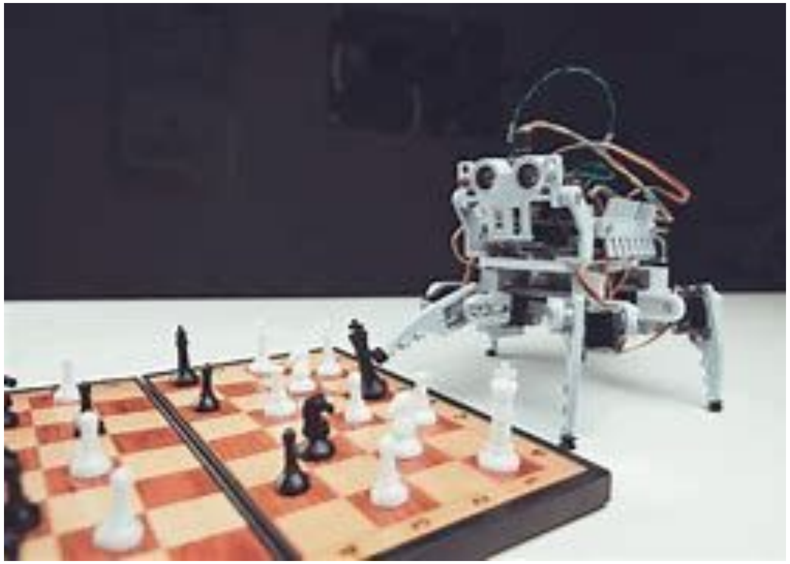
W251 | **Prabhu Narsina** | Chess Robot

Prabhu Narsina

August 2nd, 2021



Berkeley
UNIVERSITY OF CALIFORNIA

Topics

- Scope
- Data Preparation
- Approach
- Object detection and Results
- Reinforcement Learning and Results
- Next Steps

Berkeley
UNIVERSITY OF CALIFORNIA

Scope of the Project

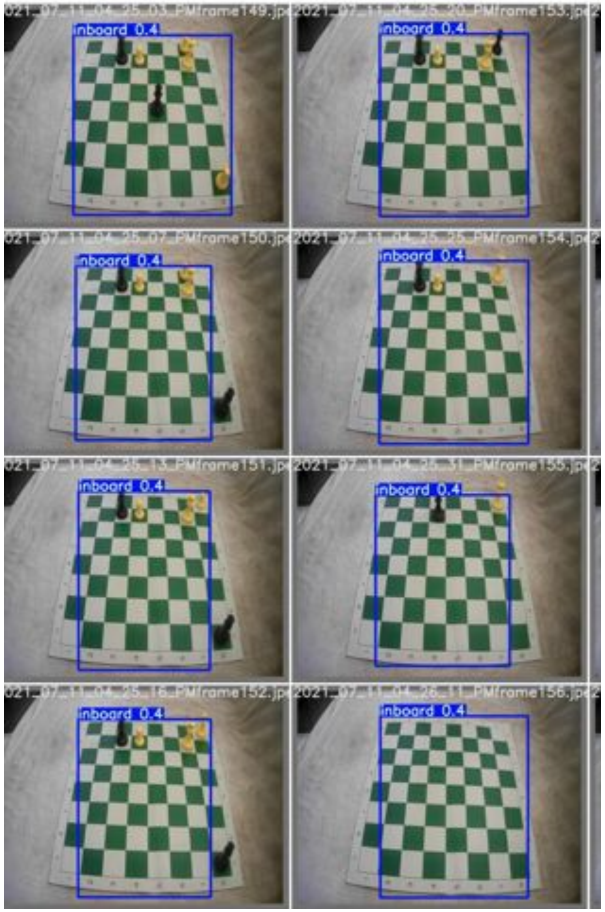
- Identify the Chess Board, Chess Pieces using Object detection
- Calculate Chess piece locations
- Should Work with a new Chess board
- Needs training with New types of pieces
- Develop Reinforcement Learning Model for Legal moves generation
  (define gym environment for chess)

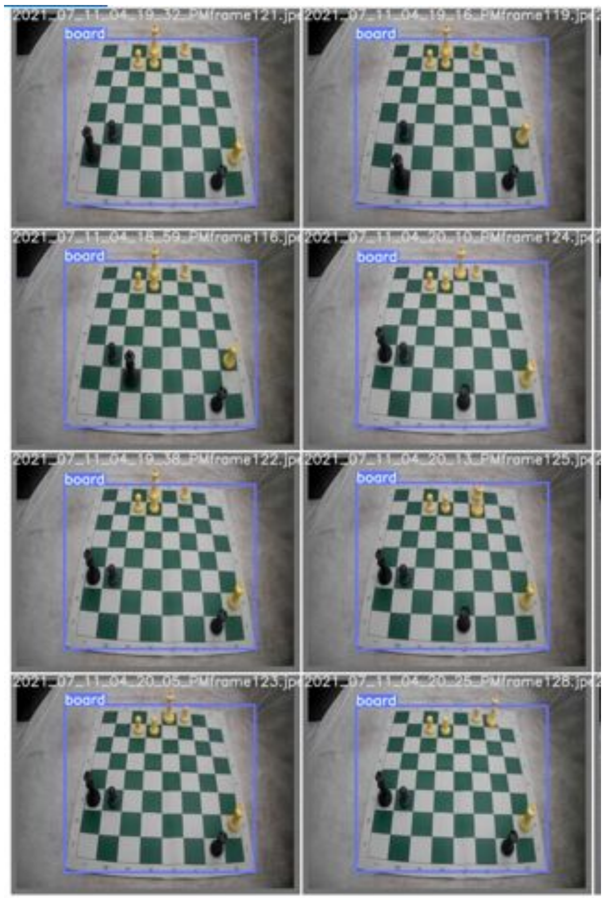
Data preparation

Data Preparation
- preexisting labels and images
- Images from 2 new boards using Jetson
  - Board (Inside and outside)
  - Chess pieces

Chess piece locations
(warp perspective matrix and trapezoid to square shape)

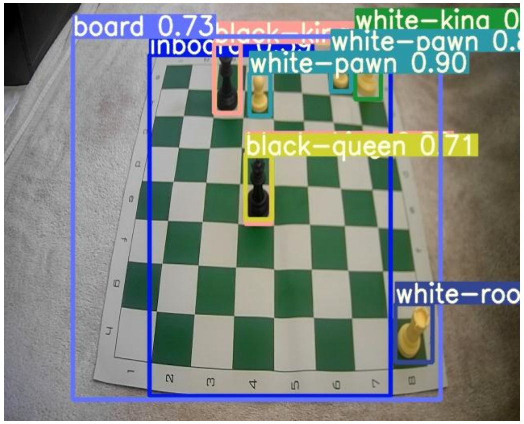
Board Labelling (prediction)

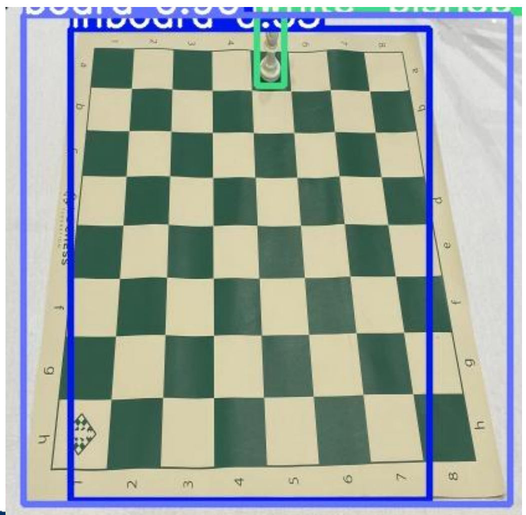


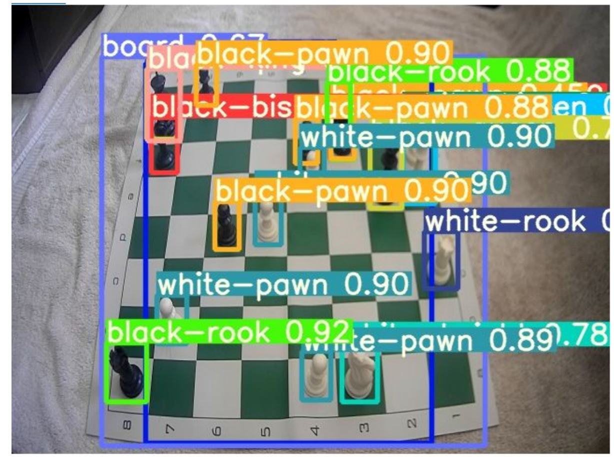


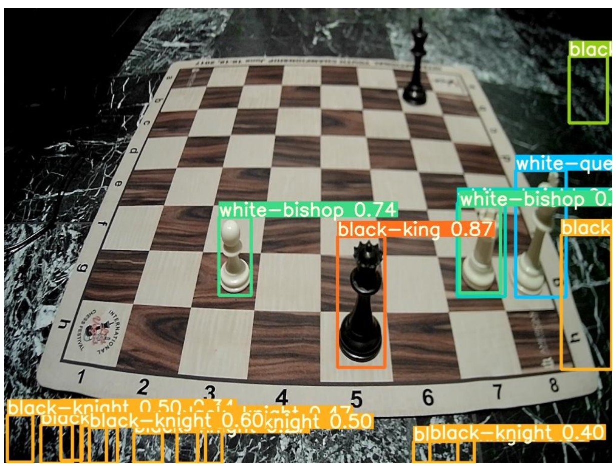
Berkeley
UNIVERSITY OF CALIFORNIA

Pieces labelling (prediction)





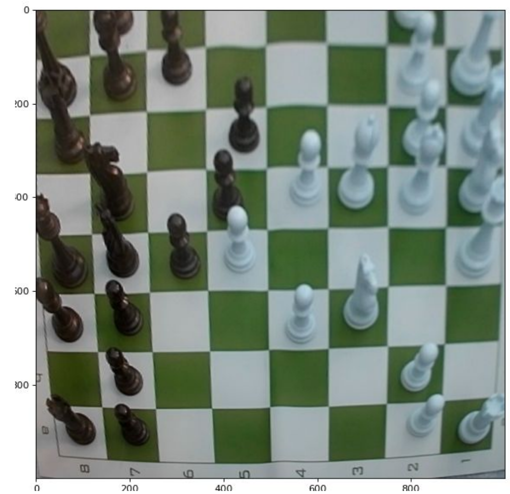




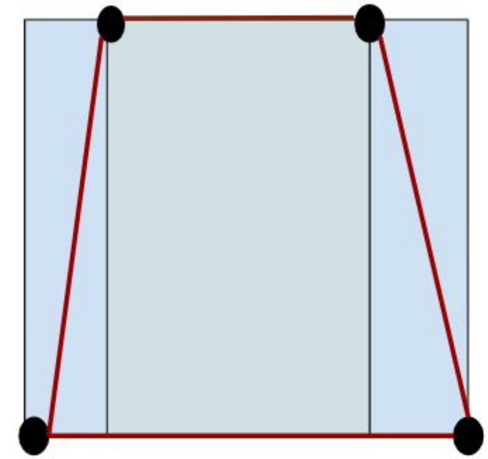
Berkeley
UNIVERSITY OF CALIFORNIA

Trapezoid to Square

Original



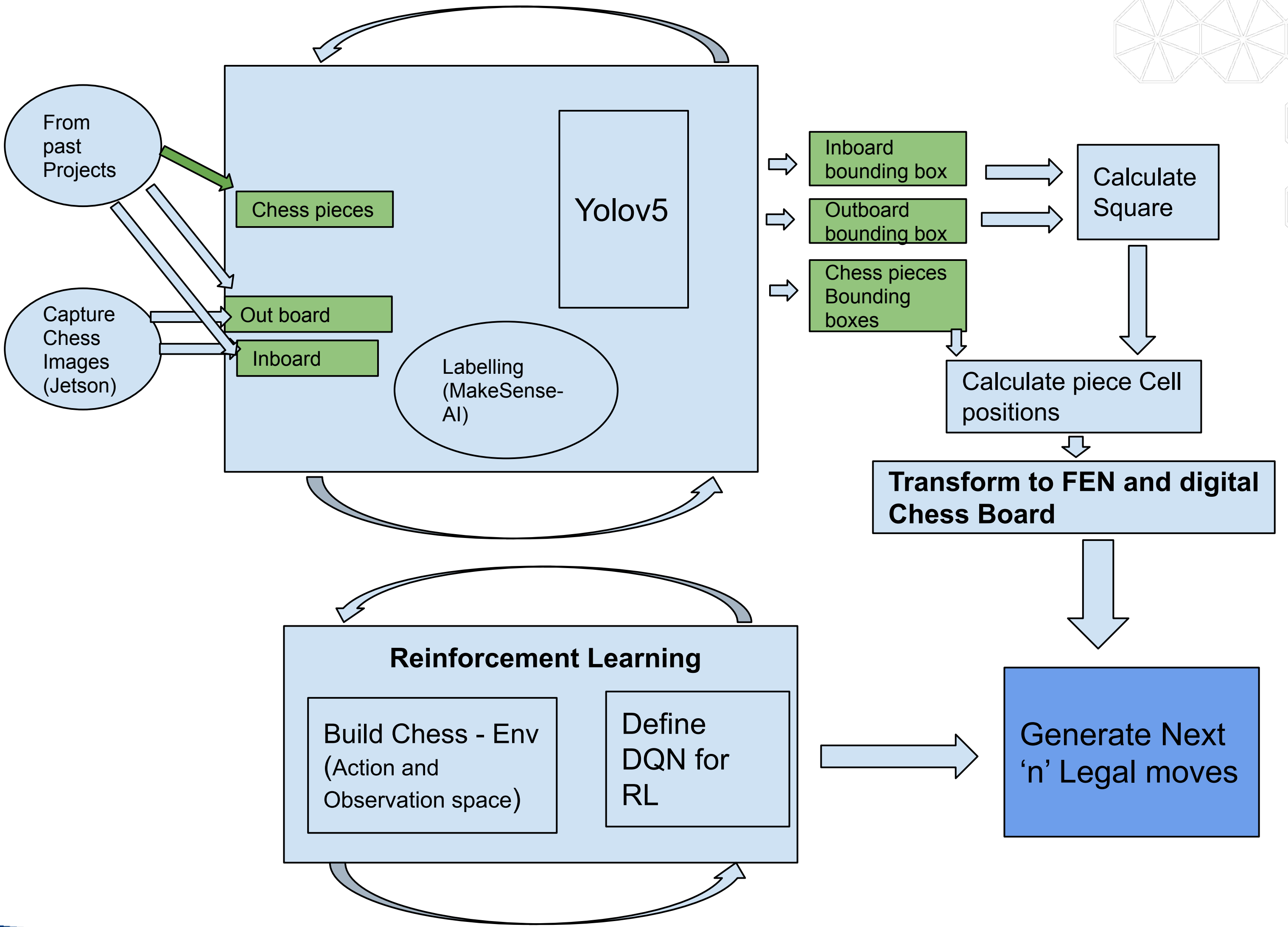
Rectified



1. Get coordinates from inboard and Outboard
2. Form trapezoid from step 1 coordinates
3. Use cv2 Perspective transform to transform to square (1000 x 1000). This also gives transformation matrix
4. For each Chess piece location, transform (dot product) using above transformation matrix to get the location on square board
5. Calculate approximate cell position based on number of cells in chess board

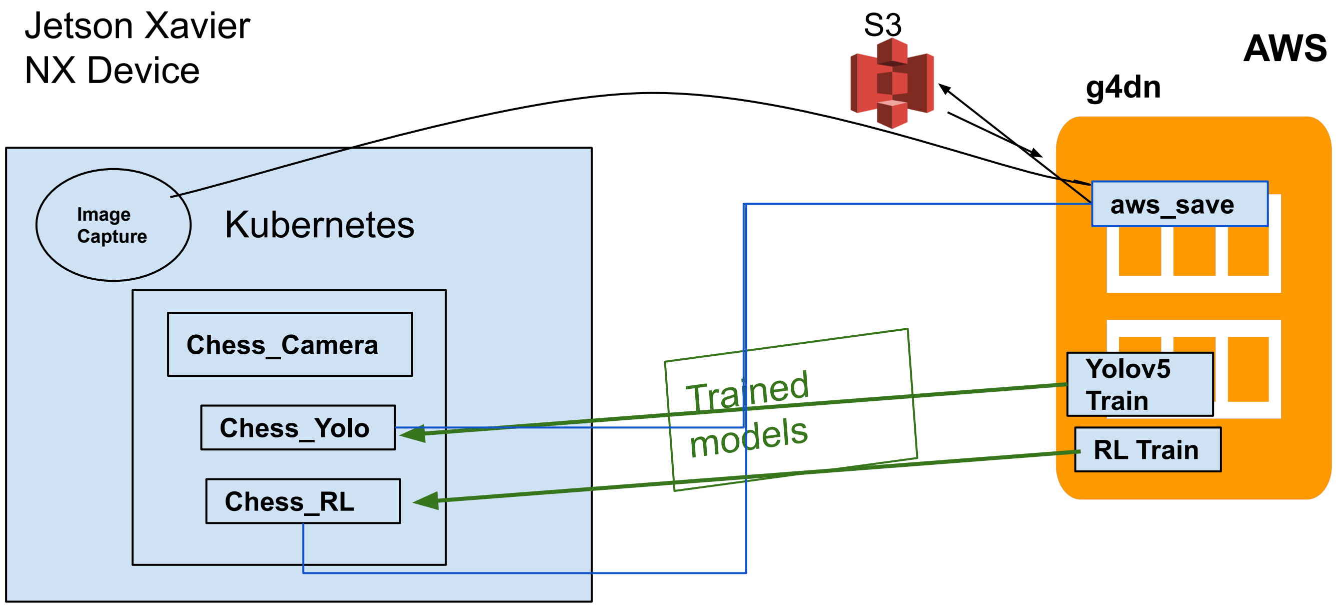
Berkeley
UNIVERSITY OF CALIFORNIA

Approach



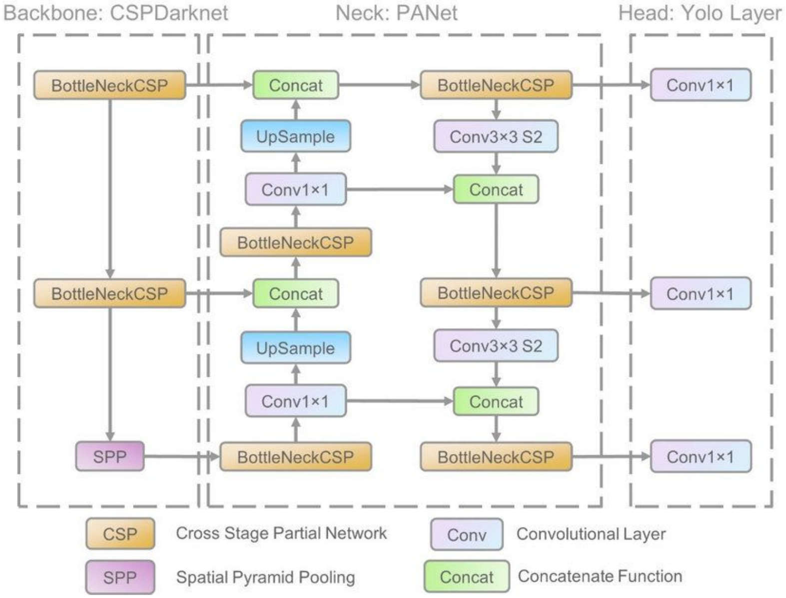
From past Projects
Capture Chess Images (Jetson)
Chess pieces
Out board
Inboard
Yolov5
Labelling (MakeSense-AI)
Inboard bounding box
Outboard bounding box
Chess pieces Bounding boxes
Calculate Square
Calculate piece Cell positions
Transform to FEN and digital Chess Board
Reinforcement Learning
Build Chess - Env (Action and Observation space)
Define DQN for RL
Generate Next 'n' Legal moves

System architecture



Yolov5 - Model

- Medium model on AWS machine
- Maximum of 200 epochs
- default LR of 1 e −5 and decay parameters
- Adam optimizer
- Batch size : 8 to 64
- use mixup option in detect
- image size 640 x 640



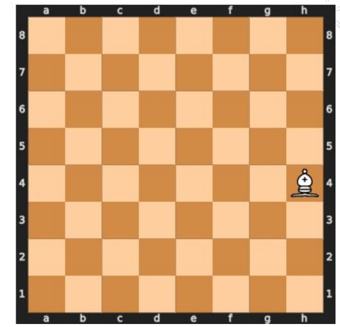
Berkeley
UNIVERSITY OF CALIFORNIA

Final Demo results

Camera

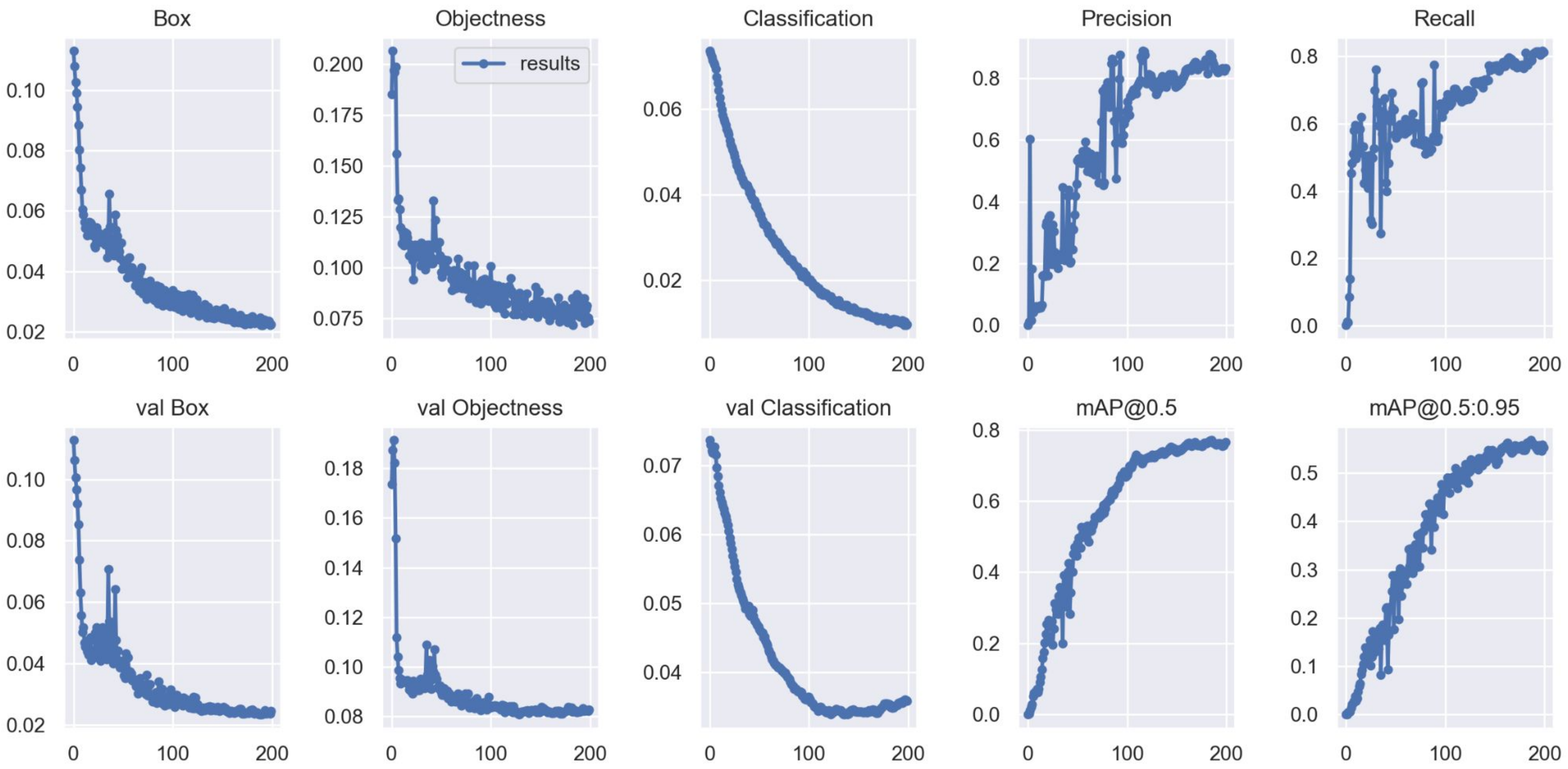
Digital

Camera



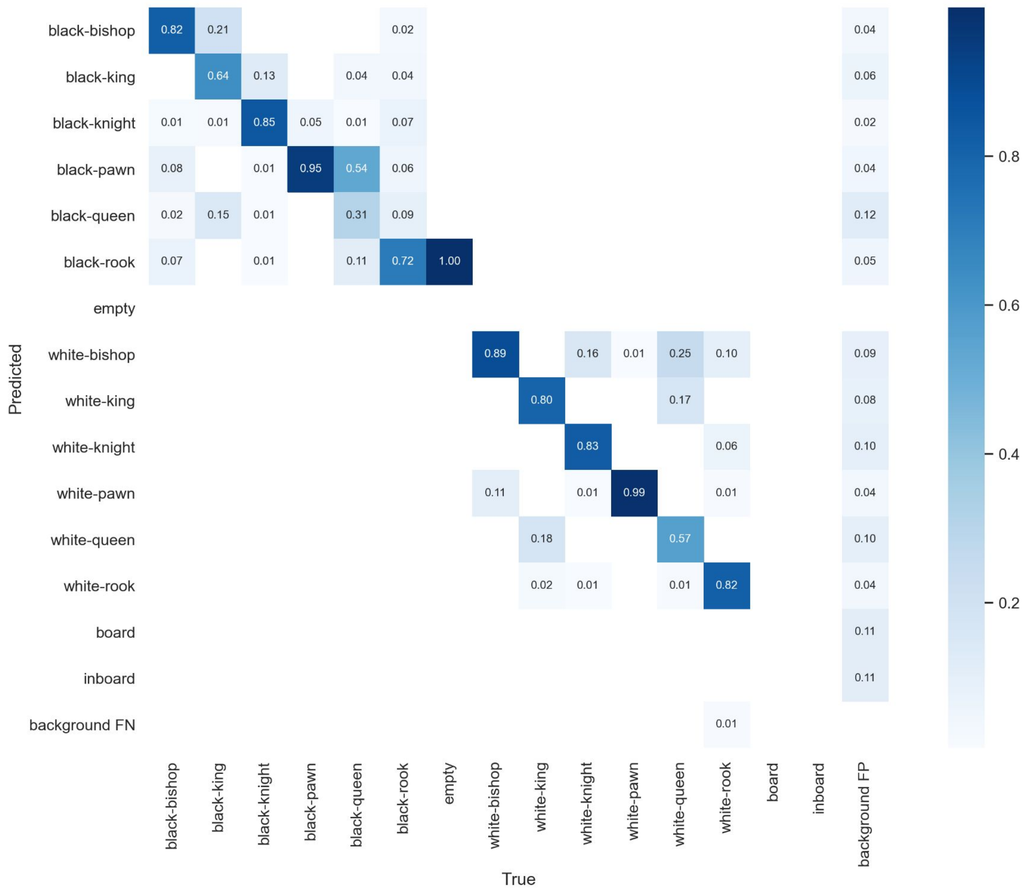
Digital

Model training metrics - Yolov5



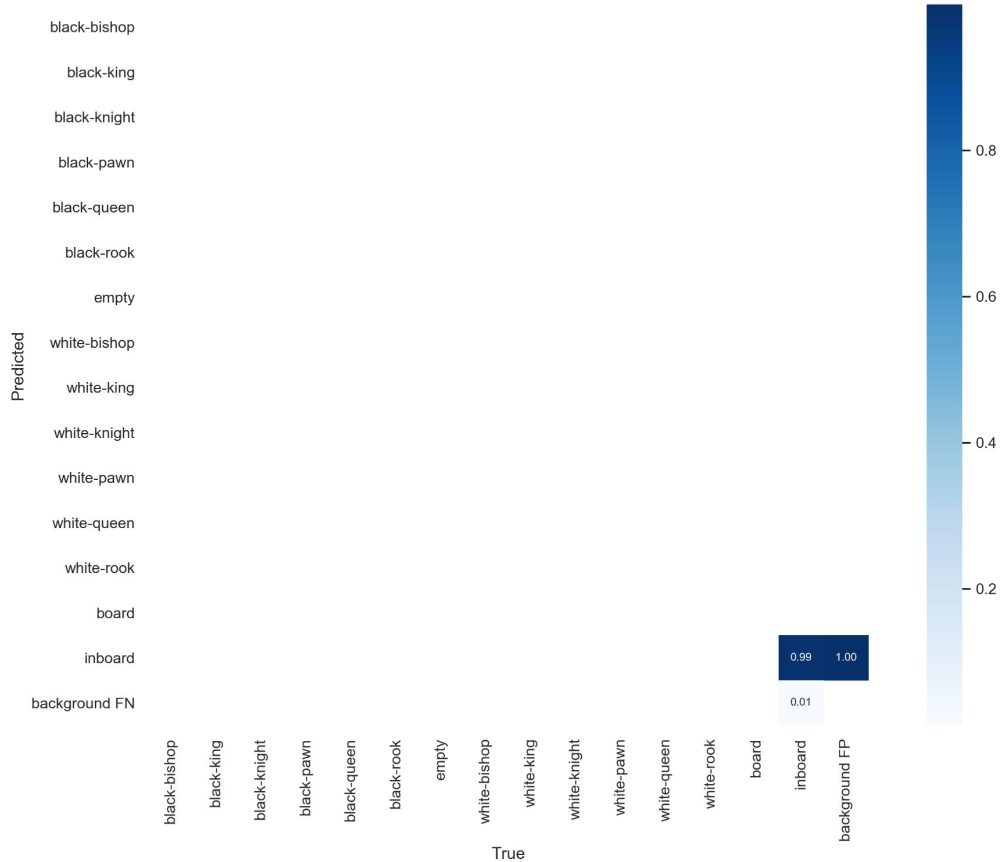
Berkeley
UNIVERSITY OF CALIFORNIA

Confusion Matrix



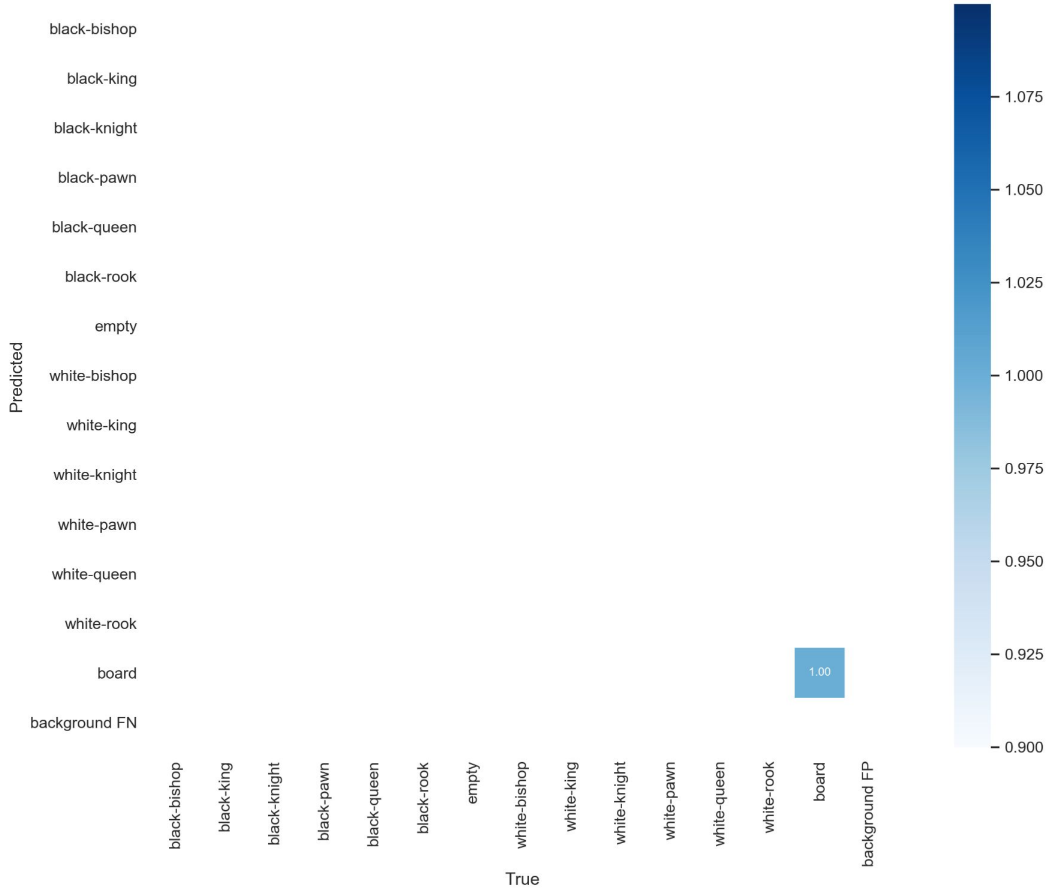
Berkeley
UNIVERSITY OF CALIFORNIA

Confusion Matrix for inboard



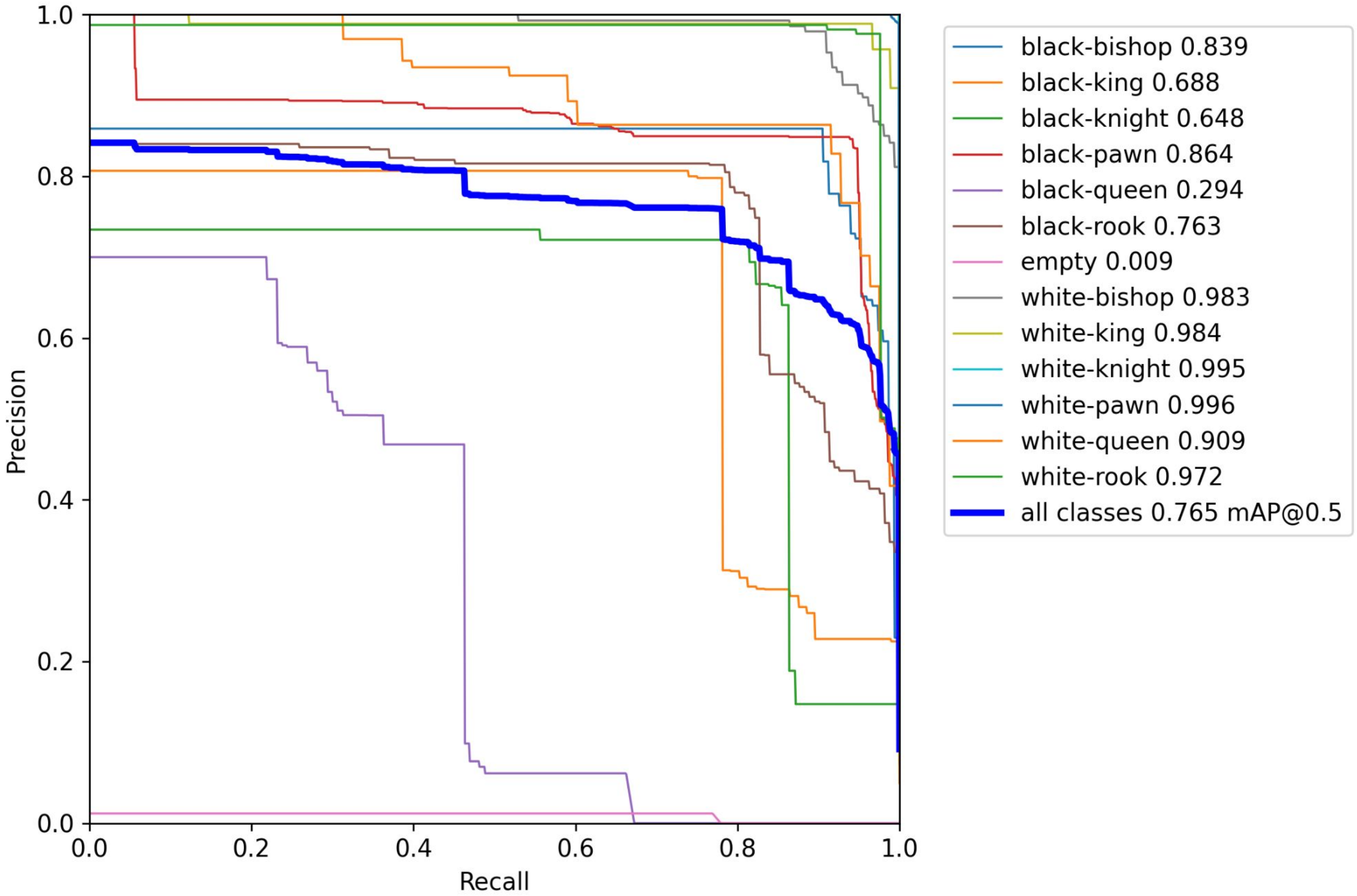
Berkeley
UNIVERSITY OF CALIFORNIA

Confusion Matrix for outboard



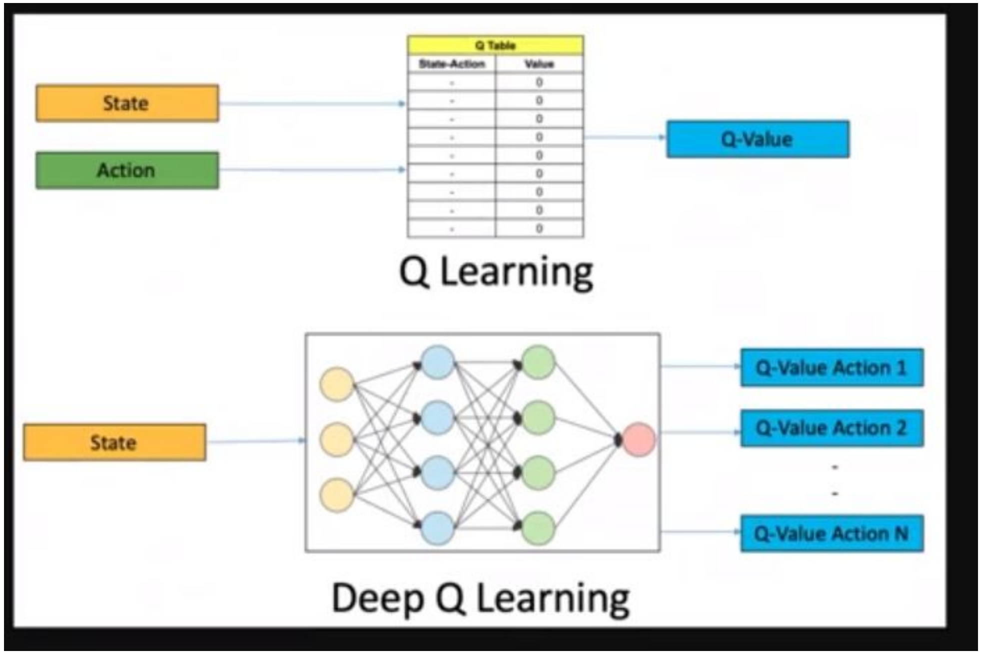
Berkeley
UNIVERSITY OF CALIFORNIA

Precision / Recall Curve



Berkeley
UNIVERSITY OF CALIFORNIA

Reinforcement learning



Chess is infinite: There are 400 different positions after each player makes one move apiece. There are 72,084 positions after two moves apiece. There are 9+ million positions after three moves apiece. There are 288+ billion different possible positions after four moves apiece.

Berkeley
UNIVERSITY OF CALIFORNIA

Legal Move - Reinforcement Learning

- Create a new Gym environment for Chess
- Define Action Space
- Define Observation Space
- Define Step function with rewards
- Tune Epsilon, Epison decay, Gamma, Batch size, Memory, definition of Done
- Define DQN with loss function and optimizer
- Create two networks based on DQN (i.e. current network and target network)
- Update the target network based on configuration

Next State $Q^{\pi}(s,a) = r + \gamma Q^{\pi}(s', \pi(s'))$

Minimize $\delta = Q(s,a) - (r + \gamma \max_a Q(s', a))$

RL - setup

Observation Space

[1,2,3,4,5,6,7,8]
[9,10,11,12,13,14,15,16]

[25,26,27,28,29,30,31,32]
[17,18,19,20,21,22,23,24]

Action space sample (Rook)

[[1,0,1],[1,0,2],[1,0,3],[1,0,4],[1,0,5],[1,0,6],[1,0,7],
[1,1,1],[1,1,2],[1,1,3],[1,1,4],[1,1,5],[1,1,6],[1,1,7],
[1,2,1],[1,2,2],[1,2,3],[1,2,4],[1,2,5],[1,2,6],[1,2,7],
[1,3,1],[1,3,2],[1,3,3],[1,3,4],[1,3,5],[1,3,6],[1,3,7]

DQN

FC, Relu and Batch Norm
loss function :Huber loss / Smooth L1 loss
Optimizer: RSMProp / AdamW (lr = 0.0001)

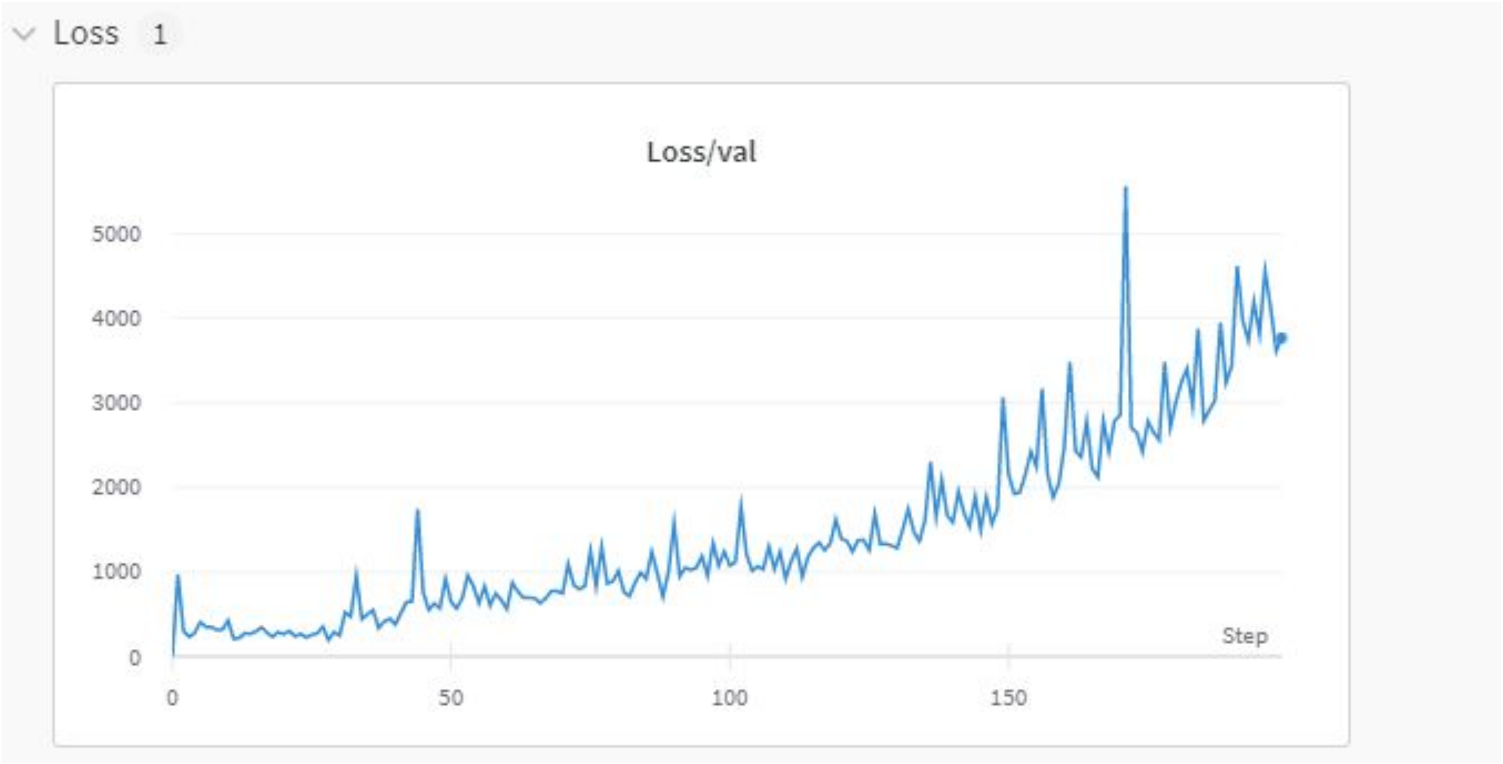
$$\Delta x_j^{(t)} = \text{momentum\_decay\_factor} \cdot \Delta x_j^{(t-1)} - \frac{\text{learning\_rate}}{\sqrt{\text{MA}\left(g_j^2\right)}} \cdot g_j^{(t)}$$

Objective

Find Legal Move

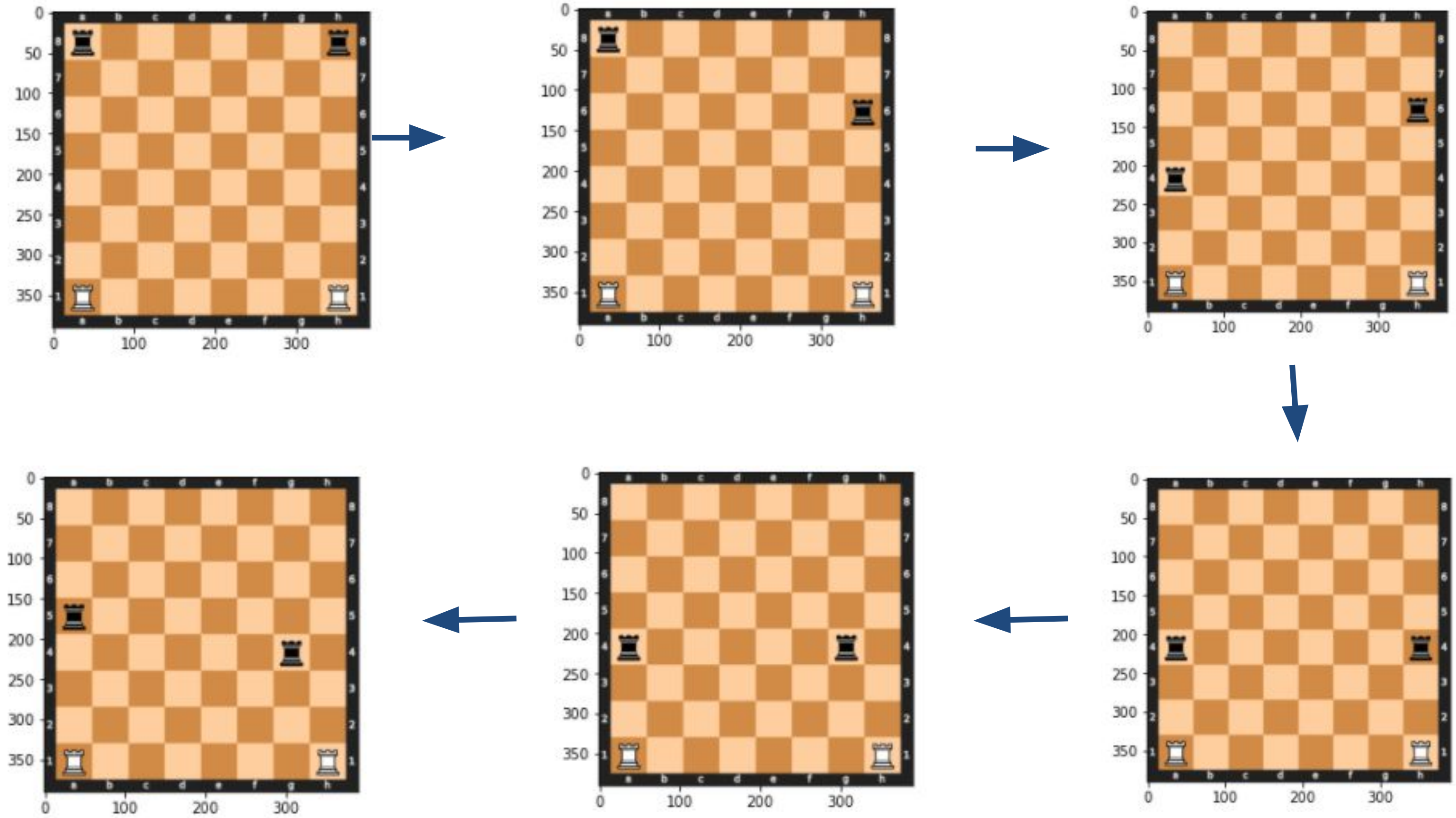
Berkeley
UNIVERSITY OF CALIFORNIA

RL Model metrics

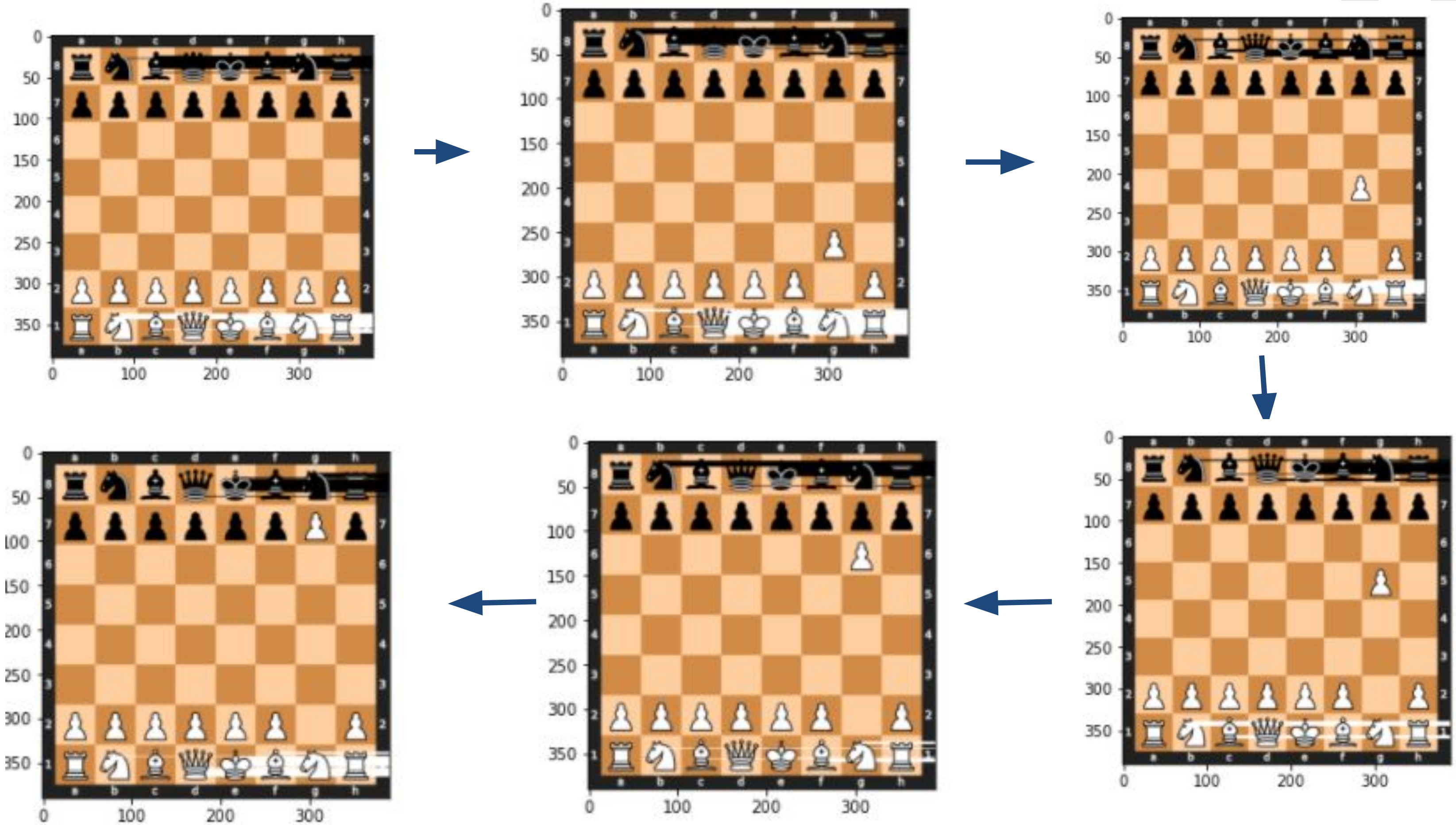


Berkeley
UNIVERSITY OF CALIFORNIA

Reinforcement Learning – Test example



Reinforcement Learning – Test example



Next Steps

- Refine Chess Identification
- Enhance to support all moves including enpasson and castling
- Leverage other open-source chess engines to rate and identify next best move
- Deploy on mobile device
- Integration with Chess clock

Berkeley
UNIVERSITY OF CALIFORNIA

References

1. ChessVision: Chess Board and Piece Recognition
2. Board Game Image Recognition using Neural Networks | by Andrew Underwood
3. Next Article: 4 Point OpenCV getPerspective Transform Example
4. Zeta36/chess-alpha-zero: Chess reinforcement learning by AlphaGo Zero methods.
5. FICS Games Database
6. Reinforcement learning (RL) 101 with Python | by Gerard Martínez
7. PyTorch for Beginners: Semantic Segmentation using torchvision
8. AlphaZero: Shedding new light on the grand games of chess, shogi and Go
9. Building Chess ID. Featuring: Computer Vision! Deep… | by Daylen Yang
10. YOLO Object Detection from image with OpenCV and Python

Questions