Chess Detection

INTRODUCTION TO DIGITAL IMAGING

Present to you by

~ Kang DB ~

Content

1. Introduction
2. Model
3. Evaluation
4. Analysis
5. Conclusion

Introduction

Introduction : Problem Statement

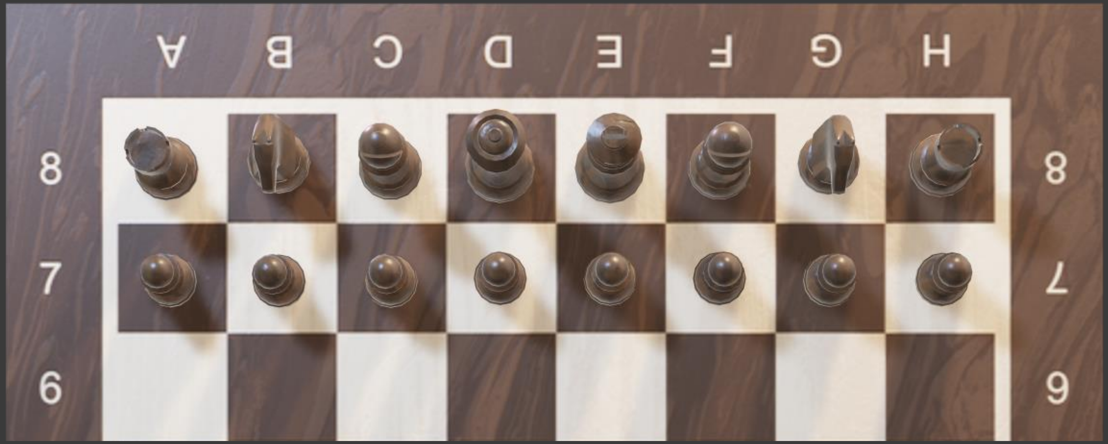
Given a video of a chessboard, use image processing to extract moves that have been played through out the game in the PGN format.

Introduction : Input

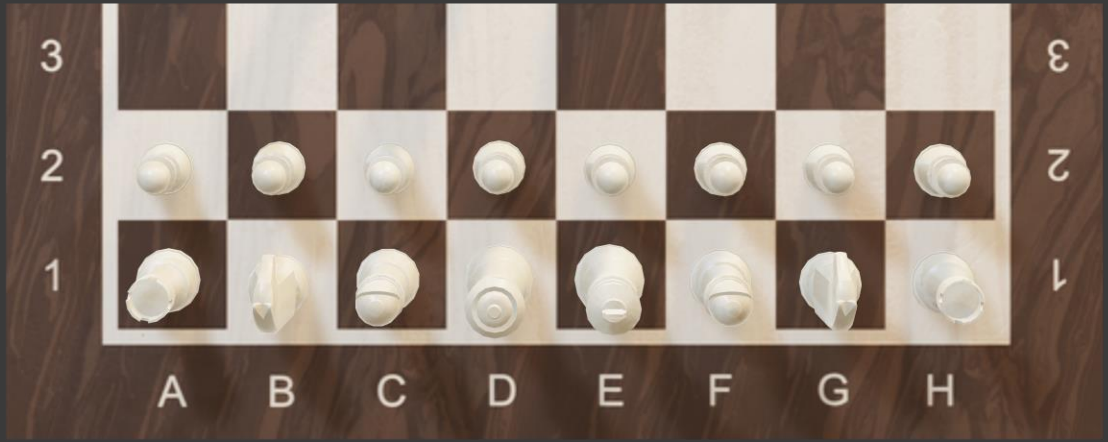
- Videos of a chess game with at least 2 moves.
- There will be hands and noises in the video
- The video may appear rotated
- The video may contain illegal moves
- You may assume the video was recorded from Black player’s perspective

Introduction : Output

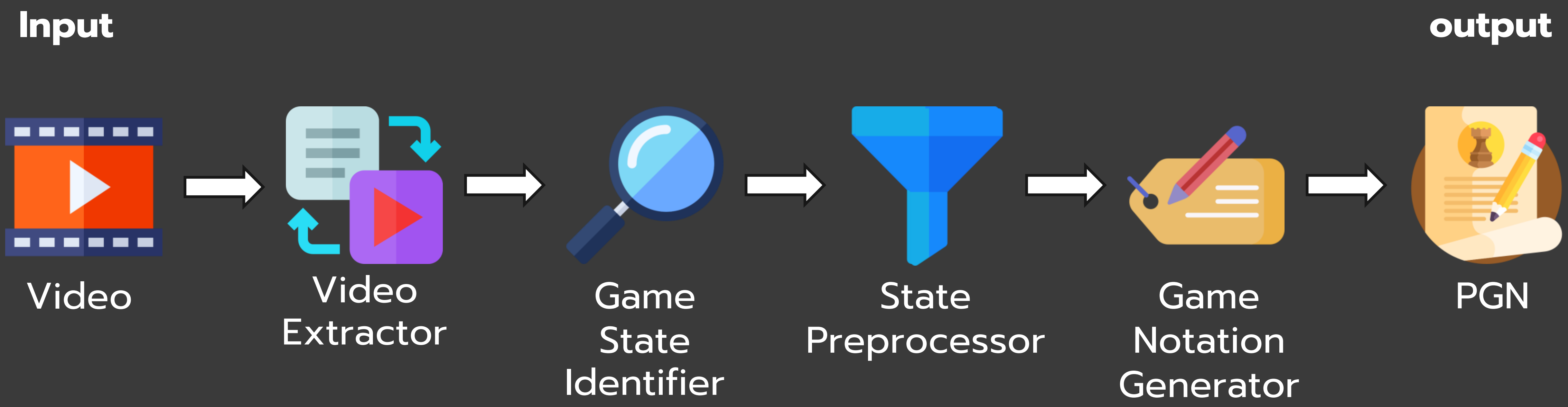
- A PGN notation of the game, starting from white.
  - If started from black, you may leave white turn as blank
- Only moves will be scored



Model



Model : Overview



Model : Video Extractor

- Capture one frame from every 3 second
- Use thresholding to determine if the board is in the new “game state”

Model : Game State Identifier

Frames

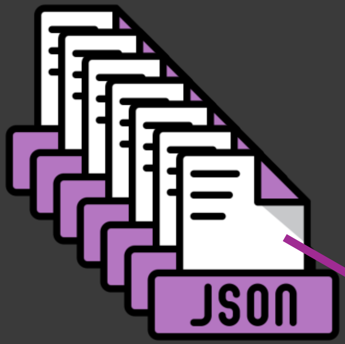
FEN Notations

- Determine the game state from image using CNN using the following steps
  - Board Localization
  - Occupancy Classification
  - Piece Classification
- We select ChessCog's pretrained model and fine tune it to better recognize the board

chesscog

Model : Transfer Learning

Many Images



Position Label from each images

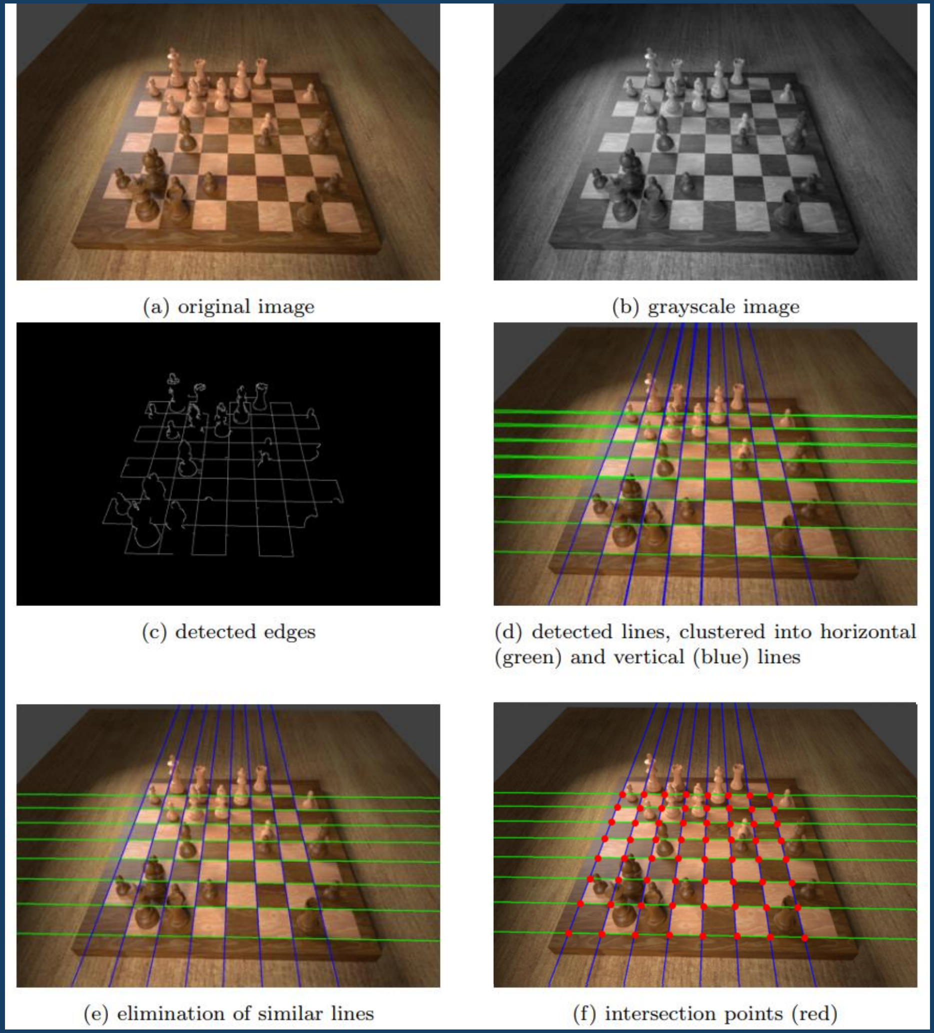
chesscog

Transfer learning

Game State Identifier


```json
{
  "white_turn": true,
  "fen": "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR"
}
```

Model : Board Localization



(a) original image

(b) grayscale image

(c) detected edges

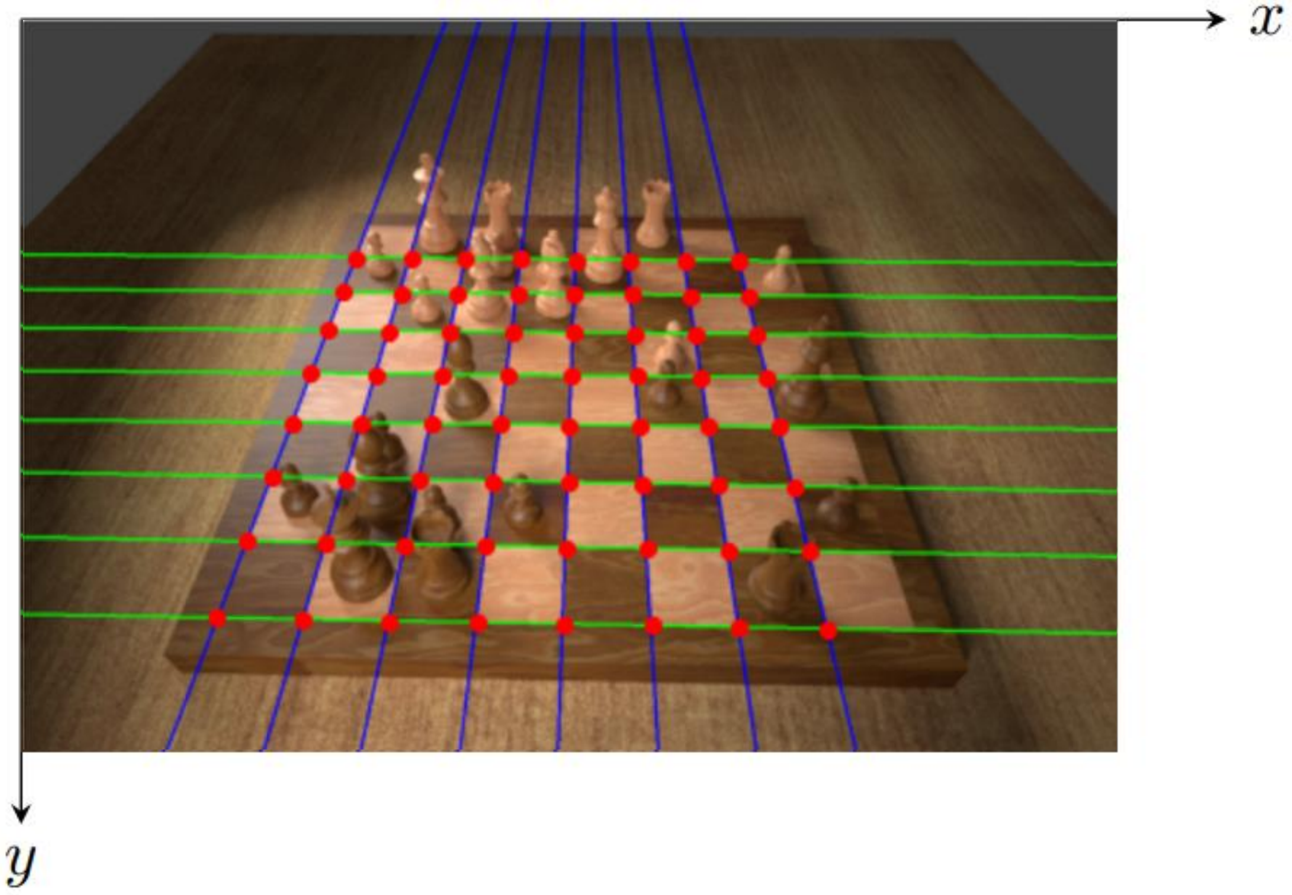
(d) detected lines, clustered into horizontal (green) and vertical (blue) lines

(e) elimination of similar lines

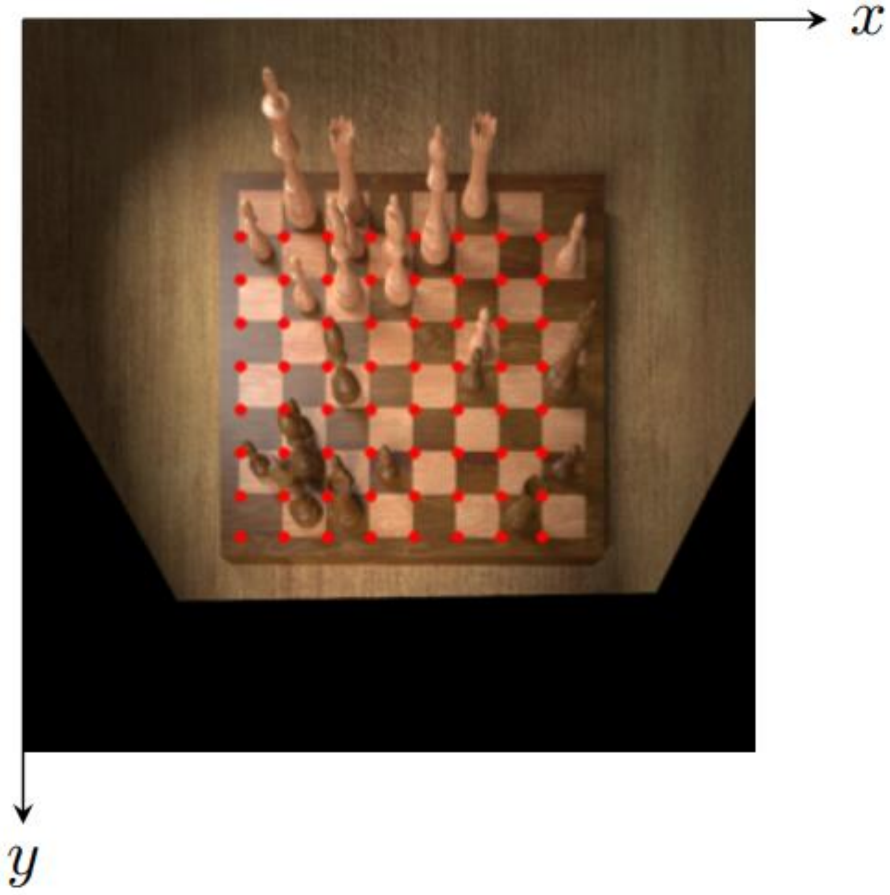
(f) intersection points (red)

- Detect corner of the board and warp the image such that it become easier to segment and process
- Hough Transform
  - Image Preprocessing -> Line Detection -> Filter&Clustering -> Intersection
- Homography
  - Finding Homography -> RANSAC -> Optimal Homography
- Missing Line
  - Define Boundary -> Handle Grid Width

Model : Board Localization

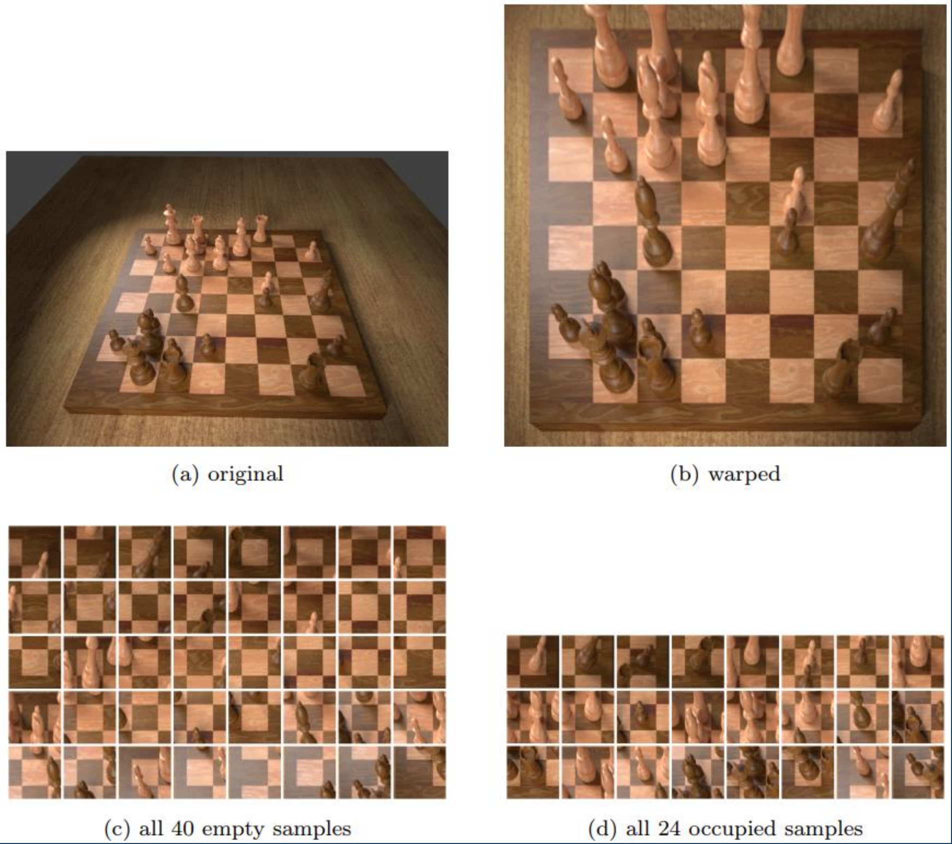


(a) original image with intersection points



(b) warped image

Model : Occupancy Classification



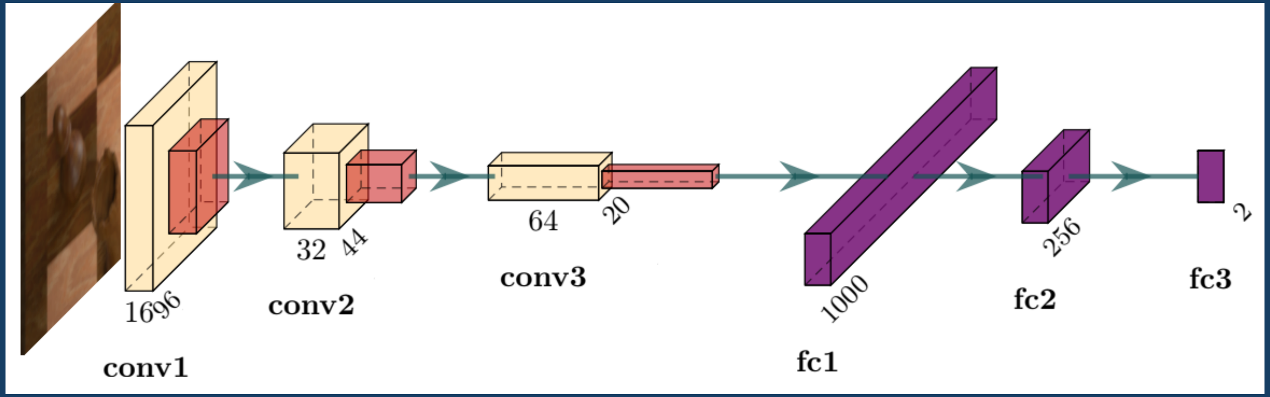
(a) original

(b) warped

(c) all 40 empty samples

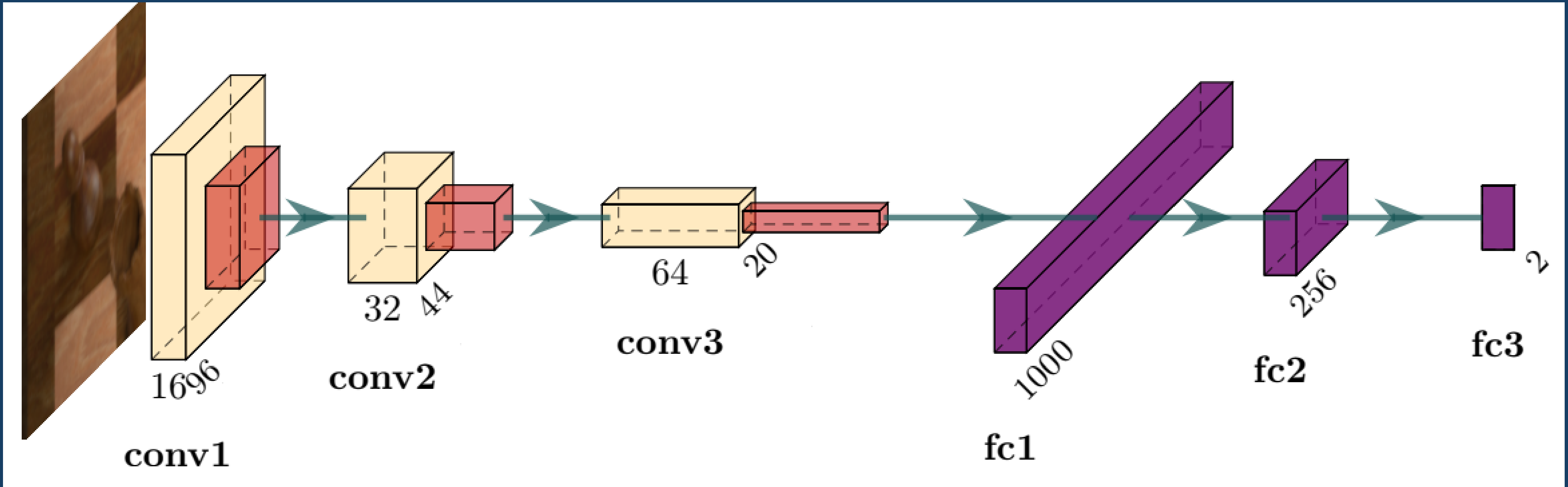
(d) all 24 occupied samples

- Warp image into 2D
- Crop each square with 50% boundary
- Using Resnet to determine occupancy



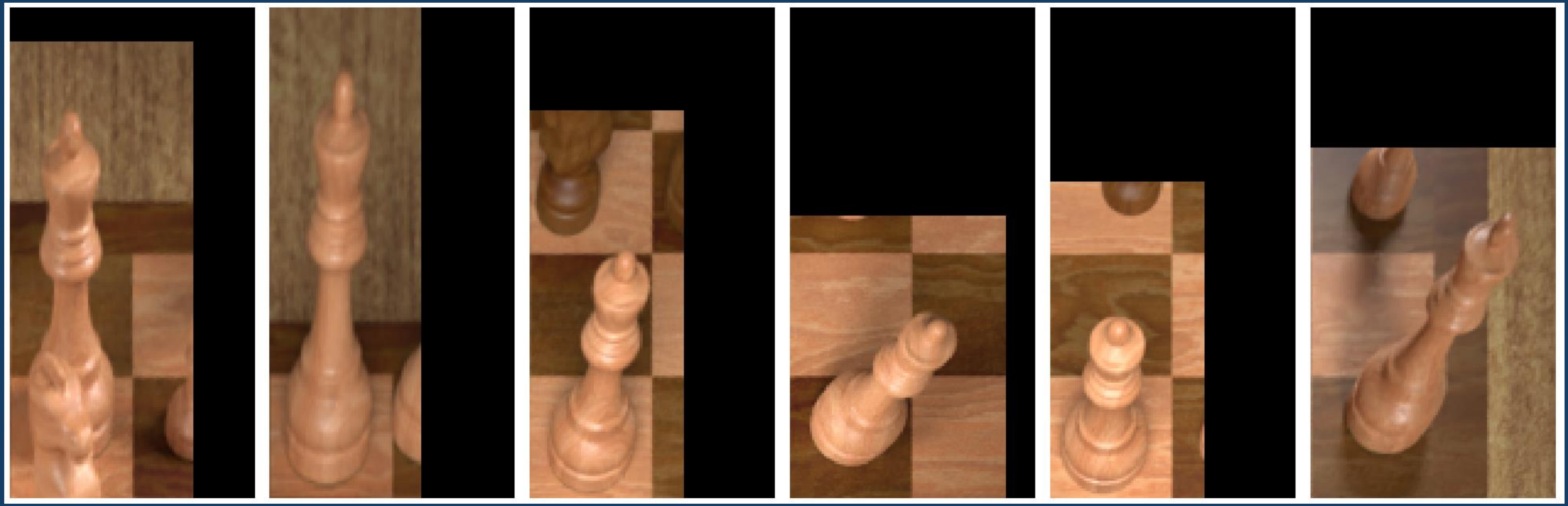
Model : Occupancy Classification

Classifier Architecture



Model : Piece Classification

- Preprocessing: Crop ROI based on rank and file for the position and perspective distortion by using linear relation to determine the width and height of the ROI
- Classifier Architecture: Use InceptionV3, two convolutional layers with ReLU activation followed by max-pooling layers, and a fully connected layer outputs a softmax classification.



Model : Game State Identifier

Frames  FEN Notations

- Determine the game state from image using CNN using the following steps
  - Board Localization
  - Occupancy Classification
  - Piece Classification
- We select ChessCog's pretrained model and fine tune it to better recognize the board

chesscog

Model : State Preprocessor

FEN Notations

Filtered FEN

- Some states are invalids from error in extraction
  - Determine the board location
  - Compare if the chess moves make sense when compare with the most recent valid board state
  - Filter out invalid state(s)

Model : State Preprocessor

It’s possible that the second image was extracted during the extraction process if the hand is too still. (Hence, exceed the comparison threshold)

Model : State Preprocessor

However, when FEN is generated, the second image will contain many artifacts. Which will be determined a faulty extraction, and will be ignored.

Model : State Preprocessor

- Some states are invalids from error in extraction
  - Determine the board location
  - Compare if the chess moves make sense when compare with the most recent valid board state
  - Filter out invalid state(s)

Model : Game Notation Generator

Filtered FEN

PGN

- Using Filtered FENs, we can easily determine the PGN move notations using the predefined algorithms
    - Translating FEN to 2D Arrays
    - Identifying Changes Between Frames
    - Determining the Move
    - Handling Special Cases

Evaluation

Evaluation : Test Dataset

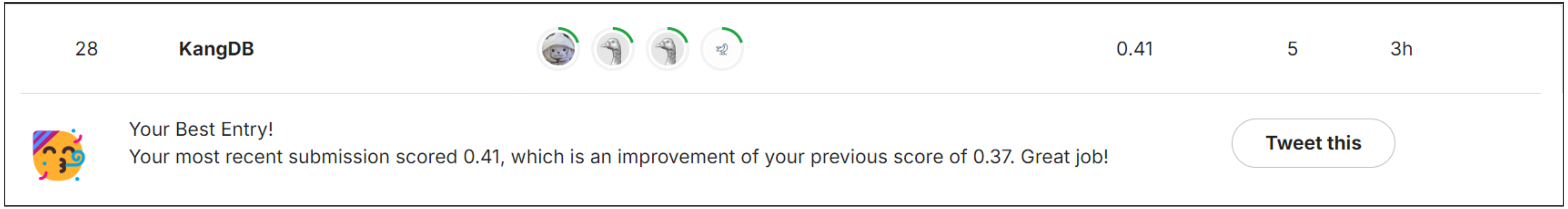
Test Data set is randomly selected from "Extra Image" Folder with manual labelling.

- A total of 30 image
- 15 from "FullBoardGame", 15 from "Single Piece Type"
- All images are assumed taken from black player's perspective
- No augmentation has been applied to any images
    - Any images in the wrong rotation have been rotated before added to the test dataset

Evaluation : Before–After Tuning

Metric	Before	After
True Positive Occupancy	588	687
True Negative Occupancy	1182	1233
False Positive Occupancy	51	0
False Negative Occupancy	99	0
Occupancy Accuracy	**92.188%**	**100.000%**
Correctly Predicted	98	666
Incorrectly Predict	78	21
Classifier Accuracy	**55.682%**	**96.943%**

Evaluation : Kaggle



Massive accuracy down!!

So what happened?

Evaluation : Kaggle

Turns out our State Preprocessor has a lot of bugs!

- The video extractor extracts too many images with hand/noise for state preprocessor to handle
- When the Game State Recognizer tried to recognize the board, it's forced to recognize hands. Hence, give a very bad result.
- When the state preprocessor try to read too many invalid images, it is failing!

Video Extractor

State Preprocessor

Evaluation : Pro/Cons Analysis

- Pros
  - Chess State recognition is very accurate (100% Occupancy, 96.943% Classifier)
  - Can be fine tuned easily. Using just small data set and short training time
    - 6 images –> 64*6 = 384 data point
- Cons
  - Model weakness is hands and object noises. This will reduce model's accuracy significantly
  - It cannot handle rotated images.
    - Due to the way pieces classifier works.
  - It will struggle with unseen data pieces
    - If fine tuning is allowed, it will perform well, terrible otherwise.
  - One small FEN mistakes will throw off the model ability to generate an accurate PGA entirely! (from that move onwards.)
  - Currently, model sometimes struggling with identifying black kings, which it confuses with the black queen.
  - Model is too slow to perform real time computation

Evaluation : Suggestion

- To handle rotate image, another model could be trained to determine the rotation of the image, and turn them back to the upright rotation
- Video Extractor algorithm to extract videos must be improved.
  - Increasing threshold might help reducing images where there are slow hand movements.
  - Increasing delay between capture might reduce the chance of hand images.
  - Another model could be trained to recognize the hands and ignore any capture with hand in it.
- State preprocessor algorithm must be improved.
  - Better handle invalid state.
  - It must be able to recognize the invalid state better.
- The model should log more. + a bonus if it can visualize a chess board for each states as a GUI.
- Increasing training data set that feature black kings and queens might reduce the misidentifying problem

Evaluation : Teamwork

Meen

- Video Extractor
- Notation Generator

Athen

- Debugger
- Pipeline Assembler
- State Preprocessor

Nac

- Researcher
- Debugger

Gain

- Transfer Learning
- Data Cleaner
- Slide

Summary

Summary

Successfully train a model that can read chess's game state and developed algorithms for video extractions and filtering

The model itself works but failed due to bad video extraction and filtering technique

Next Step

The model can be improved by increasing dataset, updating extraction algorithm, handling invalid state, and detect and auto rotate image to the right position

References

1. Czyzewski, M. A., Laskowski, A., & Wasik, S. (2020). Chessboard and chess piece recognition with the support of neural networks.
2. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. IEEE Conference on Computer Vision and Pattern Recognition, June 2009.
3. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition, June 2016.
4. Mehta, A., & Mehta, H. (2020). Augmented Reality Chess Analyzer (ARChessAnalyzer): In-device inference of physical chess game positions through board segmentation and piece recognition using convolutional neural networks. Journal of Emerging Investigators, July 2020.
5. Mohamadreza, M. (2024). Real-time chess pieces detection with YOLOv8 [Notebook]. Kaggle. Retrieved November 20, 2024, from https://www.kaggle.com/code/imtkaggleteam/real-time-chess-pieces-detection-with-yolov8/notebook.
6. Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations, San Diego, USA, May 2015.
7. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. IEEE Conference on Computer Vision and Pattern Recognition, June 2016.
8. Wölflein, G., & Arandjelović, O. (2021). Determining chess game state from an image. Journal of Imaging, 7(6), 94. https://doi.org/10.3390/jimaging7060094
9. Yang, D. (2016). Building chess ID. Medium. Retrieved November 20, 2024, from https://medium.com/@daylenyang/building-chess-id-99afa57326cd.

Thank You for Your Attention

INTRODUCTION TO DIGITAL IMAGING

Present to you by

~ Kang DB ~