Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# Optical music recognition

Master thesis

2020
Jozef Martiš

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# Optical music recognition

Master thesis

| | |
|---|---|
| Study programme: | Computer Science |
| Field of study: | Computer Science |
| Department: | Department of Computer Science |
| Supervisor: | RNDr. Zuzana Berger Haladová, PhD. |
| Consultant: | Ing. Viktor Kocur |

Bratislava, 2020
Jozef Martiš


Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

# THESIS ASSIGNMENT

| | |
|---|---|
| **Name and Surname:** | Bc. Jozef Martiš |
| **Study programme:** | Computer Science (Single degree study, master II. deg., full time form) |
| **Field of Study:** | Computer Science |
| **Type of Thesis:** | Diploma Thesis |
| **Language of Thesis:** | English |
| **Secondary language:** | Slovak |

**Title:** OMR: Optical music recognition

**Annotation:** The work is a contribution to the digitization of music notation. Using computer vision and machine learning methods it examines the possibilities of recognizing the semantics of notation from ordinary photographs of notation and subsequent creation of the music sheet in digital format.

**Aim:** Examine the current state of the OMR (Optical music recognition) area. Identify the shortcomings of existing OMR methods and suggest improvements, or create a new recognition method. Implementation of the proposed method, its subsequent validation and comparison with the State of the art.

**Keywords:** music recognition, notation, musical symbols, neural networks, object detecting

| | |
|---|---|
| **Supervisor:** | RNDr. Zuzana Berger Haladová, PhD. |
| **Consultant:** | Ing. Viktor Kocur |
| **Department:** | FMFI.KAI - Department of Applied Informatics |
| **Head of department:** | prof. Ing. Igor Farkaš, Dr. |
| **Assigned:** | 03.10.2018 |
| **Approved:** | 20.12.2018 |

prof. RNDr. Rastislav Kráľovič, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor


Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Bc. Jozef Martiš
**Študijný program:** informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
**Študijný odbor:** informatika
**Typ záverečnej práce:** diplomová
**Jazyk záverečnej práce:** anglický
**Sekundárny jazyk:** slovenský

**Názov:** OMR: Optical music recognition
*Optické rozpoznávanie notového zapisu*

**Anotácia:** Práca je príspevkom do oblasti digitalizácie notového zápisu. Pomocou metód počítačového videnia a strojového učenia skúma možnosti rozoznávania sémantiky notového zápisu z obyčajnej fotografie a následné vytvorenie zápisu v digitálnom formáte.

**Cieľ:** Preskúmať aktuálny stav v oblasti OMR (Optical music recognition). Identifikovať nedostatky existujúcich metód OMR a navrhnúť ich zlepšenie, alebo vytvoriť novú metódu rozpoznávania. Implementácia navrhnutej metódy, jej následná validácia a porovnanie so State of the art.

**Kľúčové slová:** rozpoznávanie, hudobné symboly, neurónové siete, detekcia objektov

**Vedúci:** RNDr. Zuzana Berger Haladová, PhD.
**Konzultant:** Ing. Viktor Kocur
**Katedra:** FMFI.KAI - Katedra aplikovanej informatiky
**Vedúci katedry:** prof. Ing. Igor Farkaš, Dr.

**Dátum zadania:** 03.10.2018

**Dátum schválenia:** 20.12.2018 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

# Acknowledgements

**Poďakovanie:** Ďakujem vedúcej tejto diplomovej práce, RNDr. Zuzane Berger Haladovej, PhD., ktorá si napriek svojej vyťaženosti a mnohým povinnostiam, vždy našla pre mňa dostatok času. Ďakujem za jej odborné vedenie a jej veľmi ľudský prístup voči študentom. Ďakujem konzultantovi, Ing. Viktorovi Kocurovi, za jeho nápady a za to, že mal vždy ochotu a odhodlanie so mnou riešiť problémy, ktoré sa v práci vyskytli.

Ďakujem členkám Festivalového orchestru Bratislava: Baške, Deborah a Ruth, za ich neustále povzbudzovanie, častokrát vedené humorným spôsobom, ktoré mi sprí-jemňovalo tvorenie práce. Veľmi pekne ďakujem Evičke, ktorá mi bola veľkým prínosom, vďaka nej som sa tešil z každého jedného dňa.

Ďakujem svojmu ujovi, dedkovi a celej rodine za podporu. Nakoniec, ďakujem svojej maminke. Ďakujem jej za to, že ma vychovala a vo všetkom dobrom a správnom podporovala.

# Abstract

Digitization of sheet music is a complex problem that requires its division into many smaller sub-problems. In addition to the classic, well-defined approaches and heuristics, currently in this field is increased usage of neural networks, especially for the digitization of handwritten music. In practice, however, it is very advantageous ability to recognize the camera captured printed records, created by music editor. For this reason, we present a semi-automatic method of creating a dataset of camera captured sheet music using annotations from original noise-free digital images. Object detection is a natural approach to digitizing musical symbols. In this work we show that detection of musical symbols can be relatively fast and accurate at the same time.

**Keywords: music recognition, notation, musical symbols, neural networks, object detecting**

# Abstrakt

Digitalizácia notových zápisov je komplexný problém, ktorý si vyžaduje rozdelenie na mnoho menších podproblémov. Okrem klasických dobre definovaných prístupov a heuristík, sa súčasnej dobe v tejto oblasti stále viac využívajú neurónové siete, najmä na digitalizáciu rukou písaných notových zápisov. Pre prax je však veľmi výhodná možnosť rozpoznávať odfotené vytlačené zápisy, vytvorené notovým editorom. Z toho dôvodu uvádzame poloautomatickú metódu vytvorenia datasetu odfotených notových zápisov využitím anotácii z originálnych digitálnych obrázkov bez šumu. Detekcia objektov je v prípade hudobných symbolov prirodzený prístup. V práci ukazujeme, že detekcia hudobných symbolov vie byť relatívne rýchla a presná zároveň.

**Kľúčové slová: rozpoznávanie, hudobné symboly, neurónové siete, detekcia objektov**

# Contents

# List of Figures

# List of Tables

# Introduction

With the development and expansion of music notation software arises natural need of automatically digitizing created sheets of music of human readable format. Music recognition seems to be very similar to optical character recognition, although in both cases we are trying to recognize some language, recognizing music sheets is far more complex.

In this work we look at various of approaches used in optical music recognition research field. We describe principal computer vision methods used for our development. We look at problem of musical staff removing which could be fundamental step for many notation processing methods.

Missing suitable dataset, of labeled real world data, is common problem in many machine learning areas, especially in computer vision, approaches using convolutional neural networks. We create method of semiautomatic processing photographed images to which exists clear digital original. The original picture is annotated and this annotation is used to annotate photographed pictures after processing. This way we create dataset of the real environment from the real environment, rather than simulating it.

We focuse on convolutional neural network and object detectors. We present a brief description of many different approaches of object detecting using neural networks. Training time could be problem due to, for example, lack of powerful hardware or computing cost at all. We create relatively small convolutional neural network later used for object detecting purpose of the musical symbols. On the other hand, we take existing detectors models, FasterRCNN [49] and one stage detector YOLO [34] (You only look once), to train them on DeepScore [40] dataset, well known in the optical music recognition area. We show that detecting musical symbols could be precise as well as quick.

# Chapter 1

# Current state in OMR

Music notation in connection with computer vision technologies is very active research area. There is a strong need in every day life for accurate and reliable music notation digitizing tool. At the beginning, it is very good to realize that music notation is not a music and vice versa. The final sound design depends very much on the nature of the song and in particular from a particular player. There is a difference between a swing, or a classical music. With the same section of sheet music, this section will be played noticeably differently. Although the same tones will be played, in the meaning of music theory, they will not sound for the same length and will not be given the same emphasis. Even though it is with professional musicians largely understood as having many songs actually sound, the solo parts are not necessarily always played the same. Also slow songs do not have the same form in which great emphasis is placed on expression. To preserve information on how to actually play the song, what means, it is closest to the realisation as intended by the author (unless explicitly stated freedom to the performer), there are dynamic and tempo marks. Often the marks are written instruction words (officially in italian) about how to play the next part of the song (part from the occurrence of the word). A parallel can be found in the dialogue written in the book, when we often can't imagine how the character uttered a sentence telling to someone, unless the author will not describe it further. Or, using a music notation reading program to play a music sheet, is very similar to use text-to-speech software for reading a written sentence. Although music sheets are not music, it is inseparable and very natural part of it. They store a large amount of information about the song in the format very comfortable for the player.

In this chapter, we will look at the need for development in the field of Optical music recognition (OMR) and summarize its current state. Finally, we will look at important models from the machine learning that are currently widely used in OMR.

## 1.1 Need to recognize notation

In music practice, whether in the didactic or professional, it is necessary to edit many existing full scores, or scores [1]. This adjustment was initially made by manual rewriting of the notes, understandably, more and more musicians are starting to use it differently semi-automatic accelerations using music notation reading software. The problem arises when we do not have the whole score digitized (or digitized at all). In practice, MIDI keyboards are used to speed up the input for the music software, alternatively electric keys, or a cheaper alternative, ordinary keyboard with conveniently defined keyboard shortcuts. However, this process is still requiring a lot of time. Note that it is not just about rewriting the notes themselves, the basic graphic symbols for marking the tone, including their length, but many other dynamic and melodic indications. For this reason, efforts are being made to automatize the process as much as possible.

So far, however, there is not enough universal, practical and widely applicable notation recognition tool [10]. A recognized notation is either a scan or a photograph of an existing, physical, record. Thus, the recognized notation contains a lot of noise and distortion. Of course, the better the scan and the easier the notation, the more accurately it can be recognized. There are commercial tools to offer conversion from scan to digital format. Consider, for example Smartscore [2] a Photoscore [3], which can extract the necessary information from a quality scan. However, making quality scans is a time consuming task, in addition requires hardware (specifically a scanner) that is not very handy and commonly portable. What is more, even with a very high quality scan, the tools cannot recognize handwritten music sheets to a level of practical usability [7, p. 174].

In addition, there is still huge amount of non digitized handwritten parts (a large number is found in archives). Also, still a lot of musicians, especially bandleaders from the elderly generations, editing compositions for musical bands and various musical groups, writes scores for individual instruments by hand. Software that allows you to digitize such hand-made scores, would significantly helped simplify the work of editing them. It would also, from the principle of working, brought a new way to create a digital notation, or way of entering an input into a music program. Of course, the problem recognizing handwritten notes is much more difficult than recognizing scores written in well-defined notation.

However, efforts to digitize sheet music are also made for other reasons as we stated. Digitization would bring us new preservation possibilities of physical documents, simple

[1] Naming *score* refers to a document containing information about a song, usually written for one instrument. A *full score* indicates a document containing complete information about the song (all scores in parallel/above each other, one line belongs to one score)

[2] https://www.musitek.com/

[3] https://www.neuratron.com/photoscore.htm

copying and transferring methods, and finally options for further software processing, such as search. [7, p. 173]. Another reason is the existence of a large number of songs that have never been digitized in any way (for example, only by music recording, not necessarily by digitizing notes) [32], kept only in paper drafts of the score.

## 1.2 About the difficulty of the problem

Let's take an example of music notation and its subsections, namely one measure. We will not recognize the musical key or the signature, just one measure. Suppose this measure is complete (not a pre-tact) and its length is known. Let's take a relatively frequently used four-quarter measure. Let us choose optimistic range of tones from $c_1$ to $c_3$ (lowest tone that can occur in a measure is $c_1$ and the highest is $c_3$), songs played in practice often have a much larger range of tones. Consider the four-quarter measure, the length of the tones from the whole note (covers the whole measure - played for four bars), over half (covers half the measure - played for two bars) to a sixteenth note and their combinations, so that we get four beats in sum. There are total of 3820809588459176 of such combinations (from $c_1$ to $c_3$) in the C-Major scale, each of the recognized tones can be in 15 different positions ($c_1$, $d_1$, $e_1$, $f_1$, $g_1$, $a_1$, $h_1$, $c_2$, $d_2$, $e_2$, $f_2$, $g_2$, $a_2$, $h_2$, $c_3$). This gives us $2,177825732865755910 \cdot 10^{18}$ combinations per measure. We have to notice the fact, that we have not yet considered hyphens, legates, melodic ornaments and many other elements commonly found in sheet music.

In the sheet music for instruments on which multiple tones can be played at the same time, even several tones above each other can commonly appear. In practice, scores are also used for multiple voices at once, when entries for individual voices overlap in one staff (in one musical line), see image 1.1. The existence of the most ambiguous parts must also be taken into account, where there is time of freedom for the interpreter [31, p. 72]. In addition to the above, dynamic and tempo markings, pre-tactics, keys, signatures, crosses, flats, bullets, jumps (orientations/traces in a song) and much more should be taken into account, if we want to faithfully recognize the printed or handwritten song.

## 1.3 Models and approaches used in OMR

### 1.3.1 Staff removal

The first work in this field was published in 1966 [19]. As note recognition is a very complex problem, it was necessary to look for ways to divide it into smaller subproblems. Quite common has become the removal of the music staff for better segmentation of symbols, to achieve more efficient and accurate symbol recognition [20, 31] [7]. We

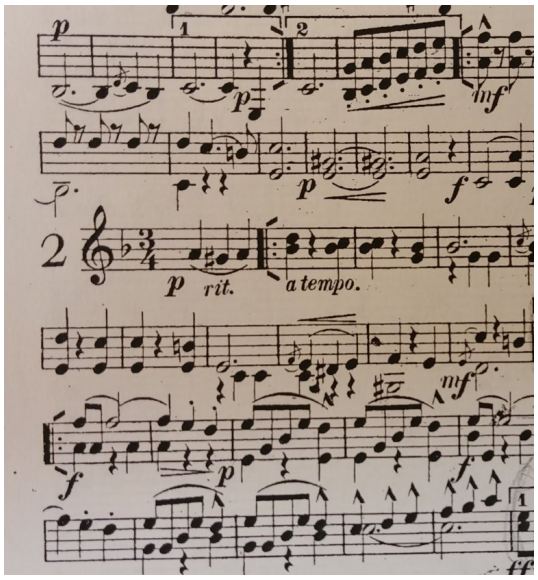



Figure 1.1: Two voices written in one score

expect the removal process not to loose no information about any symbol, from any symbol nothing will be removed. It is basically an attempt to remove pixels belonging only and only to the music staff. Methods such as projection, histograms creating, sets of candidates, graph search, or Hough transform are used to remove the music staff. The are also works in which authors recognize notes without removing the music staff to avoid problems with segmentation that may occur when the staff is removed, or the score staff is part of the procedure and thus needed for recognition [20, p. 758].

### 1.3.2 Symbol segmentation and classification methods

The most common approach of music notation recognition is hierarchical decomposition of scores. First, the entry is divided into lines according to the outline and then the graphic symbols are extracted. [7]

#### Hidden Markov Models

One of the models used is Hidden Markov Models (HMM). Works [36, 45] using HMM directly recognize notation without the need for segmentation. This approach requires having a music staff. They use a sliding window and from each are received following symptoms: number of connected components, number of components, size of the largest component, the sizes of all components together, the sizes of the smallest bounded area by components and the center of the window (gravity center). Training is performed using an extended version of the Baum-Welch algorithm [9]. This approach has achieved relatively high recognition accuracy, however only on very simple score.

#### Computer Vision Techniques

Template matching (shape similarity) is also used to classify notes [46]. In the work [47] authors deal with a robust system with the help of symbol detection and the template matching. This system is known for dealing with ambiguities in recognition symbols well. First, the notation is vertically segmented using the region-growing method and a template matching. The note symbols are then extracted again with the region-growing algorithm. In the work [26] authors use statistical momentum for detection objects, in case of ambiguity the object is classified according to the closest match.

The author in [35] presented an algorithm for recognizing printed sheets of the music. First of all, the music staff is removed. Then by evaluating the edges, bends, variety of thicknesses and significance direction (*stroke direction*, or also *stroke order*) the score is segmented into graphic primitives (lines, ellipses ...). These primitives are further classified using the k-nearest neighbor method. For each endpoint, other endpoints are searched that are collinear, or lie on a curve that results from the polynomial extrapolation of these endpoints in a segment. After classifying the primitives, there are syntactic rules applied to recreate musical symbols, it is also necessary to distinguish ambiguity.

#### Machine Learning Techniques

In addition to the Hidden Markov Models and k-nearest neighbor methods, also SVMs (Support Vector Machines) and neural networks are used in the recognition of scores. In the work [48] authors compared the above models and found the highest accuracy by classifying with SVM and k-nearest neighbor.

#### Formal grammars

As the music notation has a well-defined part of the structure, it is possible to use the [41] grammars to validate the recognized notation. Grammars have also been used directly with recognizing notation to achieve more accurate segmentation [17]. Two - level grammar approaches have also emerged, low levels for recognition of symbols as dots and lines and a high levels for musical semantics of notes. [18, 8]. There are also works dedicated to the characterization of musical notation using context-free grammars, or use of this syntactic-semantics approach with help of the lambda calculus, specifically $\lambda$ Prolog [7, p. 183].

### 1.3.3 OMR Objectives

Many works focused on the recognition of music sheets of specific notations (note fonts). A relatively common notation is Common western music notation (CWMN, or just

CMN) [14, 31], but there are also works in which authors deal with non-traditional or historical notations [5], mostly handwritten scores. New methods in machine learning and computer vision offer an opportunity to move closer to the goal of fully recognizing complete handwritten parts.

Specifically, the origin and development of convolutional neural networks bring to this area a new way of recognizing notation, or a smaller part thereof. Work [5], dedicated to medieval (specifically mensural) notation, uses convolutional neural networks. However, mensural notation is different from modern notation recordings, also even handwritten. Let us also mention the work [55] devoted to modern, handwritten, sheet music. Two convolutional neural networks are examined in this work. The first neural network is used to categorize already segmented connected components. The second network was created to distinguish a lot of different musical symbols. However, some symbols could not be always recognized by convolutional neural network due to the great similarity in shape. We will pay more attention to convolutional neural networks later.

### Gamera a Aruspix

Let us also mention the open-source framework for symbol recognition called Gamera. It is designed to work with image files, in which it can segment and classify symbols. The *C++* and *Python* languages can be used in the development environment. The individual symbols can be extracted on a basis of connected-components and classified using k-nearest neighbor. If the symbol is during extraction divided more as needed, built-in k-nearest neighbor classifier is able to rearrange the extracted symbols and then recognize them. This approach has proved necessary with recognizing very noisy image files [22].

Another existing software is Aruspix, a program on recognition of historical notations. Aruspix cuts and rotates the notation from the image, binaries and uses different heuristic approaches to reclassify each part of the picture. This preprocessing work is followed by recognizing which uses hidden Markov models [36, 45], without the need to remove the music staff. Aruspix contains a built-in editor for possibly correcting erroneous detections.

In [44], Gamera is compared with Aruspix historical notation recognition program. In this comparison, on sheet music from the 16th and 17th centuries, Aruspix achieves a higher percentage of correctly recognized symbols.

### Inconsistent OMR definition

According to [20, p. 753] recognition of printed sheets is no longer considered as problem, in contrast with handwritten records, which recognition authors consider open

problem. However, the goal of development in OMR and definition of OMR, is understood intuitively. Hardly without a clearly defined goal can be evaluated and generated some system or approach as sufficient enough [33]. In addition, especially in new approaches using deep learning, the models are trained on various datasets, which only makes difficult their objective mutual comparison and comparison state of the art in OMR area [6].

# Chapter 2

# Computer vision approaches

In this chapter we will discuss approaches that are categorized as classic computer vision. In other words, approaches, that do not use deep learning. Music scores have only in some way similar, well defined structure. For printed music sheet generated in music notation program can be applied pattern matching for example due to having symbols all the way of the same size and shape. But even only for prepossessing there is strong need to use classic computer vision tools. Years of research in this area of non deep learning approaches, brings us great set of tools for our everyday work.

## 2.1 Hough Transform

The Hough Transform algorithm serve us to find simple shapes. Algorithm was published in 1972 by R. Duda and P. Hart [23], in that time for detecting lines and curves. Now it is convenient way of finding shapes like lines, circles, ellipses and similar.

### 2.1.1 Prepossessing

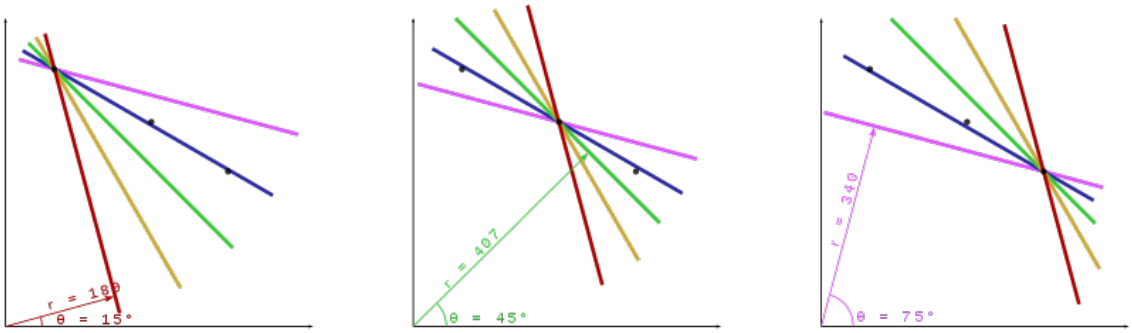
In order to correctly use Hough Transform method for finding objects, the image should be preprocessed and significant points should be extracted. The selection process depends on the character of task and its goal. In general, changes of intensities, colors etc. are the ideal case to consider. Sliding window technique can be used, for every suitable position, e. g. for every pixel, we calculate feature in base of surrounding values. After applying this step for every position, we can decide which points will be selected for further processing. The simplest way is to select points using their feature values in comparison with a threshold.

### 2.1.2 Algorithm

There is voting made for every potential point, which comes from prepossessing of the image. Every potential line passing the point is counted in the accumulator. Accumulator for one potential line stores shortest distance $r$ from point with coordinates $(x = 0, y = 0)$ and angle $\theta$ between x axis and connecting the $(0, 0)$ point with the closest point on the line. For the line can be applied following equation:

$$r = x\cos(\theta) + y\sin(\theta)$$

The line equation in form $y = ax + b$ is not used for the unpractical reason of machine computing where vertical lines could have infinite values of parameter $a$. We can then select lines on base of the voting process, for example, as in preprocessing, with using a threshold value, or we can choose top $n$, sorted by counted votes.




| θ | r |
|---|---|
| 15 | 189.0 |
| 30 | 282.0 |
| 45 | 355.7 |
| 60 | 407.3 |
| 75 | 429.4 |

| θ | r |
|---|---|
| 15 | 318.5 |
| 30 | 376.8 |
| 45 | 407.3 |
| 60 | 409.8 |
| 75 | 385.3 |

| θ | r |
|---|---|
| 15 | 419.0 |
| 30 | 443.6 |
| 45 | 438.4 |
| 60 | 402.9 |
| 75 | 340.1 |

Figure 2.1: Hough Transform algorithm. All potential lines are laid through each point and vote is incremented in the accumulator [4]

## 2.2 Edge detectors

#### Simple edge detector

In the case of detecting edges we can also come up with many ideas for solution. Good practice, with these types of problems, is to convert image to gray-scale image. In most situations, in order to find geometric primitives, intensities of gray-scale are sufficient enough. For every pixel $p_{x,y}$ in RGB color space model, we can evaluate weighted sum of color components following way:

$$p_{x,y} = w_R R_{p_{x,y}} + w_G G_{p_{x,y}} + w_B B_{p_{x,y}}$$

where $w_R, w_G, w_B$ are weights with sum 1. For gray-scale image we can use $0.299, 0.587, 0.114$ values respectively. Next step, for example, can be again to use sliding window technique. We can gain feature map where higher the number of feature is bigger the difference of two points on the original image. Problems due to a noise, low quality of photos etc. may appear. These problems are solved by the following detector.

### Canny edge detector

In the Canny edge detector [21] the first step is Gaussian filtering. Motivation is to remove noise from the image whereas characteristics of edges will be preserved. The image is smoothed applying the following Gaussian filter kernel on every pixel.

$$H_{ij} = \frac{1}{2\pi\sigma^2} exp(-\frac{(i-(k+1)^2+(j+(k+1))^2}{2\sigma^2}) \text{ where } 1 \leq i, j \leq (2k+2)$$

The size of the kernel is $(2k+1)(2k+1)$ and $\sigma$ is a free parameter. Bigger the kernel size, the more dimmed the noise level in the processed image. The detector distinguish three types of edges: vertical, horizontal and diagonal. Four filters are used to detect edges and their angles against the specific axis are rounded with the closest match of the following values: $0°, 45°, 95°$ and $135°$. In order to find only the thickest edges, we can use non-maximum suppression to thin all of the edges. With non-maximum suppression method, pixel intensities are compared in the negative and positive directions of the gradient, subsequently filtering will be applied and only pixels that are larger than neighbors pixels, based on mask and direction, will be preserved. After that step, we can filter edges based on our chosen threshold of intensity.

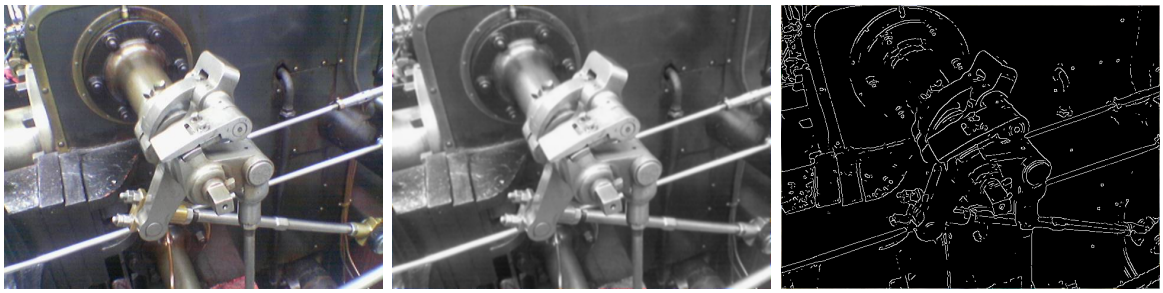


Figure 2.2: Original image [3], image after applying Gaussian filter of size 5x5 [1] and the resulting image on the end of the canny edge detecting process [2]

## 2.3 Harris corner detector

Finding corner is another common task in computer vision. Knowing position of the corners are strategically important for solving amount of diverse problems. Corners Harris detector [29] algorithm is widely used for this purpose.

### 2.3.1 Algorithm

First, we need to convert image to gray-scale color space. Next step is to build a structure tensor, which will show, for some window, the most dominant direction of image points regarding the gradient in the surrounding of the point [11].

$$M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_{(x,y)\in W} I_x^2 & \sum_{(x,y)\in W} I_x I_y \\ \sum_{(x,y)\in W} I_x I_y & \sum_{(x,y)\in W} I_y^2 \end{bmatrix}$$

where the I is an image and W is a window. Then we can compute Harris response:

$R = det(M) - k(m_1 1 + m_2 2)^2$ where $k \in [0.04, 0.06]$ is an adjustable parameter.

After that process of computing responses, we will select potential corners by comparison with our selected threshold value.

# Chapter 3

# Convolutional neural networks

In this section we will take a closer look to the general architecture of a convolutional neural network and which way to use them to solve some types of problems.

## 3.1 Development

The neural network gives us a convenient way to connect different models of machine learning together. In principle, this does not bring us added value in problem solving, Multilayer perceptron (MLP) can be simulated using several SVM [51]. However, thanks to the practicality, neural networks are gaining in popularity. An important prerequisite is the availability of the large amounts of data, which are necessary to achieve relevant results in solving many problems using neural networks. Especially, for the development of convolutional neural networks is important high availability of image (labeled) files.

As we have indicated, a convolutional neural network (CNN) is a model built against image file. CNN is trained on images, or we can say matrices, and serves for example for the classification of images. Work on the first network of this type was published in 1998, called LeNet. However, not much attention was paid to it. The turning point came in 2012, when the AlexNet network was introduced. Since then, this area of machine learning has experienced rapid development.

For better understanding, we can use fully connected network with one dimensional input layer of size $w.h$, where $w$ is the width and $h$ height of a image. In the other words, we can spread image to one dimension, concatenating lines by line. Important part of CNNs is convolutional layer, where the convolution is applied. Kernel (window) is used for moving trough image and new window is being produced. Take image X of size 3x3 and kernel W of size 2x2.

$$h_{11} = w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22}$$
$$h_{12} = w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23}$$

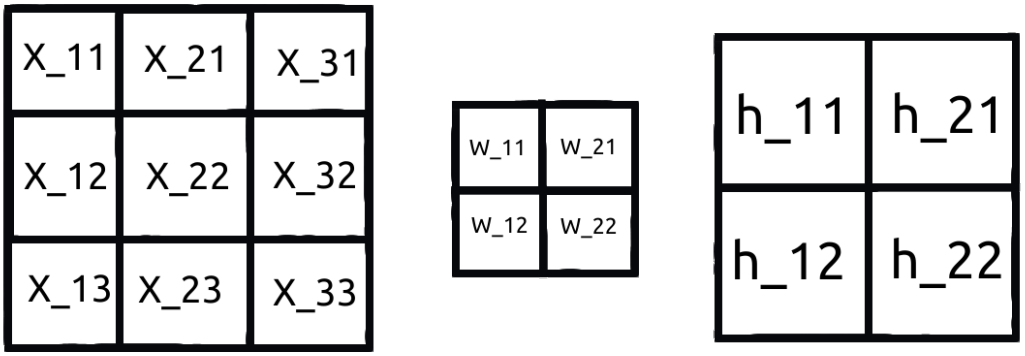



Figure 3.1: Convolution in Convolutional neural network. Kernel W of size 2x2 is applied on the input image X of size 3x3. The resulting image H will have dimension 2x2.

$$h_{21} = w_{11}x_{21} + w_{12}x_{22} + w_{21}x_{31} + w_{22}x_{32}$$
$$h_{22} = w_{11}x_{22} + w_{12}x_{23} + w_{21}x_{32} + w_{22}x_{33}$$

Applying convolution we get image H o size 2x2. See 3.1 In general:

$$h_{ij} = (\sum_{k=1}^{K_W}\sum_{l=1}^{K_H} w_{kl}x_{i+k-1,j+l-1}) + b$$

for some $0 \leq i \leq K_W$, $0 \leq j \leq K_H$ , where $X$ is the input image, $H$ is the output matrix, $W$ is the kernel, $b$, bias, what is a scalar, $K_W, K_H$ are the dimension of the kernel, width and height respectively. The shape of output feature map will be following:

$$(N_h - K_h + 1) \times (N_w - K_w + 1)$$

where $N_h$, $N_w$ are the height and width of the input image respectively.

### 3.1.1 Padding

In order not to loose information when applying convolution on the edges of the input image, we can add padding. Padding is extension of the image map beyond the original pixels. Size of the output feature can be computed following way:

$$(N_h - K_h + P_h + 1) \times (N_w - K_w + P_w + 1)$$

Where $P_h$ is the height and $P_w$ is the width of the added padding to the input image.

## 3.2 Architecture

Now we outline the general architecture of CNN. The network consists of the interconnected layers, the layers in order being:

- Input layer - image itself
- Feature map - layer representing several matrices (kernels) the same size as the image size - convolution is applied
- Pooling layer - a layer that shrinks the image by specific scale (feature maps behind this layer will be sized as the image scaled down until the next pooling layer occurs)
- Feature maps, pooling layers and dropout layers ...
- Layer transforming input from matrices (kernels) to form the next fully linked layer
- Fully connected layers - standard neural network
- Output layer

## 3.3 Training

During the training, the standard error propagation takes place, in feature maps, too. Another view can be, that there are layers like dense ones, but not fully interconnected. Which layers will be linked to us will actually determine the size of the filter.

Between layers, whether between convolutional, or fully connected, can be found *dropout* layers causing random disfunctionality of connection (with a predetermined quantity). These layers serve us to prevent overfitting, that means, the network does not learn (set parameters) against training data.

Training of such a convolutional network can be time consuming or requiring expensive computing machines hours. For training CNN with 40000 images of size 320x320 included up to 10 categories, to achieve an sufficient accuracy using a suitable, optimal network architecture and optimization function, it can take several hours to tens of hours of time when using user-oriented hardware. These depends also on many other factors used either directly in model architecture as parameters of training. The point is, we need, in ideal case, to train network just once. The network then can be used in production evaluating inputs in a fraction of training time, where inputs are evaluated multiple times repeatedly.

### 3.3.1 Error

During training we are changing weight values in a way of minimizing error e. g. loss function. Thanks to this we, can get results from network closer to ground truth. We can compute error for every training example, or after some number of them, at the end of the batch. Multiple types of loss function are widely used, like mean error, mean square error, etc. We demonstrate computing mean square error for example of convolution on the figure 3.1. Let $T$ be the ground truth matrix. Error $E$ can be evaluated the following way:

$$E = \frac{1}{2} \sum_{i \in \{(0,0),\ldots,(2,2)\}} (t_i - h_i)^2$$

### 3.3.2 Backpropagation in CNN

The main goal is to use backpropagation algorithm for training convolutional layers. We will focus only on backpropagation of convolutional layer. Let the gradient be propagated to $H$ back from parts of the network after $H$. Consider convolutional layer as shown on figure 3.1. Computing derivation of the weights can be done by following equation:

$$\begin{aligned}
\partial w_{11} &= x_{11}\partial h_{11} + x_{12}\partial h_{12} + x_{21}\partial h_{21} + x_{22}\partial h_{22} \\
\partial w_{12} &= x_{12}\partial h_{11} + x_{13}\partial h_{12} + x_{22}\partial h_{21} + x_{23}\partial h_{22} \\
\partial w_{21} &= x_{21}\partial h_{11} + x_{22}\partial h_{12} + x_{31}\partial h_{21} + x_{32}\partial h_{22} \\
\partial w_{22} &= x_{22}\partial h_{11} + x_{23}\partial h_{12} + x_{32}\partial h_{21} + x_{33}\partial h_{22}
\end{aligned}$$

## 3.4 Access to notation recognition using deep learning

Utilizing deep learning, the model that we will use for the problem solving, consist from several layers (in principle at least 3). So, if we're talking about applying CNN, we also mean using deep learning.

A general algorithm was proposed in [6] for notation recognition using deep learning. The algorithm has following steps:

#### Image preprocessing

At this stage, standard operations such as noise removal are performed, contrast adjustment, crop, transform, or binarization an image.

#### Object Detection - Music Symbols

This is the most difficult part when it is appropriate to use deep learning. We need to find and classify all relevant symbols from the image.

#### Processing - Relational understanding

Here we need to connect all the relevant symbols found together to create a correct musical notation. Due to various ambiguities (recognized symbols are at the level of stems and note headers) it is appropriate to use different syntactic semantic approaches, although most of the semantic work at the level of musical notation takes place in the next phase.

#### Encoding

The result should be a digital music file (such as MIDI or MusicXML notation) for further processing by music programs. However, it must be taken into account that many symbols will be missing, therefore formal grammars can be used. There is need to do that for all the symbols which can be clearly identified and at the same time we know that they are missing (we know some are missing).

## 3.5 Object detection

In computer vision are commonly mentioned three approaches, based on our task, see 3.2. The simplest is image classification when we want to assign one class to whole input image from our set of classes. For example set of classes is dog, cat, other and no other classes should be assigned. The second method, object detection, is about to find area on the image that we will be classifying independently. That can be explained like clipping parts of the image and the parts will be used as inputs for classification. The last method is segmentation where we want to mark all the pixels belong to object of interest on the image.

We used object detection models in order to recognize music symbols and get their position at the picture. As outlined, detecting object at the picture is about to find exact position of rectangle bounding our object of interest. Reasonable request is having the rectangle best fit, our object is not sticking up trough the edges and is impossible to scale down the rectangle. In real results of detecting object at the picture is common not having rectangle best fit, but measurements and position of it are still enough to conclude correct detecting of a object. In this chapter we will discuss about object detectors and their usage for our purpose.

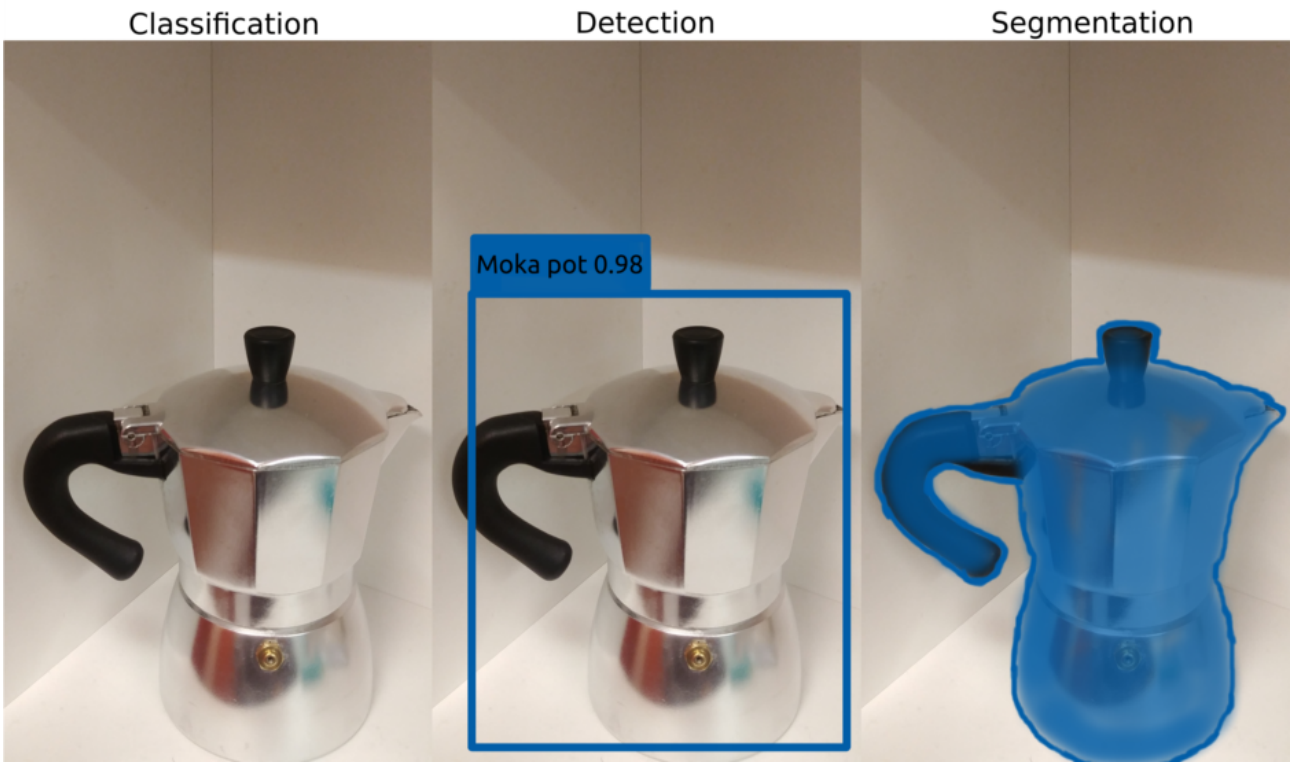



Figure 3.2: In the first case whole image is classified. The second case, object detection, we are looking for position and bounding box for object that will be classified separately. The last, segmentation, we want to have exact pixels belonging to object.

## 3.6 Faster RCNN

Faster RCNN [49] is a image object detector, an improvement of Fast RCNN [27] which is improvement of Regional based Convolutional Neural Network (RCNN) detector. Improvement against older RCNN came in speed of recognizing as well as in accuracy of classifying regions. All of mentioned detectors work in two fundamental stages. In first stage, bounding boxes are found using RPN (Region Proposal Network). Bounding boxes are interesting part of image, they surround potential objects we are looking for. These boxes will be classified in the second stage of the detector. See figure 3.3

### 3.6.1 Region Proposal Network

Aim of a Regional Proposal Network is to find rectangles bounding objects. RPN also says us for every region its objectness score. Basically RPN is determining among foreground and background of image. It is achieved using sliding window on the input image. Cropped part of the image, based on sliding window cut, is used as input for next, small subnetworks, see image 3.4.

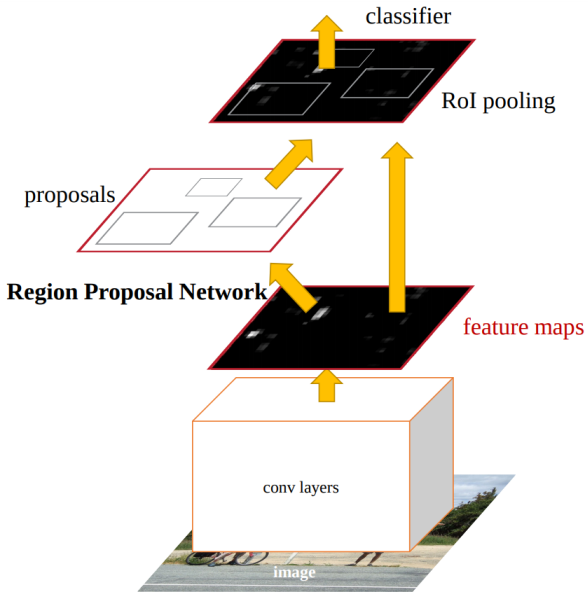



Figure 3.3: Faster RCNN network architecture [49]

## 3.7 One-stage detectors

Let us mention RetinaNet [38] network. RetinaNet belongs to category of one-stage detectors. Thanks to one-stage architecture, the detector could process far more images in the same period of time in comparison with more-stage detectors, e. g. FasterRcnn. Similarly on the working principle, to the one-stage detectors category belong, for example, OverFeat [53], Single Shot Detector (SSD) [39] and You Look Only Once (YOLO)[34].

### 3.7.1 YOLO

You Only Look Once (YOLO) detector [34], is a one-stage detector providing real time detection of objects. Unlike many other detectors, where input image is usually many times resized, YOLO was developed with an idea of applying single neural network to full image. The network split image to regions. For each region are predicted bounding boxes and probabilities of object separately. See the principles outlined on the figure 3.5.

## 3.8 U-net

U-net model [52] is a network from the set used for image segmentation. The U-net's architecture use in the first half standard model of scaling layers down, in the second, layers are re-scaling back, similarly using inverse convolution (convolution up). These backward upsized layers have parts filled with copied feature map from previous layer

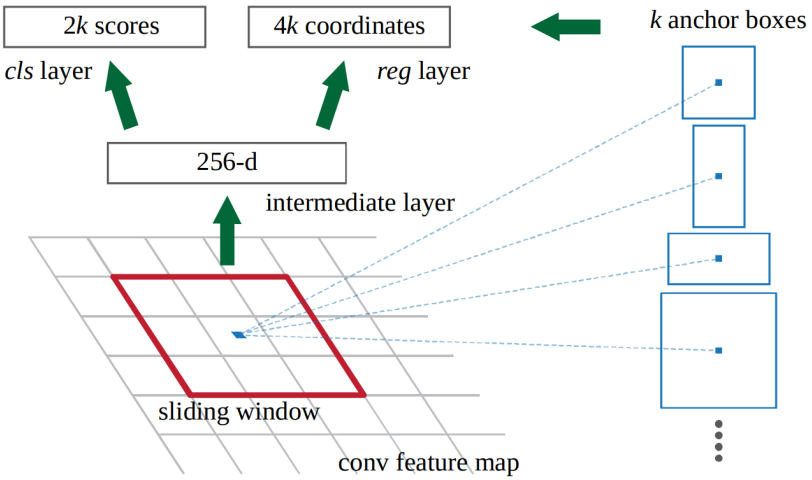



Figure 3.4: Regional Proposal Network [49]

of the same size. See the picture 3.6

## 3.9 Metric types used for detection

Metrics used for detection models use different approach than with classical machine learning problems, or at least definition of sets of True Positive and True Negative predictions. For example, predicted bounding box slightly shifted against ground truth could not be necessarily marked as wrong detection. We will present three common definitions used in the area of computer vision and object detection.

### 3.9.1 Recall

The metric tells us ratio of intersection area by area of ground truth bounding box. Standard definition of recall is

$$Recall = \frac{True\_positive}{True\_positive+False\_negative}$$

### 3.9.2 Precision

The precision metrics value is given by ratio of area of intersection by area of predicted bounding box. Precision definition is

$$Precision = \frac{True\_positive}{True\_positive+False\_positive}$$

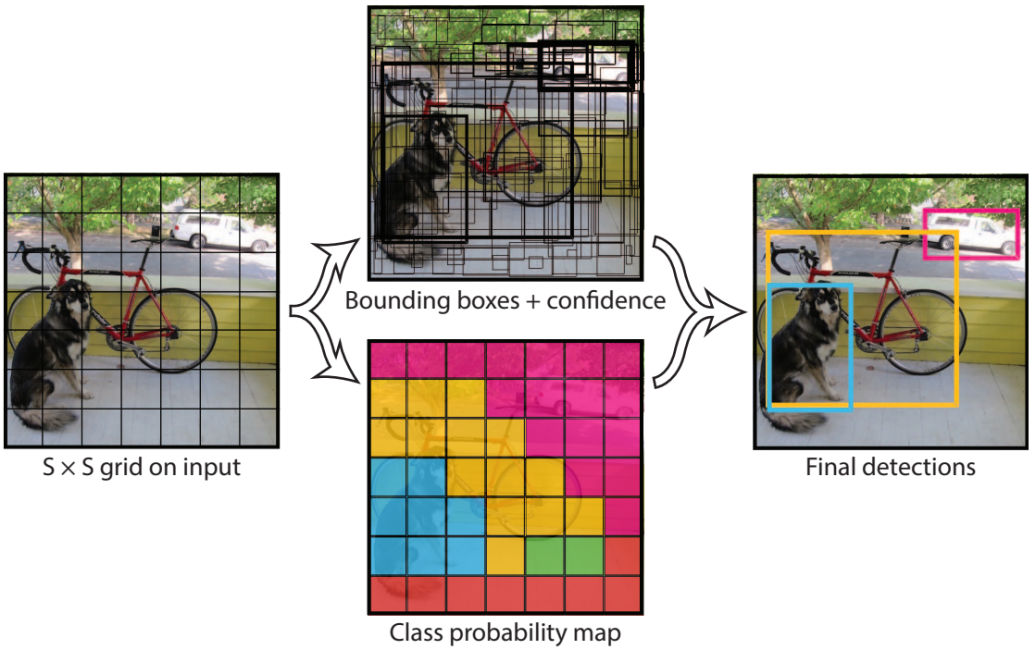



Figure 3.5: Principles of working You Only Look Once (YOLO) detector [34]

### 3.9.3 Intersection over Union

Intersection over union tells us ratio of intersection area of predicted bounding box with ground truth bounding box by area which arises from union of the two bounding boxes.

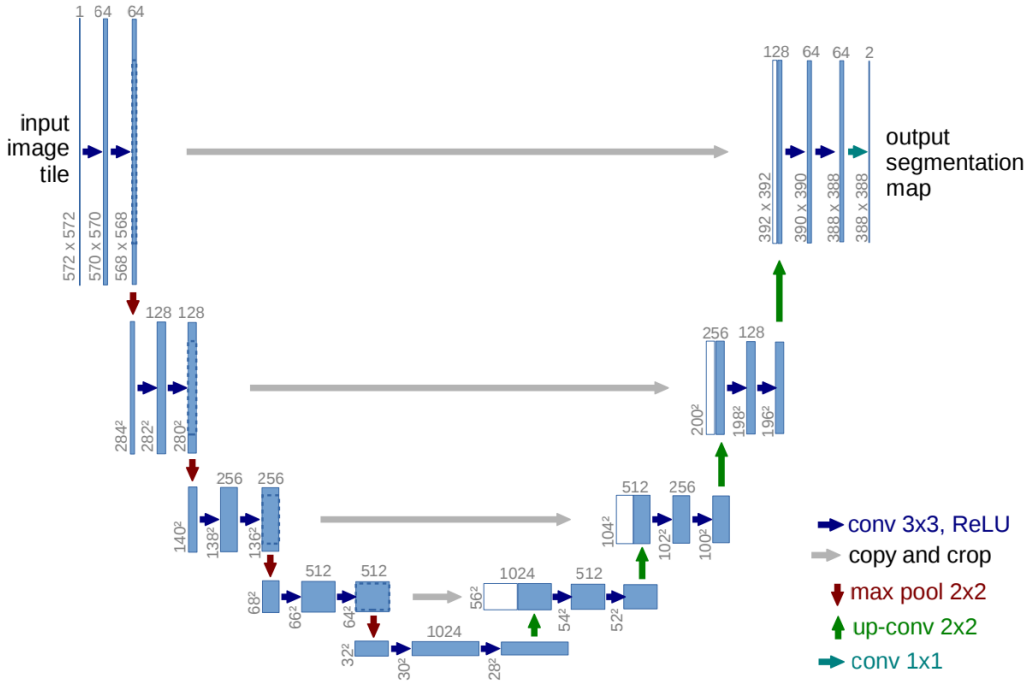



Figure 3.6: U-net architecture by [52]

# Chapter 4

# Datasets and data format

## 4.1 Data format

Object detection is active developing area where is strong need for huge amount of data, especially high quality data. Our goal is to have labeled images with position of rectangles bounding objects that we want to detect on the image. In datasets we need to describe, in some way, rectangle position and object category that rectangle is bounding. In specific cases there can be many other attributes. It is not so straightforward with object descriptions as with the pictures themselves. We will discuss two well known formats for object detection description, MS COCO [37] and Pascal VOC [24] format.

### 4.1.1 MS COCO data format

MS COCO [37] (Microsoft Common Objects in Context) is a dataset for object recognition which can be used for classification, detection as well as for segmentation. Now, we are going to focus on format in which data and annotations are stored.

#### Images in MS COCO

First of all, images and data are two separate parts. Images are stored separately, no matter where, they just need to be accessible, for practical purpose. Images are independent of annotations files. To add, dozens of gigabytes of images are available [37].

#### Annotations in MS COCO

Descriptions of images are stored in JSON format [43]. Root of JSON tree contains 4 keys, *info, [image], [annotation], [license]*. *Info* part is for general description of

whole dataset. We can describe here *year, version, description, url, date_created* attributes. Next attributes in root declare arrays of mentioned, for every image separately. In *image* we can define *id, width, height, file_name, license, flickr_url, coco_url, date_captured.*

*Annotation* class can has multiple formats. We distinguish separate formats for object detection, keypoint detection, stuff segmentation, panoptic segmentation and image captioning.

Object detection annotation consist of:

```
annotation{
    "id": int,
    "image_id": int,
    "category_id": int,
    "segmentation": RLE or [polygon],
    "area": float,
    "bbox": [x,y,width,height],
    "iscrowd": 0 or 1,
}
```

Thus we can describe bounding boxes around objects in images. *Category_id* is used to set, as expected, specific category. Category has three attributes:

```
categories[{
    "id": int,
    "name": str,
    "supercategory": str,
}]
```

It is used to store the mapping of *category_id* to names of category and supercategory.

With keypoints in computer vision we are looking for significant signs on image, called keypoints. It can be, for example, finding joints on figure or person (no necessarily real ones) in order to find abstract skelet, usually much more simpler than the real skeleton of person. Or it can be used to describe structure of face [42] in own way which can be used for multiple purpose in face recognition. Firstly, MS COCO's keypoint detection's description is an extension of object detection annotation with two additional fields.

```
annotation{
    "keypoints": [x1,y1,v1,...],
    "num_keypoints": int,
    "[cloned]": ...,
}
```

Attribute *keypoints* is an array of length $3k$, where $k$ = *total number of keypoints defined for the category*. For keypoint $k_i$ there are three values $x_i, y_i, v_i$, where

- $x_i$ = *x-position of keypoint on the image*
- $y_i$ = *y-position of keypoint on the image*
- $v_i$ = *visibility flag:*
  - 0 if keypoint is not labeled, also implies $x = 0$ & $y = 0$
  - 1 if keypoint has a label but is not visible
  - 2 if keypoint is labeled and visible

Attribute *num_keypoints* designate us the number of visible labeled keypoints for a object. Secondly, there is *categories* array of following structure:

```
categories[{
    "keypoints": [str],
    "skeleton": [edge],
    "[cloned]": ...,
}]
```

Consistently, *keypoints* is an array of keypoint names. The second attribute, *skeleton*, store edges between keypoints.

Stuff segmentation annotation is identical to object detection format except for *iscrowd* attribute, which has default value 0 in the case when we are segmenting single object in image. For writing information about segmented area we have mentioned attribute *segmentation*. The array of polygons is used in the case of single segmented object in image (i. e. *iscrowd*=0). When we need to segment multiple objects in an image, *iscrowd* is 1 in this case and we need to use RLE instead of the array of polygons due to a MS COCO's specification. Run Length Encoding (RLE) is method for compressing repeated values by value which tells us the number of value's repetition.

The last but one format, panoptic segmentation. Motivation for formation of panoptic segmentation is about effort to bring the next level of state of the art in computer vision. Panoptic segmented image has fully segmented each pixel labeled with category. Previous ways of segmentation was about to highlight one or multiple objects on the image. With goal of labeling every single pixel panoptic segmentation's approach shows out to be better in practice than previous ones. The annotation JSON format for panoptic segmentation is following:

```
annotation{
    "image_id": int,
```

```
    "file_name": str,
    "segments_info": [segment_info],
}
```

Where *image_ id* determines us id of actual annotated image. The attribute *file_ name* point to file with stored annotation. Files are per-pixel segmented and stored in PNG format. Each pixel has own unique id. Since PNG file is stored in RGB, the id of a pixel should be computed as follow:

$$id = red + green * 256 + blue * 256^2$$

The third attribute, *segments_ info* is an array of *segment_ info* class which has following structure:

```
segment_info{
    "id": int,
    "category_id": int,
    "area": int,
    "bbox": [x,y,width,height],
    "iscrowd": 0 or 1,
}
```

Each segment on the image has its own description, its own unique *id*. For example, segment can be car on image of parking lot, tires on image of one car or just coffeemaker on the shelf, see 3.2. Attribute *category_ id* point to category class, which gives us semantic description of a category. The *area* provide additional info as well as *bbox*, which has typical structure, anchor given by coordinates *x, y* and *width* and *height* of bounding box. The last attribute, *is_ crowd* indicates whether given segment is encompassed by group of objects of some category. The category format structure is very similar to detection category format unless two additional fields are added:

```
categories[{
    "id": int,
    "name": str,
    "supercategory": str,
    "isthing": 0 or 1,
    "color": [R,G,B],
}]
```

Attribute *isthing* determine us whether given category is stuff or not. The last one, *color*, servers us for consistent visualization.

The last annotation format in MS COCO we will describe, is image captioning in the dataset. The image captioning annotation format has the following structure:

```
annotation{
    "id": int,
    "image_id": int,
    "caption": str,
}
```

Every stored image have at least five belonging captions. Every caption has unique *id*. Attribute *image_ id* point to image what the caption belong to. Last attribute *caption* is raw text description. The image is described thanks to captions, semantic relations between objects or the situation on the image is given, see 4.1.

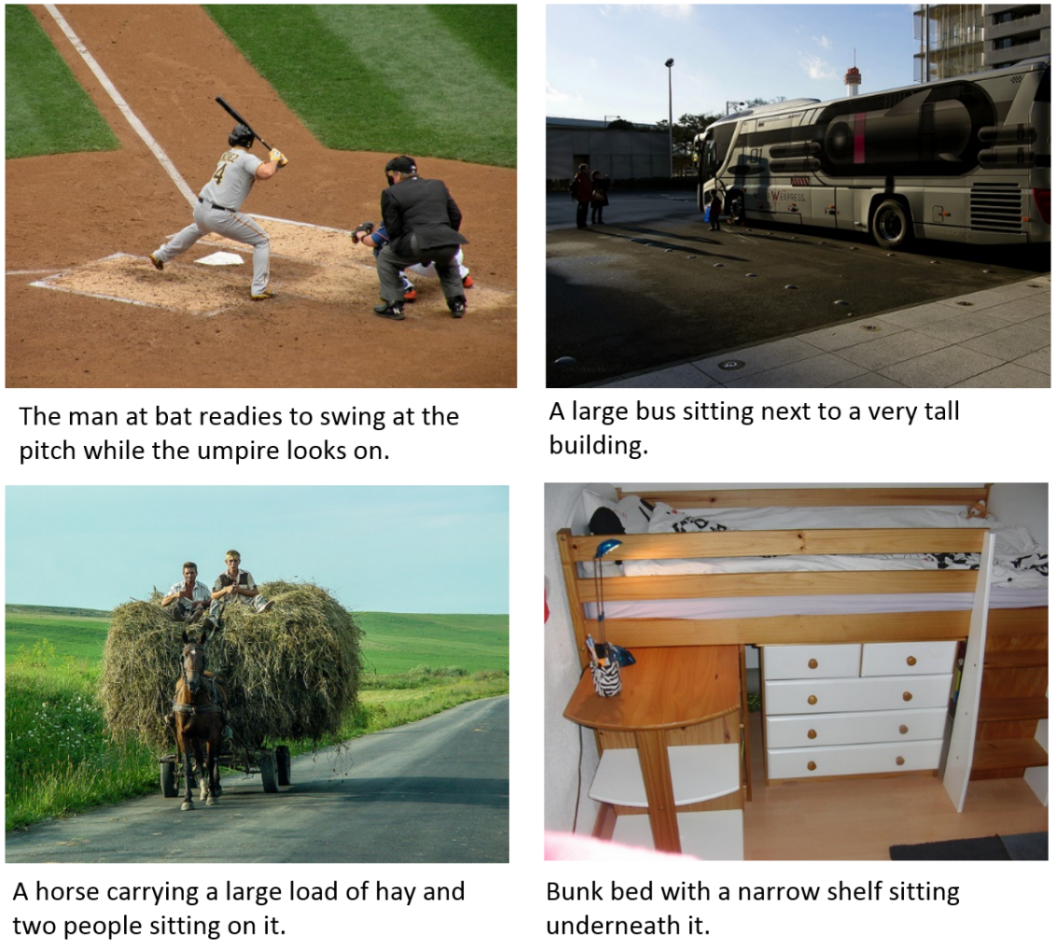



Figure 4.1: An example of image captioning [16]

### 4.1.2 Pascal VOC

Pascal Visual Object Classes (VOC)[24] is collection of datasets for online competition held in challenges. It is intended to progress state of the art in computer vision for different tasks like detection or segmentation. For every challenge a dataset is given and we are going to look at the format of the dataset. In comparison with MS COCO, Pascal

VOC store its images annotations in XML file format. The main format structure for this dataset can be noticeable on the following example of image description xml file:

```
<annotation>
    <folder>VOC2012</folder>
    <filename>2008_000200.jpg</filename>
    <source>
        <database>The VOC2008 Database</database>
        <annotation>PASCAL VOC2008</annotation>
        <image>flickr</image>
    </source>
    <size>
        <width>500</width>
        <height>375</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>person</name>
        <bndbox>
            <xmin>119</xmin>
            <ymin>76</ymin>
            <xmax>184</xmax>
            <ymax>311</ymax>
        </bndbox>
    </object>
</annotation>
```

We can observe another difference to MC COCO annotation format, bounding box in Pascal VOC format is given with two anchors, upper left and bottom right. We can describe multiple objects in one *annotation* parent. Segmentation of images is done using separate file for every image. The file is also an image where every pixel has black color if it doesn't belong to any class or color depending on what class of an object on the image has assigned. See 4.2.

## 4.2 Deepscores dataset

Deepscores is dataset [40] for detection and segmentation of printed music sheets. Labels are stored in xml files, with description of objects and their bounding boxes. Structure of deepscores annotation xml file:

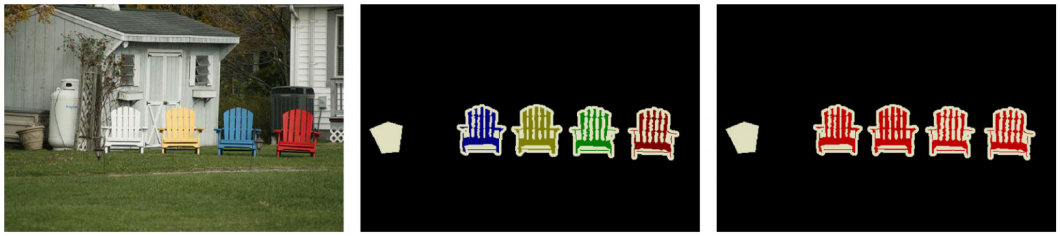


Figure 4.2: Segmentation in PASCAL VOC [24]

```
<annotation>
    <folder>Annotations</folder>
    <filename>Name of the appropriate image file</filename>
    <size>
        <width>2707</width>
        <height>3828</height>
        <depth>1</depth>
    </size>
    <segmented>0</segmented>
    <object>
    <name>category of object</name>
        <bndbox>
            <xmin>0.13591964</xmin>
            <xmax>0.14452284</xmax>
            <ymin>0.2936048</ymin>
            <ymax>0.29956971</ymax>
        </bndbox>
    </object>
    <object>
    .......
    </object>
</annotation>
```

Deepscores dataset contains tens of thousand images of generated music sheets. In average, one image contains from tens to hundreds of symbols. Deepscores also contains for every image information about segmentation.

# Chapter 5

# Approaches using computer vision techniques

## 5.1 Staff recognition and staff removing

Music staff, marked lines in a music sheet, is an important part of the notation. The different position of the note in the staff specifies different pitch, whereas the note still looks the same. Besides that, staff enable players convenient and intuitive way of real-time reading of the sheet during performance. Staff provides us complete information about absolute scale. In the other words, staff is essential part of the music notation. However, for purpose of digital processing of music sheet, staff can be unnecessary. Staff removal can be even essential part of many algorithms. We examined removing staff line using Hough transform with post-processing.

### 5.1.1 Image binarization

In first step of image processing, we binarized input image based on change color space to gray-scale using threshold. We used threshold, as the simplest method of segmentation, to divide relevant part of the image from the background. After this process, we resized the image, if needed, and again applied binarization to resized image. We binarized the image twice, in order not to loose information during resizing and having binarized image as the output of this process. Result of resizing process is image in gray-scale, values from 0 to 255, however, we need to have 0 and 1 values only. See 5.1.

### 5.1.2 Staff lines finding

Once we have binarized image of the music sheet, we can begin with staff finding process. We have one strong assumption, that the input is an image of a music sheet.

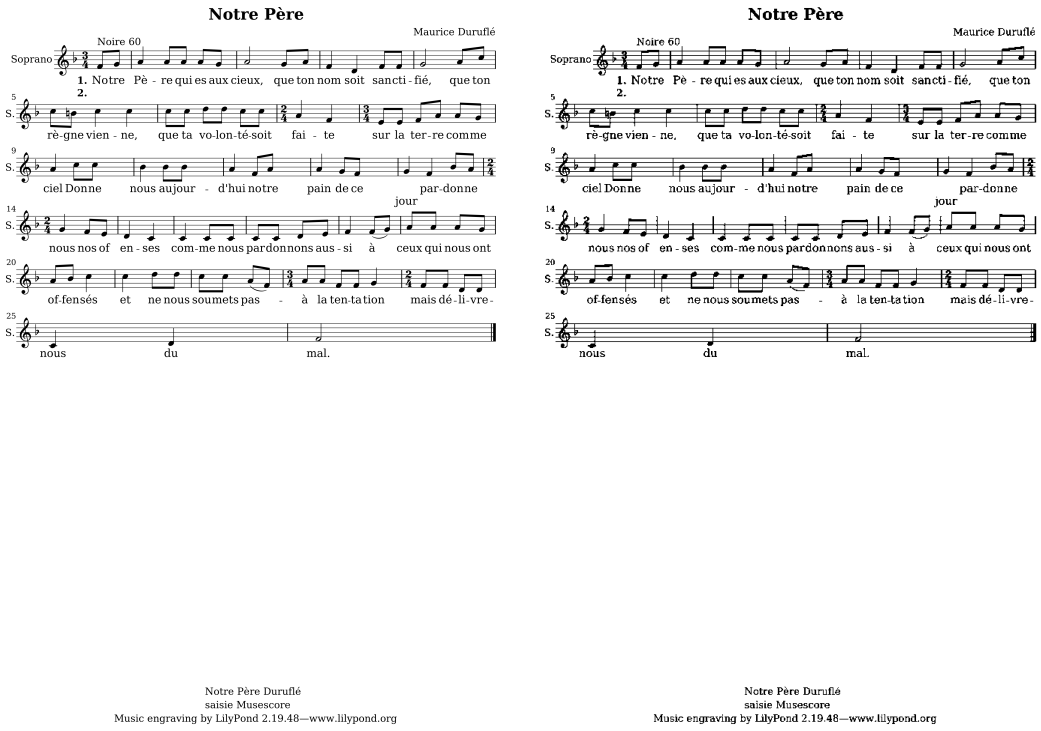



Figure 5.1: Original image [40] on the left and the image after changing to gray-scale color space and applied threshold comparison process with threshold value 161.

Every music staff line contains five visible geometric lines. We found lines using Hough transform algorithm. We select such a line, that has intersection with only of lines passing trough more than 480 relevant points on the image.

### 5.1.3 Image rotating and staff removal

When we have detected staff lines and no others, we can begin with further processing. Prerequisite of removing algorithm is having lines of the staff perpendicular with $y$-axis. In order to have the prerequisite met, we used information from accumulator of the hough line detector. All of the angles of the selected lines should be the same, however, there can be some inaccuracy in some of the line of the stuff. For having relatively right rotated image, we calculate mean angle of the lines from point $(0, 0)$ and then rotated the image around the point $(0, 0)$ by the computed angle.

Then we started with staff removal process. We began to iterate x-axis for each selected line, from Hough transform algorithm application. For each potential pixel to remove, we checked if the point is part of a straight segment consisted at least from 3 pixels. Also, we checked if no points are below or above, and there are segmented pixels right next to the point from both sides. This way we do not remove musical symbols intersecting with the line. We did this process, as well for the $y$ axis value deriving from accumulator, and also from the positions one pixel above and below the

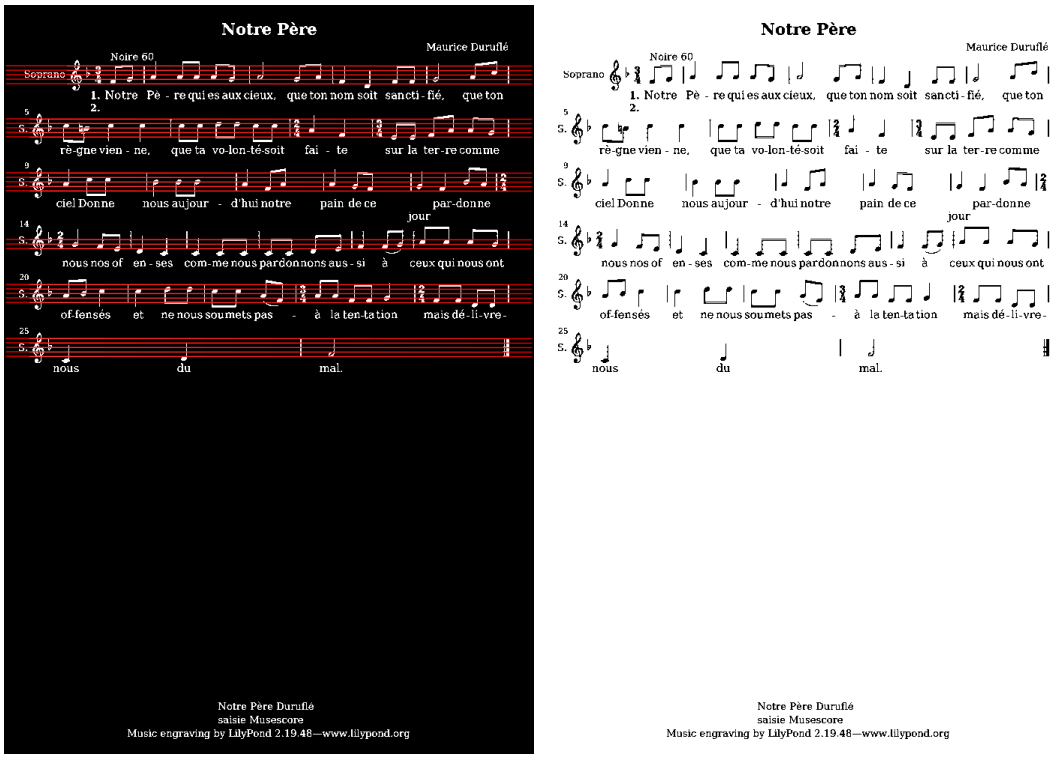



Figure 5.2: Detected staff visualisation and removed stuff in the binarized image.

$y$ values, in order to remove incorrectly rendered lines e. g. vertical shift of the line in a point. Detected and removed staff can be seen on the image 5.2.

## 5.2 Dataset augmentation

We can easily distinguish works, on the level of image processing of music sheets, to digitizing notation-software generated and to digitizing handwritten music sheets. Optical music recognition using deep learning faces problem of lack of datasets for practical use. Apart from handwritten music sheets, we want to point mainly to absence of camera photographed music sheets, generated by music notation program in well defined format. The are well labeled and segmented datasets with huge amount of data, for example [6], however many are just generated as clear digital images. Methods for adding noise to the images [15] would not have real enough result. That could be problematic for training deep model in order to work on real samples. We used another approach to this problem. Instead of attempt to make the software generated music sheets as real photographed as best we can, we used real photos of the music sheet. The photos was processed and annotated based on the original generated music sheets in digital form. We have not enough sources for creating dataset numerous enough for practical use, but we have created the method and demonstrated it on some samples.

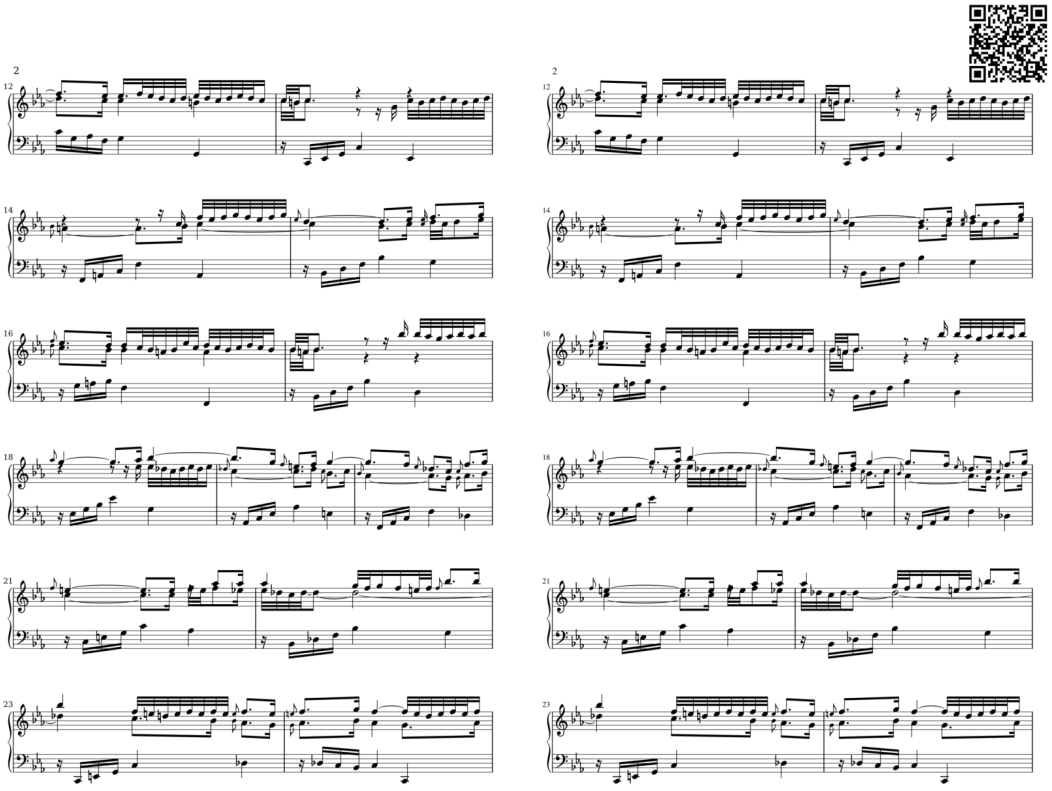


Figure 5.3: Original and extended music sheet with generated QR code.

### 5.2.1 Image labeling

In order to have the process of creating dataset automated as most as possible, we started with labeling the generated digital images. We extended the image and generate QR [54] code to the extended place. The file name of the digital image was encrypted to the QR code. See the example of original and qr-extended image on figure 5.3.

### 5.2.2 Camera calibration

Digital cameras, from the principle of working, have presence of distortion [56]. The taken pictures can be distorted, for example, in a form of fish-eye, or inverse fish eye caused by radial distortion. Although some commercial cameras have this problem partially solved and calibrated. However the motivation of manufacturer calibration is mainly aesthetic look of the image rather than full calibrated cameras taking non distorted images. For having all the images on the same level of distortion, possibly taken from multiple different devices, we need to further calibrate the camera on every device. The used method is derived from official OpenCV tutorial about camera calibration[1].

[1]OpenCV. Camera calibration tutorial:
https://docs.opencv.org/3.4/d4/d94/tutorial_camera_calibration.html

### Distortion

We will took into account two types of distortion, radial and tangential distortion. Radial factor of the distortion can be computed as following:

$$x_{distorted} = x(1 + k_1 r^2 k_2 r^4 k_3 r^6)$$

$$y_{distorted} = y(1 + k_1 r^2 k_2 r^4 k_3 r^6)$$

for some distortion coefficients $k_1, k_2, k_3$, where the point $(x_{distorted}, y_{distorted})$ is a point in distorted image of the point $(x,\ y)$ in image with removed distortion. The $r$ is euclidean distance of $(x_{distorted}, y_{distorted})$ and distortion center. Formula for tangential distortion coefficients:

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$

$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

for some $p_1, p_2$ and $r$, euclidean distance of $(x_{distorted}, y_{distorted})$ and distortion center. Our goal is to find distortion coefficients $k_1, k_2, k_3, p_1$ and $p_2$. For the values conversion to get camera focal length we use following formula:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

where $x', y'$ are coordinates on the distorted image, $x, y$ are coordinates on the undistorted image, $z = z'$ is the extension for homography coordinate system, $f_x, f_y$ are the camera focal lengths and the point $(c_x,\ c_y)$ is the point on the image, appropriate to the point of the optical center. Matrix containing $f_x, f_y, c_x,\ c_y$ is known as camera matrix.

### Coefficient findings and calibration

For the purpose of finding camera coefficients, we used open source library for computer vision, OpenCV [13]. The program was finding coefficients processing bunch of images of $9 \cdot 6$ chessboard printed on paper of A4 size and captured by the camera. It is worth to mention, that found focal length of used camera was $3.9079568780154264 \cdot 10^3$ pixel units, same for both dimensions. As well as we have camera matrix and distortion matrix, one dimensional matrix of distortion coefficients, we can remap captured images.

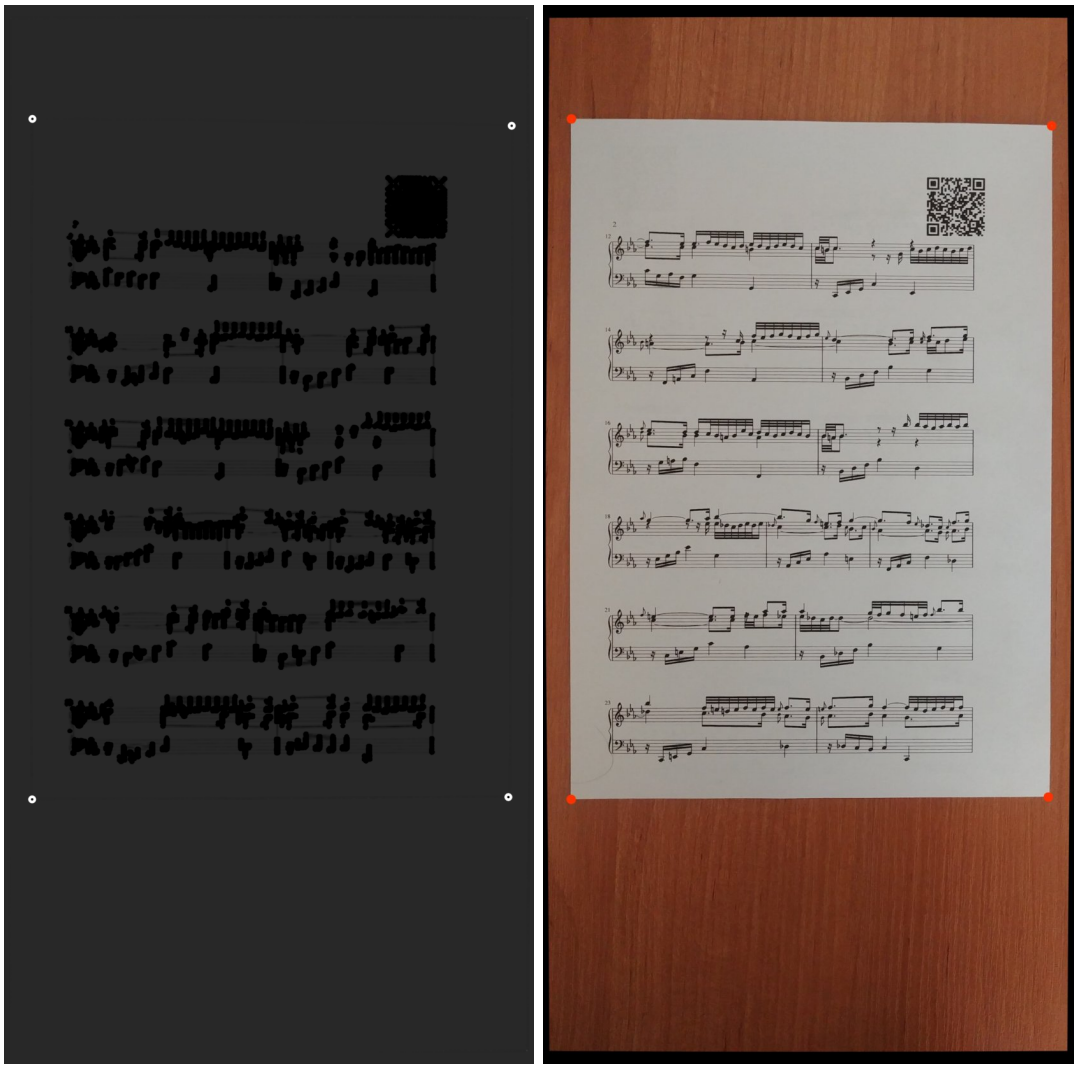


Figure 5.4: Corner found using Harris's corner detector method [29] and on the right, selected corners of the A4 printed music sheet.

### 5.2.3 Corner findings and image transforming

We have found corners on the captured image extended with generated QR code from the first part using Harris corner detector [29]. We used block size value 9, aperture 5 and free parameter of value 0.035. Parameters could be needed to adjust based on the given conditions, amount of light during image capturing, color and material of the background etc. Possible corners detected on the edges of the picture are ignored. We assumed, that the center of the captured image is inside the printed music sheet, the music sheet is on portrait orientation, as well as the captured image. Te last assumption is, the image is rotated on the surface by no more than 22.5 deg. We considered four parts of the image based on the center of the captured image. We found corner of A4 paper using finding of the most deviated corner in every of the part, top left, top right, bottom left and bottom right. See the image 5.4. Using the found corners of the

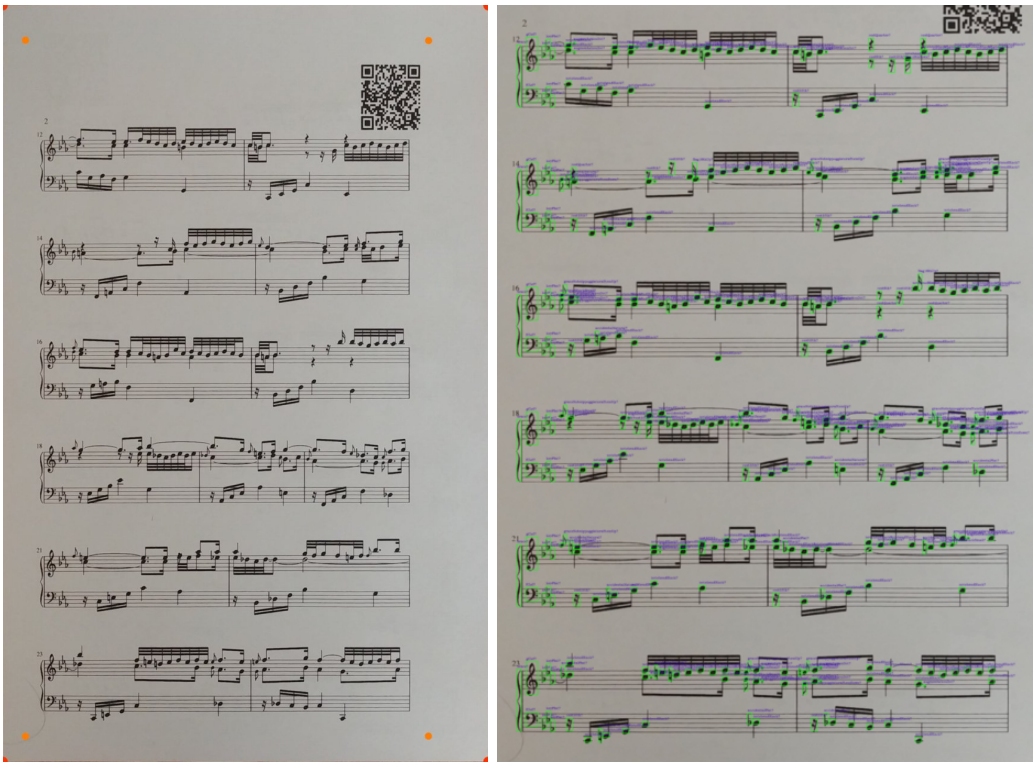


Figure 5.5: Image in the phases of processing after perspective warp of A4 sheet found on the image. The part of the QR code could be deleted, smaller in order to fit to the extending part or also height of the extending part can be increased.

sheet, we did perspective warp of the image using corners of the sheet as new corners of the image with size as the original, QR extended, digital document. Then we have to remove margins caused by the printer settings and its limitation. We used QR code corner points for this purpose. In comparison with QR code in original document, we could easily find scale factor between these two images for every dimension. Next, we can find points in processed image to be used as the new corners of a cropped image, resized to the same size, the size of original extended document. At the end, we need to remove augmented part of the image, in order to match the image exactly with the original image before extending process. As original image is annotated, we can now use this labels from annotation for our processed, camera captured, image. See the example on figure 5.5.

# Chapter 6

# Approaches using CNN

Neural networks experienced fast progress in many areas of computer vision. In the field of optical music recognition are neural networks used for recognizing hand written music sheets as well as for software generated printed music. There exists approaches differing in image prepossessing and using of neural network [5], [14], [6]. We looked at usage convolutional neural network as naive detector. Then we took existing models and detectors and examined their usage for detecting numerous music symbols on an image.

## 6.1 CNN as sliding window

We examined utilization of small model of convolutional neural network as naive detector. In the process of detecting we move on the input image with a window o certain size. The content of the windows is then used as an input to the CNN model for classification. Multiple windows of different sizes are used. The biggest window is is the same dimensions as the input layer of the CNN model. Since we worked with binarized images, missing area to the input from smaller windows will by filled by zeros.

| | | | | | | |
|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Treble clef | Bass clef | Alto clef | Sharp | Double sharp | Flat | Natural |

Table 6.1: Handwritten music symbol from the dataset [25]

### 6.1.1 Data prepossessing

We used small dataset of hundreds of handwritten music symbols [25], see table 6.1. We had available separate categorized symbols, as well as whole notation of some scores. Along with 7 categories of symbols, we have created one new category of everything but no symbols from previous categories. For this purpose we manually deleted the symbols from multiple scores. After that, we randomly cropped bunch of cut-outs from the scores, in sizes of all used windows. This way we created 701 samples of new category. Number of samples is similar to other categories.

### 6.1.2 Model

We have created simple and generalizing enough model of convolutional neural network. Architecture of the network is following:

| Input layer of the network, size 200x200 |
|---|
| Batch normalization |
| Convolutional layer, 90 filters, kernel size 10x10, L2 regularization |
| LeakyReLU and Pooling 4,4 layers |
| Dropout 0.05 and Batch normalization layers |
| Convolutional layer, 80 filters, kernel size 5x5 |
| LeakyReLU, Dropout 0.05 and Batch normalization layers |
| Convolutional layer, 80 filters, kernel size 5x5, L2 regularization |
| LeakyReLU, Dropout 0.1 and Batch normalization layers |
| Convolutional layer, 80 filters, kernel size 5x5, ReLU activation function |
| Pooling 2,2 and Batch normalization layers |
| Convolutional layer, 90 filters, kernel size 3x3 |
| LeakyReLU, Flatten and Batch normalization layers |
| Fully-connected layer, 100 neurons, L2 regularization, ReLU activation function |
| LeakyReLU, Dropout 0.1 and Batch normalization layers |
| Fully-connected layer, 80 neurons, ReLU activation function |
| LeakyReLU, Dropout 0.1 and Batch normalization layers |
| Fully-connected layer, 80 neurons, L2 regularization, ReLU activation function |
| Dropout 0.1 and Batch normalization layers |
| Fully-connected layer, 70 neurons, ReLU activation function |
| LeakyReLU layer |
| Output layer - classification, 8 neurons, softmax activation function |

In order not to have the model generalizing and not over-fitting during a training, we added multiple types of normalization as well as dropout layers. Dropout layer

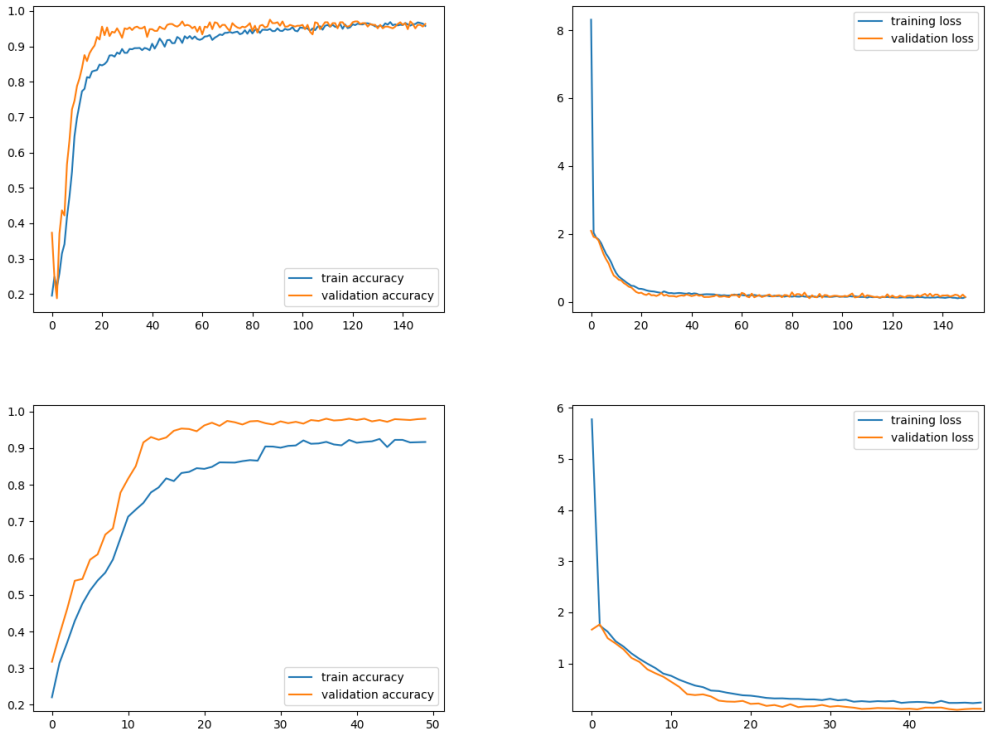



Figure 6.1: Training and validation accuracy and loss. First two images describe the whole training of 150 epochs. The next two images are from the same run, detailed to first 50 epochs of the training process.

temporarily remove connection of randomly chosen neurons. The amount of neurons to choose is optional.

### 6.1.3 Training

With completed and prepared dataset, we started with training process. During training we used image generator to augment the dataset. We used methods of augmentation like deleting mean of the image, vertical and horizontal shift, zooming the image and image rotation. The model quickly began to have training and validation accuracy over 90%. Thanks to the strong effort for generalization, validation accuracy is slightly higher than training accuracy in firsts epochs. See figure 6.1.

### 6.1.4 Detecting symbols and post processing.

In order to detect symbols on the input image of handwritten music sheet, we moved by the image with a window with a step of 32 pixels. The window was of the size 200x200, after first iteration over the image, the original size was scaled down, and rounded if needed, by $\{2, 3, 4, 5, 6\}$ for every iteration over the image. This way we

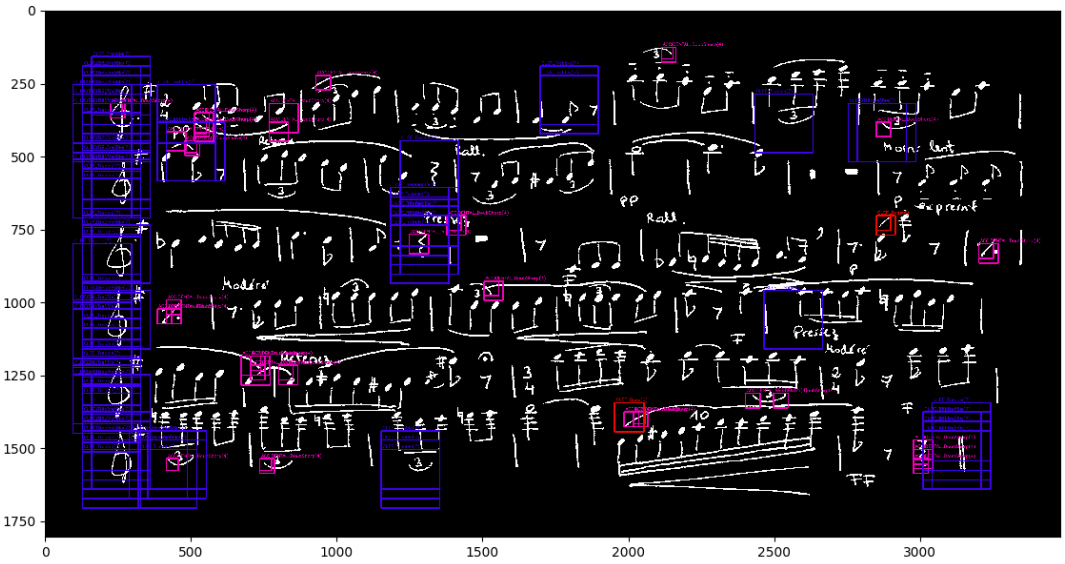



Figure 6.2: Result of sliding window detecting process using content of the window as input to the CNN.

got multiple detection of the same symbol by mutual slightly shifted windows. See the figure 6.2. For dealing with detecting conflicts, we need to further process the image. We looked at every detection coordinates and removed all of the conflicting and worse placed detection coordinates. Thus we removed all of the conflicts, and as a result the best fitted rectangle remained. See the result of this process on figure 6.3.

### 6.1.5 Possible improvements

There is a lot of possibilities for future research. For example, instead of filling the missing part of the smaller windows with zeros, the image can be scaled to fit shape of the input layer of the network. The architecture of the network can be modified in any way. Approach of naive CNN classifier can be tested on multiple varied data-sets with many more samples.

## 6.2 FasterRCNN

In this section we examined behaviour of the existing detector. We used FasterRCNN [49] for this purpose. The detector, as well as many other top detectors of state-of-the-art, is developed for detecting images with captured scenes from daily lives. Usual approach is to detect maximum tens of object in one image. However, we have different scenario, relatively noise-free and well defined pictures containing hundreds of small symbols.

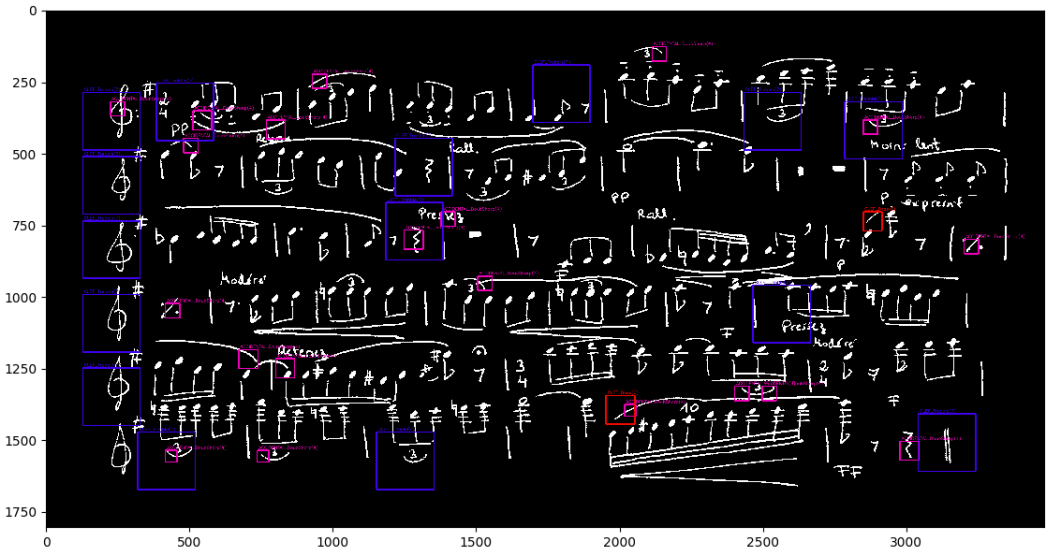



Figure 6.3: Image detection with naive CNN detector after removing conflicts of detection.

### 6.2.1 Data preparation

We used deepscores dataset [40] for FasterRCNN training. Due to a large dimension of images of music sheets, and hundreds of labeled musical symbols, training could be impractical and long lasting. We used Resnet50 [30] model as a base network, which is not adapted for our goal. Therefore, we split original images of 3000x4000 pixels to 300x300 cut-outs with overlaps of 100 pixels. his way we get many times larger dataset, with a few labeled objects in one image.

### 6.2.2 Training

We trained regional base network together with second stage detector. We used keras [28] implementation of FasterRCNN[1]. Training was divided to 250 epochs of 1000 samples. For the examples of FasterRCNN model predictions, see 6.3.

### 6.2.3 Validation

Since modified dataset consist of cropped images with mutual overlaps, we do not consider predictions on the edges. We only compare predictions and ground truth bounding boxes 20 pixels from the edges of the image. This way we mark for every pixel its ground truth and prediction. See the results on table 6.2

[1]https://github.com/kentaroy47/frcnn-from-scratch-with-keras

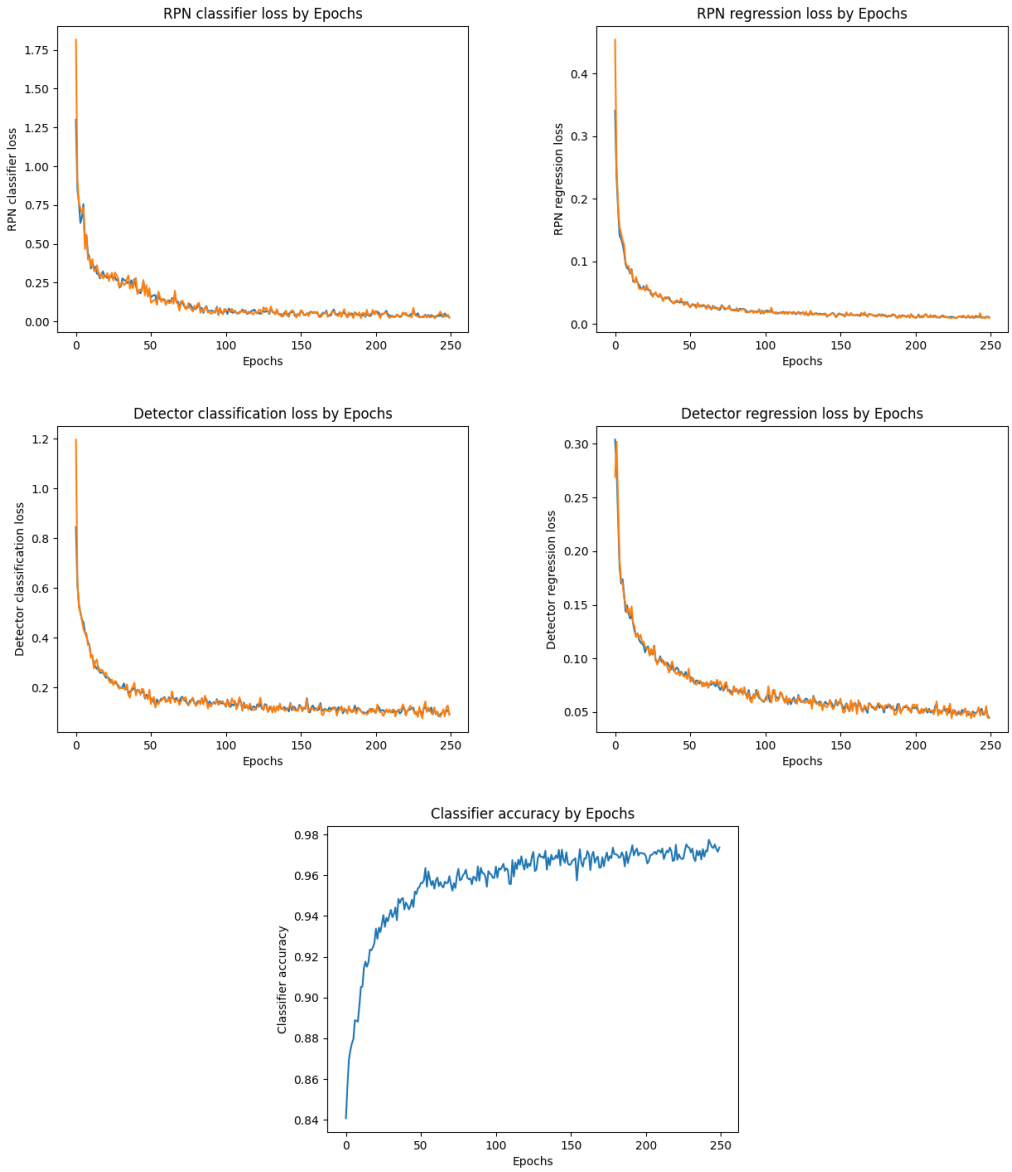



Figure 6.4: Training accuracy, training and validation loss of training process of Faster-RCNN detector.

| | |
|---|---|
| Ground truth: | 30170093 |
| Predicted: | 6396880 |
| Intersect: | 554975 |
| True intersect: | 293215 |

Table 6.2: Validation of the FasterRCNN trained model. Only bounding boxes twenty pixels from the edges were counted, both for the predictions an for the ground truth.

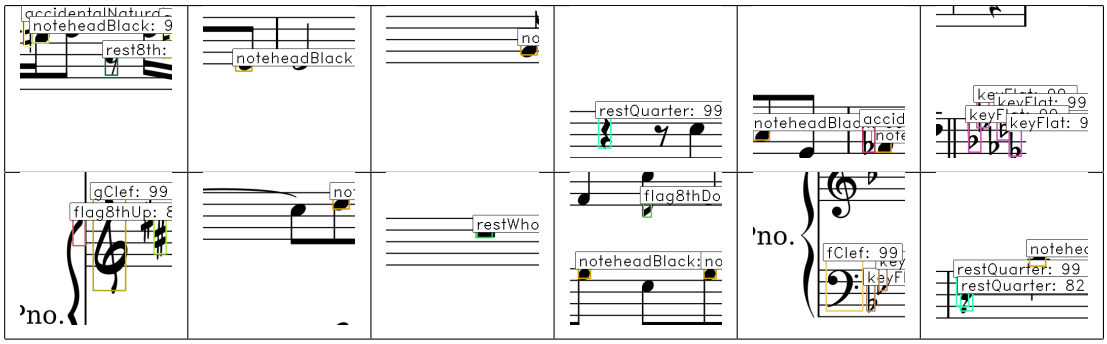



Table 6.3: Examples of the FasterRCNN trained model used on validation set.

## 6.3 YOLO

The last, one-stage detector, which behaviour on music scores we examined, was You Only Look Once (YOLO) [34] detector. YOLO detector is fast, widely used for real time detection. If YOLO detector recognize well music scores, its speed of recognition, which was in our case tenths of a second, would have many practical uses. At least, speeding the processes is always welcome.

### 6.3.1 Data preparation and training

From the principles of used YOLO architecture [12], we needed to have images the size of multiple of 32 for both dimensions. We decided to process the same Deepsocres dataset [6] the same way as for FasterRCNN model, except the size of cut-outs, which was 640x640. We statistically found expected dimensions of bounding boxes, whose number is of a similar order of magnitude of the categories. These dimensions was used in declaration of network in order to speed up training of the model. Training took up to 18000 epochs witch batch size of 64 samples. The training results was far beyond of 40% loss, we do not add any further result to the training.

### 6.3.2 Validation

Although training results denote uselessness, we did interesting observation, that many of the predictions are False positives. True positives from the predictions are in many cases covering most of the ground truth bounding boxes, false positives are many times of the wrong size of the declared category of the object. The expected dimensions of the bounding boxes of the objects, can be easily used for post processing of the predictions and filtering false predictions. The expected dimensions of bounding boxes can be statistically found as in the previous section, see 6.3.1. However, in our case was sufficient to filter up bounding boxes whose corners are not inside the image. For the results of the validation, realized the same way as in Faster RCNN section, see

| | |
|---|---|
| Ground truth: | 31321050 |
| Predicted: | 41158772 |
| Intersect: | 29938366 |
| True intersect: | 29104280 |

Table 6.4: Validation of the YOLO trained model. Only bounding boxes inside the image were counted, both for the predictions and for the ground truth.

| | FasterRCNN | YOLO |
|---|---|---|
| Image size | 300x300 | 640x640 |
| Average prediction time | 0,236401824s | 0,065157548772s |
| Average Precision | 0,045837189 | 0,822294151 |
| Recall | 0,00971873 | 0,929224276 |
| Intersection over Union | 0,008142148 | 0,684139255 |

Table 6.5: Comparison of FasterRCNN and YOLO detectors. Both models tested on the same, NVIDIA TESLA T4, hardware.

table 6.4. There is a strong expectation of higher precision rate with continuing of the training process.

## 6.4 Summary

We examined two detectors by training on images of music sheets. None of them was adjusted to our use case any way. Comparison of the detectors should be perceived with caution. Although initial dataset was the same, for the implementation reasons was created two different datasets. That means, validations datasets was different, too. There is slight difference in number of samples of the validations sets.

### 6.4.1 Results

Training of both models lasted several days for each on relatively powerful hardware. YOLO detector had more false positives than FasterRCNN. However, after filtering predictions with additional information of known area of music symbols, YOLO showed up as significantly better than FasterRCNN based on used metrics. For the comparison of these two detectors see table 6.5.

The problem of small recall together with intersection over union of FasterRCNN model could be due to many cases of non detecting musical symbol, which is exactly the opposite of trained model of YOLO detector. Next, low precision rate, which is computed only from predicted areas, can be the result of huge ratio of bounding

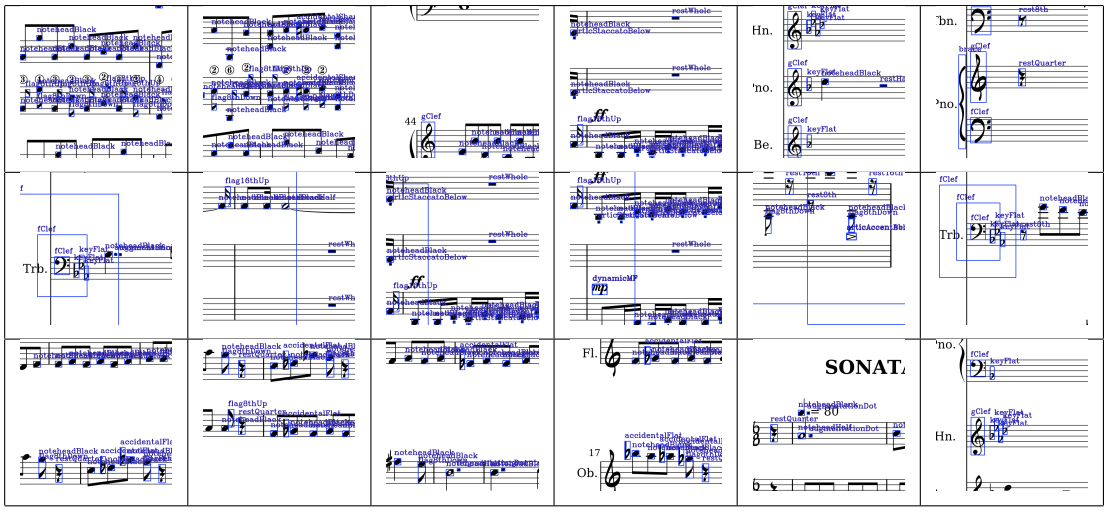


Table 6.6: Examples of the YOLO trained model used on validation set. First column consists of filtering predicted bounding boxes exceeding area of the image. Second columns presents images with unfiltered predictions, note that many predicted bounding boxes exceed the image by every side. On the last column can be seen examples of the filtered predicted bounding boxes exceeding twenty pixel from borders of the image by its any side.

box shifts against ground truth. However, shifted windows still could be acceptable, provided with possibility to correctly detect musical height of the note.

### 6.4.2 Comparison

We present comparison with other work [6] dedicated to object detection of printed and handwritten musical symbols. We look at the case where DeepScores dataset [40] is used. Result are shown in the table 6.7

### 6.4.3 Improvements

Simple heuristic in this case be used. For example, let $c$ be the size of line connecting center point of the bottom edge of the predicted bounding box with the center of the predicted bounding box. Let $h$ be the height between two lines of the staff, the same as note height. Next, let $c'$ be the half of the height of the ground truth bounding box. Then, if $|c - c'| < \frac{h}{2}$, musical position of the note can be easily determined. In the case of outlined heuristic we do not care about the horizontal shift of the predicted bounding boxes, due to a principles of music sheet notation, but the horizontal shift is counted in used metrics. Another approach of post processing, taking into account possible shifts of predicted bounding boxes against ground truth, can be comparison of the center point of the predicted bounding box with the two closest lines of the staff. Precision of the FasterRCNN model, containing predictions that are still useful, i. e.,

Our work

| Model | Average Precision (%) |
|---|---|
| YOLO | 82.2 |
| FasterRCNN | 4.6 |
| FasterRCNN* | 52.8 |

Work of A. Pacha et. al.

| Model | Average Precision (%) |
|---|---|
| FasterRCNN | 19.6 |
| RetinaNet | 9.8 |
| U-Net | 24.8 |

Table 6.7: Comparison of models with work of A. Pacha et. al. [6] on the DeepScores [40] dataset. YOLO results are computed after filtering predicted bounding boxes that exceed the image. Precision of FasterRCNN* is based on usefulness of predicted bounding boxes of trained FasterRCNN model, i. e. we can correctly identify the music symbol and its musical notation. For more see subsection 6.4.3.

we can correctly identify the music symbol and its musical notation, is 52.8339114%.

# Conclusion

To summarize, we looked at actual state in Optical Music Recognition research field. We outlined the difficulty of digitizing music sheets and mentioned many approaches in used on many levels of music sheets recognition. There are variety of application of methods from various fields. Closer focus was taken to applications of neural networks in recognition of music symbols. Due to difficulty of recognizing empirically defined handwritten symbols, recognition of handwritten music became the next heading of Optical Music Recognition. Next, we described principles of working essential algorithms in computer vision, such as detecting lines using Hough transform, edges by Canny edge detector and corners using Harris corner detector. After that we looked at convolution in convolutional layer, which brings us many applications in neural networks, mainly in computer vision. We explained differences between classification, detection and segmentation.

We introduced object detectors and outlined theirs architecture and main principles of working. After that, we introduced method of removing music staff using Hough transform and heuristics. The method could be modified any way and can have variety of uses, not only for musical sheets. We came up with the method of dataset of photographed sheets creation using labeled digital image sheets. Instead of efforts to reliably simulate real photography noise on digital image, we labelled music symbols in real, photographed, images. In addition, as in the previous case, this method can be used for variety of dataset types creation, for solving problems in many areas.

Since datasets are pillar of learning of neural network models, we looked at two of most used datasets in the area of computer vision, MS COCO [37] and PASCAL VOC [24] and well known dataset in the field of optical music recognition, DeepScores [40]. The last, we trained two detectors, FasterRCNN [49] and YOLO [34], on DeepScores dataset. Average precision of FasterRCNN model seems to be worthy of rejections, however, after taking structure of musical sheet structure and music staff into account, model precision of useful predictions is above 52%. We managed to train YOLO model to be more usable than FasterRCNN, with precision over 82%. Even though in YOLO case, we needed to filter out certainly wrong predictions. This information is known thanks to the structure of music sheet. Thus, we showed, that music scores could be recognized precisely and quickly, within a tenths of a second.

# Bibliography


[1] *Commons.wikimedia.org File:Valve gaussian.* WikiMedia Commons, Jan 2008.

[2] *Commons.wikimedia.org File:Valve monochrome.* WikiMedia Commons, Jan 2008.

[3] *Commons.wikimedia.org File:Valve original.* WikiMedia Commons, Jan 2008.

[4] *Commons.wikimedia.org File:Hough transform diagram.* WikiMedia Commons, Nov 2018.

[5] Jorge Calvo-Zaragoza Alexander Pacha. Optical music recognition in mensural notation with region-based convolutional neural networks. *Conference: 19th International Society for Music Information Retrieval Conference at Paris*, 2018.

[6] Jr. Jorge Calvo-Zaragoza Alexander Pacha, Jan Hajič. A baseline for general music object detection with deep learning. *Applied sciences*, 2018.

[7] Filipe Paszkiewicz Andre R. S. Marcal Carlos Guedes Ana Rebelo, Ichiro Fujinaga and Jaime S. Cardoso. Optical music recognition: state-of-the-art and open issues. © *Springer-Verlag London Limited*, 2012.

[8] Ciampa A Andronico A. On automatic pattern recognition and acquisition of printed music. *Proceedings of the international computer music conference, Venice*, page 245–278, 1982.

[9] Paul M Baggenstoss. A modified baum-welch algorithm for hidden markov models with multiple observation spaces. *IEEE Transactions on speech and audio processing*, 9(4):411–416, 2001.

[10] Nesi P Bellini P, Bruno I. Assessing optical music recognition tools. *Comput Music J 31:68–93*, 2007.

[11] Josef Bigun. Optimal orientation detection of linear symmetry, 1987.

[12] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[13] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[14] J. Calvo-Zaragoza and D. Rizo. End-to-end neural optical music recognition of monophonic scores. *Applied Sciences*, 2018.

[15] Jorge Calvo-Zaragoza and David Rizo. Camera-primus: Neural end-to-end optical music recognition on realistic monophonic scores. In *ISMIR*, pages 248–255, 2018.

[16] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325, 2015.

[17] Retif B Coüasnon B. Using a grammar for a reliable full score recognition system. *International computer music conference, Banff, Canada*, page 187–194, 1995.

[18] Prerau D. Computer pattern recognition of standard engraved music notation. phd thesis. *Massachussets Institute if technology, Dept of Engineering and Computer Science*, 1970.

[19] Pruslin D. Automatic recognition of sheet music. phd thesis, massachussets institute of technology. 1966.

[20] Karl Tombre David Doermann. *Handbook of Document Image Processing and Recognition.* © Springer-Verlag London, 2014.

[21] Lijun Ding and Ardeshir Goshtasby. On the canny edge detector. *Pattern Recognition*, 34(3):721–725, 2001.

[22] M. Droettboom. Correcting broken characters in the recognition of historical printed documents. *In Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries, Houston, Texas*, page 364–6, 2003.

[23] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[24] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[25] Alicia Fornés, Josep Lladós, and Gemma Sánchez. Old handwritten musical symbol classification by a dynamic time warping based method. In *International Workshop on Graphics Recognition*, pages 51–60. Springer, 2007.

[26] Taubman G. Musichand: a handwritten music recognition system, technical report. 2005.

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, Jan 2016.

[28] Antonio Gulli and Sujit Pal. *Deep learning with Keras.* Packt Publishing Ltd, 2017.

[29] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[30] Ren S. Sun J. He K., Zhang X. Deep residual learning for image recognition. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, CA, USA, 26 June–1 July*, page 770–778, 2016.

[31] J. Pokorný J. Novotný. Introduction to optical music recognition: Overview and practical challenges. *Proceedings of the Dateso 2015 Annual International Workshop on DAtabases, TExts, Specifications and Objects, vol. 1343, pp. 65–76. CEUR-WS*, 2015.

[32] Marta Kolárová Jorge Calvo-Zaragoza Jan Hajič, Alexander Pacha. How current optical music recognition systems are becoming useful for digital libraries. *Conference: the 5th International Conference*, 2018.

[33] Alexander Pacha Jorge Calvo-Zaragoza, Jan Hajič Jr. Discussion group summary: Optical music recognition. *12th IAPR International Workshop, GREC 2017, Kyoto, Japan, November 9-10*, 2018.

[34] Ross Girshick Ali Farhadi Joseph Redmon, Santosh Divvala. *You Only Look Once: Unified, Real-Time Object Detection.* 2016.

[35] Ng K. Music manuscript tracing. blostein d, kwon y-b (eds) graphics recognition algorithms and applications. *Volume 2390 of lecture notes in computer science, Springer, Berlin Heidelberg*, page 330–342, 2002.

[36] Pugin L. Optical music recognition of early typographic prints using hidden markov models. *Proceedings of the International Society for Music, information retrieval*, page 53–56, 2006.

[37] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[38] Girshick R.B. He K.-Dollár P. Lin T., Goyal P. Focal loss for dense object detection. *arXiv:1708.02002*, 2017.

[39] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[40] Jurgen Schmidhuber Marcello Pelillo-Thilo Stadelmann Lukas Tuggener, Ismail Elezi. *DeepScores – A Dataset for Segmentation, Detection and Classification of Tiny Objects*. ICPR, 2018.

[41] Hashimoto S Matsushima T, Ohteru S. An integrated music information processing system. *Proceedings of the international computer music conference, Columbus*, pages 191–198, 1989.

[42] Ajmal S. Mian, Mohammed Bennamoun, and Robyn Owens. Keypoint detection and local feature matching for textured 3d face recognition. *International Journal of Computer Vision*, 2008.

[43] Felipe Pezoa, Juan L. Reutter, Fernando Suarez, Martín Ugarte, and Domagoj Vrgoč. Foundations of json schema. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 263–273, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee.

[44] Burgoyne JA Fujinaga I Pugin L, Hockman J. Gamera versus aruspix. two optical music recognition approaches. *Proceedings of the 9th international conference on music information retrieval, Philadelphia, USA*, page 419–424, 2008.

[45] Fujinaga I Pugin L, Burgoyne JA. Map adaptation to improve optical music recognition of early music documents using hidden markov models. *Proceedings of the 8th International Society for Music, information retrieval*, page 513–516, 2007.

[46] Goecke R. Building a system for writer identification on handwritten music scores. *Proceedings of the IASTED international conference on signal processing, pattern recognition, and applications. Acta Press, Anaheim*, page 205–255, 2003.

[47] Goecke R. Robust and adaptive omr system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP J Appl Signal Process*, 2007.

[48] Cardoso J Rebelo A, Capela G. Optical recognition of music symbols. *Int J Doc Anal Recognit*, page 13(1):19–31, 2010.

[49] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[50] Girshick R. Sun J. Ren S., He K. Towards real-time object detection with region proposal networks. *In Advances in Neural Information Processing Systems 28, Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA*, page 91–99, 2017.

[51] Samy Bengio Ronan Collobert. Links between perceptrons, mlps and svms. *ICML '04 Proceedings of the twenty-first international conference on Machine learning*, page 23, 2004.

[52] Brox T. Ronneberger O., Fischer P. Convolutional networks for biomedical image segmentation. in medical image computing and computer-assisted intervention. *Proceedings of the 18th International Conference, Munich, Germany, 5–9 October 2015; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Springer International Publishing: Cham, Switzerland*, page 234–241, 2015.

[53] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[54] Tan Jin Soon. Qr code. *Synthesis Journal*, 2008:59–78, 2008.

[55] Cuihong Wen, Ana Rebelo, Jing Zhang, and Jaime Cardoso. A new optical music recognition system based on combined neural network. *Pattern Recognition Letters*, 58:1–7, 2015.

[56] Juyang Weng, Paul Cohen, and Marc Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):965–980, 1992.