НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

National Research University Higher School of Economics
Faculty of Computer Science
HSE and UoL Double Degree Programme in Data Science and Business Analytics

Programming Project
Make a knight's move!

Done by Dosaev Savelii Yuryevich in year 2022, group БПАД202
Supervisor: Starichkov Nikita Yuryevich, director for University Affairs, 1C

Moscow, Russia

Description of subject category

This is a programming project dedicated to helping new players, who utilise a real, physical chessboard, to learn chess by creating application that analyses a given position.

Object of the course work (of my part):
Image recognition

Subject of the course work (of my part):
Grid detection and perspective warp on a chessboard
Chess piece detection using Convolutional Neural Network

PROJECT'S SIGNIFICANCE/ACTUALITY

Reasons for increased chess popularity:

1. Chess broadcasts from professional players
2. Netflix series 'Queen's gambit'
3. Improved interface of online chess websites
4. Covid-19

Fig. 1: Chess.com activity

PROJECT'S GOAL AND AIMS

PROJECT'S GOAL

The goal of this project is to create an android application that will capture a real chessboard, recognize chess position, transfer it tothe app, analyze the position and give advice.

PROJECT'S OBJECTIVES

1. Build front-end side of the application
2. Build server side of the application
3. Use chess engine and chess rules to correct the prediction
4. Implement chessboard capture and chess piece detection – my objective

BASIC CONCEPTS,TERMS, DEFINITIONS

Chess engine - a computer software that analyzes and evaluates chess positions

Convolutional neural network (CNN) - subclass of neural network, based on the shared-weight architecture of the convolution kernels or filters that slide along input features

Dataset - raw collection of data.

Forsyth–Edwards Notation (FEN) - a common notation for defining a certain chess board situation. The goal of FEN is to offer all of the information needed to restart a game from a specific point. Example:
1q1r4/r3b1k1/p1B1Q1p1/1pp5/2P2P2/2N3P1/PP3P2/R5K1

Saddle (lattice) point - a point in image that belongs to the intersection of 4 adjacent chess tiles.

ANALYSIS OF EXISTING DECISIONS AND SOLUTIONS

Three main works:

Chessboard and chess piece recognition with the support of neural networks

Maciej A. Czyzewski, Artur Laskowski, Szymon Wasik

Chess Position Recognition from a Photo

Zoltán Orémuš

Convert a physical chessboard into a digital one

Saurabh B.

FUNCTIONAL STANDARDS AND STACK

Functional standards:

Execution time: less than 10 seconds per image

Accuracy of chessboard capture algorithm: 95% or more

Accuracy of piece detection: 95% or more

Stack:

Python 3.7.9 with Visual Studio Code

Numpy, python-cv, keras, sklearn

CHOICE OF MODELS, METHODS OR ALGORITHMS

Chessboard capture algorithm

⇩

Chess piece detection algorithm

⇩

Chess rules

⇩

Chess engine

Chess position as FEN

DESCRIPTION OF THE CHOSEN MODEL

Chessboard capture algorithm

Geometric approach VS neural network

Pros of geometry:
No need for train data
Explainability

Pros of neural network:
Easier to understand
Generally faster
More robust

Geometry method of chessboard capture

Steps:

1. Find and prune lattice points
2. Find and 'sanity check' contours
3. Find the vector space where chessboard lies
4. Warp the board with regards to vector space
5. Crop everything except the chess board

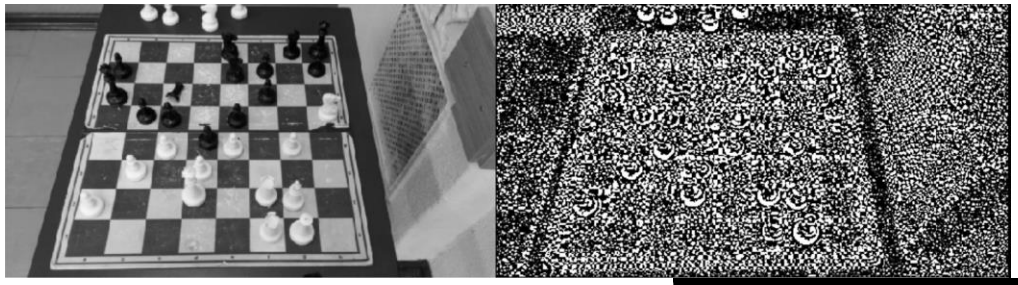
DESCRIPTION OF THE CHOSEN MODEL

Geometric chessboard capture algorithm

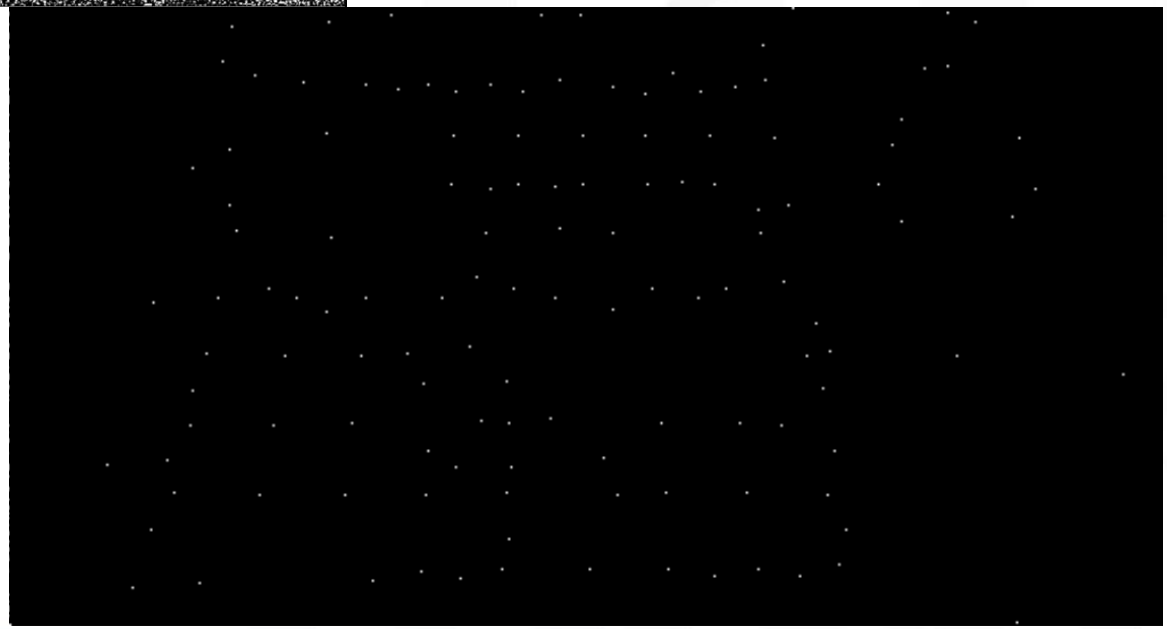
Image gradient vs line detection

DESCRIPTION OF THE CHOSEN ALGORITHM

To find lattice points, I used Sobel method to get image gradient in terms of x and y



Pruning gradients yield this ->



DESCRIPTION OF THE CHOSEN ALGORITHM

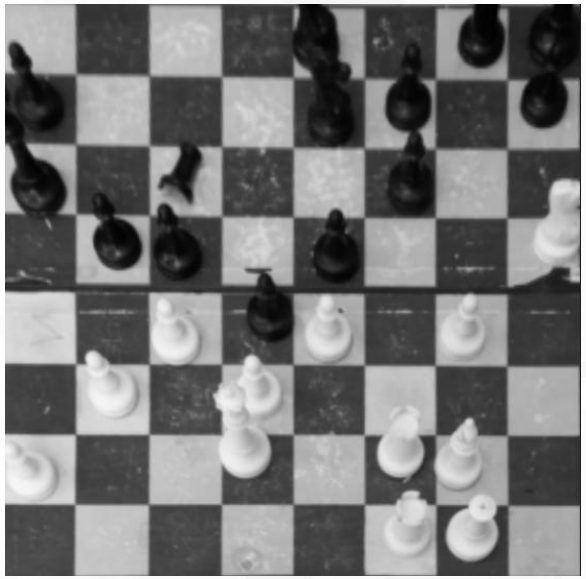
There is existing method findContours in cv2 to obtain all contours of an image

<- Initial contours

Pruned contours ->

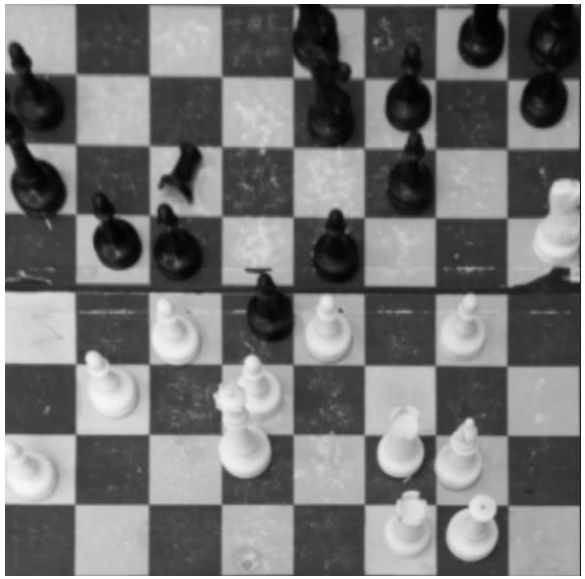
DESCRIPTION OF THE CHOSEN ALGORITHM

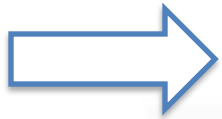
Knowing linear operator, we can warp and crop an image to get a final result

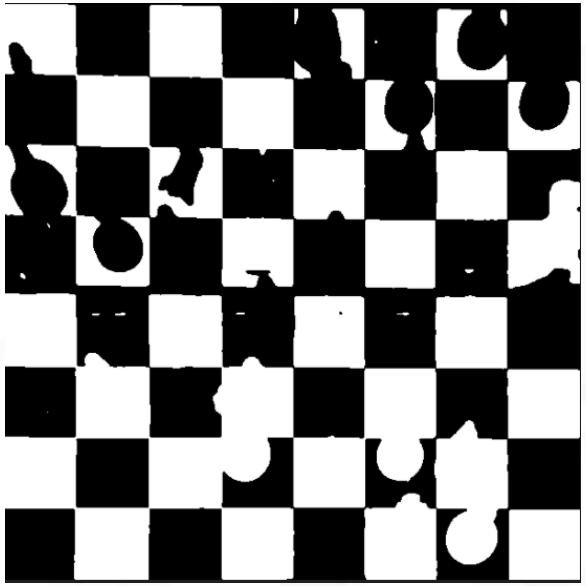


DESCRIPTION OF THE CHOSEN ALGORITHM

How to check whether the orientation is right? Using binary filter, of course!

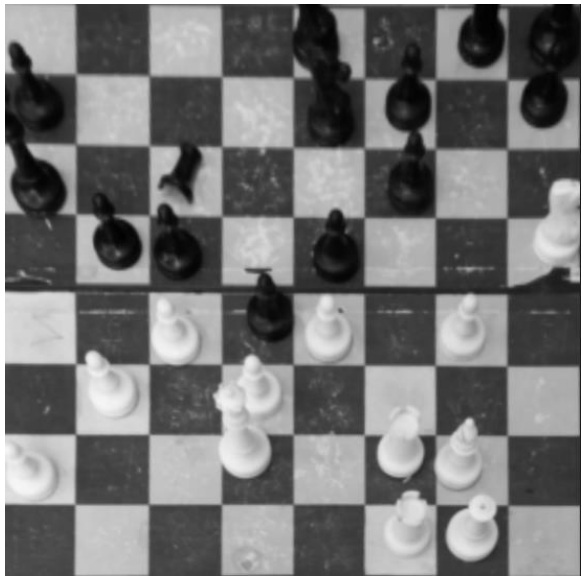






DESCRIPTION OF THE CHOSEN ALGORITHM

How to check whether the orientation is right? Using binary filter, of course!



64 chess tiles

Convolutional neural network

FEN string

DESCRIPTION OF THE CHOSEN ALGORITHM

Description of CNN created:

Transfer learning from InceptionResNetV2
Last 200 layers were re-trained
Trained for 15 epochs, no improvements later
Approximately one hour to train
97% accuracy validation

PROJECT/PROGRAMM TEST

Input: path to any image, regardless of resolution
Output: FEN string

```
import time
start_time = time.time()
fen = main(filename)
print(fen)
print('time elapsed: ', time.time() - start_time)
```

```
Processing 2b1k1r1-1pqp1pPp-5r2-4P3-8-3B2P1-1PP4P-1K1R3R.JPG
2b1k1r1/1pqp1pPp/5r2/4P3/8/3B2P1/1PP4P/1Q1R3R
time elapsed:  8.670201778411865
```

MAIN RESULTS OF THE PROJECT

As a result of my work, I have a chessboard capture algorithm, goal of which is to highlight the board and remove everything else. This algorithm is very robust, executes only in 1 to 2 seconds and has amazing accuracy of around 97%

I also created a procedure to verify the orientation and to prepare the chessboard to create CNN. Chess tiles obtained that way are fed to CNN, and its output is used to construct FEN string. CNN takes a considerable time to predict tiles of approximately 6 to 7 seconds

All the code and model that I have created can be integrated to the chess analytical application, which is the overall goal of my team.

POSSIBILITIES FOR CONTINUATION OF THE WORK

Possible upgrades for board capture algorithm:

1. Better fine-tuning of parameters
2. Switching from geometric approach to neural network

Possible upgrades for chess piece recognition:

1. Train model on big amount of diverse data
2. Find a better base model for transfer learning

LIST OF REFERENCES

1. https://github.com/samryan18/chess-dataset
2. https://is.muni.cz/th/meean/Master_Thesis.pdf
3. https://arxiv.org/pdf/1708.03898.pdf
4. https://docs.opencv.org/3.4/d7/da8/tutorial_table_of_content_imgproc.html
5. https://tech.bakkenbaeck.com/post/chessvision
6. https://docs.opencv.org/4.x/d3/d05/tutorial_py_table_of_contents_contours.html
7. https://keras.io/api/applications/
8. https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html

Thank you for your attention!

Saveliy Dosaev
Savelii.dosaev@gmail.com

Moscow, 2022