

Project 1: Chessboard Square Classification and Board-State Reconstruction (Real Data)

Description:

Given real chessboard images, the goal is to classify each of the 64 board squares into one of the possible piece classes (white pawn, black knight, etc.).

The system should then reconstruct the chess diagram (based on the FEN notation) based on the per-square predictions.

You will receive labeled frames with the corresponding chessboard state (piece-square positions only, occlusions are not labeled), as well as PGN labeled games.

The PGN files contain the full game state information, which can be used to easily generate additional frame level labels. (You have all board states from the PGN throughout the game, but you don't know which state corresponds to each frame.)

Additional notes:

1. **You don't have to** use the PGN games, but you might find it useful.
2. **You are allowed** to use more datasets from the internet if you found it useful. (i.e, free datasets that you can use <https://universe.roboflow.com/>).

3. **You are allowed** to leverage the temporal nature of the video (i.e., consecutive frames) to generate additional labeled data from the PGN files.

For example, you can do so either by manually labeling frames or by applying any algorithm of your choice, classical methods, self-/unsupervised approaches, your own trained models, or off-the-shelf models that will label your data.

4. **Your classifier must not rely on temporal information** (e.g., what happened in previous or subsequent frames). Its only input is a single static image of the board, and its output is the per-square classification.

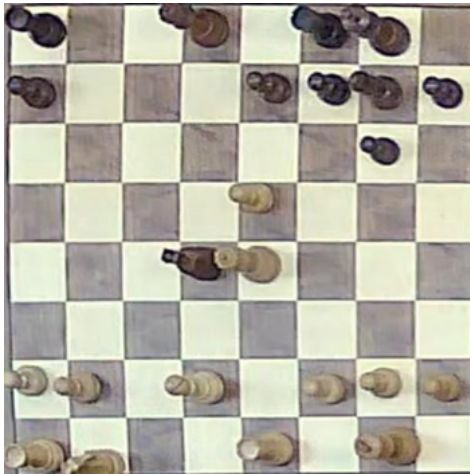
5. **Learn about Out-of-Distribution (OOD) methods**, it might be useful. Understand their core principles, and apply them to handle occlusions. Specifically, if a square is occluded, your overall algorithm should output “**unknown**” for that square. This does not have to be done by the neural network itself, it can also be handled through pre or post-processing of the model's predictions or via pre-processing (before you apply your model).

6. **You don't have to** use OOD, you can use any idea to solve the occlusion.

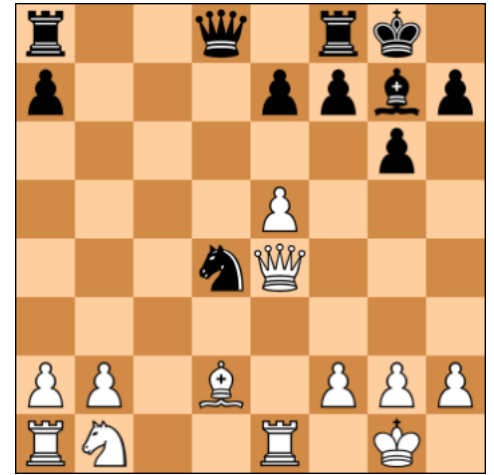
Key Components:

- Multi-class classification **per square**
- Handling occlusions and uncertainty (Out of Distribution / dustbin class) **per square**
- Classifying the entire board state and producing a (FEN) and board image
- Robustness to new games.

Example:



--> Classification --> FENN -->



You will be provided with ZIP files for each game.

Each ZIP contains an **images/** folder and either a **game.csv** file or a **PGN** file with the corresponding labels.

The **game.csv** labeling works as follows: for every distinct board state in the game, there is a specific frame that corresponds to that state.

1	from_frame	to_frame	fen
2	200	200	rnbqkbnr/pppppppp/8/8/8/PPPPPPPP/RNBQKBNR
3	588	588	rnbqkbnr/pppppppp/8/8/3P4/8/PPP1PPPP/RNBQKBNR
4	620	620	rnbqkbnr/ppp1pppp/8/3p4/3P4/8/PPP1PPPP/RNBQKBNR
5	840	840	rnbqkbnr/ppp1pppp/8/3p4/2PP4/8/PP2PPPP/RNBQKBNR
6	856	856	rnbqkbnr/ppp2ppp/4p3/3p4/2PP4/8/PP2PPPP/RNBQKBNR
7	872	872	rnbqkbnr/ppp2ppp/4p3/3p4/2PP4/2N5/PP2PPPP/R1BQKBNR
8	896	896	rnbqkb1r/ppp2ppp/4pn2/3p4/2PP4/2N5/PP2PPPP/R1BQKBNR
9	936	936	rnbqkb1r/ppp2ppp/4pn2/3p2B1/2PP4/2N5/PP2PPPP/R2QKBNR
10	1024	1024	rnbqk2r/ppp1bppp/4pn2/3p2B1/2PP4/2N5/PP2PPPP/R2QKBNR
11	1040	1040	rnbqk2r/ppp1bppp/4pn2/3p2B1/2PP4/2N1P3/PP3PPP/R2QKBNR
12	1124	1124	rnbq1rk1/ppp1bppp/4pn2/3p2B1/2PP4/2N1P3/PP3PPP/R2QKBNR
13	1152	1152	rnbq1rk1/ppp1bppp/4pn2/3p2B1/2PP4/2N1PN2/PP3PPP/R2QKB1R
14	1172	1172	rnbq1rk1/ppp1bpp1/4pn1p/3p2B1/2PP4/2N1PN2/PP3PPP/R2QKB1R
15	1188	1188	rnbq1rk1/ppp1bpp1/4pn1p/3p4/2PP3B/2N1PN2/PP3PPP/R2QKB1R
16	1216	1216	rnbq1rk1/nnn1hnn1/4n2n/3n4/2PPn2R/2N1PN2/PP3PPP/R2QKB1R

The **from_frame** and **to_frame** fields are identical. They both indicate the frame number in which the board appears in the state described by the **fen** column. (frames might have the same board state, so we chose only 1 for each state).

Some occlusions examples:



You should use python packages to generate the chess diagram given a fen.

Good luck!