MNR: MUSCIMA Notes Recognition. Using Faster R-CNN on handwritten music dataset.

Federico Magnolfi, *federico.magnolfi2@stud.unifi.it*
Stiven Metaj, *stiven.metaj@mail.polimi.it*

***Abstract*—This paper presents a project about the use of Region-based Deep Learning methods to perform Optical Music Recognition (OMR) on handwritten music images. The choice of a dataset to manipulate is critical; in this case, the new MUSCIMA++ (a subset of CVCMUSCIMA) is chosen: it offers primitives and high-level notation objects on 140 music scores. Authors, after a preprocessing of the data, have used a Faster R-CNN to obtain information about notes inside music pentagrams. The results are promising: both mAP (mean Average Precision) and our distance metric improve proportionally with the time of the network training. Moreover, we notice that a pretrained network (even if the pretrain is performed on quite different data) is faster than a non-pretrained one, which instead, after the same time, keeps learning more than the first.**

***Index Terms*—region based, RPN, faster r-cnn, convolutional neural network, deep learning, optical music recognition, OMR, MUSCIMA**

I. Introduction

An open issue about Document and Data Mining is OMR (Optical Music Recognition). OMR, in fact, is the attempt to recognize objects in music sheets to replay them in digital standard forms (like MIDI files).

This type of recognition can be likened to OCR (Optical Character Recognition) for the music notation writing system; however, it is harder to fulfill because of the lack of tight standards that, instead, we can easily find in handwritten or digital text pages. Moreover, in music sheets, the same music can be represented by different forms (Fig. 1).

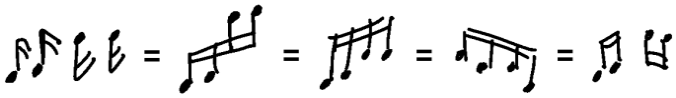

Fig. 1. Example of equivalent rhythm and notes in different forms.

For the task a well-known dataset about OMR problems is used: the CVCMUSCIMA dataset; especially a subset of it, called MUSCIMA++, brings a lot of benefits thanks to its large number of annotations.

In addition, to perform the recognition of these elements many options are available; in particular, we have decided to use image preprocessing tasks (like image segmentation and resize) and a Convolutional Neural Network achieving object detection and classification.

II. Datasets

A. CVC-MUSCIMA

The CVC-MUSCIMA database contains handwritten music score images, with 1,000 music sheets written by 50 different musicians. All of them are adult musicians, in order to ensure that they have their own characteristic handwriting style. Each writer has transcribed the same 20 music pages, using the same pen and the same kind of music paper (with printed staff lines). This dataset and its ground truth have been specially designed for writer identification and staff removal tasks, but in this study, we are interested in a different task, which is recognition of music notes position inside pentagrams [1].

The dataset contains also two other versions of the images. There are images with only staffs and images without them. Moreover, in order to permits more powerful uses, CVCMUSCIMA contains also distorted versions of the images.

B. MUSCIMA++

MUSCIMA++ has annotations for 140 images from the CVC-MUSCIMA dataset. It contains 91255 symbols, consisting of both notation primitives and higher-level notation objects, such as key signatures or time signatures. There are 23352 notes in the dataset, of which 21356 have a full notehead, 1648 have an empty notehead, and 348 are grace notes. For each annotated object in an image, it's provided both the bounding box and a pixel mask that defines exactly which pixels within the bounding box belong to the given object [2].

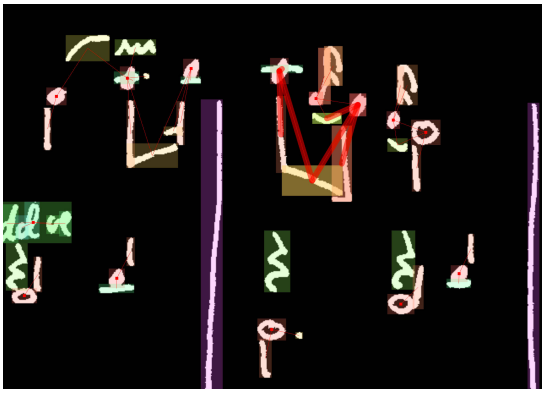

Fig. 2. MUSCIMA++ annotations example visualized on an image without staffs. Each colored rectangle represents the bounding box for an object, and the color represents the class of the object. All this information is available in XML format.

In this project, we use the recent MUSCIMA++ ground truth (version 1.0), suitable for musical symbol detection (localization, classification) and notation reconstruction. Examples of annotations are displayed graphically in Figure 2.

All annotations are stored in various XML files, which structure is similar to the following:

```
<CropObjectList>
<CropObjects>

<CropObject xml:id="MUSCIMA_W-01_N-10_0">
<Id>0</Id>
<MLClassName>notehead-full</MLClassName>
<Top>372</Top>
<Left>494</Left>
<Width>29</Width>
<Height>20</Height>
<Outlinks>730 575</Outlinks>
</CropObject>

<CropObject xml:id="MUSCIMA_W-01_N-10_230">
<Id>230</Id>
<MLClassName>notehead-empty</MLClassName>
<Top>1000</Top>
<Left>536</Left>
<Width>35</Width>
<Height>29</Height>
<Outlinks>263</Outlinks>
</CropObject>

</CropObjects>
</CropObjectList>
```

III. Task

The target of this study is the challenge to recognize notes position inside the pentagram. Given a note, the possible positions for it are: 1st, 2nd, 3rd, 4th or 5th line, 1st, 2nd, 3rd or 4th space, and above or under staffs (Fig. 3).

Fig. 3. An example to present the classes we used to identify notes inside a pentagram.

This information isn't directly available in MUSCIMA++, but we have obtained them by counting lines above and below a particular note, starting from its head, which position is annotated in original ground truth.

IV. Detection

Given an image, the process of locating and classifying objects inside the image is called detection. In the last few years, the most successful approaches to object detection are based on deep learning, and in particular on R-CNNs (Region-based Convolutional Neural Networks) [3] (Fig. 4). For this project, we have chosen to use a Faster R-CNN architecture with a ResNet-50 [**?**] backbone and we picked the Facebook implementation in maskrcnn-benchmark [4] for PyTorch machine-learning library.

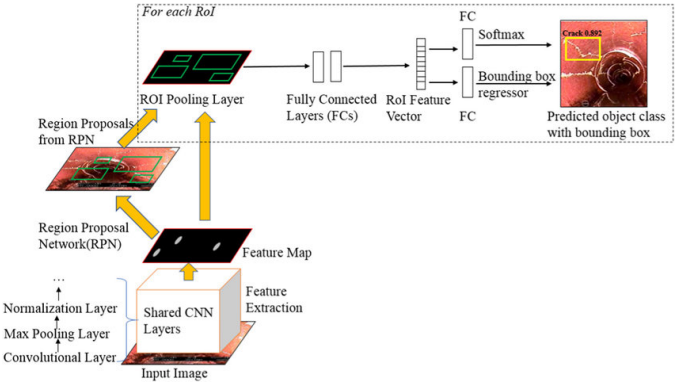

Fig. 4. A quick way to see how Faster R-CNNs work. The 2 major parts of that are the Region Proposal, also called the "backbone" of the network, and the Classificator that allows calculating the classes probabilities for each ROI (Region Of Interest) proposed.

V. Data conversion

In order to be able to use maskrcnn-benchmark for our scope, we needed to convert the dataset in a compatible format. In fact we need appropriate images for the neural network input and, moreover, we had to decide which of change the data structure or change how data are read from the network was more correct.

In particular 2 datasets are used in most cases of object detection projects:

- PASCAL VOC (Visual Object Classes): this dataset provides standardized image data sets for object class recognition. It contains 11,530 images with 27,450 ROI annotated objects inside; it has 20 classes in total like "person" or "car" [5].
- COCO (Common Objects in COntext): this dataset is a large image dataset designed for object detection, segmentation, person keypoints detection, stuff segmentation, and caption generation. It's made by more than 200000 images, representing complex everyday scenes of common objects (80 different object categories) in their natural context. The dataset is partitioned into training, validation and test sets: for each set, annotations are provided in a unique json file [6].

Understanding these datasets is primary to use most of the online libraries that perform Object Detection and Classification, in fact using them is now a standard. We decide, finally, to change the MUSCIMA++ structure to have a COCO style dataset.

A. Patches

Instead of using all the 140 images of the dataset, we decide to shred them into many smaller images, to which we refer as "patches". We have achieved this purpose with the following steps:

- Separate staffs by analyzing horizontal pixel projection (Fig. 5).
- Partition each staff in a bunch of sequential non-overlapping staffs.
- For each staff, pick other patches in a random position, in the same number as the previous point (for example if we have 8 sequential non-overlapping patches in one staff we pick another 8 random patches on the same staff).

The last step has been done in order to decrease the chances of losing a note during the conversion: in fact, a note is added to ground truth only if it's enough included inside the patch.

Finally, the created dataset contains a total of 21244 patches.

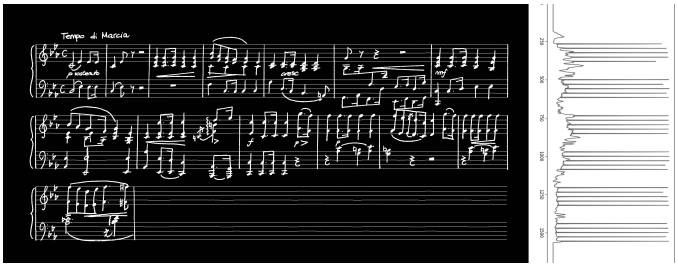

Fig. 5. A visualized horizontal projection example. Applying a threshold we can divide the image in different staffs.

B. Notes positions

The reconstruction of the position of a note starts from its notehead center: from this point (calculated from information in XML file inside MUSCIMA++ dataset [7]), we count lines above and below the note by looking into the corresponding image with only staffs (after an image preprocessing to assure that lines are not interrupted).

We then combine this information to calculate the identifier of the class, which are: 1 (US: under staffs), 2 (L1: 1st line), 3 (S1: 1st space), 4 (L2: 2nd line), ... and so on until 11 (OS: over staffs) (Fig. 6). Number 0 is avoided because it is the background identifier in the detection process.

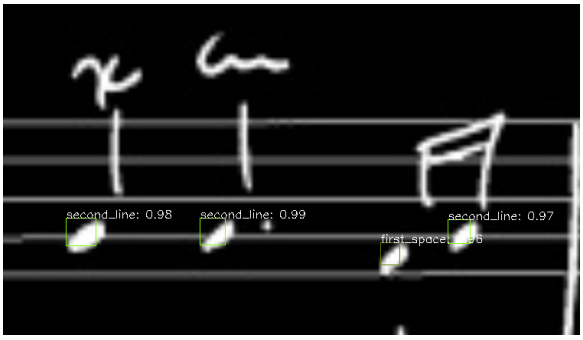

Fig. 6. A visualized prediction example.

VI. Experiments

The experiments are divided into two major steps: prepare the data for the network and train the network considering what we want to read and to evaluate in output.

The first step is already explained in the previous paragraphs. Let's spread out the second step describing the metrics we used.

A. Metrics

To compare different algorithms, in object detection is very common to use the mean Average Precision metric (mAP) [8], for different IoU (Intersection over Union) thresholds; since it's very common, we use this metric. But it's not obvious that this metric is a good measurement in Optical Music Recognition, considering that the desired high-level task is often generated MIDI files ready to be played. In fact, if the final goal of OMR is obtaining the MIDI file, it's intuitive that it doesn't really matter to predict notes bounding boxes with a high IoU w.r.t. ground truth boxes, but we simply want that the note will be played in the right moment.

For this reason, we define a specific metric for our experiments, which tries to calculate a distance between the sequence of notes obtained from the ground truth, and the predicted sequence. We made the following assumptions:

- A sequence of notes is a list of lists of notes: the main list identifies the passage of time, while each internal list contains all the notes that must be played in the same time instant.
- Note duration is not relevant for distance.
- If a note is wrong, doesn't matter how much it's wrong (for an easier first attempt of train).
- Distance between two sequences of notes is symmetric.

Of course, all of these assumptions are questionable, but despite this, we believe that the distance we define is a more indicative metric than mAP.

VII. Results

In this section, we show the results after training the network for 75 epochs considering both mAP and Average Distance between different sequences of notes. In particular, 4 metrics are presented:

- AD: Average Distance.
- AP: mean Average Precision for IoU from 0.50 to 0.95 with a step size of 0.05.
- AP50: Average Precision considering boxes having at least IoU equal to 0.50.
- AP75: Average Precision considering boxes having at least IoU equal to 0.75.

On the x-axis of the following plots, is reported the number of mini-batches. Considering that our dataset has 21244 patches and that we used mini-batches of 16 images, we can retrieve the corresponding epoch number multiplying by 16 and dividing by 21244 the number of mini-batches.

On the y-axis is reported the corresponding value of the metric.

A. Average distance

The Average Distance between true and predicted notes sequences presents a good improvement in the case of the non-pretrained model, while the distance is always pretty low in the case of the pretrained model (Fig. 7).

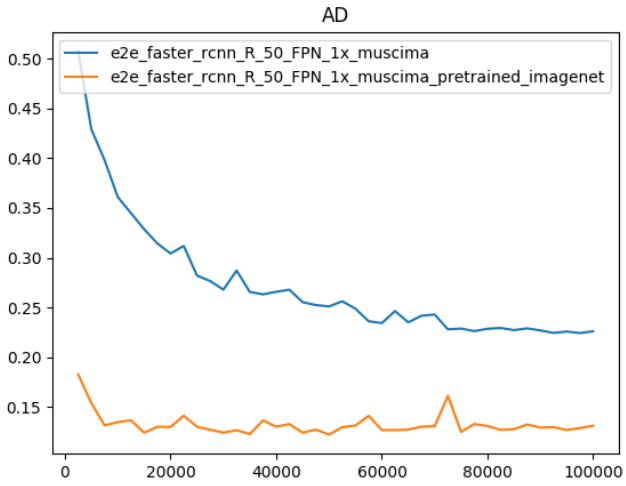

Fig. 7. Comparing Average Distance between pretrained and non-pretrained network.

B. Average Precisions

As regards the Average Precisions, similar results are observed: the AP of the pretrained model quickly grows to the maximum value, while the AP of the non-pretrained model grows slowly without reaching similar values to the other variant (Fig. 8, 9, 10).

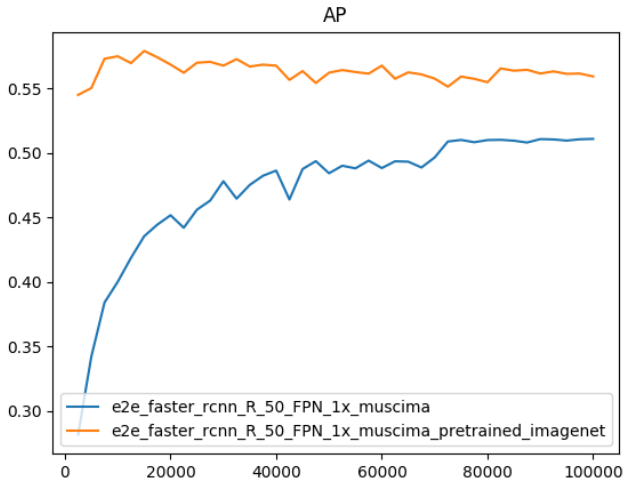

Fig. 8. Comparing Average Precision between pretrained and non-pretrained network.

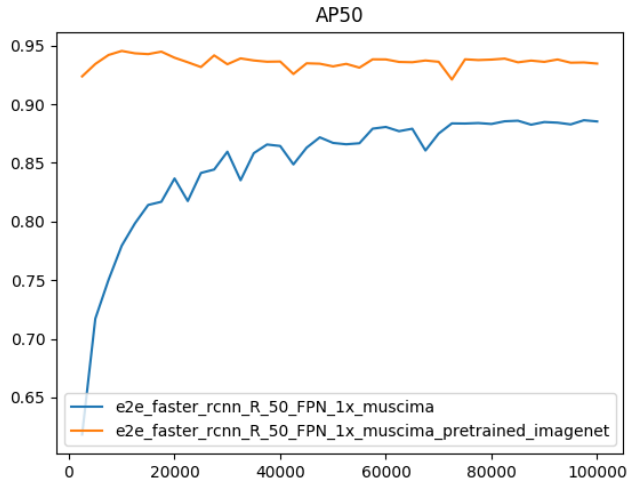

Fig. 9. Comparing Average Precision (considering boxes having at least IoU equal to 50) between pretrained and non-pretrained network.

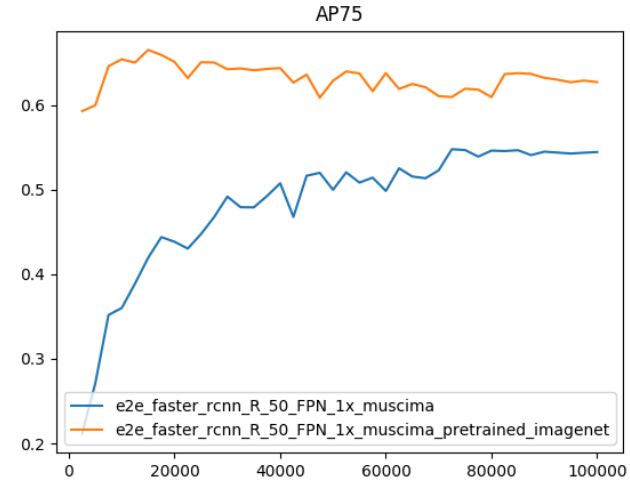

Fig. 10. Comparing Average Precision (considering boxes having at least IoU equal to 75) between pretrained and non-pretrained network.

VIII. Conclusions

We can see that the pretrained network reaches very quickly the best performances, a sign that transfer learning [9] works very well even in very different contexts. Instead, the non-pretrained network improves more slowly but continues to improve even after a long time, a sign that it needs more time to learn.

However, we have obtained a good first result: it is possible to use standard object detection libraries for the notes detection in music sheets. Even if these libraries work better and are pretrained with common images with objects like people or cars, neural networks can learn to detect notes by their position inside the pentagram with good accuracy.

IX. Future Works

There a lot of future possible works about OMR and particularly this project:

- Try a different backbone (like ResNet-101 or ResNet-152 [**?**]) to see if there are improvements.
- Extend the detection to all the notes, comprising also out of staffs notes.
- Define a better distance metric, considering how much the network is wrong (when it is).
- Use also distorted images offered by CVCMUSCIMA to improve network performances.
- Use data augmentation techniques to further enlarge the dataset.

Acknowledgment

The authors would like to thank Prof. Simone Marinai for the advice and the time spent with us during the project.

References

[1] The cvc-muscima database. [Online]. Available: http://www.cvc.uab.es/cvcmuscima/index_database.html
[2] Ufel muscima++ dataset. [Online]. Available: https://ufal.mff.cuni.cz/muscima
[3] A. Pacha, H. Eidenberger, K.-Y. Choi, B. Coüasnon, and Y. Ricquebourg", "Handwritten music object detection: Open issues and baseline results"."
[4] Faster r-cnn and mask r-cnn in pytorch 1.0. [Online]. Available: https://github.com/facebookresearch/maskrcnn-benchmark
[5] The pascal visual object classes homepage. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/
[6] Coco homepage. [Online]. Available: http://cocodataset.org
[7] J. H. c jr. and P. Pecina, "The muscima++ dataset for handwritten optical music recognition."
[8] H. Muller, W. Muller, D. M. Squire, S. Marchand-Maillet, and T. Pun, "Performance evaluation in content-based image retrieval: overview and proposals."
[9] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning."