
NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING
CS-GY 6643 COMPUTER VISION SPRING 2021

FINAL PROJECT

CHESS IMAGE RECOGNITION USING TRADITIONAL COMPUTER
VISION TECHNIQUES AND NEURAL NETWORKS

Name: Tianyi Zhao netID: tz1330
Name: Zhe Luan netID: zl3368

Date: May 16 2021

I Introduction

Tencent's Fox Go, a popular go chess game platform which has one of the most powerful go Ai FineArt, has recently released a mobile application which can recognize go chessboard and go pieces. It can output a 2D image of the go-chessboard and help judge the game. Inspired by this application, we want to implement a chess board and chess pieces recognition application.

II Experiment & Result

We break this project to two parts, chessboard recognition and chess pieces recognition and positioning. The first part should be straightforward in computer vision. We use traditional CV techniques such as edge detection and Hough transformation.

Different from go pieces, chess pieces recognition is more complicated since there are 6 types of chess pieces of both black and white pieces, so we intend to use some deep learning techniques such as CNN to do this part.

II.1 Chessboard Coordinates

We use traditional computer vision techniques to build the chessboard coordinates from an original empty chessboard. First the original chessboard images are transformed to grayscale images and smoothed. Then we use canny edge detection and Hough transform to find the lines.

The set of lines should be classified to horizontal and vertical sets. We use image K-means clustering with $k=2$ to classify the lines generated by in the previous steps. Then we can build the chessboard coordinates based by calculating the intersections between each pair of lines in the two sets.

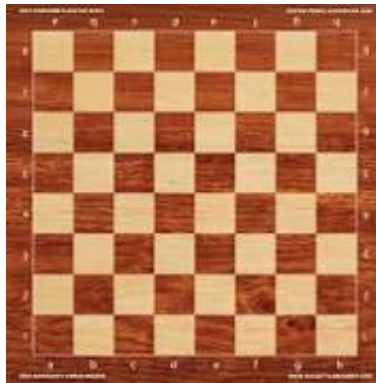


Figure 1: Original chessboard (1).



Figure 2: Grayscale and Gaussian blur.

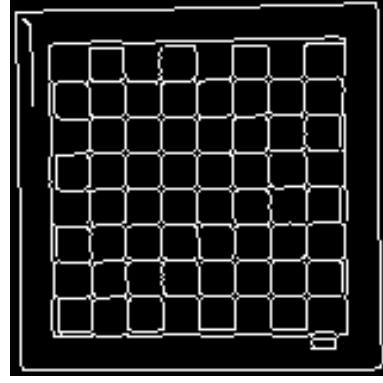


Figure 3: Canny edge detection.

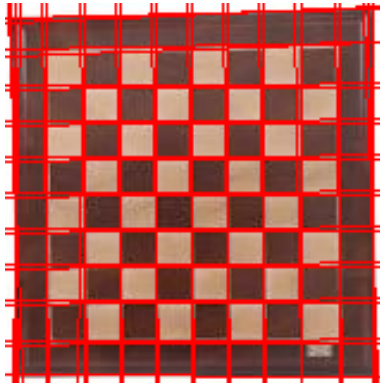


Figure 4: Hough transform.

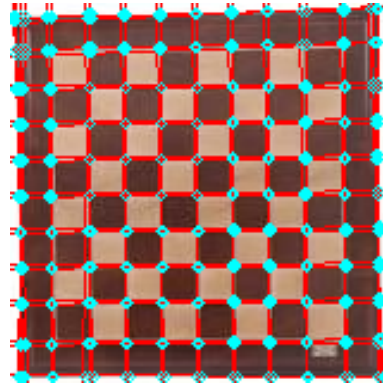


Figure 5: Clustering and intersections.

With these methods, we can have good results for different angles of chessboard as the figure shown below.

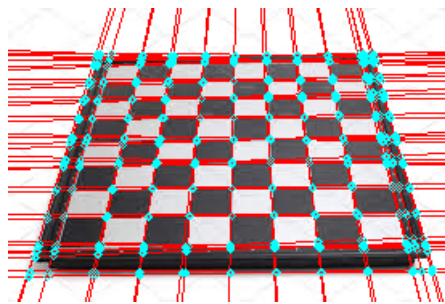


Figure 6: Chessboard coordinates with different camera angles.

When we experimenting on chessboard with pieces, we can generate good coordinates for most cases. However, for each chessboard with different camera angles, we need to manually adjust the parameters of hough transform and line filters.

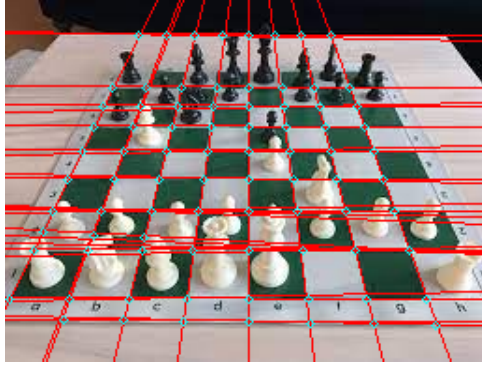


Figure 7: Chessboard with pieces.

II.2 Chess Piece Classification

Our dataset consists of 2406 images from twelve types of chess pieces, which are bishops, kings, queens, knights, pawns, rooks in both black and white, and empty cells as the thirteenth piece type. As the example images below show that there are images that have only one chess piece that need to be classified, and also images contain chess pieces. Each type of chess piece has images taken from multiple angles.



Figure 8: Examples of images in the dataset.

class	number of images
black bishop	200
black king	103
black knight	171
black pawn	194
black queen	185
black rook	198
empty	212
white bishop	215
white king	110
white knight	205
white pawn	199
white queen	198
white rook	216
total	2406

We randomly select 20 images from each class and form a validation dataset of 260 images. (validation dataset with approximately 10

Convolutional Neural Network has proven to have great performance in image classification tasks, so the team first chose to build a CNN model and train it with our dataset. The CNN model has three convolutional layers, corporate with three max pooling layers to learn the images and extract the useful information and two fully connection layers with Relu activation function to perform predictions of the class of the piece.

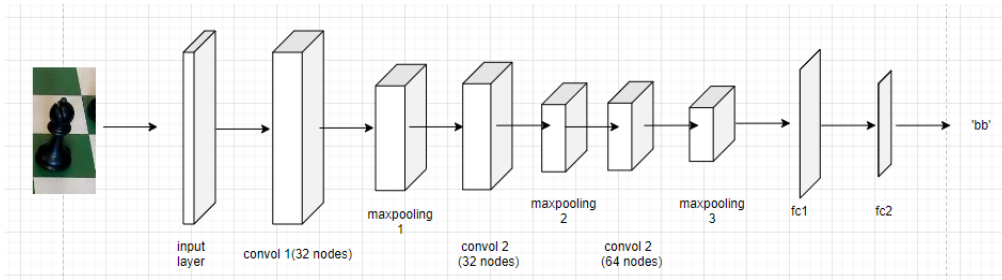


Figure 9: Structure of CNN model.

To build the model, the team chose to use Keras as it provides a high level deep learning framework which is easy to use and set up on google colab where we can access free GPU recourse. When we feed the data of the image, we rescale the image into size of 224 *224 pixels. Keras data generator module provides the option to alter the data by randomly rotating the image in a given range of angle. We use

this feature with setting the range of the rotation to be 0-30 degree, to increase the generalization performance of our model.

The team also has a concern that given the relatively small size of the accessible dataset, the learning problem might not be learnable for a deep CNN model. So the alternative solution is to utilize the pretrained computer vision models like VGG and Resnet to build a transfer learning model. VGG and Resnet are very deep convolutional networks models with many hidden layers and huge amounts of parameters and they are trained on Imagenet. Transfer learning is a technique that a model trained on a similar dataset as that of the current problem can be used to solve the current problem. Following the standard approach of the transfer learning, the team froze all the parameters in the pretrained models. “Freeze” means that we turn the parameters in the pretrained model to be not learnable so that it would not get updated during the later training. We also remove the fully connected layers, since we are going to train our own fully connected layers to do the predictions. Then we add two fully connected layers on top of the processed pretrained model, then we train the model with our dataset.

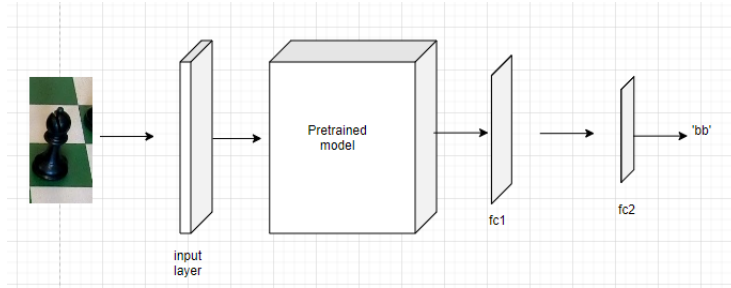


Figure 10: Structure of the transfer learning model.

In the standard approach of transfer learning, there should be one more step of “fine-tuning”, which means we set the parameters inside the pretrained model to be learnable and train the whole model with a very small learning rate. But due to lack of data, we decide not to do this step to prevent overfitting the data.

After the experiment, the performance of the simple CNN model matches the concern of the team. The categorical prediction accuracy on the validation dataset of the model converges around 0.4 in 30 epochs. The model achieved 0.44 as the highest test accuracy. This performance is even worse than random guess which implies that this learning problem is not learnable for the model in the current setting. Following the plan, the team switched to the transfer learning.

We build three models with the same structure of transfer learning model discussed in the previous section by using three different pretrained models: VGG16, VGG19 and Resnet50. The test accuracy of model build with Resnet is unexpect-

edly bad. The training accuracy and test accuracy converged at the 5th epoch at 0.156 and 0.123. Compared to this, the model uses VGG16 and the model uses VGG 19 get the similar performance in 30 epochs as the figure below shows.

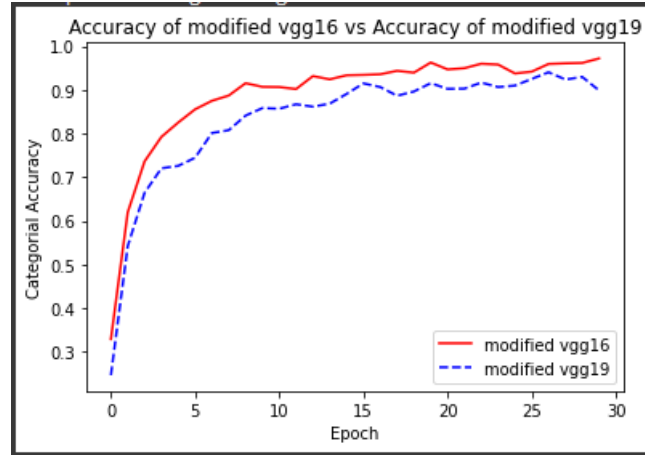


Figure 11: Test accuracy of the model using VGG16 and the model using VGG19.

According to the figure, we can tell that the transfer learning model using VGG16 outperforms the one using VGG19. The best test accuracy of 0.83 is obtained at the 28th epoch, and after that the overfitting occurs.


Therefore, the team chose to use the transfer learning model using VGG16 as our model in chess piece classification.

class	Precision	Recall	F1-score	Support
black bishop	0.58	0.9	0.71	20
black king	0.94	0.85	0.89	20
black knight	0.94	0.8	0.86	20
black pawn	0.94	0.8	0.86	20
black queen	0.93	0.7	0.8	20
black rook	0.85	0.85	0.85	20
empty	1	0.95	0.97	20
white bishop	0.77	0.85	0.81	20
white king	1	0.9	0.75	20
white knight	0.93	0.7	0.8	20
white pawn	0.86	0.9	0.88	20
white queen	0.79	0.95	0.86	20
white rook	0.62	0.9	0.73	20

The classification report shows that the model has some trouble in classifying black bishops. This might be due to the fact that the bishops have some sort of

similar shape with the pawns with some angles to the focal axis. This explanation is reasonable since the precision for white bishops is also lower than the average precision.

The confusion matrix of the predictions on the validation data is showing below:

		Predicted classes													
		black bisho p	black king	black knigh t	black pawn	black quee n	black rook	empt y	white bisho p	white king	white knigh t	white pawn	white quee n	white rook	
		black bisho p	180	10	0	0	0	0	0	0	0	0	0	10	
		black king	10	170	0	0	10	10	0	0	0	0	0	0	
		black knigh t	10	10	160	0	0	20	0	0	0	0	0	0	
		black pawn	40	0	0	160	0	0	0	0	0	0	0	0	
		black quee n	50	0	0	0	140	0	0	0	0	0	0	10	
		black rook	20	0	10	0	0	170	0	0	0	0	0	0	
Actual Class es		empt y	0	0	0	0	0	0	190	0	0	10	0	0	
		white bisho p	0	0	0	0	0	0	0	170	0	0	10	20	
		white king	0	0	0	0	0	0	0	20	120	0	0	40	
		white knigh t	0	0	0	0	0	0	0	10	0	140	10	40	
		white pawn	0	0	0	0	0	0	0	20	0	0	180	0	
		white quee n	0	0	0	0	0	0	0	0	0	0	0	190	10
		white rook	0	0	0	0	0	0	0	0	0	0	10	10	180

III Conclusion

In this project, we have first used traditional computer vision techniques to recognize chessboard coordinates with different camera angles. With the coordinates we transform the image into many small patches and use the small images to do chess piece classification. By experimenting on different vision networks, we found

that VGG16 has a good performance. Then we can combine the coordinates of the chess piece and the classification result to create the visualization of the final chess board.

IV Further Work

In chessboard recognition, we need to manually adjust the hough transform parameters for each set of images. A possible way to solve this problem is to scan the parameters and do multiple experiments to figure out if the output number of points is reasonable (between 60 to 100). But if the set of images contains one with no pieces on the chessboard, we can always get the correct coordinates.

In chess piece classification, there is an obvious problem that, when we divide the transformed chess board into small patches of images containing information of each cell, we are very likely to have several segments of multiple chess pieces inside one small image. And this problem will decrease the accuracy of the classification. One possible work around of this problem is to separate the task of chess piece detection and classification into two subproblems. That is, we first use a machine learning model like YOLO to detect where we have chess pieces on the board, then we map its coordinates to the board by the transformed board coordinates. For this model we need to input a picture of the chess board taken from above of the board. It does not have to have the focal axis perfectly aligned with the normal vector of the plane of the chess board, we can set up an acceptable range of angles. Then we take a picture of the chess board in the same way we have in our current dataset, then we use the coordinates we got from the previous task, and crop each chess piece out from the original image and classify them.

V Reference

1. ChessVision: Chess Board and Piece Recognition. Jialin Ding. Stanford University.

https://web.stanford.edu/class/cs231a/prev_projects_2016/CS_231A_Final_Report.pdf

2. Chess Piece Recognition Using Oriented Chamfer Matching with a Comparison to CNN. Youye Xie, Gongguo Tang, William Hoff. Colorado School of Mines, Golden, Colorado USA.

<https://par.nsf.gov/servlets/purl/10099572>

3. Determining Chess Game State From an Image. Georg Wölflein, Ognjen Arandjelovic.

<https://arxiv.org/pdf/2104.14963.pdf>