Optical Music Recognition using Object Detection and a Convolutional Neuronal Network

Onur Kocahan and Friedrich Hartmann

March 31, 2021

1 Introduction

The goal of Optical Music Recognition (OMR) is to reproduce a sequence of document-written musical notation by a machine. Since the language of musical notation contains hundreds of symbols, this problem becomes really complex.

We aim to treat only a fragment of the musical notation language, namely the fragment of American and European folk music. It has a huge restriction on the alphabet. For instance, it does not contain chords. Furthermore, it is more appropriate of our use case. Because we want to apply OMR to an application that photocopies a tune of a song-book and transforms it later to an audio file. This can be used to learn how to sing this tune.

Available OMR Datasets such as DeepScores [Tuggener, 2021] contain around 30 000 000 sheets of written music with close to a hundred millions of small objects. This would really blow up our project. That's why we generate our own dataset.

We further follow the general framework to OMR that contains according to Rabela et al. [2012] the following steps.

1. image pre-processing

2. recognition of the musical symbols;

3. reconstruction of the musical information in order to build a logical description of musical notation; and

4. construction of a musical notation model to be represented as a symbolic description of the musical sheet.

The most challenging task in this framework is point 2. An common approach to that is to use neuronal networks. A baseline for that is given by Pacha et al. [2018]. The authors applied Faster R-CNN, RetinaNet and U-Net on various datasets. The performance on DeepScores which is not hand-written and therefore, easier to be trained, is not promising as the mean average precisions of the intersection over union ratios are 19.6%, 9.8%, and 24.8%, respectively.

An other approach by Metaj and Magnolfi [2021] is more promising. But they have really good results around 95% only for a selection where the bounding boxes have at least an intersection over union ratio of 0.50.

That's why we invent a new approach on the object detection that is not based on a neural net.

We further apply a convolutional neuronal network to classify the outputs of our objects detection mechanism which are then be used to generate an audio file.

Implementation details can be read up in our GitHub project `https://github.com/lutacluny/Sheet-Music-Recognition`.

2 Musical Notation Background

This section is about giving the reader a short summary of musical notation.

There exist various styles of writing music, but we focus only on the modern staff notation which is characterized by a staff line.
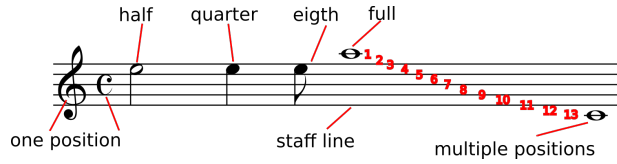
The position of the note head determines its name.

Figure 1: Musical notation simplified

Figure 2: Abstracting a tune

It can be on the line, between two lines and above or under the staff using ledger lines. Therefore, limiting the ledger lines to one upper and one lower, a line can hold 13 different notes.

The value of a note is specified by its shape and defines its duration which is given relatively to the beat. That means that the note value *quarter* has the length of a quarter beat.

A visualization of the explanations above is given in Figure 1. Other musical symbols do not depend on its vertical position at the line.

3 Training Data Sampling

We build a highly flexible framework to generate a database that contains image files which can be used to train a neuronal net. It is build of musical symbols extracted as vector graphics by the help of the tool abc2mps S. [2021].

As one generated image file per musical symbol is not sufficient for training, we introduced several parameters for data augmentation. These are output dimension, scaling, rotation, horizontal shift, vertical shift and the respective number of outputs in the ranges defined by these parameters. It is for example possible to get 5 output files with respect to the scaling in the range 80% to 120%.

Our database supports at the moment up to 107 different musical symbols. Therefore, the number of training examples is $107 \cdot N$, where N is the number of augmented files per symbol.

4 Object Detection

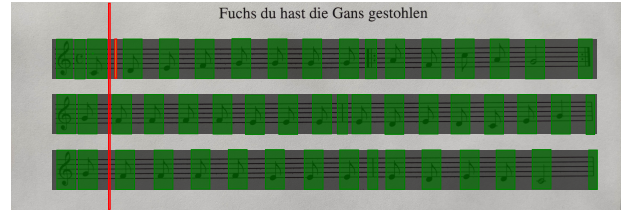In this section we describe our approach on the object detection. As the input is a single image file, the general task is to split this image into several images, such that each of the resulting images contain one and only one musical symbol which can be recognized by a machine. The first step is to transform the image from RBG into black and white.

Our idea is further based on the observation that tunes of American and European folk music, written in the modern staff notation, contain recurrent patterns which is illustrated in Figure 2.

As it can be observed, each line, colored black, has the same height and width. Furthermore, the distance between two lines remains the same. That gives the possibility to identify first the most upper line and then calculate the position of the other lines. This can be achieved by representing the image as a matrix and identifying a column, that contains only staff lines. Because such a column has a unique pattern on the distances between the lines, which can be easily calculated on the matrix. The red colored line illustrates this fact in Figure 2.

After segmentation of the lines we split each line into symbols or groups of symbols, until each of the resulting segments contain one and only one symbol. To accomplish this, we take advantage of the fact that a column of the respective matrix representation associated with a musical symbol, has a significantly bigger amount of black pixels. The reference value of a column without a musical symbol is calculated identifying a column that matches the orange line in Figure 2. This is performed the same way as for the red marked column.

The success of this procedure depends on the photo quality. It is mandatory that the staff lines are parallel to the edges of the photocopy. But in practice, this not really an issue, since most cameras have a

note detection accuracy in %	< 50	< 90	< 100	100
	1	11	65	487

Figure 3: For example, it can be observed that on 54 images up to 10% of the symbol is not visible anymore.

grid implemented.

Furthermore, there are two hyper-parameters that needs to be chosen appropriated. Like we explained before, what matters is the amount of black pixels per column. So, there needs to be a thresh hold that separates columns containing musical symbols from those containing only staff lines. This is given as ϵ_{black} in per cent. The second one ,δ, is the separate width, given as a fraction of the images width. Only columns marked as containing a note, for which its distance exceeds the separate width, are split.

It turned out that $\epsilon_{black} = 60\%$ and $\delta = 1/120$ in the first run and $\epsilon_{black} = 140\%$ and $\delta = 1/10$ in the second shows desirable results. Our test images are split into 543 different images. It holds further, that each new image contain one and only one musical symbol. The quality is shown in Figure 3.

It has to be finally admitted that choosing the right values for ϵ_{black} and δ is challenging. Finding a promising strategy on that, which performs good apart form our test suite, is not part of this project.

5 Image Classification

We tried various models for image classification, but only 2 of them show competitive results. These models we present are GoogleNet and ResNet50 with transfer learning techniques. We trained them on our data set from section 2.

In this section, we evaluate the performance on our test suite. It contains the 543 images extracted by our object detection mechanism as described before. We show further the influence of the hyper-parameters on the model accuracy.

Shift(x,y)	Rotation(min,max)	Scaling	Accuracy
(0.10,0.05)	(-2,4)	0.1	%38
(0.15,0.05)	(-2,4)	0.1	%32
(0.15,0.15)	(-2,4)	0.1	%42
(0.15,0.15)	(-2,4)	0.2	%42
(0.25,0.15)	(-4,4)	0.2	%48
(0.25,0.15)	(-4,4)	0.3	%52
(0.25,0.25)	(-6,6)	0.3	%43

Figure 4: The database generation parameters effect the classification accuracy.

Optimizer	Number of Epocs	Mini-batch Size	Accuracy
sgdm	3	64	%48
rmsprop	3	64	%46
adam	3	64	%49
adam	3	32	%44
adam	3	128	%52
adam	4	128	%41
adam	2	128	%38

Figure 5: The effect of choosing optimizer, number of epochs and mini-batch size on the classification precision

Dataset We fixed one model and compared different choices on the respective hyper-parameters. The results are shown in Figure 4. It can be observed that the accuracy differs from 32 % up to 52%. Therefore, those hyper-parameters has to be chosen with special care, since they have a big influence on the accuracy. We see later, the dataset has indeed the biggest influence on the final performance.

Optimizer, Mini-batch Size, Epochs Next, we analyze parameter such as optimizer, mini-batch size and numbers of epochs. Figure 5 shows that apart from training only 2 epochs, their influence on the best performing dataset differs only in the range of approximately 10 %. In order to compare the effect of the optimizer, mini-batch size and the number of epochs independently, each test differs only on a single parameter.

Learning Rate	Accuracy
0.001	%27
0.003	%36
0.005	%52
0.006	%45
0.007	%42
0.01	%40
0.02	%35
0.03	%50

Figure 6: Relation between learning rate and accuracy.

Model	Epochs	Training Time	Result
GoogleNet	3	10:27	%52
ResNet50	3	16:14	%45
GoogleNet	4	14:49	%42
ResNet50	4	19:31	%39
GoogleNet	5	10:14	%32
ResNet50	5	10:14	%48

Figure 7: Comparison of GoogleNet and ResNet50

Learning rate According to our observation in terms of hyper-tuning the model, the second most important parameter to increase the accuracy is the learning rate. It heavily influences over-fitting and under-fitting. This is important, since our dataset tends to over-fit. To find the best spot, we let the other parameters remain unchanged. Therefore, we can analyze the learning rate independently. Results are shown in Figure 6.

Model selection We analyze GoogleNet and Resnet50 in terms of epochs, training time and accuracy after they where finely-tuned. The results are shown in Figure 7. It can be observed that training time of GoogleNet is less, but it performs slightly better than ResNet50.

Other To evaluate the accuracy, not only the parameters mentioned before have to be considered. We noticed, that our models have difficulties on 14 of the

symbols. Such symbols occur 41 times in our test suite. Thus, we decreased the size of the test suite from 543 to 502. It remain only 93 different output labels to classify.

Furthermore, we think that a convolutional neuronal net designed for image classification, does not match totally the requirements for note classification. The first one aims to detect shift invariant features in horizontal and vertical direction. For example, if the image contains a cat, the exact position of that cat does not matter. But this does not hold for notes, because the vertical position of a note determines its name (See Section 2).

6   Post Processing

The final step is to transform the output labels into an audio file. This is performed by first converting the output labels into a format that can be processed by PySynth g4brielvs [2021] which is the tool that generates the audio file.

References

g4brielvs. PySynth. `https://tomita.readthedocs.io/en/latest/`, March 2021.

Stiven Metaj and Federico Magnolfi. MNR_MUSCIMA_Notes_Recognition. `https://github.com/StivenMetaj/MNR_MUSCIMA_Notes_Recognition`, March 2021.

Alexander Pacha, Jan jr. Hajič, and Jorge Calvo-Zaragoza. A baseline for general music object detection with deep learning. *Applied Sciences*, 2018.

Ana Rabela, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, and Jaime S. Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 2012.

Lee S. abs2mps. `https://github.com/leesavide/abcm2ps/issues`, March 2021.

Lukas Tuggener. DeepScores. `https://tuggeluk.github.io/deepscores/`, March 2021.