handle for a detector with a big input image size.

We also do not consider variable sized symbols like beam, tie, slurs or barline because of their extreme varying size ratio and very simple shapes. Models like line detection models should be more suitable in order to recognize these symbols.

Finally, we propose to leave as future work the definition of meta classes like *timeSig12over8* which annotates a time signature composed of stacked digits 1, 2 and 8 with a global bounding box around the three symbols. Again, this choice is done in order to simplify the task of the detector.

Using this set of guidelines, we select a set of 55 classes covering most commonly used symbols in orchestral and piano scores, see Figure 3 for an overview of the most common music symbols.

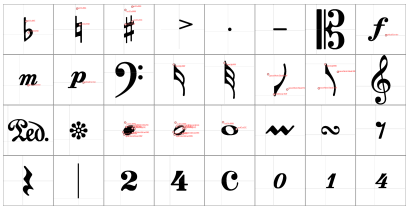


Fig. 3: Common music symbols chosen in the class set composed of accidentals, clefs, ornaments, dynamic signs, tuplets and note heads.

Data Augmentation: In association with the annotation, MuseScore produce an image for each page of a music score. However, these images will be clean, without any kind of noise or deformation. Because our end goal is to use our trained model on real historical scores, we plan to apply common noise and deformation with an open-source software called DocCreator [2]. Noise and deformations can be applied to documents while keeping the spatial annotations like bounding boxes synchronized with the modified document.

Detection Model: For our detection model, we use a regular Faster R-CNN as presented and implemented by [5]. The Faster R-CNN is a two stage detector using a region proposal network (RPN) and a region classifier. The RPN is trained in order to predict possible regions containing an object. The region classifier predicts the class of the object contained in a proposed region while also refining its bounding box. The transition between the RPN and region classifier is done by using an ROI pooling operation which is able to crop a sub-region of the RPN output feature map using the bounding boxes produced by the RPN.

In order to feed the music score to the network, we will first reuse the same strategy as [1] and crop the music score along the stafflines. However, we would like to expand this input size to its maximum and eventually evaluate the performance of the detector applied to a whole page of a music score.

Dataset Description: The dataset is constituted by searching the MuseScore database for scores matching the creative-common zero (CC0) license (equivalent to public domain license). We refine our search by filtering composer known for classical or romantic music like Mozart, Vivaldi, Beethoven or Haydn. Our training dataset is constituted of 48 scores, producing a total of 636 pages and 278797 symbols with 46 different classes. Some previously considered classes like the ‘fermata‘ or keyboard specific symbols are missing and we are looking into expanding our current dataset in order to cover more symbols. We split our dataset at the page level into a train and test set, keeping 70% for training and 30% for testing.

Quantitative and Qualitative Results: We plan to report quantitative results in term of mean Average Precision on a synthetic test set, but also produce some qualitative results on a few manually annotated real historical music scores. We hope that these results will show that a deep learning detector is able to transfer its learned knowledge from synthetic music scores to real historical music scores.

References

[1] A. Pacha, K.-Y. Choi, B. Couasnon, Y. Ricquebourg, R. Zanibbi, and H. Eidenberger, “Handwritten Music Object Detection: Open Issues and Baseline Results.” IEEE, pp. 163–168. [Online]. Available: https://ieeexplore.ieee.org/document/8395189/

[2] DocCreator an Application to generate sythetic document images for performance evaluation and retrainning. [Online]. Available: http://doc-creator.labri.fr/

[3] “MuseScore is an open source and free music notation software. For support, contribution, bug reports, visit MuseScore.org. Fork and make pull requests!” [Online]. Available: https://github.com/musescore/MuseScore

[4] SMuFL. [Online]. Available: https://www.smufl.org/

[5] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, “Speed/accuracy trade-offs for modern convolutional object detectors.” [Online]. Available: http://arxiv.org/abs/1611.10012