



## Assignment of master's thesis

<b>Title:</b>	A Tool for Digitalizing Handwritten Chess Notation Sheets
<b>Student:</b>	Bc. Jana Maříková
<b>Supervisor:</b>	Ing. Karel Klouda, Ph.D.
<b>Study program:</b>	Informatics
<b>Branch / specialization:</b>	Knowledge Engineering
<b>Department:</b>	Department of Applied Mathematics
<b>Validity:</b>	until the end of summer semester 2021/2022

### Instructions

The thesis aims to build a tool that helps chess players transform their handwritten chess notation sheets to digitalized form (ideally to the Portable Game Notation (PGN)). This tool should automate the transformation as much as possible, using computer vision (OCR) and machine learning techniques.

The tool with a web UI should support the following process:

- Upload a photo with a handwritten chess notation sheet.
- Detect the sheet on the photo and transform it into a rectangle with no background.
- Use OCR tools to recognize handwritten chess moves notation. Highlight non recognized moves and enable the user to correct them manually.
- Save the final result in the PGN format.





**FACULTY  
OF INFORMATION  
TECHNOLOGY  
CTU IN PRAGUE**

Master's thesis

# **A Tool for Digitalizing Handwritten Chess Notation Sheets**

*Bc. Jana Maříková*

Department of Applied Mathematics  
Supervisor: Ing. Karel Klouda, Ph.D.

May 6, 2021



---

## **Acknowledgements**

I would like to express gratitude to my supervisor Ing. Karel Klouda, Ph.D. for his motivation and advice. His guidance helped me in both implementing solution and writing this thesis. Last but not least, I would like to thank my family for their support during my study.



---

## Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No.121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 6, 2021

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2021 Jana Maříková. All rights reserved.

*This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).*

### **Citation of this thesis**

Maříková, Jana. *A Tool for Digitalizing Handwritten Chess Notation Sheets*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021. Also available from: <http://marikja7.pythonanywhere.com/>.



---

## Abstrakt

Tato práce se zabývá implementací nástroje na konverzi šachového partiáře do digitální podoby za pomoci OCR a technik strojového učení. Šachový partiář je dokument, do kterého zapisuje hráč své a soupeřovy tahy v průběhu partie. Nejdříve je popsána šachová terminologie a prozkoumána existující řešení. Poté jsou prozkoumány metody obecného OCR systému a na závěr je popsáno implementované řešení společně s jeho vyhodnocením.

**Klíčová slova** OCR, šachy, šachový partiář, PGN, algebraická notace, CNN

---

## Abstract

This thesis aims to create a tool that would automate converting a photo of a chess score sheet into digitalized form with the help of OCR and machine learning techniques. The score sheet is a paper document where players write down their and opponents' moves. First of all, the chess terminology and existing solutions are introduced. Then the description of a general OCR system is stated, and, finally, the implementation of the system and its evaluation are given.

**Keywords** OCR, chess, score sheet, PGN, algebraic notation, CNN



---

# Contents

<b>Introduction</b>	<b>1</b>
<b>1 Short Introduction to Chess</b>	<b>3</b>
1.1 Algebraic Notation . . . . .	3
1.2 Score sheet . . . . .	5
1.3 PGN format . . . . .	6
<b>2 Existing Solutions</b>	<b>7</b>
2.1 Checklist Based Convertor . . . . .	7
2.2 Reine . . . . .	8
2.3 CheSScan . . . . .	9
2.4 Summary . . . . .	10
<b>3 OCR</b>	<b>11</b>
3.1 Introduction to OCR . . . . .	11
3.2 Image Acquisition . . . . .	13
3.3 Preprocessing . . . . .	14
3.3.1 Image Binarization . . . . .	14
3.3.2 Noise Reduction . . . . .	15
3.3.3 Skeletonization . . . . .	16
3.3.4 Region of Interest . . . . .	16
3.4 Segmentation . . . . .	17
3.4.1 Holistic . . . . .	18
3.4.2 Explicit Segmentation . . . . .	18
3.4.3 Implicit Segmentation . . . . .	19
3.5 Feature Extraction . . . . .	20
3.5.1 Statistical . . . . .	20
3.5.2 Structural . . . . .	20
3.5.3 Global Transformations and Moments . . . . .	20
3.6 Character Classification . . . . .	21

3.6.1	Support Vector Machines (SVM)	21
3.6.2	K-Nearest Neighbors (K-NN)	21
3.6.3	Random Forests (RF)	21
3.6.4	Convolution Neural Networks (CNN)	22
3.6.5	Comparison of Classification Methods	23
3.7	Postprocessing	24
3.7.1	Manual Error Correction	24
3.7.2	Lexicon-Based Error Correction	24
3.7.3	Context-Based Error Correction	24
3.8	Existing OCR Systems	24
<b>4</b>	<b>Implementation</b>	<b>27</b>
4.1	Technology	27
4.1.1	Flask	27
4.1.2	React	27
4.1.3	PostgreSQL	27
4.1.4	Libraries	28
4.1.5	Pythonanywhere	28
4.2	System Functionality	29
4.3	Score Sheet Conversion Process	30
4.3.1	Preprocessing	31
4.3.2	Move Segmentation	34
4.3.3	Character Segmentation	35
4.3.4	Character Classification	37
4.3.5	Game Creation	40
<b>5</b>	<b>Evaluation</b>	<b>45</b>
5.1	Dataset	45
5.2	Move Segmentation	46
5.3	Character Segmentation	47
5.4	Character Classification	48
5.5	Move Prediction	50
5.6	Time Complexity	52
5.7	Possible Improvements	52
	<b>Conclusion</b>	<b>55</b>
	<b>Bibliography</b>	<b>57</b>
<b>A</b>	<b>Examples from dataset</b>	<b>63</b>
<b>B</b>	<b>Model Comparison</b>	<b>69</b>
<b>C</b>	<b>PGN game statistic</b>	<b>75</b>

<b>D</b>	<b>Examples of application</b>	<b>77</b>
<b>E</b>	<b>Acronyms</b>	<b>81</b>
<b>F</b>	<b>Contents of enclosed CD</b>	<b>83</b>



---

# List of Figures

1.1	Squares . . . . .	4
1.2	Chess pieces . . . . .	4
1.3	Score sheets . . . . .	5
1.4	Example of the PGN format . . . . .	6
2.1	Example of required score sheet . . . . .	7
2.2	Reine score sheet . . . . .	8
2.3	Score sheet converted into PGN . . . . .	8
2.4	CheSScan . . . . .	9
3.1	OCR division . . . . .	12
3.2	Different types of the image . . . . .	13
3.3	Image binarization . . . . .	14
3.4	Skeletonization . . . . .	16
3.5	Division of character segmentation . . . . .	17
3.6	Vertical projection . . . . .	18
3.7	Connected components . . . . .	19
3.8	Classification methods . . . . .	21
3.9	A neuron . . . . .	22
3.10	Comparison of classification methods . . . . .	23
4.1	System overview . . . . .	29
4.2	Score Sheet Conversion Process . . . . .	30
4.3	Implemented steps of preprocessing . . . . .	31
4.4	Perspective transformation . . . . .	32
4.5	Move segmentation – <i>score sheet A</i> . . . . .	34
4.6	Character segmentation – <i>score sheet A</i> . . . . .	35
4.7	Character segmentation – <i>score sheet B</i> . . . . .	36
4.8	LeNet-5 architecture . . . . .	37
5.1	Predicted game length subtracted from PGN game length . . . . .	46

5.2	Move length . . . . .	47
5.3	Character classification . . . . .	48
5.4	Confution matrix for char prediction . . . . .	49
5.5	Move prediction . . . . .	50
5.6	Levenshtein distance between the actual move and first-guess move	51
5.7	Time complexity . . . . .	51
A.1	Category A – <i>score sheet A</i> . . . . .	64
A.2	Moves from PGN corresponding with examples from Category A ( <i>score sheet A</i> ) and Category L . . . . .	64
A.3	Category A – <i>score sheet B</i> . . . . .	65
A.4	Moves from PGN corresponding with examples from Category A ( <i>score sheet B</i> ) . . . . .	65
A.5	Category B . . . . .	66
A.6	Category C . . . . .	67
A.7	Category L . . . . .	68
B.1	Customized LeNet-5 model summary . . . . .	70
B.2	LeNet-5 . . . . .	71
B.3	Customized LeNet-5 . . . . .	72
B.4	AlexNet . . . . .	73
C.1	Average length of move in game . . . . .	75
C.2	Length of move in game . . . . .	76
C.3	Distribution of characters in games . . . . .	76
D.1	Template upload . . . . .	77
D.2	Template upload – rotation . . . . .	78
D.3	Template upload – move area selection . . . . .	78
D.4	Template select . . . . .	79
D.5	Game conversion . . . . .	79



---

## List of Tables

4.1	Character classification – <i>score sheet A</i> . . . . .	38
4.2	Character classification – <i>score sheet B</i> . . . . .	39
4.3	Candidate moves – <i>score sheet B</i> . . . . .	39
4.4	Candidate moves – <i>score sheet A</i> . . . . .	41
4.5	Game creation – <i>score sheet A</i> . . . . .	42
4.6	Game correction – <i>score sheet B</i> . . . . .	43



---

# Introduction

There are about fifty official chess tournaments every year in the Czech Republic. Players are obligated to write down their and opponents' moves into the paper document called a score sheet. The tournament organizers are bound to transcribe these score sheets into the computer and send them to Czech Chess Organization.

Official reasons aside, the players themselves also benefit from writing down their games being able to analyze them later.

Nowadays, the transcription into the computer is usually done with the help of some chess program which offers the graphical interface with a chessboard. The user is moving pieces, and the program is storing the moves in the standardized system called algebraic notation. These programs are usually not free.

Although there is a lot of chess software for chess players to improve their skills, only a few are created to help with converting paper score sheets into a digitalized form.

The goal of this thesis is to create program which would help with digitalizing score sheets by implementing a Character Recognition System (OCR). The OCR is a technology that converts images of handwritten or printed text into the text as represented in a given system.

The thesis consists of five sections. The basic chess terms necessary for proper understanding of this thesis are introduced in Section 1. Investigation of existing solutions follows in Section 2. The theoretical background and current state-of-art of Optical Character Recognition are described in Section 3. Finally, the implementation of the system is introduced in Section 4, and in Section 5, the implemented system is evaluated, and possible improvements suggested.



---

# Short Introduction to Chess

Basic terms and standards from chess tournaments, relevant for understanding this thesis's idea, are introduced in this chapter.

First of all, the system used to record chess moves, so-called algebraic notation, are presented in Section 1.1. Score sheets that serve for chess notation recording are described in Section 1.2. Finally, the PGN format is explained in Section 1.3.

## 1.1 Algebraic Notation

Almost every player must write down their and opponent's moves in a standard chess tournament. It serves several purposes:

- **Disputes** – Some rules depend on the number of moves or repetitions of the same position. In disputes during the game, a referee can look into a notation and restore the game.
- **Administration** – In the vast majority of standard chess tournaments, a team that organizes the tournament must transcript all games and send them in the form of PGN files to the Czech Chess Organization FIDE (the world international chess organization).
- **Analysis** – The overwhelming majority of players store their games in a chess database to replay and analyze them later to learn from their mistakes.

Over time, various approaches to record moves have been used, but the only one that lasts is the syntax system called algebraic notation, which has become standard.

In general algebraic notation describes a piece that moves and a square where to move. Every piece has its letter abbreviation, which usually represents the first letter of its name capitalized. In chess, a pawn is not understood

## 1. SHORT INTRODUCTION TO CHESS

---

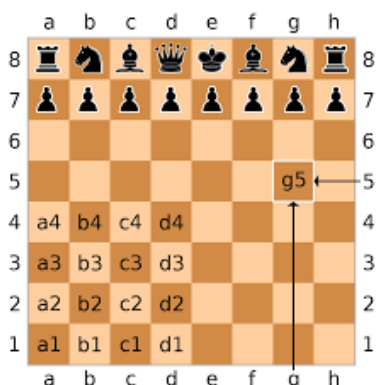


Figure 1.1: Squares

ENG	CZ
K – King	K – Král
Q – Queen	D – Dáma
R – Rook	V – Věž
B – Bishop	S – Střelec
N – Knight	J – Jezdec

Figure 1.2: Chess pieces

as a piece, so when a pawn is moving, a letter abbreviation is omitted, and only the target square is noted. Piece names and letters representing them are listed in Table 1.2 in Czech and English. Square coordinates consist of a lowercase letter from a to h representing a column and a number from 1 to 8 representing a row. Examples of square coordinates are shown in Figure 1.1. For example, move Bf6 means that bishop goes to square f6.

There are some more complex moves, which can be broken down into the following categories:

- **Captures** – When there is an opponent piece on the target square symbol x is added after the piece letter. For example Bxf6 means that bishop captures an opponent piece at square f6. When a pawn captures a piece, a column letter is written instead of a piece letter. For example exd4 means that pawn from column e captures an opponent piece at square d4.
- **Ambiguity** – When two (or more) identical pieces can move to the same target square, it is necessary to distinguish between them by adding a column letter or a row digit to specify the origin of the piece that moves. This letter/digit is added right after the piece letter. For example Jbd7 means that knight from column b moves to d7 square.
- **Castling** – There are also some special moves such as 0-0 for the king-side castling and 0-0-0 for the queenside castling.
- **Special symbols** – According to FIDE rules, it is mandatory to note a check with + symbol and checkmate with # or ++ symbols [1], but a lot of players do not do that.

The figure shows three different chess score sheet templates. Each template includes a header section for player information (names, ratings, date, round, desk) and a main table for recording moves. The middle and right sheets also include a table for recording the result of the game (win, draw, loss) and a section for the arbiter's signature and date.

Figure 1.3: Score sheets

## 1.2 Score sheet

A score sheet is a form used by players to write down their and opponents moves, usually in algebraic notation. The score sheet is usually in paper form, there are also electronic score sheets, but there are quite expensive and not approved by FIDE, which means that these score sheets can not be used in official chess tournaments. The format of the score sheet is not defined by any standard, so nearly every chess club or chess tournament presents its own version. However every score sheet must store a similar set of information. Hence the score sheet generally consists of:

- **Header part** – This part contains relevant information about players as their names and ratings, game data such as a tournament name, a round, a desk, and a date of the game.
- **Moves part** – This part contains cells with numbers. A number is a move number. Moves are written into the cells. A white piece move is recorded into the left-hand side cell, a black piece move is recorded into the right-hand cell (with the same number).

The result is written down below the moves, and both players must sign both score sheets, indicating the result of the game when the game is finished. Finally, the score sheet is handed over to an arbiter.

In Figure 1.3 some commonly used score sheets in Czech Republic are shown. Score sheets from other countries differ by a logo, number of moves per score sheet page, position of the move number (sometimes separates white and black moves) but the basic format of the moves part is always very similar.

## 1. SHORT INTRODUCTION TO CHESS

---

```
[Event "Pohár města Říčany 2020"]
[Site "Městské kulturní středisko U labutě"]
[Date "2020.08.24"]
[Round "4.3"]
[White "Le, Anh Dung"]
[Black "Petr, Jakub"]
[Result "0-1"]
[WhiteElo "1616"]
[BlackElo "2236"]
[ECO "B10"]
[PlyCount "0"]
[EventDate "2020.08.22"]
[Opening "Caro-Kann defence"]

1. e4 c6 2. b3 d5 3. Bb2 dxe4 4. Nc3 Nf6 5. Qe2 Bf5 6. O-O-O Nbd7 7. h3 Qa5
8. g4 Bg6 9. Kb1 O-O-O 10. Bg2 h5 11. f4 exf3 12. Nxf3 hxg4 13. hxg4 Rxh1
14. Rxh1 Nxc4 15. Rh4 Ngf6 16. Ra4 Qb6 17. d4 e6 18. Rc4 Kb8 19. Na4 Qc7 20. Nc5
Nb6 21. Nxb7 Nxc4 22. Nxd8 Na5 23. Ne5 Qxd8 24. Bxc6 Qb6 25. Qh2 Bd6 26. Nd7+
Kc7 27. Ne5 Nxc6 0-1
```

Figure 1.4: Example of the PGN format

### 1.3 PGN format

The abbreviation PGN stands for *Portable Game Notation*. It is the standard format for storing chess games in ASCII text files. The vast majority of chess software recognizes this format. The PGN format is also a standard way to publish chess games in books and also on the Internet (see *The Week in Chess*<sup>1</sup>). A chess tournament organizer must use this format to publish all played games. Abbreviations of chess pieces are different for different languages, so also the PGN format differs for different languages. A chess game in the PGN format is shown in Figure 1.4 in English.

---

<sup>1</sup><https://theweekinchess.com/>



# Existing Solutions

In this chapter, existing solutions are described and reviewed to see whether a new solution can bring additional value to players and/or organizers of tournaments. Criteria like functionality and useability are used for evaluation.

## 2.1 Checklist Based Convertor

This application<sup>2</sup> requires a special score sheet with preprinted symbols as seen in the demo example<sup>3</sup> and in Figure 2.1, where the example score sheet is shown. The pre-printed symbols are suitable for English notation. A user is assumed to mark off the symbols that appear in a move. The application provides a command-line interface. The only functionality that offers is to upload one score sheet. The PGN file is automatically downloaded after the score sheet upload, with no option to correct misclassified moves.

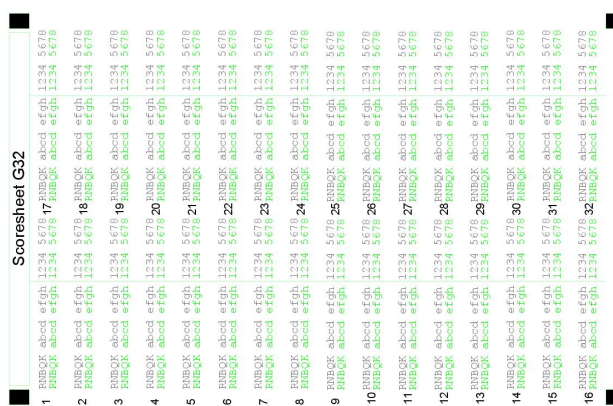


Figure 2.1: Example of required score sheet

<sup>2</sup><https://1drv.ms/u/s!AkjJ015BouadhTLLJQKkoDYbCvR6>

<sup>3</sup><https://www.youtube.com/watch?v=nc2n-UjrzpY>

## 2. EXISTING SOLUTIONS

WHITE:													BLACK:												
1	e	4			d	5							26	k	x	f	2		k	e	7				
2	e	x	d	5	q	x	d	5					27	N	e	6		k	f	6					
3	N	c	3		Q	a	5						28	N	x	d	8		k	e	7				
4	d	4			N	c	6						29	f	8										
5	N	f	3		N	f	6						30												
6	B	c	4		e	6							31												
7	B	d	2		B	e	7						32												
8	O	-	O		O	-	O						33												
9	N	b	5		Q	b	6						34												
10	B	f	4		R	d	7						35												
11	q	e	2		e	f	e	7					36												
12	R	f	e	1	k	a	j	8					37												
13	R	a	d	1	a	6							38												
14	B	x	c	7	a	a	5						39												
15	B	x	a	5	a	k	b	5					40												
16	B	x	d	8	B	x	d	8					41												
17	B	b	3		R	e	7						42												
18	B	x	b	5	h	6							43												
19	q	x	b	7	e	5							44												
20	Q	x	d	7	R	x	d	7					45												
21	d	x	e	5	R	x	d	1					46												
22	R	x	d	1	N	d	4						47												
23	N	x	d	4	N	e	4						48												
24	e	6			N	x	f	2					49												
25	e	x	f	7	k	f	8						50												

Figure 2.2: Reine score sheet

Figure 2.3: Score sheet converted into PGN

## 2.2 Reine

The idea of converting score sheets to digital form using Optical Recognition System was mentioned for the first time in a medium post by Marek Smigielski in May 2017 [2]. He did not finish the implementation of his idea, but he published source code on GitHub<sup>4</sup>. The source code makes clear that it is suitable only for a specific type of a score sheet.

Reine<sup>5</sup> is a free web application created two years later with published source code<sup>6</sup>. It is inspired by Smigielski’s work, especially by the used score sheet. The main disadvantage of this application is that a user is assumed to use a provided score sheet. This score sheet has special symbols in corners to better deal with perspective transformation and a separate cell for every letter, see Figure 2.2, and it provides bad results when characters do not fit the cells. Therefore, the application is not suitable for a general chess tournament when players are in time press and want to write down their moves as quickly as possible. The requirement to use a specific and actually unusual form of the score sheet is also very limiting.

<sup>4</sup><https://github.com/smigielski/pgn-reader-poc>

<sup>5</sup><https://www.reinechess.com/>

<sup>6</sup><https://github.com/Messier-16/Reine-Chess-Scoresheet-Scanner>



Figure 2.4: CheSScan

## 2.3 CheSScan

CheSScan<sup>7</sup> is a mobile application for Android and iOS which allows user to load the score sheet photo and it returns possible moves, which can be then downloaded as the PGN or replayed on the board. When some moves are not recognized correctly, three other possible moves to change the move are offered. Moves can also be changed by moving pieces on the board. CheSScan also offers to analyze the game by Arasan Chess<sup>8</sup>. Arasan Chess is an open-source chess engine that shows the best moves in position and the position evaluation.

The additional functionality that CheSScan offers are to load a photo of the board from a book or a screen and transform it into the FEN format which is standard for storing chess positions in chess programs.

The description does not specify suitability for various notation languages. Performed tests indicate that only English notation is supported in the current version.

Another disadvantage is that the application accepts only one score sheet. There are two situations when the user wants to convert multiple score sheets for one game. The same score sheets (score sheets from both players) for a higher quality of conversion or the case when the game was longer than supported by the score sheet and the game must be recorded on multiple score sheets. None of these options is supported.

In Figure 2.4 the examples from the CheSScan are shown.

<sup>7</sup><https://chesscan.com/>

<sup>8</sup><https://www.arasanchess.org/index.shtml>

### 2.4 Summary

Although there are numerous chess applications for game analysis, player training, and chess games databases, only three applications for converting chess score sheets were found. This fact itself sufficiently illustrates the complexity of the problem.

Two of the score sheet processing applications require a special score sheet to simplify the problem. The special score sheet is, in general, not suitable for chess tournaments, and only a few players will use them as expected. One of the score sheet processing applications (Reine) is no longer available as of 3/10/2021. Therefore, the only suitable application is CheSScan, but the usability of this application is limited by chess notation language (English).

The existence of a single usable application and in fact the application that is usable only in English-speaking countries make sufficient space for a new application. The analysis shows that such an application cannot rely on a specific score sheet format and must and able to process score sheets currently used in chess tournaments.

---

# OCR

Various approaches and current state-of-art in Optical Character Recognition are presented in this chapter. The overall introduction to OCR is described in Section 3.1. The individual steps that the general OCR system consists of are introduced in Sections 3.2 – 3.7. At the end of this chapter in Section 5.7 the existing OCR systems are explored.

## 3.1 Introduction to OCR

OCR stands for *Optical Character Recognition*. OCR is a technology that converts images of handwritten or printed text into the text as represented in a given system. It was originally meant to be used only for a printed text, nowadays it can be used for a handwritten text as well. [3]

The research went in many directions during past years. This section is based on [4] and discusses different types of OCR systems that emerged as an outcome of this research. These systems can be categorized based on image acquisition mode, character connectivity, and font restrictions.

Based on the type of input the systems can be categorized as:

- **Machine printed character recognition systems** – Systems that are used for the recognition of a text written by a computer. The complexity of a machine printed text recognition is given by a huge number of fonts that can be used.
- **Handwriting recognition systems** – Systems that are used for the recognition of handwritten texts.

### 3. OCR

---

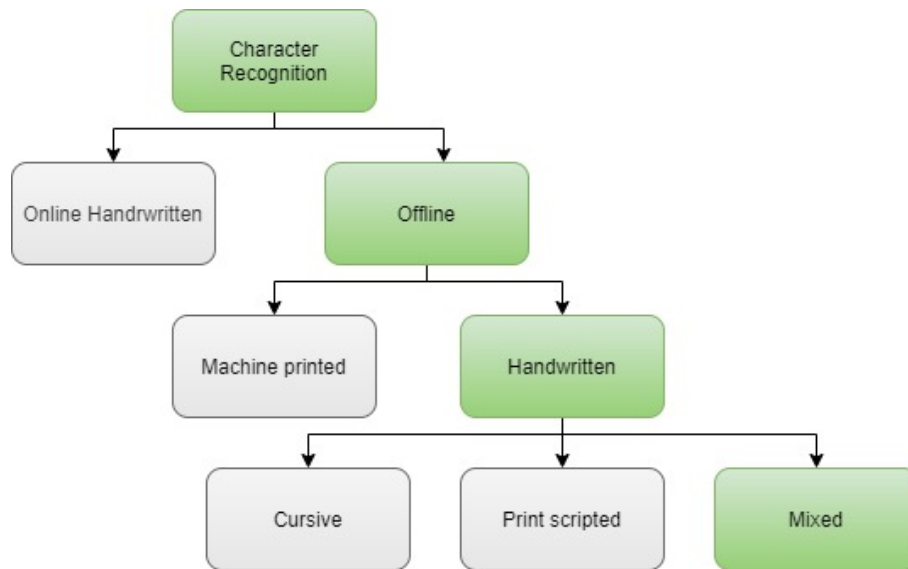


Figure 3.1: OCR division

Character Recognition of handwritten text is a complex problem as it must deal with different styles of writing and different pen movements. There are also two types of handwritten style:

- **Cursive handwritten** – A style of handwriting with rounded letters that are joined together.
- **Print-script handwritten** – A style of handwriting that uses simple unjoined letters resembling printed lettering.

Handwriting recognition systems can be further divided into subcategories based on the time they access an input (time when the system starts to process the input):

- **Online** – Online Character Recognition system processes the text in real-time while a user is writing a character. They are less complex as they can benefit from temporal or time-based information as a speed, velocity, or writing direction.
- **Offline** – Offline Character Recognition system operates on static data. The input is usually given in form of the image containing text.

The division of OCR is shown in Figure 3.1. The basic stages of offline handwritten print script input document OCR based on [4] are described in the following sections.

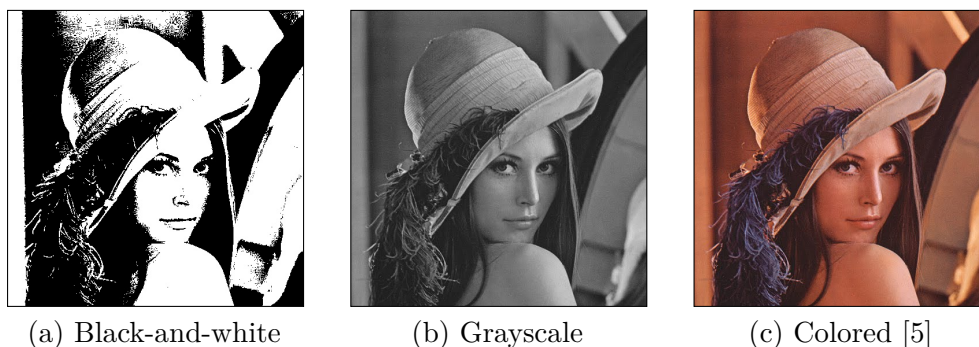


Figure 3.2: Different types of the image

## 3.2 Image Acquisition

Image acquisition is the first step of any OCR system when the digital image of the document with a text is obtained. In offline OCR systems, the two most common ways to acquire digital images are with a digital camera or a scanner.

From a mathematical point of view, a digital image  $P$  is a two-dimensional matrix of intensity values  $p_{ij}$  usually referred to as pixels. The pixel is a shortage for *picture element* and is assumed to be square with sides of length one, with the pixel with value  $p_{ij}$  centered at the point  $(i, j)$ .

There are three basic types of image based on color [6], these types of images are shown in Figure 3.2:

- **Black-and-white** – The simplest form of an image where  $p_{ij}$  is a single value from the set  $\{0, 1\}$  where 1 represents the white color and 0 represents the black color.
- **Grayscale** – The value  $p_{ij}$  is an integer with a value between 0 and 255 denoting shade of gray.
- **Colored** – Each  $p_{i,j}$  is a vector of three or more values. Most of the colors can be generated by red, green, and blue. The most common format is the RGB color model where each  $p_{i,j}$  is a vector of three values,  $p_{ij} = (r_{ij}, g_{ij}, b_{ij})$ , that denotes the amount of red, green and blue at the point  $(i, j)$ .

One of the advantages when representing an image as a two-dimensional matrix is the possibility to manipulate the image by performing mathematical operations. Some basic operations for increasing the quality of the image is discussed in the next section.

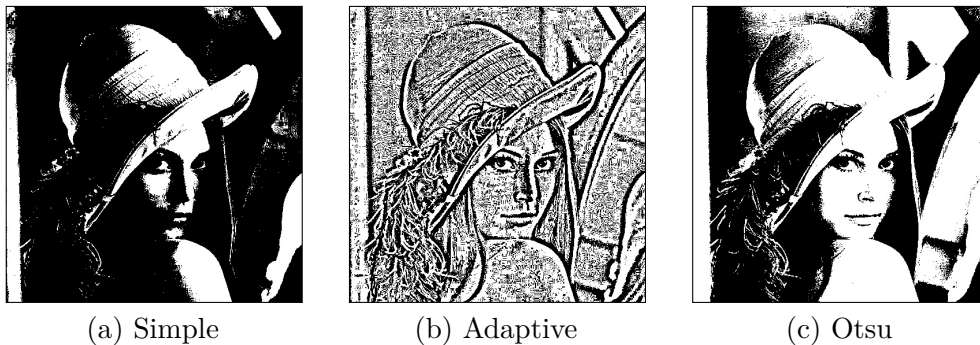


Figure 3.3: Image binarization

### 3.3 Preprocessing

The success rate of OCR is in correlation with the image quality. The purpose of image preprocessing is to remove non-relevant parts and improve the quality.

Image preprocessing is highly dependent on a type of input document, the purpose of character recognition, and the chosen algorithm for character segmentation and classification. Here only the preprocessing steps relevant to the expected input are described.

#### 3.3.1 Image Binarization

Image binarization is a conversion of an image into a black-and-white form, also called a binary image. This step is required to unite different colors of pen used for writing. The most common binarization techniques are: [7]

- **Simple Thresholding** – Pixels are compared to a global threshold  $T$ :

$$p_{x,y} = \begin{cases} 1, & \text{If } p_{x,y} \geq T \\ 0, & \text{otherwise} \end{cases}$$

- **Adaptive Thresholding** – The image is divided into smaller regions, and for every region, a local threshold is computed. The different methods can be used for calculating the local threshold. For example, the mean of neighborhood area can be used.
- **Otsu Binarization** – Automatically determines an optimal global threshold value from the image histogram. A detailed description of the Otsu method is in [8].

The resulting binary images for these techniques are shown in Figure 3.3.



### 3.3.2 Noise Reduction

Noise refers to unwanted pixel patterns or errors in the pixel values which negatively affects the output [9]. The goal of the noise reduction process is to remove these patterns and correct errors in pixel values.

Based on the characteristics, noise can be divided into following categories: [10]

- **Ruled Line Noise** – Some documents contain lines and boxes which need to be removed. Several problems with line removal occur – lines can be broken, have different thicknesses, and be crooked when a camera acquires the image. To crooked lines noise is also referred to as *Stroke Like Pattern Noise*. Some letters contain a straight line (5, 7, 1, ...) that should not be removed.
- **Marginal Noise** – Dark shadows that appear in the vertical or horizontal margins of an image are called marginal noise. It is often the result of scanning a thick document.
- **Salt And Pepper Noise** – This noise usually consists of a small group of pixels. It is often caused by the bad quality of acquisition media or by dirt on the document.
- **Background Noise** – This category comprises uneven contrast, show-through effects, interfering strokes, and background spots. The degradation of a document can cause this type of noise.

Based on the type of noise, three main approaches can be used for noise removal, according to [11]. The approaches are following:

- **Filtering** (masks, Gaussian filter, etc) – Each pixel  $p_{x,y}$  is calculated based on its neighbors and used filter.
- **Morphological Operations** (erosion, dilation, etc.) – Morphological transformations are simple operations based on the binary image form. It expects two inputs – the original image and the structuring element or kernel, which decides the nature of the operation. Two basic morphological operators are erosion and dilation. A detailed explanation of morphological operation is in [12].
- **Component Labelling** – This technique groups pixels based on pixel connectivity. For a detailed explanation of pixel connectivity, see [13]. Components with a sum of pixels below a chosen threshold are removed.



(a) Original image      (b) Skeletonized image

Figure 3.4: Skeletonization

### 3.3.3 Skeletonization

Every person has a different style of writing, so different widths of stroke occur. To make the width of the stroke uniform, skeletonization can be used. Skeletonization is the result of thinning and it returns binary image eroded to its innermost level. The skeleton of the image should reflect the structure of an object's shape. There are two types of skeletonization algorithms based on [14]:

- **Iterative** – Removing the contour pixels iteratively till a width of one pixel is reached.
- **Non-iterative** – Produce a skeleton directly without investigating all of the individual pixels.

An example of skeletonization can be seen in Figure 3.4.

### 3.3.4 Region of Interest

Region of Interest (ROI) represents the relevant parts of the image. In the OCR system, the ROI are usually text areas. According to [15], ROI methods are based on:

- **Templates** – Defines a set of locations where information can be retrieved for a specific document layout.
- **Extraction Rules** – ROI are determined by some general knowledge of document such as the rectangular shape, color of borders, etc.
- **Machine Learning** – A model that can recognize text areas is created. This method works well especially for machine printed text as can be seen in the study [16].

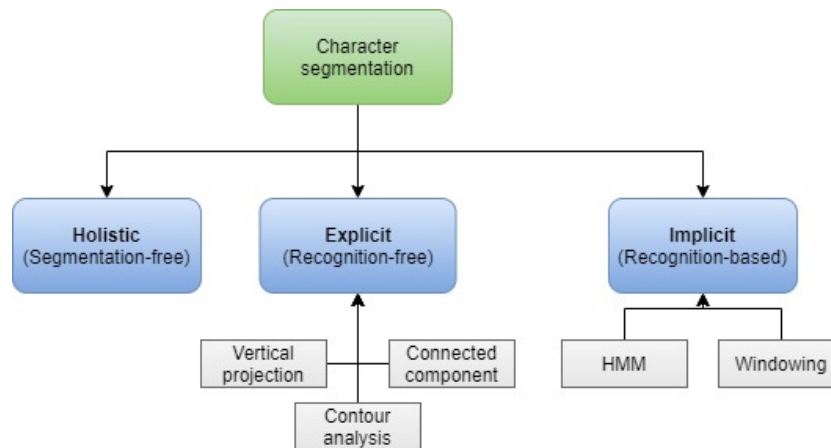


Figure 3.5: Division of character segmentation

### 3.4 Segmentation

There are three levels of segmentation in the general OCR: [17]

- **Lines** – Separation of individual lines from document. The most common approach for line segmentation is Horizontal projection where the peaks in projection indicate the line. [18]
- **Words** – Words segmentation tries to parse concatenated text to infer where word breaks exist. Vertical projection is often used for this process, where the trough indicates breakpoints.
- **Characters** – Character segmentation is a process that tries to decompose an image of a sequence of characters into individual symbols or the smallest units of a language.

In the context of this work, words are represented by moves and every move is in a different cell. The only relevant segmentation is character segmentation in this case. Different techniques of character segmentation are introduced in this section.

There is no united division of segmentation approaches. In studies dealing with segmentation, the division is usually based on the stage of recognition included in the process. The division chosen in this work was based on [19] and [20]. The authors divided segmentation approaches into three *pure* categories – *holistic*, *implicit*, and *explicit* as shown in Figure 3.5. The authors also suggest that there is a fourth category called *hybrid* that combines at least two of the techniques from the *pure* approaches.

### 3. OCR

---

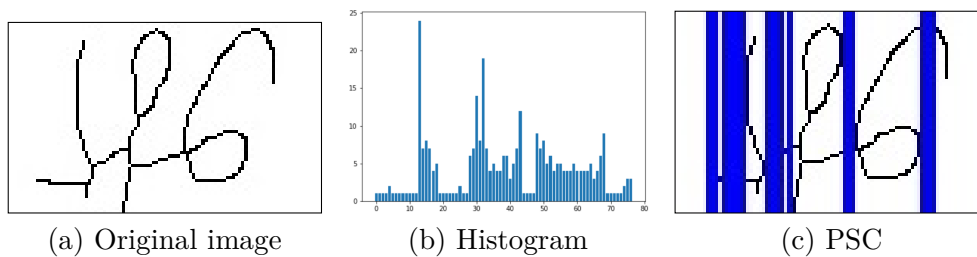


Figure 3.6: Vertical projection

#### 3.4.1 Holistic

Holistic also known as the *segmentation-free* approach is based on segmenting words and then classifying the whole word as a unit. This approach is usually restricted to a predefined lexicon and requires a huge dataset of words in that specific lexicon as training on word samples is required. Thus this approach is more suitable for OCR with a statically predefined lexicon, which is not likely to be changed, e.g., bank cheque recognition. [17]

The holistic recognition can also be used for online handwritten OCR for a personal computer or notepad, when recognition can be fine-tuned due to personal handwriting style.

In studies dealing with OCR, the holistic approach is more often used for cursive Arabic texts than texts in the Latin alphabet as can be seen in [21] and [22].

#### 3.4.2 Explicit Segmentation

Explicit segmentation is also called *recognition-free segmentation*. The word is segmented into individual characters which are then classified – the segmentation process and the classification process do not interact. [17]

The most common techniques from the recognition-free family are introduced in this section. These methods are not stand-alone and usually must be combined or supplemented for better performance.

These methods often seek to find *dissection* of the image. Dissection in this context is the decomposition of the image based on some general features of the character such as width, height, number of pixels, etc.

The most common methods from explicit segmentation are:

- **White space and pitch** – In machine printing, characters are often separated by vertical columns of white pixels and each character has a fixed width. Fix width and height can be forced by creating a separate cell for every character in a handwritten case. [20] Thus, white spaces and pitch provide a basis for estimating segmentation points.

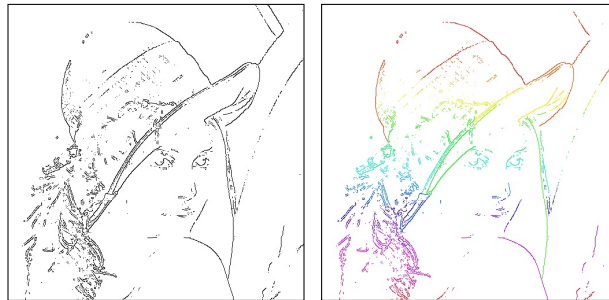


Figure 3.7: Connected components

- **Vertical histogram** – The vertical histogram approach seeks to find PSC (*Potential Segmentation Column*). Vertical histogram, also called *vertical projection* or *projection analysis*, is a simple process of counting black pixels in each column. PSC are columns where the sum of black pixels is below some defined threshold. This technique can detect white spaces and slightly touching chars.

This approach is useful mainly in good quality machine printing. There are some downsides of this approach when using handwritten text – when a word is slanted, or characters are overlapped, this approach gives inaccurate results. The example of the Vertical Histogram approach can be seen in Figure 3.6. PSC are indicated by blue color. The threshold value for PSC was set on one in this case.

- **Connected component** – The connected component method determines the connected black region called component. This method provides good results when dealing with printed characters. When used in handwritten text recognition, this method is often supplemented by some splitting and merging techniques to avoid over-segmenting and pre-segmenting. [20] Examples of image processed by connected component analysis is in Figure 3.7.

### 3.4.3 Implicit Segmentation

Implicit segmentation also called *recognition-based*, is based on segmentation and recognition at the same stage of OCR. Image is split into many overlapping sub-images independent of the content – in contrast to the recognition-free approach, which determines sub-images based on their features. Then the system searches the sub-image for components that match classes in its alphabet. [23]

The most common method in implicit segmentation is based on HMM (*Hidden Markov Models*). For a detailed explanation of the HMM and how works in the segmentation process see [24].

## 3.5 Feature Extraction

The feature extraction reduces dimensionality by extracting the most relevant information from the image. The output of the feature extraction stage is a feature vector. The form of the vector depends on the method used for classification. However, the goal is the same – extract a set of features to maximize the recognition rate with the least possible number of elements. Feature extraction methods are based on three types of features which are described in the following subsections. [25]

### 3.5.1 Statistical

The features are derived from the statistical distributions of pixels from an image. The most common methods in this category are:

- **Zoning** – The character image is divided into  $n \times m$  zones of predefined size. Local characteristics such as pixel density, number of vertical lines, etc., are extracted from each zone to a feature vector.
- **Projection Histograms** – The image of a character can be represented as a 1D signal by projection histogram. Projection histograms count the number of pixels in each column and row of a character image. The drawback of this method is the dependence on a slant of the character and noise.
- **Profiles** – Feature vector consist of number of pixels between the bounding box of the character and the character.

### 3.5.2 Structural

The structural method identifies structural features of a character as topological and geometric properties of the character. Structural features can be, for example, the number of vertical lines, number of horizontal lines, endpoints, crosspoints, etc.

### 3.5.3 Global Transformations and Moments

Methods from this class are often based on the Fourier Transformation of the contour of the image. The first  $n$  coefficients of the Fourier Transformation are considered to be an  $n$ -dimensional feature vector that represents the character. For detailed explanation of Global Transformations and Moments see [26].

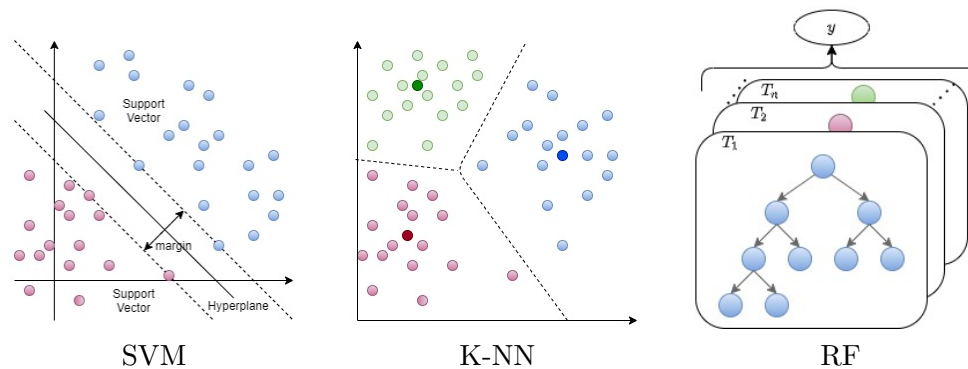


Figure 3.8: Classification methods

## 3.6 Character Classification

Basic methods for multiclass classification based on learning from examples are described in this section.

### 3.6.1 Support Vector Machines (SVM)

SVM is a supervised learning algorithm. The SVM's objective is to find a hyperplane in an  $n$ -dimensional space that maximizes the separation between two classes. The same principle is utilized for multiclass classification after breaking down the multiclassification problem into multiple binary classification problems. A detailed description of the SVM method can be found in [27].

### 3.6.2 K-Nearest Neighbors (K-NN)

K-NN is a non-parametric supervised machine learning algorithm used for both classification and regression. The K-NN algorithm assumes that data from one class exist in close proximity. It means that a distance between two points from the same class is usually smaller than a distance between two points belonging to different classes. The distance is usually measured by Euclidean or Hamilton distance. [28]

### 3.6.3 Random Forests (RF)

Random forests, also called *random decision forests*, uses multiple (usually a large number of) decision trees trained on different parts of the same training set. The classification of a sample is given by votes of individual decision trees. A decision tree is a tree-like model with decision rules in nodes. [29]

### 3. OCR

---

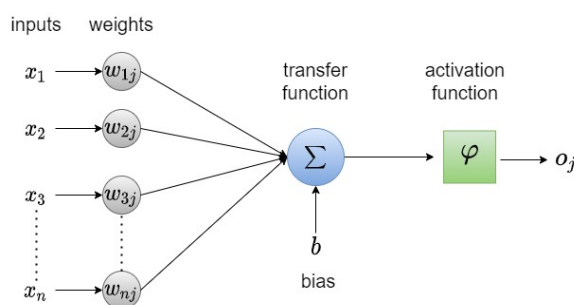


Figure 3.9: A neuron

#### 3.6.4 Convolution Neural Networks (CNN)

Neural Network (NN) imitates processes in a brain by modeling the biological neural system. The smallest unit of the brain is a neuron as it is in NN. The NN neuron can be represented by

$$y_k = \varphi \left( \sum_{j=0}^m w_{kj} x_j + b \right),$$

where  $w_{kj}$  are parameters and  $\varphi$  is an activation function.

Every activation function (or non-linearity) takes a single number and performs a mathematical operation on it. There are several activation functions used in practice:

- **Sigmoid** –  $\sigma(x) = 1/(1 + e^{-x})$ ,
- **Tanh** –  $\tanh(x)$ ,
- **ReLU** –  $\max(0, x)$ .

The overview of other activation functions and their comparison can be seen in [30]. Models are organized into distinct layers of neurons, where neurons are put together into an acyclic graph.

Convolutional Neural Network is a type of NNs, where images on the input are assumed. The following layers are used to build CNN. [31]

- **Convolutional layer** – This type of layer detects features in image based on a given filter.
- **Pooling layer** – The goal of this type of layer is to reduce the spatial size of the representation by some defined pooling operation.
- **Fully-Connected layer** – All neurons from the previous layer co-create the input to neurons of the next layer, and neurons in one layer are not connected.

A detailed explanation of Convolution Neural Networks can be found in [32].



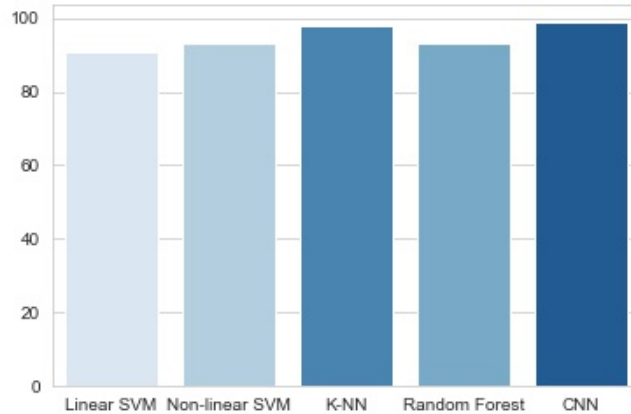


Figure 3.10: Comparison of classification methods

### 3.6.5 Comparison of Classification Methods

The comparison of models based on the mentioned methods are introduced in this section. All models were trained on the MNIST dataset [33], which is a handwritten digit dataset. MNIST contains 60.000 training samples and 10.000 testing samples with a shape  $28 \times 28$ .

- **SVM** – The implementation of the SVM model from [34] was considered. The preprocessing steps deskewing and normalization between 0-1 were implemented. Two models were tested – linear with **91%** accuracy and non-linear (RBF kernel,  $C = 1$ , default value of gamma) with **93%** accuracy.
- **K-NN** – The implemented K-NN model from [35] was considered. The best accuracy of **98%** was obtained with  $k = 11$ .
- **RF** – The RF model’s accuracy with fine-tuned parameters was around **93%** as proposed in [36].
- **CNN** – The CNN model’s accuracy depends mainly on the model architecture (types and number of layers, activation function, etc.). The proposed CNN architecture is discussed in Section 4.3.4. The best-found accuracy obtained by the CNN model was around **99%**. [37]

The comparison of methods in terms of accuracy can be seen in Figure 3.10.

## 3.7 Postprocessing

The goal of postprocessing is to find and correct errors in predicted words. Several postprocessing methods based on [38] are introduced in this section.

### 3.7.1 Manual Error Correction

The resulting output of the OCR system is corrected by humans, also known as *proofreading*.

One of the Manual Error Correction applications was Project Gutenberg, the volunteering project to digitalize books. Several people did proofreading and correcting of OCR errors several times.

### 3.7.2 Lexicon-Based Error Correction

This method uses the comparison of words in the lexicon with predicted words and computes distance. The distance can be measured by: [39]

- **Levenshtein Distance (LD)** – The minimum number of operation (deletion, insertion, and substitution) required to transform from word  $a$  to word  $b$ .
- **Longest Common Subsequence Distance (LCSD)** – Similar to Levenshtein distance but does not include substitution operation. Substitution must be replaced by delete and insert operation.

The drawback of Lexicon-Based methods is the need to compare each word in a document with each word in the provided lexicon which can be computationally costly.

### 3.7.3 Context-Based Error Correction

Context-Based Error Correction techniques are based on statistical language modeling and word  $n$ -grams. In general, the likelihood that a word sequence appears is counted. For the detailed explanation, see [40].

## 3.8 Existing OCR Systems

The area of application of the OCR systems is significant. Besides the classical text documents, there are more specific applications as automatic number plate recognition, traffic sight recognition, passport recognition, defeating CAPTCHA anti-bot systems, and many more.

The wide usage of OCR and its application in specific areas imply many existing OCR systems. The leading open-source OCR system is currently the Tesseract developed by HP which is now partially funded by Google under

the Apache license. The System is intended for machine print documents and can be trained on different types of fonts as proposed in [41].

Despite the good accuracy rate in recognition of machine-printed text, the system provides poor results for handwritten ones as shown in [42].



---

# Implementation

This chapter describes the implemented system. The used technology is introduced in Section 4.1, and the implemented stages of the proposed OCR system are detailed in Section 4.2.

## 4.1 Technology

The final system was developed as a frontend-backend web application. The frontend was created in React. The backend was developed in Flask, and PostgreSQL was chosen for the database. The final application was deployed on `pythonanywhere.com` platform.

### 4.1.1 Flask

The Flask is a lightweight WSGI web framework written in Python. [43] WSGI stands for *Web Server Gateway Interface*. It is a specification that defines how web applications can be bound together and how a web server communicates with web applications to process a request. [44]

### 4.1.2 React

React is a declarative and flexible javascript library for building a user interface. The complex UI is composed of small and isolated pieces of code called *components*. [45]

### 4.1.3 PostgreSQL

PostgreSQL is an open-source object-relational database witch supports both SQL (relational) and JSON (non-relational) querying. One of advantages of using PostgreSQL is high scalability in the quantity of data it can handle and the number of concurrent users it can accommodate. [46]

### 4.1.4 Libraries

Some of the most essential used libraries and APIs are described below:

- **OpenCV** – Stands for *Open Source Computer Vision Library*, and it is an open-source computer vision and machine learning software library with Python and C/C++ interface. The library provides more than 2500 optimized algorithms which can be used for face detection, identify objects, classify human actions in videos, and many more. [47]
- **Scikit-image** – Python open-source library for image processing. The OpenCV does not provide all practical algorithms such as Sauvola Thresholding. Therefore Scikit-image was used as a supplement to the OpenCV library.
- **TensorFlow** – The most common library in the development of Deep Learning models. It was created by Google, it is open-source, and it provides APIs in majority of programming languages including Python and C/C++. [48]
- **Keras** – High-level library written in Python running on top of TensorFlow. Keras is used to build neural networks without worrying about the mathematical aspects of tensor algebra, numerical techniques, and optimization methods. [48]
- **Python-chess** – Python chess library which provides valid move generation, move validation, and support for standard formats as PGN and FEN. It also includes the stockfish chess engine interface with an estimated rating of about 3516 (the strongest human chess player has a 2847 rating). [49]
- **Tinypng** – A library that provides API for compression of images in the format of jpg and png without the quality loss. Tinypng uses the technique of *quantization* which reduces the number of colors by combining similar colors. [50]

### 4.1.5 Pythonanywhere

The application was deployed on pythonanywhere<sup>9</sup> which is an online IDE and web hosting service based on Python with support for PostgreSQL and MySQL. [51]

The major drawback of the pythonanywhere is an incompatibility with the latest version of Tensorflow (r2.4). The version of Python, Keras, and TensorFlow have to be downgraded to solve this issue.

---

<sup>9</sup><http://marikja7.pythonanywhere.com/>

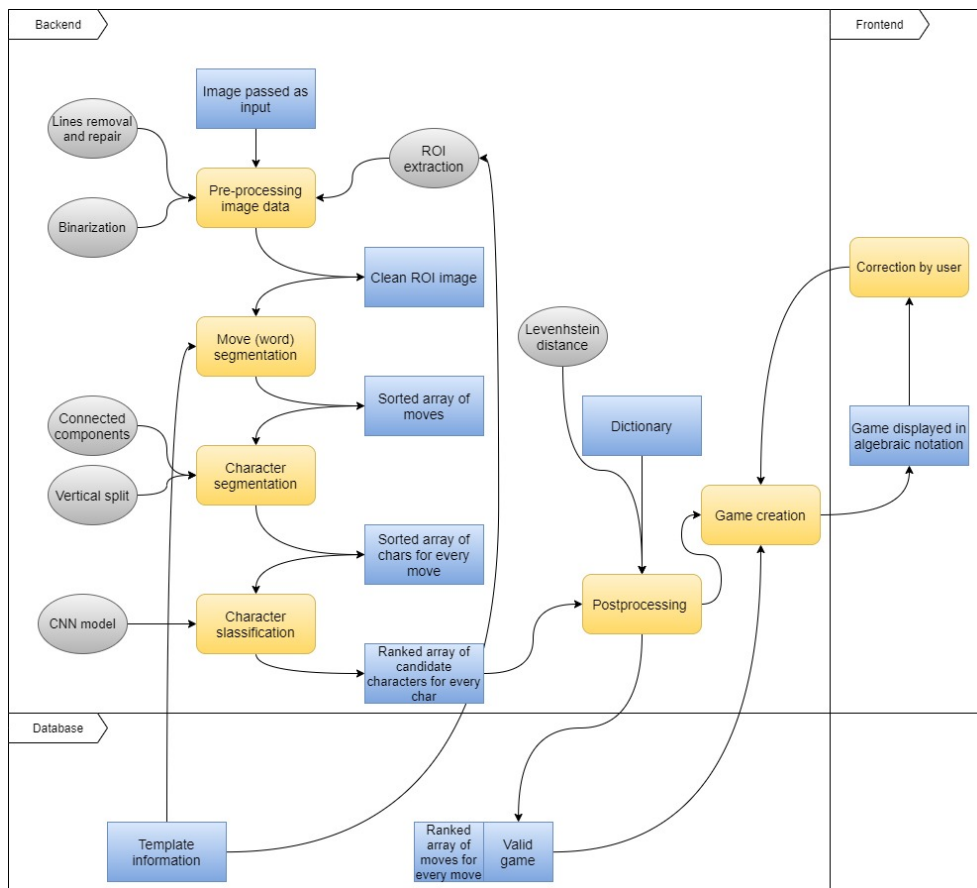


Figure 4.1: System overview

## 4.2 System Functionality

The system was implemented to fulfill the following functional requirements:

- **Score sheet upload** – User can upload one photo or scan of a score sheet.
- **Replay game** – After the score sheet is uploaded, the system displays the chessboard along with the game in English algebraic notation. User can replayed the game on the chessboard.
- **Correct move** – The user is allowed to correct misclassified moves. The game is recomputed and returned after that.
- **Download PGN file** – The user can download the game in PGN format.

The overview of implemented system is shown in Figure 4.1.

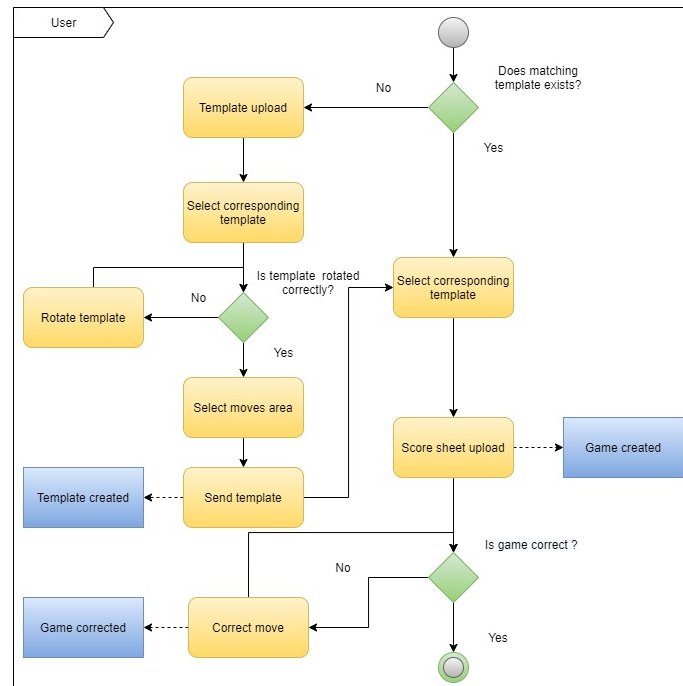


Figure 4.2: Score Sheet Conversion Process

### 4.3 Score Sheet Conversion Process

The user is expected to manage two main steps to obtain the score sheet in digitalized form:

- **Template upload** – For the ROI extraction, the template method was used. This method firstly requires uploading an empty score sheet. The user has to rotate the template and select the area with moves. A box detection algorithm similar to the one introduced in [45] is applied to determine cells designated for writing moves. The template upload process examples from the application are shown in Figures D.1, D.2, and D.3.
- **Score sheet upload** – The central part of the OCR starts after the score sheet upload. The individual steps of the score sheet upload process are described in this section.

The activity diagram describing required user actions to obtain the converted score sheet is in Figure 4.2.

For a better insight into the functionality of implemented OCR system, two model score sheets are used to show the impact of individual steps of the OCR system. The term *score sheet A* is used for the score sheet shown in Figure A.1, the second score sheet in Figure A.3 is labeled as *score sheet B*.



### 4.3. Score Sheet Conversion Process

dt	g6	21	Kz2	Vl7	41	Va7+	Kx9+
ct	Sa7	22	Kd2	Jl6	42	Dc7+	Ka8
Jc3	d6	23	Jr2	Vl2	43	Dx4+	Dl6
rt	z5	24	Vg1	Vl8	44	Ka3	Vl8
Jl3	Jc6	25	Kc2	Vl7	45	Dl5	Ka7
d5	Jd+	26	Kl3	Va7	46	Dg5	
Sz2	c5	27	Vc1	Sd7	47		
O-0	Jl6	28	Dl1	Dz8	48		0:1
Vz1	O-0	29	c5	Rxc5	49		
R4	R6	30	Rxc5	Sxat	50		
L3	Jr8	31	Ka2	dxcs	51		
Sd3	l5	32	Sxc5	Sl5	52		
Jxdt	cxdt	33	Sxl8	Sxd3	53		
Jr2	lt	34	Sxg7	Dat	54		
l3	Vl7	35	Kl2	Dl5	55		
at	q5	36	Ka3	Kxg7	56		
Sa3	l5	37	Vc7	Ka6	57		
Kl2	at	38	Jx4+	Rx4+	58		
Jg1	Sz8	39	Dc1	Da6	59		
Rxgt	Rxgt	40	Kl3+	Vl8	60		
RTIFICATION		STAR		Cas:			

(a) Perspective transformation

dt	g6	21	Kz2	Vl7	41	Va7+	Kx9+
ct	Sa7	22	Kd2	Jl6	42	Dc7+	Ka8
Jc3	d6	23	Jr2	Vl2	43	Dx4+	Dl6
rt	z5	24	Vg1	Vl8	44	Ka3	Vl8
Jl3	Jc6	25	Kc2	Vl7	45	Dl5	Ka7
d5	Jd+	26	Kl3	Va7	46	Dg5	
Sz2	c5	27	Vc1	Sd7	47		
O-0	Jl6	28	Dl1	Dz8	48		0:1
Vz1	O-0	29	c5	Rxc5	49		
R4	R6	30	Rxc5	Sxat	50		
L3	Jr8	31	Ka2	dxcs	51		
Sd3	l5	32	Sxc5	Sl5	52		
Jxdt	cxdt	33	Sxl8	Sxd3	53		
Jr2	lt	34	Sxg7	Dat	54		
l3	Vl7	35	Kl2	Dl5	55		
at	q5	36	Ka3	Kxg7	56		
Sa3	l5	37	Vc7	Ka6	57		
Kl2	at	38	Jx4+	Rx4+	58		
Jg1	Sz8	39	Dc1	Da6	59		
Rxgt	Rxgt	40	Kl3+	Vl8	60		
RTIFICATION		STAR		Cas:			

(b) Sauvola Thresholding

dt	g6	21	Kz2	Vl7	41	Va7+	Kx9+
ct	Sa7	22	Kd2	Jl6	42	Dc7+	Ka8
Jc3	d6	23	Jr2	Vl2	43	Dx4+	Dl6
rt	z5	24	Vg1	Vl8	44	Ka3	Vl8
Jl3	Jc6	25	Kc2	Vl7	45	Dl5	Ka7
d5	Jd+	26	Kl3	Va7	46	Dg5	
Sz2	c5	27	Vc1	Sd7	47		
O-0	Jl6	28	Dl1	Dz8	48		0:1
Vz1	O-0	29	c5	Rxc5	49		
R4	R6	30	Rxc5	Sxat	50		
L3	Jr8	31	Ka2	dxcs	51		
Sd3	l5	32	Sxc5	Sl5	52		
Jxdt	cxdt	33	Sxl8	Sxd3	53		
Jr2	lt	34	Sxg7	Dat	54		
l3	Vl7	35	Kl2	Dl5	55		
at	q5	36	Ka3	Kxg7	56		
Sa3	l5	37	Vc7	Ka6	57		
Kl2	at	38	Jx4+	Rx4+	58		
Jg1	Sz8	39	Dc1	Da6	59		
Rxgt	Rxgt	40	Kl3+	Vl8	60		
RTIFICATION		STAR		Cas:			

(c) Lines removal

dt	g6	21	Kz2	Vl7	41	Va7+	Kx9+
ct	Sa7	22	Kd2	Jl6	42	Dc7+	Ka8
Jc3	d6	23	Jr2	Vl2	43	Dx4+	Dl6
rt	z5	24	Vg1	Vl8	44	Ka3	Vl8
Jl3	Jc6	25	Kc2	Vl7	45	Dl5	Ka7
d5	Jd+	26	Kl3	Va7	46	Dg5	
Sz2	c5	27	Vc1	Sd7	47		
O-0	Jl6	28	Dl1	Dz8	48		0:1
Vz1	O-0	29	c5	Rxc5	49		
R4	R6	30	Rxc5	Sxat	50		
L3	Jr8	31	Ka2	dxcs	51		
Sd3	l5	32	Sxc5	Sl5	52		
Jxdt	cxdt	33	Sxl8	Sxd3	53		
Jr2	lt	34	Sxg7	Dat	54		
l3	Vl7	35	Kl2	Dl5	55		
at	q5	36	Ka3	Kxg7	56		
Sa3	l5	37	Vc7	Ka6	57		
Kl2	at	38	Jx4+	Rx4+	58		
Jg1	Sz8	39	Dc1	Da6	59		
Rxgt	Rxgt	40	Kl3+	Vl8	60		
RTIFICATION		STAR		Cas:			

(d) Lines repair

Figure 4.3: Implemented steps of preprocessing

#### 4.3.1 Preprocessing

The steps involved in preprocessing stage are derived from the expected appearance of the score sheet. The application can use different forms of score sheet as long as some requirements are fulfilled. The typical format of the score sheet expected as the input is described in Section 1.2.

The preprocessing stage consists of four steps – perspective transformation, binarization, line removal and line repair. Each of these steps is described in this section in detail.

The example of output from each preprocessing step is shown in Figure 4.3. The final outcome of this step is shown in Figure 4.3d. These outputs corresponds with *score sheet A* used as an input.

## 4. IMPLEMENTATION



Figure 4.4: Perspective transformation

### Perspective transformation

Users are allowed to upload not only scanned score sheet, but also a photo of the score sheet. The photo can be taken from different angles; therefore, the score sheet's perspective transformation must be applied. The perspective transformation consists of the following steps:

- **The key points features detection** – The key points need to be detected on both the score sheet and the selected template. For the detection of the key points, a SIFT (*scale-invariant feature transform*) algorithm was used. For a detailed explanation of how SIFT works, see [52]. The drawback of SIFT is time complexity. The fastest alternative is ORB (*Oriented FAST and Rotated BRIEF*). However, it provided significantly worse results than SIFT when tested on score sheets.
- **The key points features comparison** – The template and score sheet's key points are compared by FLANN (*Fast Library for Approximate Nearest Neighbors*). It contains optimized algorithms for fast nearest neighbor search in large datasets and high dimensional features. For FLANN explanation, see [53].
- **Homography matrix** – A homography relates to any two images of the

same planar surface in space. The homography matrix  $H$  is a  $3 \times 3$  matrix that maps the point  $(x_1, y_1)$  in one image to the corresponding point  $(x_2, y_2)$  in another image. At least four corresponding points between them must be known to calculate the homography between two images. [54] These points are obtained by SIFT algorithm from the previous step.

- **Points transformation** – Points from the uploaded score sheet are recomputed by the Homography matrix to their new location:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}.$$

Key points counted by SIFT, matching points and corners of the score sheet calculated by the Homography matrix can be seen in Figure 4.4.

### **Binarization**

Sauvola thresholding is used as it provides the best results along with Otsu binarization supplemented by component labeling noise removal. In the Sauvola Thresholding, several thresholds  $T(x, y)$  are calculated for every pixel by

$$T(x, y) = m(x, y) \left[ 1 + k \left( \frac{\delta(x, y)}{R} - 1 \right) \right],$$

where  $m(x, y)$  and  $\delta(x, y)$  are mean and standard deviation of the pixels within a window of size  $w \times w$  given as a parameter,  $R$  is the maximum value of the standard deviation, and  $k$  is a bias which takes positive value in the range  $(0.2, 0.5)$ . [55]

### **Lines removal and repair**

The lines removal and repair step aims to remove and repair vertical and horizontal lines that create the table. The first step of this process is to define a vertical and horizontal kernel. Lines are detected by morphological operation closing using the defined kernels.

The detected contours are filled with white color (or the color chosen for the background). After this step, gaps in letters that go down below the line can occur. The kernel with opposite aspect ratios for closing is defined to fill these gaps.

d4	g6	21	Kx2	Vx7	41	Vx7+	Kx9?
c4	g7	22	Kd2	Jl6	42	Dc7+	Ka6
Jc3	d5	23	Jx2	Vx2	43	Dx4	Dl6
r4	r5	24	Va1	Vx8	44	Ka3	Vx8
Jl3	Jc6	25	Kc2	Vx7	45	Dl5	Ka7
d5	Jd4	26	Kx3	Va7	46	Dg5	
Sx2	c5	27	Vc1	Sx7	47		
O-O	Jl6	28	Dl1	Dx8	48		
Vx1	O-O	29	c5	Rxc5	49		
R4	R6	30	Rxc5	Sxat	50		
R3	Jx8	31	Ka2	dx5	51		
Sd3	l5	32	Sxc5	Sx5	52		
Jx4	cxd4	33	Sx8	Sxd3	53		
Jx2	l4	34	Sx7	Dat	54		
R3	Vx7	35	Kx2	Dx5	55		
a4	a5	36	Ka3	Kx7	56		
Sa3	R5	37	Vc7	Ka6	57		
Kl2	a4	38	Jx4	Rx4	58		
Va1	Sx8	39	Dc1	Dc6	59		
Rx4	Rx4	40	Kx3	Vx8	60		

0:1




Figure 4.5: Move segmentation – score sheet A

### 4.3.2 Move Segmentation

Move cells are computed and stored in the format of left upper coordinates, width, and height after the template upload, as mentioned in Section 4.2. After the score sheet upload, these cells are restored and scaled by the formulas:

$$x_{\text{scale}} = \frac{\text{score sheet width}}{\text{template width}}, \quad y_{\text{scale}} = \frac{\text{score sheet height}}{\text{template height}}$$

to match the score sheet size.

Possible use of cursive letters going below the line is taking in consideration and therefore bigger area is extracted. For the purpose of this thesis the following terms are defined:

- **Valid move area** – The area that corresponds with the exact cell intended for the player to write a move.
- **Extracted move area** – The area that is extracted in the system and then processed by the character segmentation step. The extracted area is in shape of width = cell.width \*  $x_{\text{scale}}$  and height = cell.height \*  $\frac{3}{2} * y_{\text{scale}}$ .

The critical step in the move segmentation stage is to detect the end of the game. Players make this task difficult by writing the result right below the moves and not leaving empty cells between the moves and the game's result. The extracted move area is considered a move when the sum of the black pixels is greater than some chosen cell area percentage.

The result of the move segmentation is in Figure 4.5. Moves considered valid are bordered with blue color.

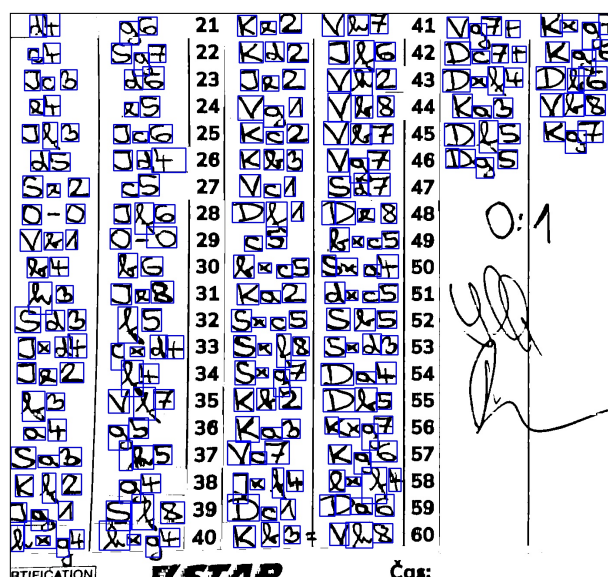


Figure 4.6: Character segmentation – score sheet A

### 4.3.3 Character Segmentation

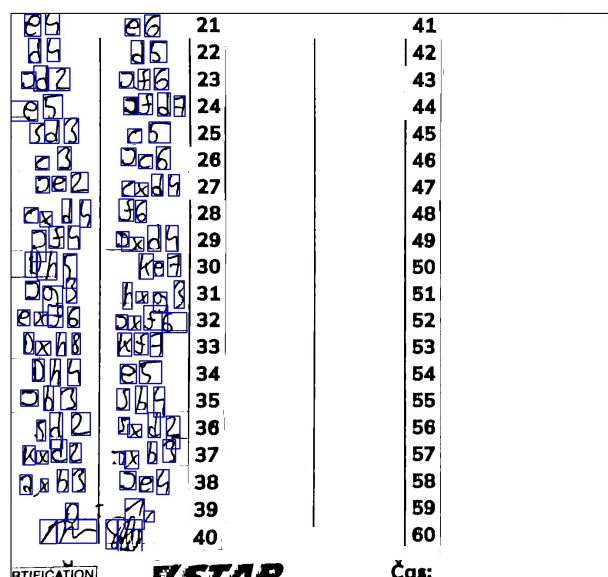
One of the advantages of working with algebraic notation instead of general text is word length and lexicon restriction. Chess moves can contain two to five chars.

The Kaggle dataset [56] which contains 3.5 million chess games in PGN format was used to evaluate PGN characteristics such as the summary of move lengths in games, the average move lengths and the distribution of chars, digits, and symbols. The statistics based on these characteristics are shown in Appendix C. These characteristics were used to create basic assumptions about the text content of moves in score sheets.

The first step of the character segmentation process is to count overall characteristics of moves in the score sheet – the average sum of black pixels, the average width, and the average height. These values are used to determine threshold values.

The core of the segmentation process is the evaluation of individual components – which basically means to distinguish between three possibilities – a component is stand alone character, the component is a noise, or a component is a part of a character. The component evaluation consist of the following steps:

1. **Component removal** – Parts that do not belong to the move must be removed (parts that are from an upper move or a below move). Components that start below the *valid move area* are removed.

Figure 4.7: Character segmentation – *score sheet B*

2. **Horizontal split** – This step aims to split component which is touching another character from below the line. The component higher than a defined threshold is split based on the horizontal histogram.
3. **Vertical merge** – When one component is below the other component, components are merged.
4. **Vertical split** – Components wider than a defined threshold are split by vertical line based on the vertical histogram.
5. **Noise removal** – All components with a sum of black pixels below some defined threshold are removed.

The result of the character segmentation process corresponding with *score sheet A* is shown in Figure 4.6. The result of the character segmentation process corresponding with *score sheet B* can be seen in Figure 4.7.

In the *score sheet A*, the example of over-segmenting occurs – in black 39 move, symbol D is segmented as two characters. It is caused by big difference of its size compared to other characters.

In the second example with *score sheet B* symbols are of constant size, so there is no over-segmenting, although the char e in the 4th white move is segmented with the part of the line, which was incorrectly left behind by the preprocessing step.

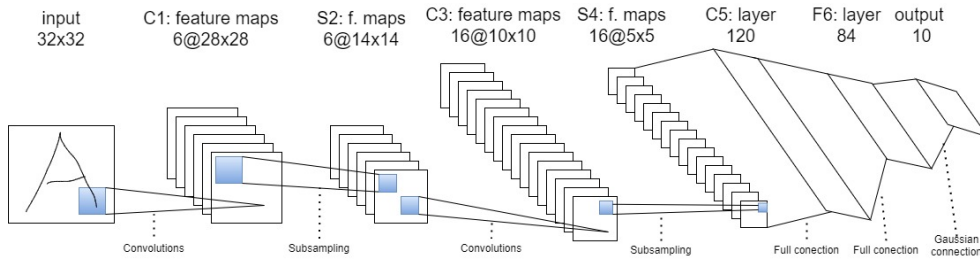


Figure 4.8: LeNet-5 architecture

#### 4.3.4 Character Classification

The CNN model was chosen for character classification. Three architectures were tried – LeNet-5, customized LeNet-5 and AlexNet.

LeNet was one of the first proposed architectures of a convolutional neural network. Nowadays, it is considered as *hello world* in the machine learning field. LeNet was originally designed for numbers from 0 to 9. There are several versions of the original LeNet. The most used is LeNet-5. The architecture of the LeNet-5 model can be seen in Figure 4.8.

The customized LeNet-5 model is taken from [37]. Accuracy on the MNIST dataset is 99.7%. It is based on LeNet-5 architecture with the following improvements:

- Single  $5 \times 5$  filters are replaced by two stacked  $3 \times 3$  filters.
- A convolution with stride two pooling layers are replaced by convolution with stride 2.
- Sigmoid is replaced by ReLU.
- Batch normalization and Dropout are added.
- More feature maps (channels) are added.

AlexNet is a convolutional neural network with eight layers. It was originally designed to classify the colored images which could belong to one of 22000 categories. Therefore the used AlexNet has to be customized for gray-scale images and smaller category sets. For the detailed explanation of AlexNet, see the original paper [57].

The models were trained on digits 0-9, lower-case characters a-h, and upper-case characters D, K, V, S, J, X, O from the EMNIST dataset [58]. EMNIST is the extension of the MNIST dataset containing all letters from the Latin alphabet. There are also datasets which contains special symbols from algebraic notation such as -, + and #. The decision was made not to train the model on these symbols because, even though it is a convention, many players do not bother with writing them.

#### 4. IMPLEMENTATION

---

#	W	B	#	W	B	#	W	B
1	d4	g6	21	K22	R87	41	Rg7f	Kxg7
2	cR4	Bg7	22	Kd2	Nx5	42	Qc7f	Kgg
3	Nc3	dg	23	N22	Rx2	43	Qx24	Q26
4	24	25	24	Rg1	Rx8	44	Ka3	xx8
5	N23	Nc6	25	Kc2	R87	45	Qe5	Kg7
6	d5	NdK	26	K83	Rg7	46	N235	
7	B22	c5	27	Rch	Bd7			
8	OO	Nx6	28	Qg1	NQ28			
9	R2h	Ob	29	c5	8xc5			
10	84	85	30	xxc5	Bxa4			
11	a3	N28	31	Ka2	dx5			
12	Bd3	x5	32	Bxc5	Bx5			
13	Nxd4	5xd4	33	Bx28	5xd3			
14	N22	x4	34	Bxg7	Qa4			
15	x3	Rx7	35	Kx2	Qx5			
16	a4	g5	36	Ka3	Kxg7			
17	Bab	f25	37	xc7	Kg6			
18	Kx2	g4	38	2Kx4	2x84			
19	Ngh	B88	39	Qc1	NNa6			
20	8gxg4	8xg4	40	K23	R88			

Table 4.1: Character classification – *score sheet A*

The accuracy of CNN models often improves with the size of the dataset. Data augmentation is used to expand the training part of the dataset. It is a technique for generating new images by modifying the existing ones. The modifying operation often includes shifting, flipping and zooming. [59]

The models were trained during 45 epoch with 64 batches. LeNet-5 and AlexNet model architectures have to be slightly modified to fit the 23 classes of characters that can score sheets written in algebraic notation contain. The validation accuracy of models was following:

- **LeNet-5** – 90.610%
- **Customized LeNet-5** – 97.186%
- **AlexNet** – 96.975%

The customized LeNet-5 model was chosen because of its highest testing accuracy. The summary of chosen model architecture is shown in Appendix in Figure B.1. The comparison of the training loss vs. validation loss and training accuracy vs. validation accuracy of the trained models is shown in Appendix B.



#	W	B
1	e4	e6
2	d4	d5
3	Nd2	Nf6
4	f5	Nfd7
5	5d3	c5
6	c3	Nc5
7	Ne2	cxd4
8	cxdB	f6
9	Kf4	Qxd4
10	1h5	Ke7
11	Ng3	bxQ3
12	exN6	Qx56
13	Qxh8	Kf7
14	Qh4	e5
15	Nb3	NbB
16	5d2	5xd3
17	Kxd2	7xb3
18	2xh3	NeB

#	W	B
1	e4	e6
2	d4	d5
3	Nd2	Nf6
4	f5	Nfd7
5	Bd3	c5
6	c3	Nc5
7	Ne2	cxd4
8	cxd5	f6
9	Kf4	Qxd4
10	Nh5	Ke7
11	Ng3	bx3
12	exf6	Qxf6
13	Qxh8	Kf7
14	Qh4	e5
15	Nb3	Nb5
16	Bd2	Bxd3
17	Kxd2	Qxb3
18	Nxh3	Ne5

Table 4.2: Character classification – Table 4.3: Candidate moves – *score sheet B*

The model returns all possible characters along with their probabilities. The most probable prediction of individual segmented characters corresponding with score sheet in Figure 4.6 are shown in Table 4.1. It is assumed that characters in score sheets are written in Czech algebraic notation and then translate to English algebraic notation, based on Table 1.2. The symbols in Table 4.1 are in English algebraic notation. The green color indicates correctly recognized characters; the red color indicates incorrectly recognized ones.

As can be seen in Table 4.1, even if the chars extraction is of a good quality, the model has a problem with recognizing the cursive written symbols as e, which is always misclassified as 2; then b, h and f which are usually classified as 8, 2 or x. The possible cause of this behavior is that in English-speaking countries, cursive writing is not so common. [60] This results in small representation in the EMNIST dataset and that is why the cursive letters are so poorly recognized.

The score sheet with only print handwritten characters can be seen in Figure 4.7. From the corresponding Table 4.2 of predicted characters it is clear that print handwritten characters provide better results. In the score sheet there is also an example of mistake the player made – in move 11 he wrote 3 instead of 6 (Ng3 should be Ng6 and hxg3 should be hxg6).

```

1 # Game
2 [
3   # 1. white move
4   [
5     # array of prediction for the 1.character in 1.move
6     [('d', 0.9997), ('x', 0.0002), ..., ('7', 2.26e-09)],
7     # array of prediction for the 2.character in 1.move
8     [('4', 0.9992), ('h', 0.0003), ..., ('0', 1.53e-09)]
9   ],
10  # 1. black move
11  [
12    [('g', 0.9989), ('8', 0.0003), ..., ('R', 4.73e-09)],
13    [('6', 0.8080), ('5', 0.1313), ..., ('4', 3.16-07)]
14  ],
15  . . . . .,
16  # 45. move black
17  [
18    [('K', 0.9991), ('x', 0.0008), ..., ('6', 1.05e-06)],
19    [('g', 0.9850), ('a', 0.0120), ..., ('7', 1.84e-08)],
20    [('7', 0.9112), ('1', 0.0889), ..., ('3', 1.20-07)]
21  ]
22  # 46. move white
23  [
24    [('N', 0.9981), ('x', 0.0010), ..., ('6', 1.23e-07)],
25    [('2', 0.9990), ('x', 0.0005), ..., ('e', 1.84e-08)],
26    [('3', 0.9212), ('g', 0.0724), ..., ('c', 1.10-07)],
27    [('5', 0.9732), ('B', 0.0267), ..., ('4', 3.79e-11)]
28  ]
29 ]

```

Listing 4.1: Example of input to the game creation step – *score sheet A*

### 4.3.5 Game Creation

The goal of the Game creation step is to reconstruct the played game from predicted characters. The shorten example of the input to this step corresponding with the *score sheet A* is shown in Listing 4.1. As can be seen, the input is in the form of a three-dimensional array – the outer array represents the whole game, every move is represented as an two-dimensional array with every classified character represented as an array of possible characters along with their probabilities. The outputs and inputs to the individual steps of OCR system are also shown in Figure 4.1.

For the purpose of this thesis, the term *first-guess move* is defined as a string consisting of all most probable predicted characters in the move. The *first-guess moves* are shown in Table 4.1 (*score sheet A*) and in Table 4.2 (*score sheet B*).

The algorithm for game creation consists of three main steps – candidate moves generation, game creation, and game correction, described in this section.

#	W	B	#	W	B	#	W	B
1	d4	g6	21	Ka2	Rb7	41	Rg7	Kxg7
2	c4	Bg7	22	Kd2	Nf5	42	Qc7	Kg5
3	Nc3	d6	23	Na2	Ra2	43	Qxd4	Qxg1
4	d4	d5	24	Rg1	Rg8	44	Ka3	Rxd8
5	Ne3	Nc6	25	Kc2	Rg7	45	Qe5	Kg7
6	d5	Nd4	26	Kg3	Rg7	46	Nxg5	
7	Ba2	c5	27	Rc1	Bd7			
8	O-O	Nf6	28	Qd1	Nxd8			
9	Rh1	O-O	29	c5	Kxc5			
10	g4	g5	30	Kxc5	Bxa4			
11	a3	Na8	31	Ka2	dxg5			
12	Bd3	f5	32	Bxc5	Bg5			
13	Nxd4	exd4	33	Bxe8	Bxd3			
14	Ng2	f4	34	Bxg7	Qa4			
15	g3	Rd7	35	Kg2	Qd5			
16	a4	g5	36	Ka3	Kxg7			
17	Ba3	Bg5	37	Rc7	Kg6			
18	Ka2	g4	38	Nxg4	Qxd4			
19	Ng1	Bf8	39	Qc1	Nxb5			
20	Qxg4	fxg4	40	Ke3	Rg8			

Table 4.4: Candidate moves – *score sheet A***Candidate moves generation**

The goal of this step is to obtain syntactically correct moves ranked due to their score based on the predicted characters. First-guess move  $M$  from all first predicted characters  $m_1, \dots, m_n$  is created. Levenshtein distance  $l$  with modified operation cost is computed between  $M$  and all syntactically correct moves  $Z_1, \dots, Z_n$ . The cost  $C$  of insert, delete and substitute operation is set as follow:

$$C_{\text{ins}}(m_i) = 1, \quad C_{\text{del}}(m_i) = 1 - P(m_i), \quad C_{\text{sub}}(m_i, z_i) = P(m_i) - P(z_i),$$

where  $P(m_i)$  is the predicted probability of the char from first-guess move  $M$  on position  $i$  and  $P(z_i)$  is the probability of char from  $Z$  on position  $i$ . The score  $s$  of the candidate move  $Z$  is computed as:

$$s(Z) = \frac{\sum_{i=1}^n P(m_i)}{n} - l(Z, M).$$

Candidate moves with highest score corresponding with *score sheet A* can be seen in Figure 4.4. Candidate moves with highest score corresponding with *score sheet B* are shown in Figure 4.3.

#	W	B	#	W	B	#	W	B
1	d4	g6	21	Ke2	Rb7	41	Ne8	Rh8
2	c4	Bg7	22	Ne8	Rh8	42	Ng7	Rg8
3	Nc3	d6	23	Ng7	Rg8	43	Ne8	Qa6
4	a4	a5	24	Rg1	Rb8	44	Ng7	Rf8
5	Nf3	Nf6	25	Bc2	Rb7	45	Ne8	Kd7
6	d5	Nb4	26	Rg3	Rb7	46	Ng7	
7	Nd2	c5	27	Ne8	Rh8	47		
8	Nb5	Nf6	28	Qg1	Qd8	48		
9	Rb1	Qd7	29	Ng7	Rg8	49		
10	g4	g5	30	Qxc5	bxa4	50		
11	h3	Ng8	31	Ne8	dxs5	51		
12	Nc7	Kg8	32	Ng7	Bf5	52		
13	Ne8	Nh6	33	Ne8	Rh8	53		
14	Nxg7	Rg8	34	Ng7	Rg8	54		
15	f3	Ra7	35	Ne8	Rh8	55		
16	e4	e5	36	Ng7	Ke7	56		
17	Bd3	b5	37	Ne8	Rg8	57		
18	Kf2	Rh8	38	Ng7	Rh8	58		
19	Ne8	Rg8	39	Ne8	Qa8	59		
20	Ng7	Nxg4	40	Ng7	Rg8	60		

Table 4.5: Game creation – *score sheet A***Game creation**

The most likely move is to pick in every step of the game. When some move is not predicted correctly, there is a high probability that the rest of the game is going to be reconstructed incorrectly as can be seen in Table 4.5 where the created game corresponds with *score sheet A*. To filter very bad moves, chess engine stockfish was tried, but within a reasonable time limit (around 0.2s) it returns nonsense values.

Some other methods for game creation were tried during the development of the application, but the time complexity of the system with these attempts was unmanageable. One of them was to construct tree with more possibilities tried in steps where the score of a move is below some defined threshold. The other was to fix the moves with scores higher than some specified threshold and then fill the gap between the fix moves.

Candidate moves with information about the current main line are stored and created game is presented to a user. The example of the application after the score sheet upload, when the chessboard is return is shown in Figure D.5. The examples of created game corresponding with *score sheet B* is shown in Table 4.6 in first two columns.

### 4.3. Score Sheet Conversion Process

#	W	B		W	B		W	B		W	B
1	e4	e6									
2	d4	d5									
3	Nd2	Nf6									
4	e5	Nfd7									
5	Bd3	c5									
6	c3	Nc6									
7	Ne2	cxd4									
8	cxd4	f6									
9	Nf4	Nxd4									
10	Nh5	Ke7		Dh5	Ke7						
11	Nf3	Rg8		Nf3	Rg8		Ng6	hxg6			
12	exf6	Nxf6		exf6	Nxf6		exf6	Nxf6			
13	Nh5	Kf7		Qxh7	Kf7		Qxh8	Kf7			
14	Nxg7	e5		Qh4	e5		Qh4	e5			
15	Nb3	Nb5		Nh3	Nb5		Nb3	Nb5			Bb4
16	Bd2	Rh8		Bd2	Rh8		Bd2	Be7		Bd2	Bxd2
17	Ne8	Rg8		Qxh8	Be7		Qh8	Qxh8		Kxd3	Naxb3
18	Ng7	Ne4		Qg8	Kxg8		Bxg6	Kg8		axb3	Ne4
19											
20											

Table 4.6: Game correction – *score sheet B*

#### Game correction

The user can correct wrong moves by moving piece on the board. After each such a correction the game is recomputed. The example of corrections made by user is shown in Table 4.6, corresponding to *score sheet B*. Three correction were needed in this case to obtain the correct game. Firstly, the move 10.Nh5 was corrected to Dh5, then the move 11.Nf3 was changed to Ng6 and finally the black 11th move Nb5 was corrected to Bb4.



---

# Evaluation

The evaluation of the system is introduced in this chapter. The created dataset of score sheets is described in Section 5.1. The evaluation of the individual steps of converting score sheets into the PGN file are presented in Sections 5.2 – 5.5. The time complexity of the system is described in Section 5.6. Finally, in Section 5.7 possible improvements to increase the accuracy of the implemented system are described.

## 5.1 Dataset

There is no public dataset with photos of score sheets. Therefore a dataset for testing and evaluation was created. The dataset contains two tournaments – *Ricany-2020* and *Ricany-2019* with 430 photos of score sheets and corresponding PGN files. For better evaluation purposes, score sheets were manually divided into four categories:

- **A** – Score sheets of good quality which do not contain any additional information. Characters are not overlapping but can be slightly touching.
- **B** – Score sheets of medium quality. They can contain overlapping chars, some scratches or additional information written by a player.
- **C** – Score sheets that are hardly readable by humans.
- **L** – Score sheets that are not in Czech algebraic notation.

The evaluation provided in this section is based on score sheets and the PGN files from this dataset. There is no guarantee that the PGN file is complete and does not contain any incorrectly transcribed move. Some errors in the dataset were detected and corrected in the evaluation stage.

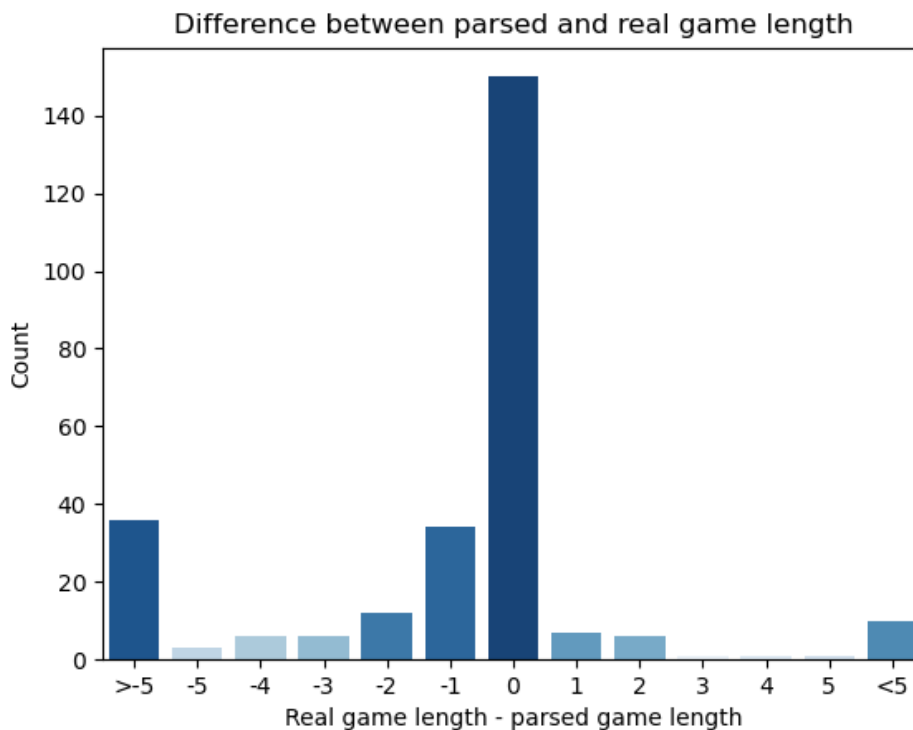


Figure 5.1: Predicted game length subtracted from PGN game length

## 5.2 Move Segmentation

The accuracy of move segmentation is evaluated in this section. The difference between the actual game length and parsed game length is shown in Figure 5.1. The move segmentation is not language-dependent; therefore, score sheets from all categories from tournament *Ricany-2020* were used for move segmentation evaluation. The total number of used score sheets is 273. Factors that lead to incorrect move segmentation can be:

- **Result and signature** – The most common reason for the longer game prediction is the result written right after moves. The result and signature part are then incorrectly considered as moves.
- **Skipped move** – Sometimes players forget to write down the move. When they realize that, they leave empty cells and continue to write the rest. The system could incorrectly recognize that as the end of the game.

The graph also includes the detected 32 games that are not transcribed in the entire length.



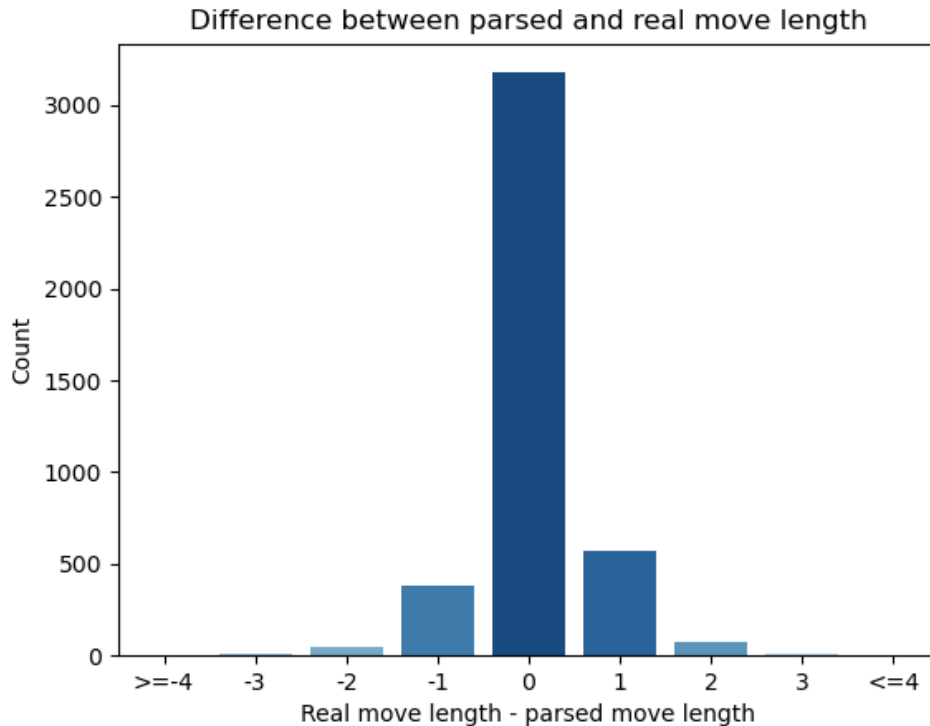


Figure 5.2: Move length

### 5.3 Character Segmentation

Character segmentation was evaluated with use of score sheets from *category A* which contains 60 games segmented into 4264 moves. The difference between the real move length and parsed move length is shown in Figure 5.2. Factors that can cause incorrect move segmentation can be:

- **Special symbols** – Some people write special symbols (+, =, #) and some do not – in the PGN file are symbols + and # always written.
- **Additional information** – Some players write some additional information into the score sheet as a time spend on move, that can be wrongly considered as a part of a move.
- **Different characters size** – The system depends on the proper size of characters. When a player writes some characters significantly bigger than others, the system tends to over-segment them.
- **Overlapping characters** – The system often fails to correctly segment overlapping characters.

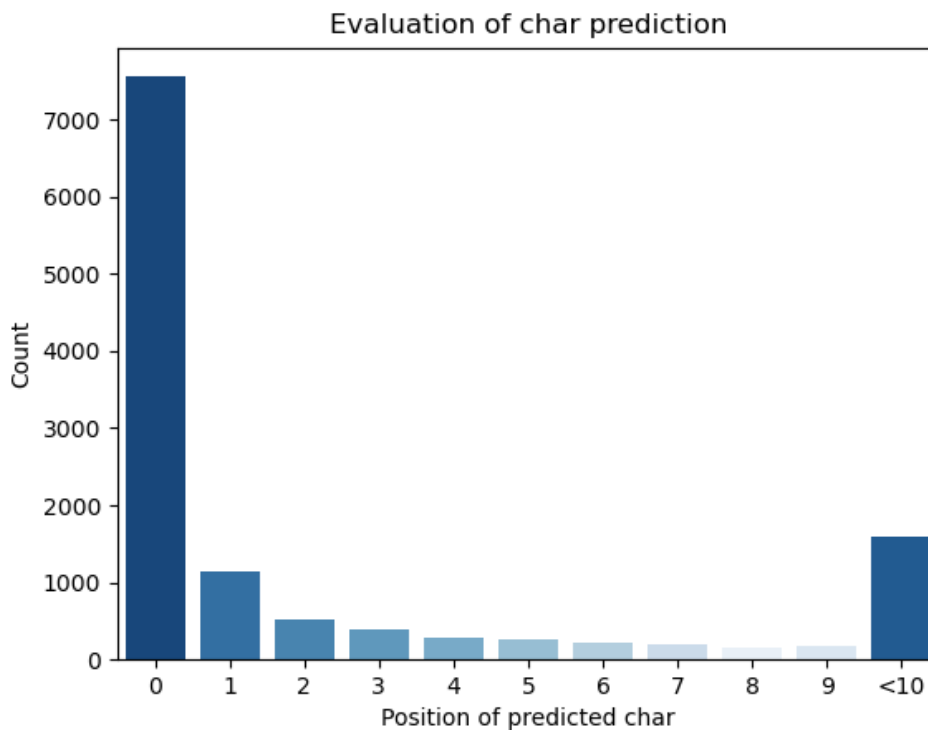


Figure 5.3: Character classification

## 5.4 Character Classification

Character classification was evaluated with use of score sheets from *category A* which contains 60 games and is segmented into 4264 moves and 12224 characters. The graph of the position of the actual character in the array of possible characters returned by a CNN model is shown in Figure 5.3. In Figure 5.4 there is shown a confusion matrix of predicted and real chars. Factors that can cause incorrect character classification can be:

- **Cursive letters** – The model fails to recognize handwritten cursive letters such as **b**, **e**, **f** and **h** as described in Section 4.3.4.
- **Error in segmentation** – The classification step depends on the previous step of char segmentation. Poorly segmented characters lead to incorrect classification.
- **Player’s mistake** – Players sometimes write wrong move. The example of this case was shown in Figure 4.7.

## 5.4. Character Classification

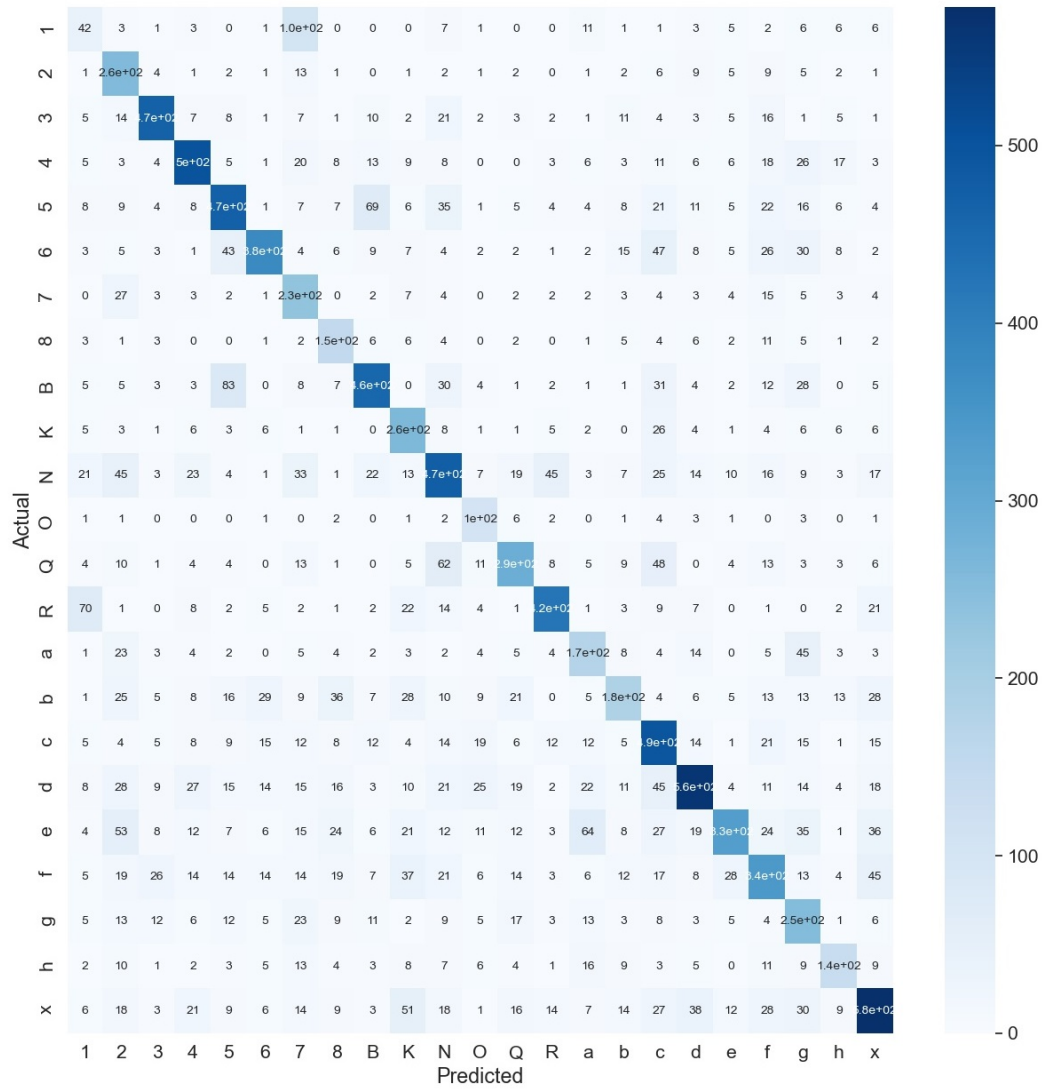


Figure 5.4: Confusion matrix for char prediction

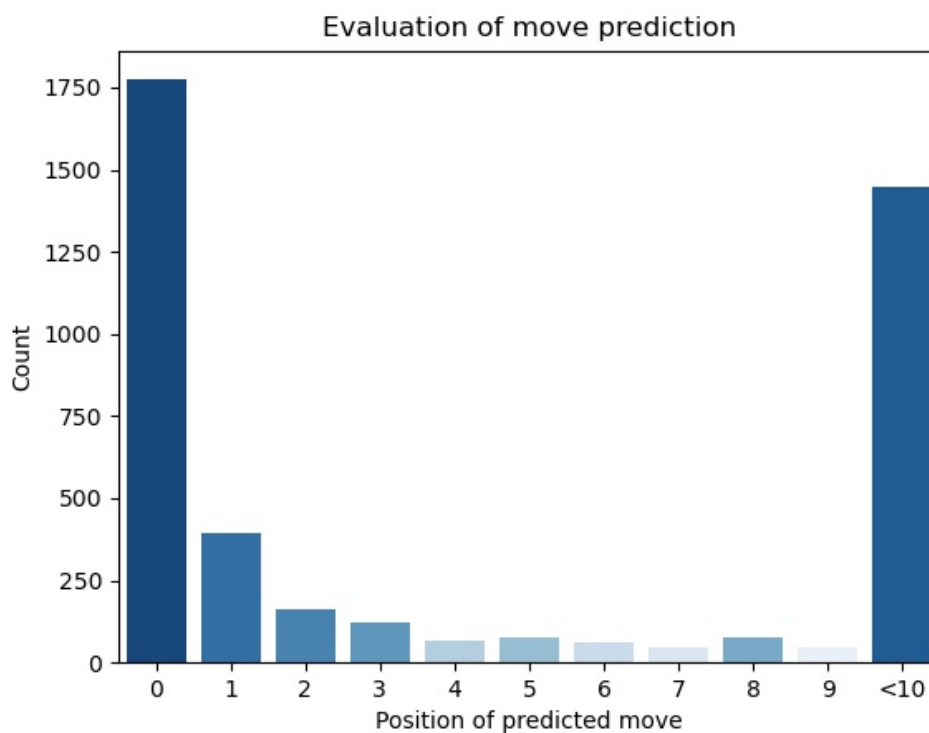


Figure 5.5: Move prediction

## 5.5 Move Prediction

The evaluation of the moves created by candidate moves generation is shown in this section. The output of candidate moves generation step is an array of moves sorted by their score. The position of the actual move in the candidate move array is shown in Figure 5.5. The Levenshtein distance between the actual move and first-guess move is in Figure 5.6. The reason for the bad result of this step could happen by fact that when some character is predicted wrongly but there are lots of other possibilities.

*Hypothetical Example:* The move Nb5 occurs in a game, and b is wrongly classified as 3. It is clear that something went wrong – either the move was segmented incorrectly and the move should be pawn move with the number on the second position or 3 was misclassified. With the high classification probability N and 5, the misclassification of 3 is more likely. But there are still eight options that need to be taken into account – Na5, Nb5, ..., Nh5.

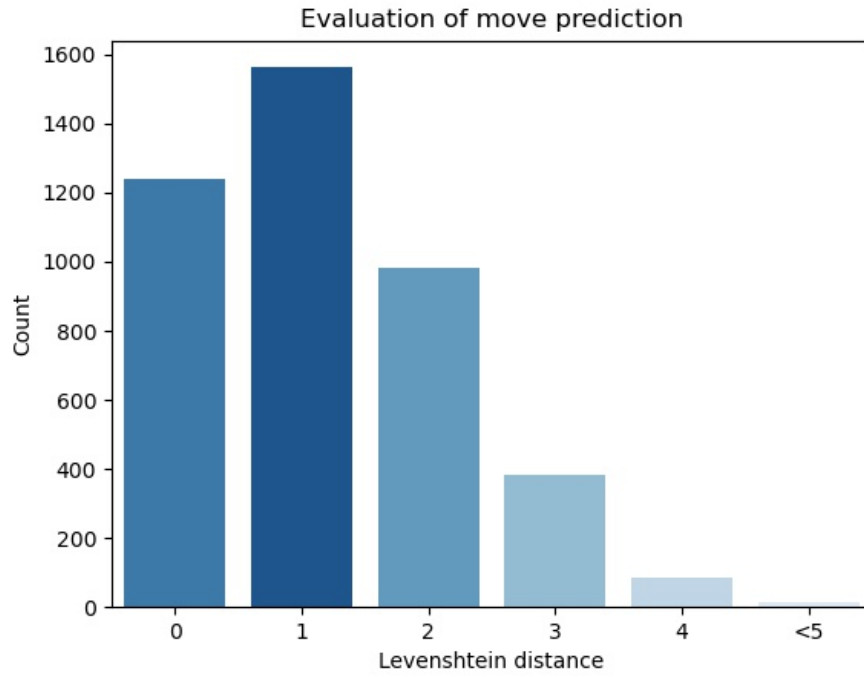


Figure 5.6: Levenshtein distance between the actual move and first-guess move

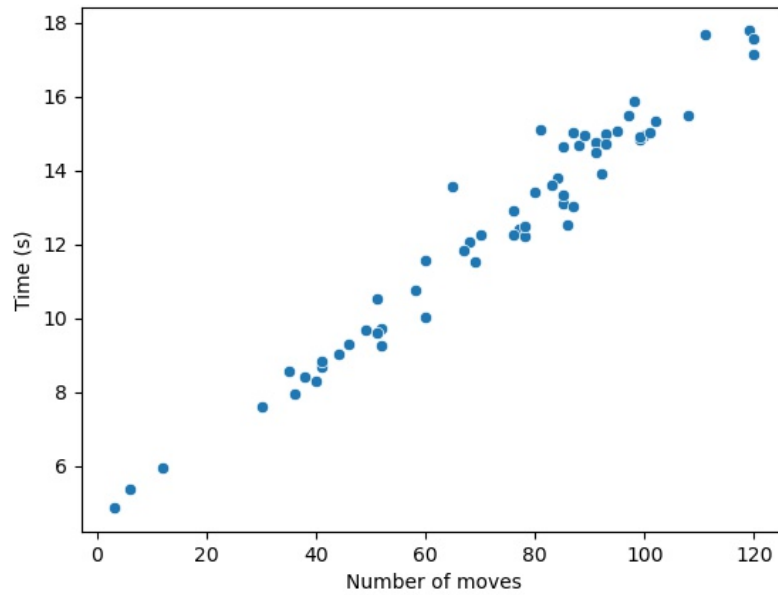


Figure 5.7: Time complexity

## 5.6 Time Complexity

The time complexity of individual steps was measured on 65 games with average 65.6 moves per game and 191.98 chars per game:

- **Preprocessing** – The average runtime of preprocessing step is around 3.4 seconds. The most expensive part of this step is the SIFT algorithm that takes around 3 seconds. The time complexity of this step depends only on the size of the template and the uploaded score sheet.
- **Character segmentation** – The complexity of the character segmentation is given by the number of moves and characters. The average runtime of this step is around 1.54 seconds.
- **Character classification** – The complexity of character classification step is given by number of characters. The average runtime is around 4.4 seconds.
- **Game creation** – The game creation step runtime is around 4.4 seconds and depends on a number of moves.

The full duration of process is around 14 seconds, the dependence of the time complexity on the number of moves is shown in Figure 5.7. The time complexity was measured on the notebook with 16GB RAM and 4-cores i7 8th Generation. The deployed application on pythonanywhere is significantly slower.

## 5.7 Possible Improvements

With the system as complex as this, there is a place for many improvements – whether functional ones or ones improving accuracy. Some suggestions are introduced in this section.

- **Multilingual** – The actual version of the system supports only the Czech algebraic notation. The model predicts the chars with respect to Czech algebraic notation and then translates them into English algebraic notation. The new model could be trained on chars that can occur in other specific languages and add the translation between that language and English algebraic notation to support more languages. In this case, there would be an additional step added; the user would choose the language of the score sheet before uploading it. Automatic recognition would also be an option but would lead to lower accuracy.
- **More score sheets uploaded** – In the current version of the application uploading only one score sheet is supported. The support for two same score from both players from the same game sheets upload could lead to better accuracy.

- **Opening library** – The chess game consists of three parts – opening, middle game, and end game. Nowadays, the opening part is highly theoretical, and players usually have some opening repertoire that they usually know very well – sometimes, the known and analyzed position goes even beyond the 20 moves. Games from a database could improve the prediction of the moves in the opening part of the game. The games in the database should be organized into a tree structure to optimize searching time.





---

## Conclusion

The goal of this thesis was to create a tool that would help players with converting score sheets to digitalized form. The tool should allow uploading a photo or a scan of the score sheet. After uploading, the tool should detect the score sheet in the image, segment individual moves and convert them into the algebraic notation with the possibility to correct wrongly classified moves by the user. The user should also be allowed to download the game as a PGN file.

All these requirements were successfully accomplished. The tool was implemented as a web application that was deployed on a server. The user can upload the score sheet then an application explores a chessboard along with the game in algebraic notation. The user can replay the game on the chessboard, correct wrong moves by moving pieces on the board and download the game as the PGN file.

Three existing applications with the same goal were tried. Two of them simplify the task by using the particular version of the score sheet. The third application was available only as a mobile app and only for English algebraic notation. For some specific cases they can be helpful, but they are not sufficient for broader use.

Various approaches and the current state-of-the-art in the OCR were investigated to find the proper methods for developing the application. Along with that, the dataset containing 430 photos of score sheets with corresponding PGN files was created to evaluate the final application.

During the design and implementation of the application, the author of this thesis had some ideas for improvements outside the scope of this thesis. These improvements are mentioned at the end of the thesis for the future extension of the application. With these improvements, the assumption is that the application would be a valuable tool to optimize time spent on converting score sheets into a PGN file.



---

# Bibliography

- [1] FIDE LAWS of CHESS [Online]. [cit. 2021-02-03]. Available from: <https://www.fide.com/FIDE/handbook/LawsOfChess.pdf>
- [2] Smigielski, M. From chess score sheet to ICR with OpenCV and image recognition [Online]. 2017 [cit. 2021-02-03]. Available from: <https://medium.com/@mareksmigielski/from-chess-score-sheet-to-icr-with-opencv-and-image-recognition-f7bed2cc3de4>
- [3] Christensson, P. OCR Definition *TechTerms* [Online]. 2018 [cit. 2021-03-03]. Available from: <https://techterms.com/definition/ocr>
- [4] Islam, N.; Islam, Z.; et al. A Survey on Optical Character Recognition System. *CoRR*, volume abs/1710.05703, 2017, 1710.05703. Available from: <http://arxiv.org/abs/1710.05703>
- [5] Wazirali, R.; Slehat, S.; et al. Objective Quality Metrics in Correlation with Subjective Quality Metrics for Steganography. 07 2015. Available from: [https://www.researchgate.net/figure/Secret-message-hidden-in-different-color-level-of-Lenna\\_fig3\\_287195029](https://www.researchgate.net/figure/Secret-message-hidden-in-different-color-level-of-Lenna_fig3_287195029)
- [6] Digital images and image formats *Universitetet I Oslo* [Online]. [cit. 2021-02-03]. Available from: <https://www.uio.no/studier/emner/matnat/math/MAT-INF1100/h08/kompendiet/images.pdf>
- [7] Image Thresholding *OpenCV* [Online]. [cit. 2021-02-03]. Available from: [https://docs.opencv.org/master/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html)
- [8] Otsu thresholding—image binarization *HBY coding academic* [Online]. 2019 [cit. 2021-03-05]. Available from: <https://medium.com/@hbyacademic/otsu-thresholding-4337710dc519>

- [9] Bansal, K.; Kumar, R. K-Algorithm A Modified Technique for Noise Removal in Handwritten Documents. *CoRR*, volume abs/1306.1462, 2013, 1306.1462. Available from: <http://arxiv.org/abs/1306.1462>
- [10] Subudhi, R.; Sahu, B.; et al. A Novel Noise Reduction Method For OCR System. 12 2013.
- [11] Pulak Purkait, J. Optical Handwritten Character/Numeral Recognition. Available from: <https://www.isical.ac.in/~vlrg/sites/default/files/Pulak/Off-Line%20Handwritten%20OCR.pdf>
- [12] Morphological Image Processing. [Online]. [cit. 2021-03-06]. Available from: <https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm>
- [13] Fisher, R.; Perkins, S.; et al. Pixel Connectivity. [Online]. 2003 [cit. 2021-03-10]. Available from: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/connect.htm>
- [14] Abu-Ain, W.; Abdullah, S. N. H. S.; et al. Skeletonization Algorithm for Binary Images. *Procedia Technology*, volume 11, 2013: pp. 704–709, ISSN 2212-0173, doi:<https://doi.org/10.1016/j.protcy.2013.12.248>, 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013. Available from: <https://www.sciencedirect.com/science/article/pii/S2212017313004027>
- [15] How to Get an 80% ROI over Your OCR Implementation *docdigitizer* [Online]. [cit. 2021-03-11]. Available from: <https://www.docdigitizer.com/2019/07/18/calculate-roi-of-ocr-initiatives/>
- [16] Hines, K. Learning to Read: Computer Vision Methods for Extracting Text from Images. *CapitalOne*. [Online]. 2019 [cit. 2021-02-04]. Available from: <https://www.capitalone.com/tech/machine-learning/learning-to-read-computer-vision-methods-for-extracting-text-from-images/>
- [17] Kaur, A.; Baghla, S.; et al. Study Of Various Character Segmentation Techniques For Handwritten Off-Line Cursive Words: A Review. *International Journal of Advances in Science, Engineering and Technology (IJASEAT)*, pp. 154-158, Volume-3, Issue-3.
- [18] Ptak, R.; Żygadło, B.; et al. Projection-Based Text Line Segmentation with a Variable Threshold. *International Journal of Applied Mathematics and Computer Science*, volume 27, 03 2017, doi:10.1515/amcs-2017-0014.
- [19] Brodowska, M. Oversegmentation methods for character segmentation in off-line cursive handwritten word recognition : an overview. *Schedae Informaticae*, volume 20, 2012: pp. 43–65.

- 
- [20] Casey, R.; Lecolinet, E. A Survey of methods and strategies in character segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, volume 18, 08 1996: pp. 690 – 706, doi:10.1109/34.506792.
- [21] Märgner, V.; Abed, H. *Guide to OCR for Arabic Scripts*. SpringerLink : Bücher, Springer London, 2012, ISBN 9781447140726. Available from: <https://books.google.cz/books?id=1BASqp1ImRIC>
- [22] Nashwan, F. M. A.; Rashwan, M. A. A.; et al. A Holistic Technique for an Arabic OCR System. *Journal of Imaging*, volume 4, no. 1, 2018, ISSN 2313-433X, doi:10.3390/jimaging4010006. Available from: <https://www.mdpi.com/2313-433X/4/1/6>
- [23] Choudhary, A. A Review of Various Character Segmentation Techniques for Cursive Handwritten Words Recognition. *Int J Inf Comput Technol*, 2014, 4.6: 559-564.
- [24] Ploetz, T.; Fink, G. Markov models for offline handwriting recognition: A survey. *IJDAR*, volume 12, 12 2009: pp. 269–298, doi:10.1007/s10032-009-0098-4.
- [25] TAWDE, Gaurav Y.; KUNDARGI, Jayshree. An overview of feature extraction techniques in ocr for indian scripts focused on offline handwriting. *International Journal of Engineering Research and Applications*, 2013, 3.1: 919-926.f.
- [26] Vijay, P.; Yumnam, J. A study on method of feature extraction for Handwritten Character Recognition. *Indian Journal of Science and Technology*, volume 6, 03 2013: pp. 174–178.
- [27] Wang, L. *Support vector machines: theory and applications*, volume 177. Springer Science & Business Media, 2005.
- [28] Harrison, O. Machine Learning Basics with the K-Nearest Neighbors Algorithm. *Towards Data Science*. [Online]. 2018 [cit. 2021-04-03]. Available from: <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
- [29] The Ultimate Guide to Decision Trees for Machine Learning *keboola* [Online]. 2020 [cit. 2021-04-03]. Available from: <https://www.keboola.com/blog/decision-trees-machine-learning>
- [30] Himnashu, S. Activation Functions : Sigmoid, ReLU, Leaky ReLU and Softmax basics for Neural Networks and Deep Learning. [Online]. 2019 [cit. 2021-04-03]. Available from: <https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e>

- [31] Isaksson, M. Four Common Types of Neural Network Layers. *Towards Data Science*. [Online]. 2020 [cit. 2021-04-11]. Available from: <https://towardsdatascience.com/four-common-types-of-neural-network-layers-c0d3bb2a966c>
- [32] Stanford. Convolutional Neural Networks (CNNs / ConvNets) [Online]. 2021 [cit. 2021-04-15]. Available from: <https://cs231n.github.io/convolutional-networks/>
- [33] Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, volume 29, no. 6, 2012: pp. 141–142.
- [34] Digit Recognition using SVM with 98% accuracy [Online]. 2018 [cit. 2021-04-15]. Available from: <https://www.kaggle.com/sanesanyo/digit-recognition-using-svm-with-98-accuracy>
- [35] k-NN for MNIST Classification *Back-Prop* [Online]. 2019 [cit. 2021-04-03]. Available from: [https://www.back-prop.com/deep\\_learning/knn/mnist/2019/05/16/knn\\_classifier/](https://www.back-prop.com/deep_learning/knn/mnist/2019/05/16/knn_classifier/)
- [36] BERNARD, Simon; ADAM, Sébastien; HEUTTE, Laurent. Using random forests for handwritten digit recognition. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. IEEE, 2007. p. 1043-1047.
- [37] Deotte, C. Accuracy=99.75% using 25 Million Training Images!! *25 Million Images! [0.99757] MNIST* [Online]. 2018 [cit. 2021-02-03]. Available from: <https://www.kaggle.com/cdeotte/25-million-images-0-99757-mnist>
- [38] Bassil, Y.; Alwani, M. OCR Post-Processing Error Correction Algorithm using Google Online Spelling Suggestion. *CoRR*, volume abs/1204.0191, 2012, 1204.0191. Available from: <http://arxiv.org/abs/1204.0191>
- [39] Edward, M. Measure distance between 2 words by simple calculation *Towards Data Science* [Online]. 2018 [cit. 2021-03-05]. Available from: <https://towardsdatascience.com/measure-distance-between-2-words-by-simple-calculation-a97cf4993305>
- [40] Afi, H.; Qiu, Z.; et al. *Using SMT for OCR Error Correction of Historical Texts*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, 962–966 pp. Available from: <https://www.aclweb.org/anthology/L16-1153>
- [41] Andreas, M. M. Simple OCR with Tesseract *Towards Data Science* [Online]. 2020 [cit. 2021-04-15].

- 
- [42] Rakshit, S.; Basu, S. Recognition of Handwritten Roman Script Using Tesseract Open source OCR Engine. *CoRR*, volume abs/1003.5891, 2010, 1003.5891. Available from: <http://arxiv.org/abs/1003.5891>
- [43] Flask. [Online]. [cit. 2021-04-15]. Available from: <https://palletsprojects.com/p/flask/>
- [44] What is WSGI? [Online]. [cit. 2021-04-16]. Available from: <https://wsgi.readthedocs.io/en/latest/what.html>
- [45] A JavaScript library for building user interfaces *React*. [Online]. [cit. 2021-04-16]. Available from: <https://reactjs.org/>
- [46] What is PostgreSQL? [Online]. [cit. 2021-04-15]. Available from: <https://aws.amazon.com/rds/postgresql/what-is-postgresql/>
- [47] About *OpenCv*. [Online]. [cit. 2021-04-13]. Available from: <https://opencv.org/about/>
- [48] Chandra, R. The What's What of Keras and TensorFlow *Keras and TensorFlow* [Online]. 2019 [cit. 2021-04-12]. Available from: <https://www.upgrad.com/blog/the-whats-what-of-keras-and-tensorflow/>
- [49] Glass, K. python-chess: a chess library for Python [Online]. 2014 [cit. 2021-04-20]. Available from: <https://python-chess.readthedocs.io/en/latest/>
- [50] Smart PNG and JPEG compression *voordmedia*. [Online]. [cit. 2021-04-20]. Available from: <https://tinypng.com/>
- [51] pythonanywhere. [Online]. [cit. 2021-04-20]. Available from: <https://www.pythonanywhere.com/>
- [52] Ning, M. SIFT(Scale-invariant feature transform) *Towards Data Science Upgrade* [Online]. 2019 [cit. 2021-03-11]. Available from: <https://towardsdatascience.com/sift-scale-invariant-feature-transform-c7233dc60f37>
- [53] Feature Matching with FLANN *OpenCv*. [Online]. [cit. 2021-03-11]. Available from: [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_feature2d/py\\_matcher/py\\_matcher.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html)
- [54] Agarwal, S. Homography - And how to calculate it? [Online]. 2020 [cit. 2021-03-11]. Available from: <https://medium.com/all-things-about-robotics-and-computer-vision/homography-and-how-to-calculate-it-8abf3a13ddc5>

- [55] Singh, T. R.; Roy, S.; et al. A New Local Adaptive Thresholding Technique in Binarization. *CoRR*, volume abs/1201.5227, 2012, 1201.5227. Available from: <http://arxiv.org/abs/1201.5227>
- [56] 3.5 Million Chess Games *Kaggle: Your Machine Learning and Data Science Community* [Online]. 2019 [cit. 2021-04-09]. Available from: <https://www.kaggle.com/milesh1/35-million-chess-games>
- [57] Krizhevsky, A.; Sutskever, I.; et al. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, volume 25, 2012: pp. 1097–1105.
- [58] Cohen, G.; Afshar, S.; et al. EMNIST: an extension of MNIST to handwritten letters. 2017. Available from: <http://arxiv.org/abs/1702.05373>
- [59] Brownlee, J. How to Configure Image Data Augmentation in Keras [Online]. 2019 [cit. 2021-04-09]. Available from: <https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/>
- [60] Berger, T. What We Lose With the Decline of Cursive *edutopia* [Online]. 2017 [cit. 2021-03-05]. Available from: <https://www.edutopia.org/article/what-we-lose-with-decline-cursive-tom-berger>



## Examples from dataset

## A. EXAMPLES FROM DATASET

1/1



### KLUB ŠACHISTŮ ŘÍČANY 1925



 BIOMEDICA
  ŘÍČANY

Utkání: OPEN ŘÍČANY 2020

Soutěž: KPSŠS Datum: 22.8 Kolo: 1

 JAROMÍR SKÁLA	 SEBASTIAN PLISCHKI
FIDE ELO <u>1891</u> N-ELO	FIDE ELO <u>2356</u> N-ELO

	BÍLÝ	ČERNÝ	BÍLÝ	ČERNÝ	BÍLÝ	ČERNÝ		
1	d4	g6	21	Kx2	Vx7	41	Vx7+	Kxg7
2	c4	Sa7	22	Kd2	Jx6	42	Dc7+	Kg6
3	Jc3	d6	23	Jx2	Vx2	43	Dx4	Dx6
4	e4	e5	24	Vg1	Vx8	44	Ka3	Vx8
5	Jx3	Jc6	25	Kc2	Vx7	45	Dx5	Kg7
6	d5	Jd4	26	Kx3	Va7	46	Dg5	
7	Sx2	c5	27	Vc1	Sd7	47		
8	O-O	Jx6	28	Dx1	Dx8	48	0:1	
9	Vx1	O-O	29	c5	bxc5	49		
10	fx4	fx6	30	bxc5	Sxa4	50		
11	h3	Jx8	31	Ka2	dx5	51		
12	Sd3	fx5	32	Sxc5	Sfx5	52		
13	Jxd4	cx4	33	Sfx8	Sxd3	53		
14	Jx2	fx4	34	Sxg7	Dat	54		
15	fx3	Vx7	35	Kx2	Dx5	55		
16	a4	g5	36	Ka3	Kxg7	56		
17	Sa3	fx5	37	Vc7	Kg6	57		
18	Kx2	g4	38	Jx4	fx4	58		
19	Jg1	Sx8	39	Dc1	Dx6	59		
20	fxg4	fxg4	40	Kx3	Vx8	60		



STAR

Čas:

karna@tria.cz, www.tria.cz

Figure A.1: Category A – score sheet A

1. d4 g6 2. c4 Bg7 3. Nc3 d6 4. e4 e5 5. Nf3 Nc6 6. d5 Nd4 7. Be2 c5 8. O-O Nf6  
 9. Rb1 O-O 10. b4 b6 11. h3 Ne8 12. Bd3 f5 13. Nxd4 cxd4 14. Ne2 f4 15. f3 Rf7  
 16. a4 g5 17. Ba3 h5 18. Kf2 g4 19. Ng1 Bf8 20. hxg4 hxg4 21. Ke2 Rh7 22. Kd2  
 Nf6 23. Ne2 Rh2 24. Rg1 Rb8 25. Kc2 Rb7 26. Kb3 Rg7 27. Rc1 Bd7 28. Qf1 Qe8  
 29. c5 bxc5 30. bxc5 Bxa4+ 31. Ka2 dxc5 32. Bxc5 Bb5 33. Bxf8 Bxd3 34. Bxg7 Qa4+  
 35. Kb2 Qb5+ 36. Ka3 Kxg7 37. Rc7+ Kg6 38. Nxf4+ exf4 39. Qc1 Qa6+ 40. Kb3 Rh8  
 41. Rg7+ Kxg7 42. Qc7+ Kg6 43. Qxf4 Qb6+ 44. Ka3 Rb8 45. Qf5+ Kg7 46. Qg5+ 0-1

Figure A.2: Moves from PGN corresponding with examples from Category A (score sheet A) and Category L

115



## KLUB ŠACHISTŮ ŘÍČANY 1925



 BIOMEDICA
  ŘÍČANY

Utkání: OPEN ŘÍČANY A

Soutěž: Datum: Kolo: 1

♙ MEJZR J.	♜ SKÝPALA O.
FIDE ELO	N-ELO
FIDE ELO	N-ELO

	BÍLÝ	ČERNÝ		BÍLÝ	ČERNÝ		BÍLÝ	ČERNÝ
1	e4	e6	21			41		
2	d4	d5	22			42		
3	d2	f6	23			43		
4	e5	f7	24			44		
5	d3	c5	25			45		
6	c3	c6	26			46		
7	e2	cx d4	27			47		
8	cx d4	f6	28			48		
9	f5	dx d4	29			49		
10	dh5	ke7	30			50		
11	g3	hxg3	31			51		
12	exf6	xxf6	32			52		
13	dxh8	kf7	33			53		
14	dh4	e5	34			54		
15	g3	sb4	35			55		
16	sd2	sxd2	36			56		
17	kxd2	xxb3	37			57		
18	axb3	de4	38			58		
19			39			59		
20			40			60		





Čas:


Figure A.3: Category A – score sheet B

1. e4 e6 2. d4 d5 3. Nd2 Nf6 4. e5 Nf7 5. Bd3 c5 6. c3 Nc6 7. Ne2 cxd4 8. cxd4 f6 9. Nf4 Nxd4 10. Qh5+ Ke7 11. Ng6+ hxg6 12. exf6+ Nxf6 13. Qxh8 Kf7 14. Qh4 e5 15. Nb3 Bb4+ 16. Bd2 Bxd2+ 17. Kxd2 Nxb3+ 18. axb3 Ne4 0-1


Figure A.4: Moves from PGN corresponding with examples from Category A (score sheet B)



A. EXAMPLES FROM DATASET

1/9



## KLUB ŠACHISTŮ ŘÍČANY 1925





 BIOMEDICA
  ŘÍČANY

Utkání:  Pohár města Říčany OPEN A

Soutěž: ..... Datum: 22.8.2020 Kolo: 1

♔ Strnad Michal	♚ Staněk Ondřej
FIDE ELO	N-ELO
FIDE ELO	N-ELO

	BÍLÝ	ČERNÝ		BÍLÝ	ČERNÝ		BÍLÝ	ČERNÝ
1	e4	c6	21	Vh1	g5	41		
2		d4	22	Dd1	kg6	42		
3	exd5	cxr5	23	Ve5	d2	43		
4		c4	24	f4	f6	44		
5	Jc3	Jc6	25	EG+	kh5	45		
6		Jf3	26	Vg1	Dc4	46		
7	cxr5	Jax5	27	Ve2	Vh8	47		
8		Db3	28	Dd6	Dc4	48		
9	gxf3	e6	29	Vh2+	Vh3	49		
10		Dxb7	30	Dgbt		50		
11	Sb5+	Jxb5	31			51		
12		Dcb+	32			52		
13	Dxb5	Vb8	33	1-0		53		
14		Jxb+	34			54		
15	Dc2	Kf6	35			55		
16	Sd2	Sb4	36	Strnad		56		
17	0-0-0	Dc8	37			57		
18	kb1	Df5	38			58		
19	Ka1	Sd2	39			59		
20	Dxd2	Ved8	40			60		





Čas:


[www.sachyricany.cz](http://www.sachyricany.cz)
[www.openricany.cz](http://www.openricany.cz)



Figure A.5: Category B

1/3





## KLUB ŠACHISTŮ ŘÍČANY 1925



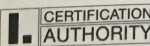
 BIOMEDICA
  ŘÍČANY


Utkání: .....

Soutěž: ..... Datum: ..... Kolo: .....

 KOPER 1971	 MORAVEC
FIDE ELO	N-ELO
FIDE ELO	MORAVEC N-ELO 2236

	BÍLÝ	ČERNÝ		BÍLÝ	ČERNÝ		BÍLÝ	ČERNÝ
1	d4	d5	21	JF7	g5	41	KF1	KF2
2	c4	c9	22	Kg1	g4	42	Vg2	Vg3
3	cd	cd	23	Jg3	sg6	43	KF1	KF2
4	JF3	c6	24	Jg3	Dx6	44	Vg2	Vg3
5	JF3	JF6	25	Jg2	Jg2	45	Vg2	Jg
6	g5	g7	26	JF7	Vg3	46		
7	c3	Jb17	27	b7	JF6	47	0	1
8	sd3	h6	28	g5	Vg8	48		
9	ch4	Jh5	29	bc	bc	49		
10	Sx6	Dxc	30	Vb6	Vg8	50		
11	oo	oo	31	Dg2	kh	51		
12	Dc2	Vb7	32	Vb7	Sh5	52		
13	h3	Vf8	33	Jg2	Vg8	53		
14	Vg6	o5	34	Jg3	Vg8	54		
15	g4	b6	35	Dc2	Jh4	55		
16	h3	Jf4	36	Vg3	Vg8	56		
17	Sx6	dl	37	Xg	Jh	57		
18	Jg2	Sx5	38	JF7	g9	58		
19	Vg1	Jg5	39	Dx6	Jg8	59		
20	kh1	Jh4	40	Vg7	Jh2	60		






Čas:


www.sachyricany.cz
www.openricany.cz


Figure A.6: Category C

A. EXAMPLES FROM DATASET




# KLUB ŠACHISTŮ ŘÍČANY 1925







BIOMEDICA



ŘÍČANY


Utkání: .....

Soutěž: ..... Datum: 22.8 Kolo: 1

 Skala, 7	 Plisckai, 5
FIDE ELO      N-ELO	FIDE ELO      N-ELO

	BÍLÝ	ČERNÝ		BÍLÝ	ČERNÝ		BÍLÝ	ČERNÝ
1	d4	g6	21	Med	Tb7	41	Tg7	Kg7
2	c4	Lg7	22	Kd2	Sx6	42	Dg7	Lg6
3	Sa3	d6	23	Sd7	Tb2	43	Dx7d	Dg7
4	e4	e5	24	Tg1	Tb8	44	Kg3	Tb5
5	Sf3	Sd6	25	Kd8	Tb7	45	Df5+	Kg7
6	d5	Sd4	26	Kf3	Tg7	46	Dg5	Kg7
7	Ld2	c5	27	Tc1	Ld7	47		
8	0-0	Sf6	28	Df1	Dd8	48	0:1	
9	Tb1	0-0	29	c3	Bxc5	49		
10	b4	b6	30	Bxc5	Lxc4	50		
11	h3	Se8	31	hax	dxc5	51		
12	Ld3	f5	32	Lxc5	Lb5	52		
13	Sd4	Ecd4	33	Lxc8	Ld3	53		
14	Se2	f4	34	Lxc8	0d4	54		
15	f3	Td7	35	Lb2	0b5	55		
16	ad4	g5	36	Ka3	Kxc7	56		
17	La3	h5	37	Td7	Kg6	57		
18	Kf2	g4	38	Sxf4	exf4	58		
19	Sg1	Lb8	39	Dc4	Dd6	59		
20	h5	h5	40	Kf3	Td8	60		

I. CERTIFICATION AUTHORITY



Čas: 0

Figure A.7: Category L

# Model Comparison

## B. MODEL COMPARISON

---

```

Model: "sequential_25"
Layer (type)                Output Shape                Param #
=====
conv2d_94 (Conv2D)          (None, 26, 26, 32)         320
batch_normalization_35 (Batc (None, 26, 26, 32)         128
conv2d_95 (Conv2D)          (None, 24, 24, 32)         9248
batch_normalization_36 (Batc (None, 24, 24, 32)         128
conv2d_96 (Conv2D)          (None, 12, 12, 32)         25632
batch_normalization_37 (Batc (None, 12, 12, 32)         128
dropout_25 (Dropout)        (None, 12, 12, 32)         0
conv2d_97 (Conv2D)          (None, 10, 10, 64)         18496
batch_normalization_38 (Batc (None, 10, 10, 64)         256
conv2d_98 (Conv2D)          (None, 8, 8, 64)           36928
batch_normalization_39 (Batc (None, 8, 8, 64)           256
conv2d_99 (Conv2D)          (None, 4, 4, 64)           102464
batch_normalization_40 (Batc (None, 4, 4, 64)           256
dropout_26 (Dropout)        (None, 4, 4, 64)           0
conv2d_100 (Conv2D)         (None, 1, 1, 128)          131200
batch_normalization_41 (Batc (None, 1, 1, 128)          512
flatten_24 (Flatten)        (None, 128)                 0
dropout_27 (Dropout)        (None, 128)                 0
dense_42 (Dense)            (None, 23)                  2967
=====
Total params: 328,919
Trainable params: 328,087
Non-trainable params: 832

```

Figure B.1: Customized LeNet-5 model summary



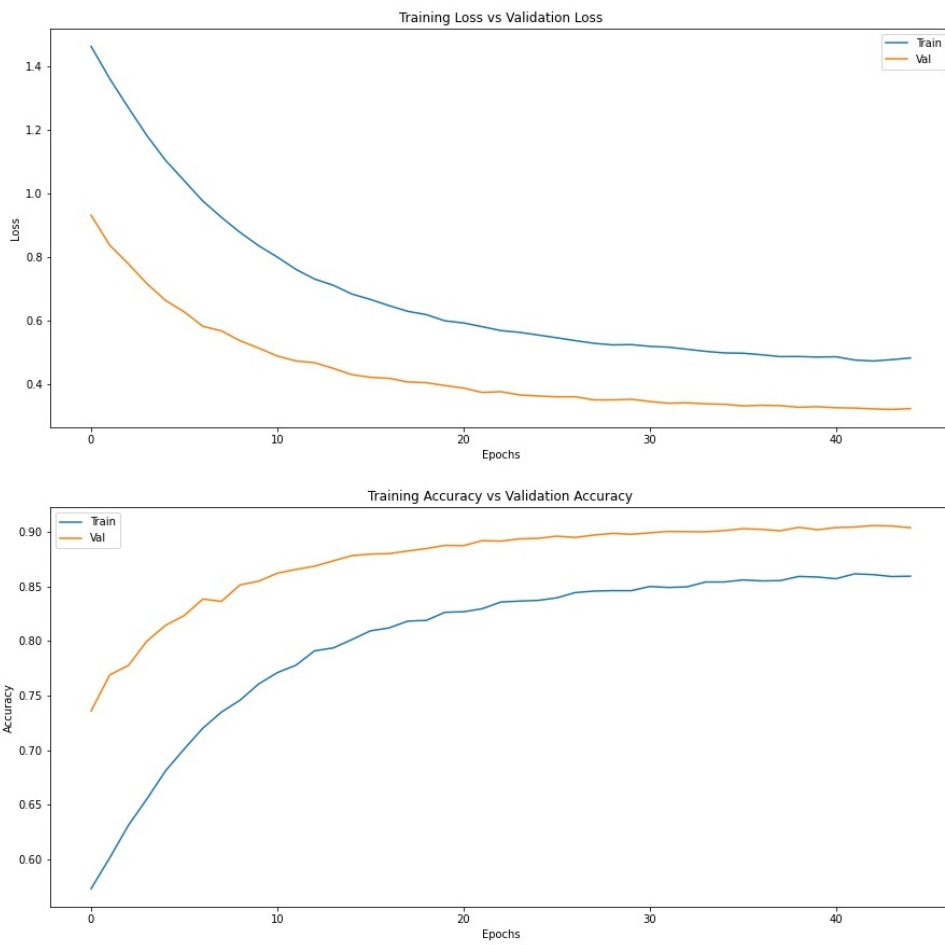


Figure B.2: LeNet-5

## B. MODEL COMPARISON

---

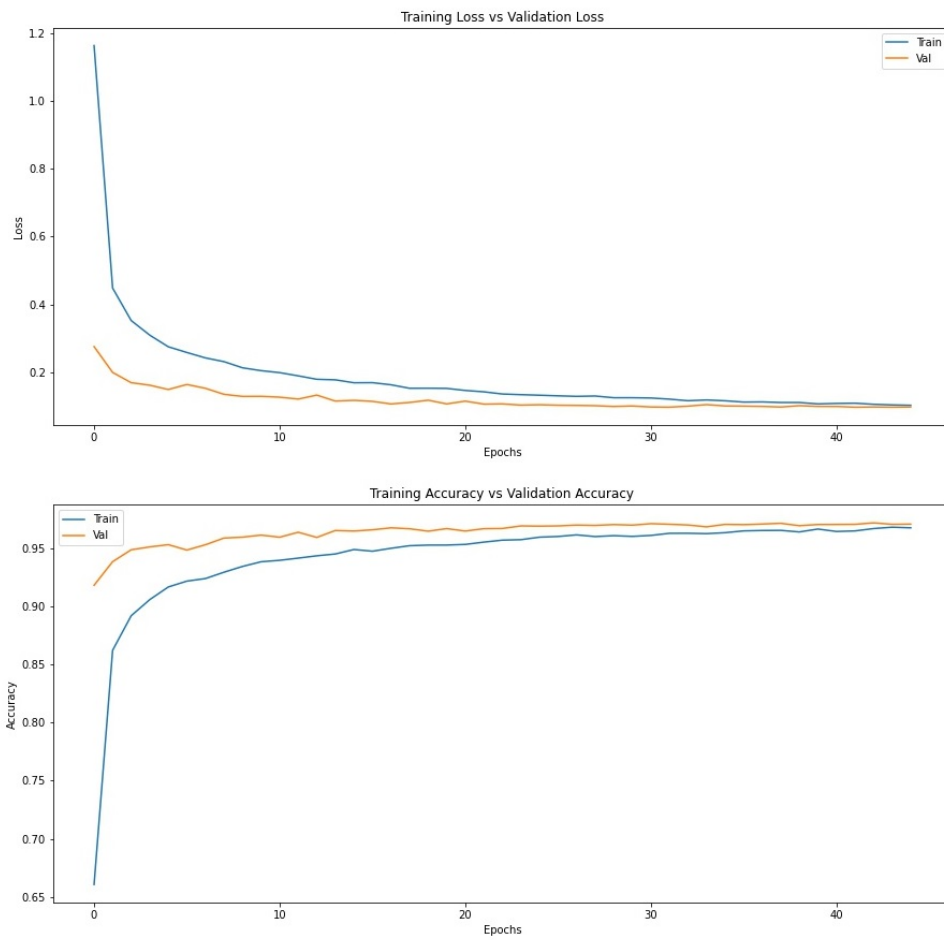


Figure B.3: Customized LeNet-5

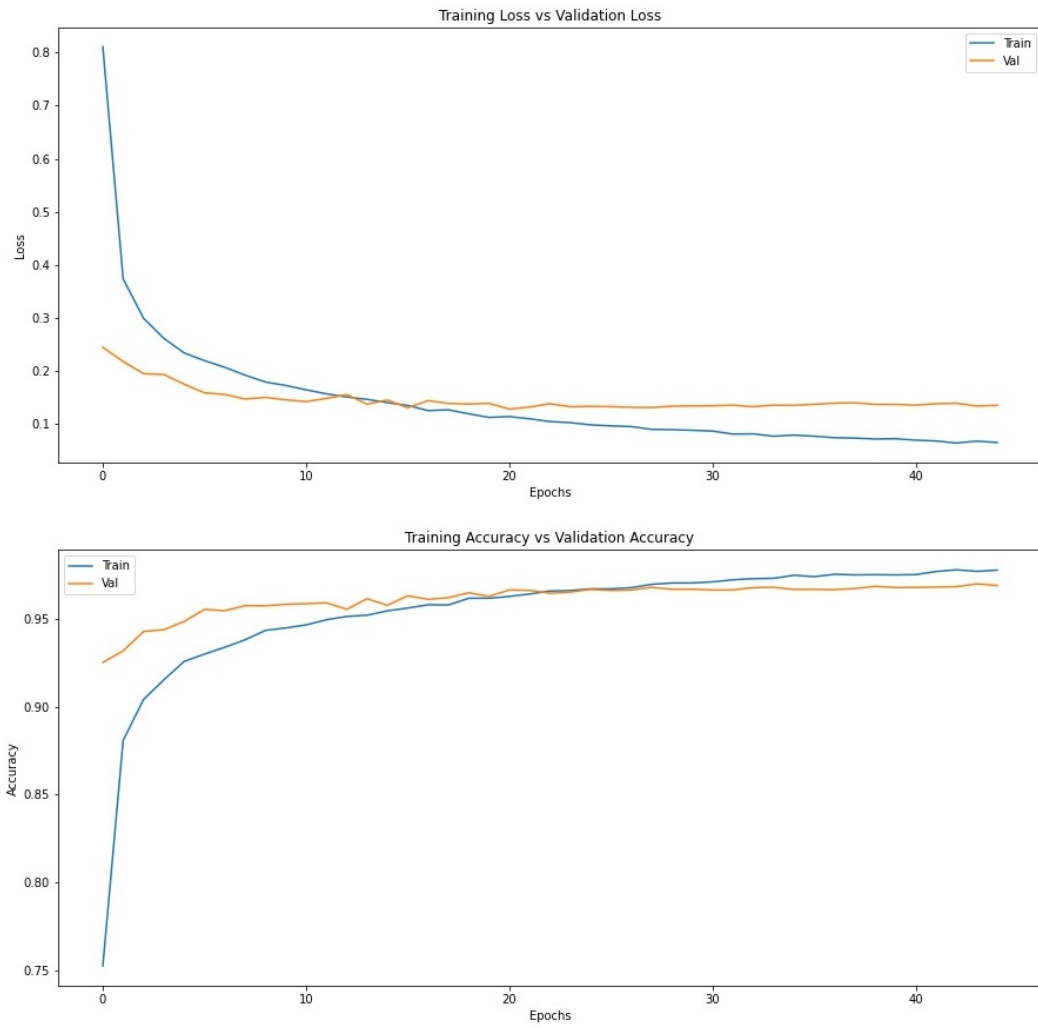


Figure B.4: AlexNet



---

## PGN game statistic

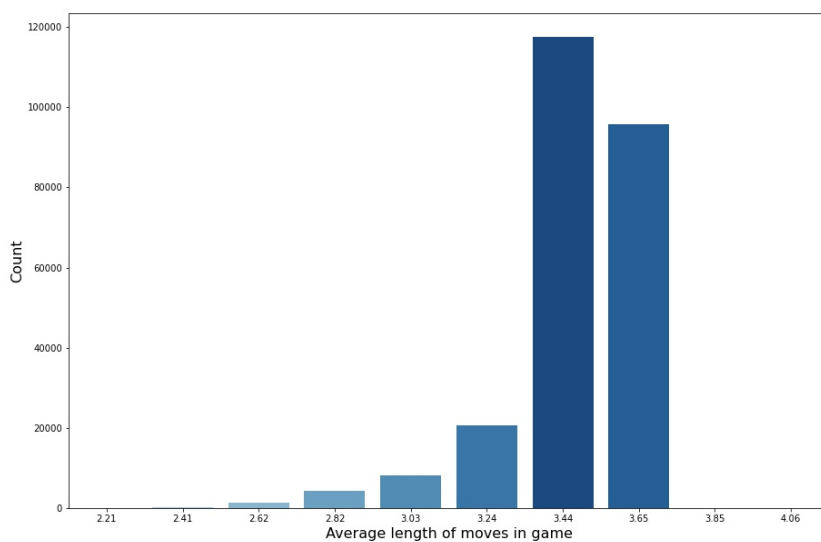


Figure C.1: Average length of move in game

### C. PGN GAME STATISTIC

---

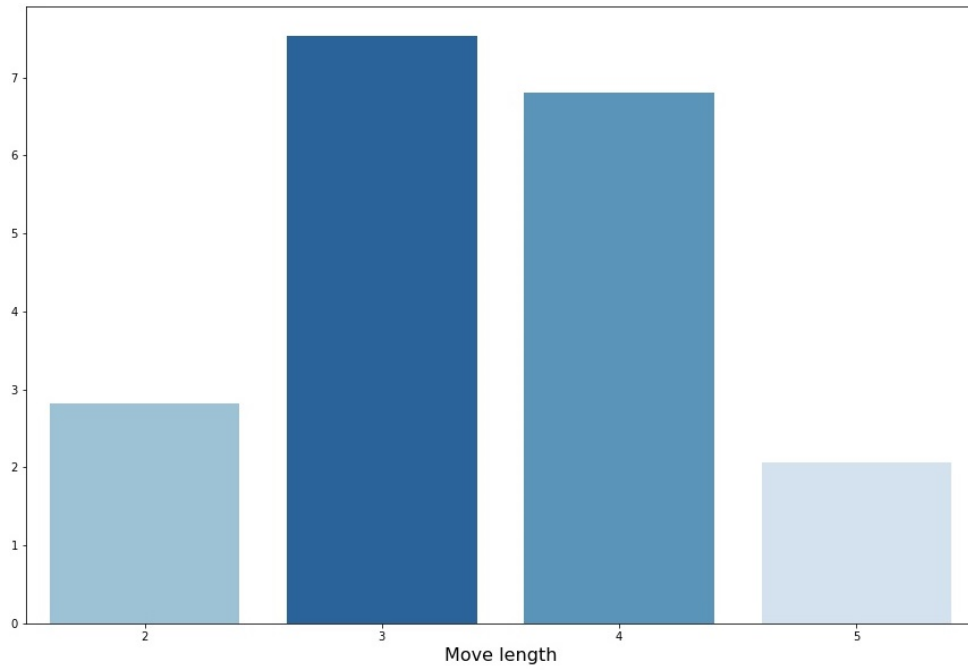


Figure C.2: Length of move in game

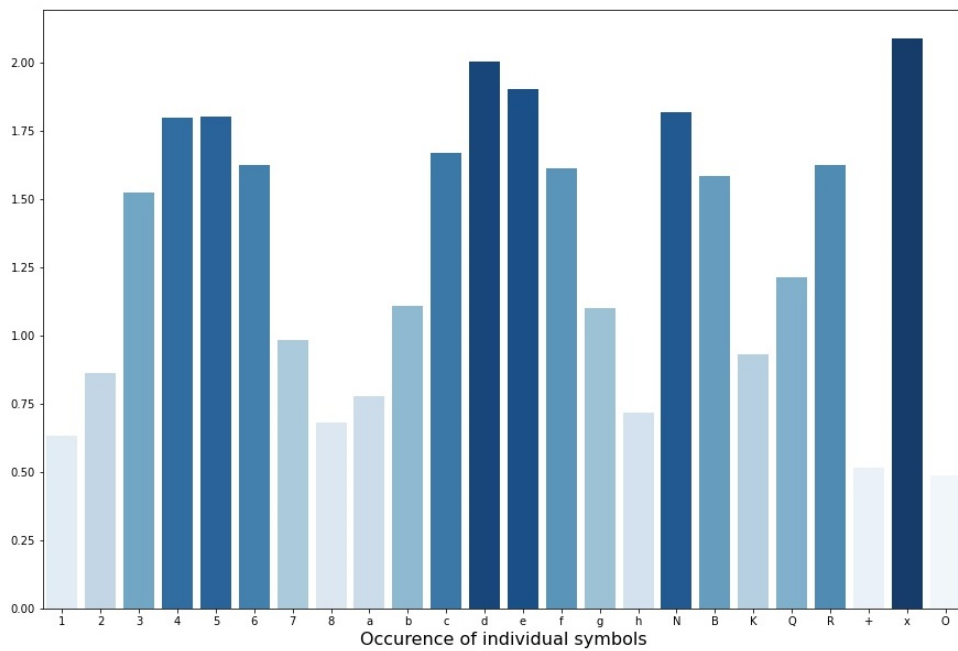


Figure C.3: Distribution of characters in games

## Examples of application

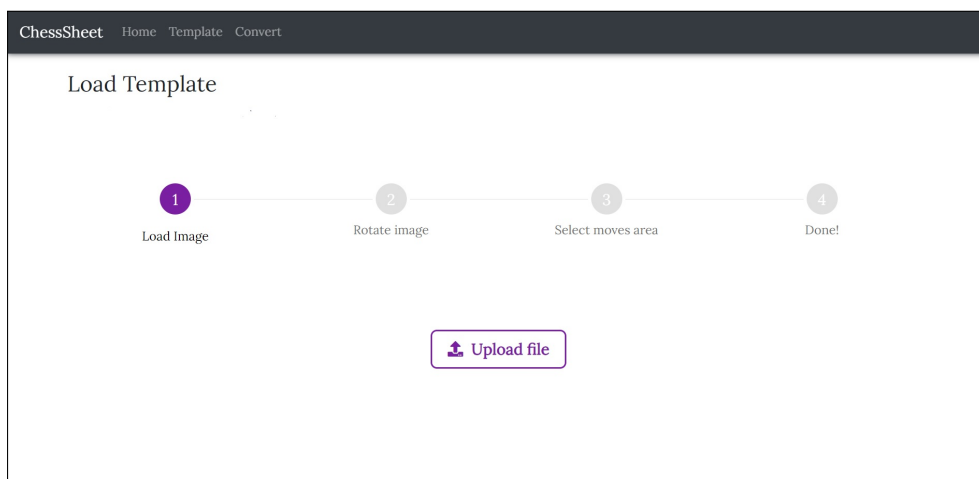


Figure D.1: Template upload

## D. EXAMPLES OF APPLICATION

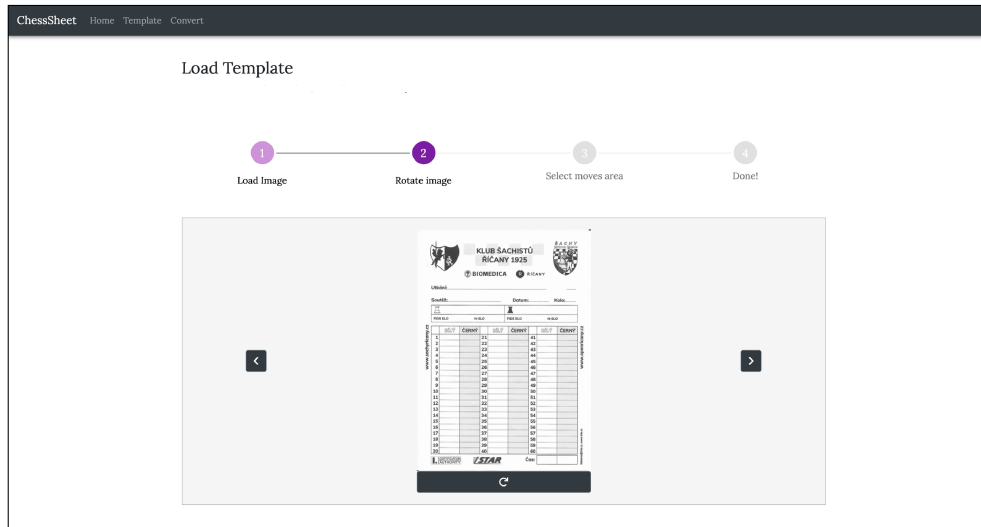


Figure D.2: Template upload – rotation

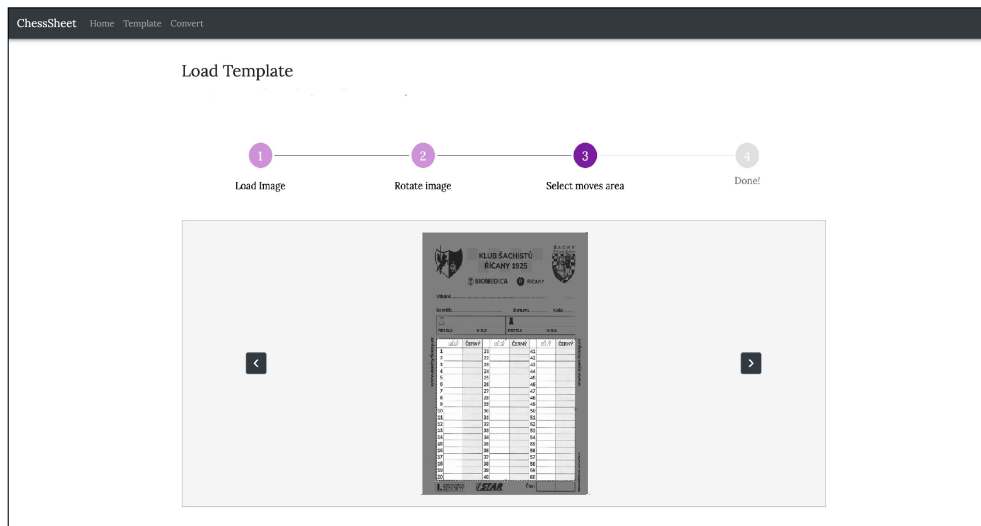


Figure D.3: Template upload – move area selection



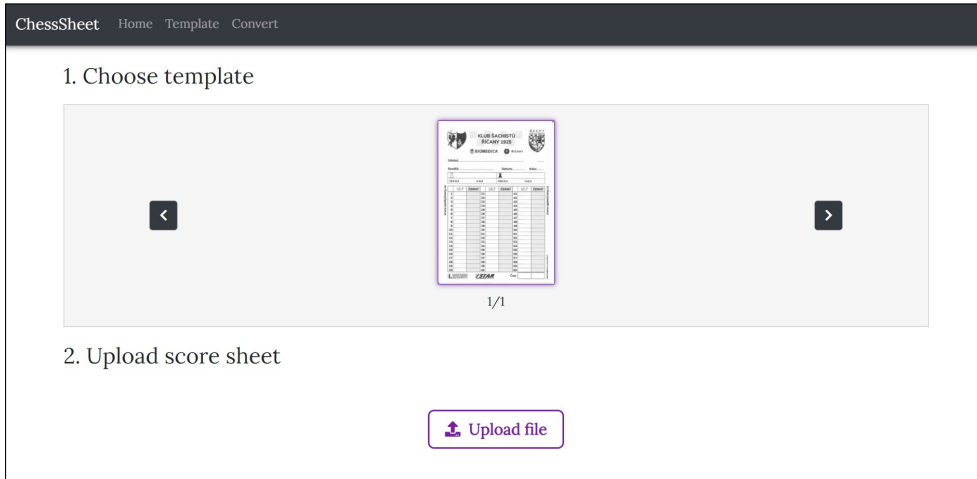


Figure D.4: Template select

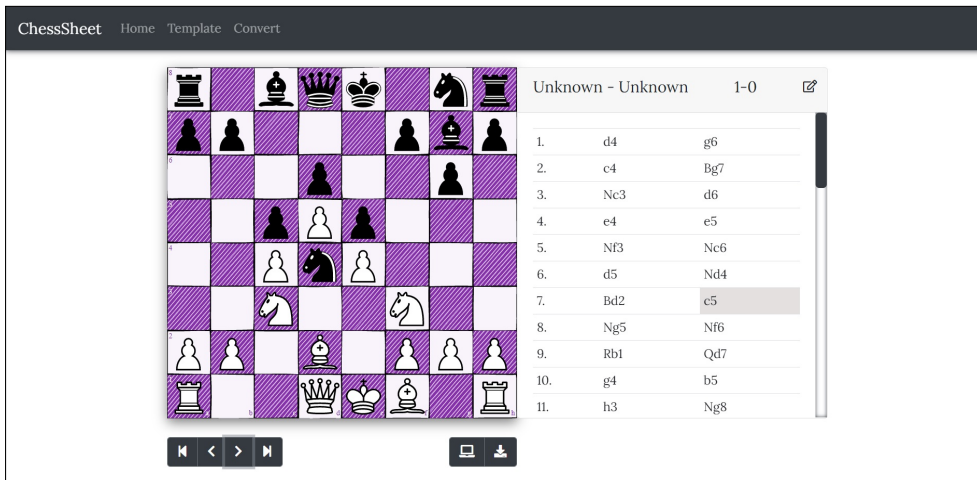


Figure D.5: Game conversion



---

## Acronyms

**PGN** Portable Game Notation

**FIDE** Fédération Internationale des Échecs

**OCR** Optical Character Recognition

**FEN** Forsyth–Edwards Notation

**RGB** Red, Green, Blue

**ROI** Region of Interest

**PSC** Potential Segmentation Column

**SVM** Support Vector Machine

**K-NN** k-Nearest Neighbours

**RF** Random Forest

**NN** Neural Networks

**CNN** Convolutional Neural Networks

**MNIST** Modified National Institute of Standards and Technology database

**WSGI** Web Server Gateway Interface

**OpenCV** Open Source Computer Vision Library

**SIFT** Scale-Invariant Feature Transform

**ORB** Oriented FAST and Rotated BRIEF

**FLANN** Fast Library for Approximate Nearest Neighbors

**EMNIST** Extended MNIST



---

## Contents of enclosed CD

	readme.txt .....	the file with CD contents description
	dataset .....	the file with score sheets
	src .....	the directory of source codes
	readme.md .....	guideline to run the app
	frontend .....	source codes of frontend
	backend .....	source codes of backend
	notebooks .....	scripts
	thesis .....	the directory of $\text{\LaTeX}$ source codes of the thesis
	text .....	the thesis text directory
	thesis.pdf .....	the thesis text in PDF format