CVChess: A Deep Learning Framework for Converting Chessboard Images to Forsyth–Edwards Notation

Luthira Abeykoon
Student# 1010410397
luthira.abeykoon@mail.utoronto.ca

Darshan Kasundra
Student# 1010292434
darshan.kasundra@mail.utoronto.ca

Gawtham Senthilvelan
Student# 1009977704
gawthaman.senthilvelan@mail.utoronto.ca

Ved Patel
Student# 1010140447
ved.patel@mail.utoronto.ca

Abstract

Chess has experienced a large increase in viewership since the pandemic, driven largely by the accessibility of online learning platforms. However, no equivalent assistance exists for physical chess games, creating a divide between analog and digital chess experiences. This paper presents CVChess, a deep learning framework for converting chessboard images to Forsyth-Edwards Notation (FEN), which is later input into online chess engines to provide you with the best next move.

Our approach employs a convolutional neural network (CNN) architecture to perform piece recognition from smartphone camera images. The system processes RGB images of a physical chess board through a multistep process: image preprocessing using the Hough Line Transform for edge detection, projective transform to achieve a top-down board alignment, segmentation into 64 individual squares, and piece classification into 13 classes (6 unique white pieces, 6 unique black pieces and an empty square) using a CNN. We train and evaluate our model using the Chess Recognition Dataset (ChessReD), containing 10800 annotated smartphone images captured under diverse lighting conditions and angles. The resulting classifications are encoded as an FEN string, which can be fed into a chess engine to generate the most optimal move. —-Total Pages: 8

1 Introduction

Chess has seen a dramatic resurgence in popularity, with tournament viewership rising by 55% from 2020 to 2024 (ChessWatch, 2024). A key factor behind this surge can be attributed to the growing presence of chess in live streaming, which has made learning the game more accessible than ever. People are interested in chess, and more importantly, people want to get better at chess. While online platforms provide learning methods such as the 'hint' button on Chess.com, which recommends the optimal move at a given game state, no such convenience exists for physical chess games.

Our team proposes an easy-to-use tool which can quickly inform users of the most optimal move to make. By using neural networks to detect piece positions from images of physical chess boards at random game states, we can generate a Forsyth-Edwards Notation (FEN) string, a standardized representation of a chessboard. This string can easily be fed into a chess engine to compute the most optimal move. Figure 1 outlines this process. Our system acts as a real-world 'hint' button, providing guidance for players of all skill levels. Deep learning works well for this project due to the object-detection abilities of neural networks, particularly in complex spatial settings such as a physical chessboard.

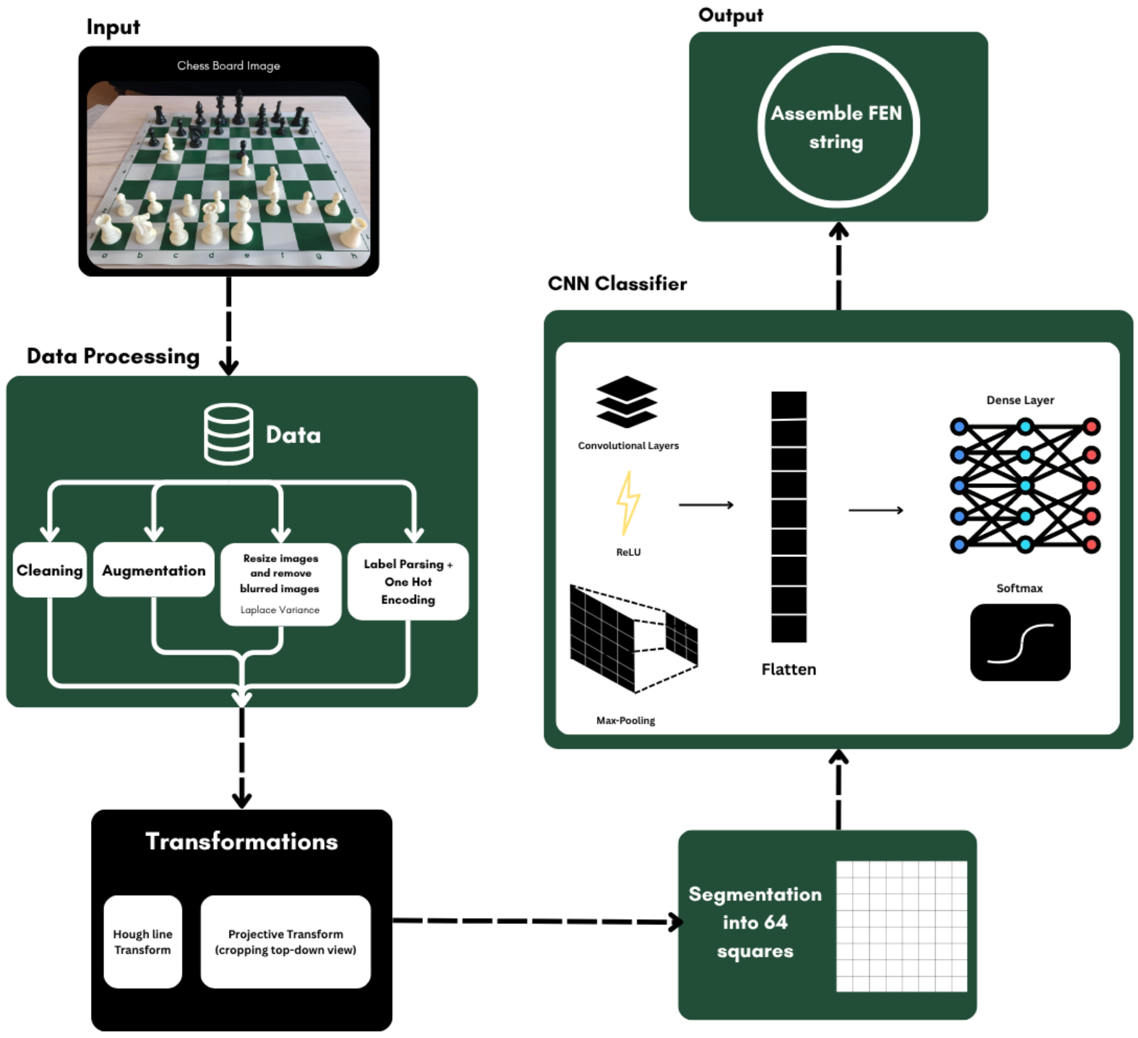


Figure 1: Overview of our model pipeline for real-world chessboard recognition.

2 Background and Related Work

Much research has been conducted on detecting chess pieces on physical chess boards and keeps on improving as we speak. These studies explore different methods of detecting the board such as using CNNs, x/y gradients, and much more. This section will explore key research in computer vision and its application to identifying chess boards.

2.1 End-to-End Chess Recognition (Masouris & van Gemert, 2023)

Masouris and van Gemert's work in End-to-End Chess Recognition utilizes the ResNeXt-101 CNN to process the raw images. Their approach skips traditional neural networks and rather trains the model to implicitly learn where the chess board is, understand the spatial layout of the 64 squares and classify what's in each square (12 unique pieces - 6 black & 6 white plus empty). This classification is then converted to FEN notation. They correctly classified 15.26% of ChessRed's test images which is 7x better than any other current state-of-the-art.

2.2 Efficient Chess Vision – A Computer Vision Application (Wu, 2022)

Wu's approach in Efficient Chess Vision – A Computer Vision Application avoids CNNs until after board identification. His method uses a gradient-based technique where the input image is converted to greyscale and x, y gradients are calculated to identify chess board boundaries. Each square within the board undergoes separate identification and pieces are classified into 13 distinct classes before conversion to FEN format. The model achieved 99.5% accuracy using a medium CNN.

2.3 Chessboard and Chess Piece Recognition With the Support of Neural Networks (Stojanovic et al., 2017)

The approach in this paper is similar to those discussed above as it employs CNNs to process input images. The chess board identification follows three-steps: detecting straight lines, locating line intersections, and clustering lattice points to identify the chess board. Their piece recognition relies on color and shape descriptors, along with physical properties such as height and area for classification. The results are then converted to FEN format. This model achieved 95% accuracy in piece detection.

2.4 Chess-CV (Rizo-Ramirez, 2021)

The method in Chess-CV (a GitHub baked project) projects the chess board onto a 2D plane and employs edge detection algorithms to identify the board. The board is divided into 64 individual squares, where each square is fed into a trained CNN for piece classification. The pieces are classified into 13 distinct classes, and this information is stored in a NumPy array of FEN strings representing the board position. The model achieved 88.9% accuracy on validation data.

2.5 ChessVision: Chess Board and Piece Recognition (Ding et al., 2016)

The approach in ChessVision: Chess Board and Piece Recognition avoids neural networks entirely, and instead utilizes computer vision and machine learning techniques such as SVM and Scale-Invariant Feature Transforms. Their method requires manual selection of the chess board's four corners. For piece recognition, they use variable-sized bounding boxes tailored to different pieces and use SVM classifiers to individually analyze each of the 64 squares. Their output is a visual representation of the board instead of a FEN string. This model achieved 95% detection accuracy and 85% classification accuracy.

3 Dataprocessing

The Chess Recognition Dataset (ChessReD) (de Winter et al., 2022) contains a collection of annotated chess board images captured using smartphone cameras, namely the iPhone12, Huawei P40 Pro and Samsung Galaxy S8, to simulate real-world chess recognition scenarios. Images were collected by replaying 100 real chess games selected using 100 unique Encyclopedia of Chess (ECO) opening codes. Using Portable Game Notation (PGN) data, each game was physically reconstructed move by move on a standard chess board, with an image captured after every move. To introduce visual diversity, images were captured from top-down and oblique angles, under both natural and artificial lighting. In total, this dataset includes 10800 images, each with a FEN string (ground truth) to describe the state of the chess board in algebraic notation. Additionally, a smaller subset of data, ChessReD2K, includes approximately 2,078 images across 20 games, with extra metadata such as bounding boxes and board corner annotations.

In addition to using this public dataset, we will collect a custom dataset using a standard Staunton chessboard. Photos will be taken under various lighting conditions and angles. The images will be labeled manually verified FEN strings and used to test model generalization or further fine-tuning.

The following preprocessing steps will be applied by the team:

- **Cleaning:**
  - Images with extreme blur, bad lighting will be filtered using Laplacian variance and brightness thresholds.
  - FEN string will be validated to ensure legal board states.
  - Misaligned annotations, NAN values, and unreadable boards will be removed.
- **Image Preprocessing:** All images will be resized to 512x512 pixels and normalized using ImageNet mean and standard deviation values.
- **Label Parsing and One Hot Encoding:** FEN strings provided with each image will be parsed into a 64 element label array, corresponding to the 8x8 board squares. Each square will be encoded as a one-hot vector over 13 classes (12 piece types and empty square), resulting in a (64, 13) label tensor, which aligns with the model output.
- **Grid Mapping:** Each image will be treated as an 8x8 grid of squares (chessboard dimensions). The model will output predictions for each square, which will be compared against the one hot encoding labels.
- **Augmentation:** To simulate real-world noise and improve generalization, we will apply the following augmentations during training:
  - Random brightness and contrast transforms
  - Perspective distortion transforms
  - Addition Gaussian noise/blur
  - Motion blur
  - Orientation transformations (horizontal and vertical rotations by 90 degrees)
- **Splitting the dataset:** The data will be split into common ratios in accordance with training, testing, and validation of 80:10:10 and 70:10:20. The team will also apply a 60:20:20 split in order to match the existing dataset configuration.

In addition to the ChessReD dataset and to improve model robustness, the team will combine and concatenate other datasets:

- **Fenify-3D** (Notnil, 2022): A mix of 30,000 synthetic board renders and 9500 crowdsourced real-world images, all labeled with FEN. This dataset offers diversity in lighting and camera angles but may have inconsistent annotations due to its crowdsourced nature.
- **Synthetic Chess Board Image** (TheFamousRat, 2023): This dataset consists of algorithmically generated boards with perfectly annotated piece positions in various unique backgrounds. YThese renders give us the ideal conditions and layout variety but they don't include real-world artifacts such as glare, and reflections.

By blending this with the ChessReD, we have a balance between realistic capture conditions and perfect annotations, helping the model generalize across both noising camera feeds and ideal layouts. For further augmentation techniques such as slicing and shuffling could also be employed as necessary.

4 Architecture

There are several key components the model architecture must address. Firstly, we must have some form of image preprocessing where the input is an RGB image of a physical chessboard (captured by the user's camera) and the output would be a cropped top-down view of the board. During this, we would need to convert the image to grayscale and use some method of corner detection, specifically the Hough Line Transform, to detect the outer corners of the board and the internal 8x8 grid lines. We will also using a projective transform matrix, as shown in the ChessVision project (Ding et al., 2016) to create a top-down view. We will then segment the board into its 64 individual square images, where the input would be the top-down view of the board, and the output is the individual squares.

This is then followed by some form of piece recognition where the input would be 64 cropped square images and the output would be the classification of each square (e.g. "white rook", "black queen", "empty"). Past works have gone about this differently, where they separated this step into two intermediary stages, where, for example, a black bishop and a white bishop would first be classified as a "bishop" and then later classified as either a white or black piece. We have chosen to keep this as one step as we believe it would allow our model to remain more straightforward, which would prove useful when debugging. For piece recognition, we will train a CNN classifier with 13 classes, being the 6 unique white pieces, 6 unique black pieces and 1 empty square. The input shape to the CNN will be 3 x 64 x 64 (RGB image of a single square).

This will be fed into the first layer, which is a convolutional layer with the ReLU activation function and a max pooling operation. In the convolutional layer, we extract the basic edges, textures and colours, where a 3x3 kernel slides over the image. The activation function introduces non-linearity, enabling the model to learn complex patterns during piece recognition, and the max pool helps reduce the need for computational power. We will repeat this process for 3-4 more layers to learn the different intricacies of each piece. The resulting 3D feature map will be flattened into a 1D vector and passed through fully connected layers for final classification into one of the 13 output classes.

Once the pieces have been classified, they can easily be converted to FEN notation which can be fed into a stockfish engine to return the next best move. We will use an open-source stockfish chess engine which implements a convenient stockfish class in Python.

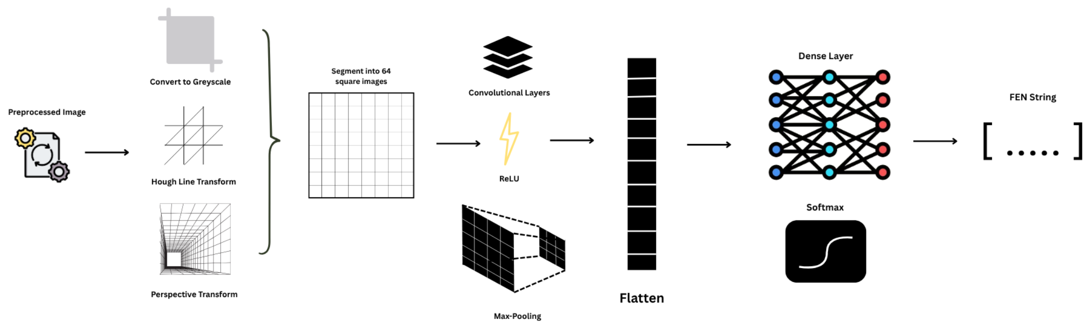


Figure 2: Overview of our model architecture.

5 Baseline Model

To build a solid baseline for FEN generation, we will use a simple CNN designed for square-level classification. The preprocessing will follow a similar pipeline to our main model, meaning board detection, alignment, square segmentation into an 8x8 array, and normalization. The CNN will classify each square into one of 13 classes and will consist of two convolutional layers with ReLU activations, followed by 2 fully connected layers. This setup will produce a 13-dimensional vector for each square in the 8x8 array. This prediction will then be concatenated and converted into a FEN string. This closely aligns with previous work done with LiveChess2FEN, a framework for classifying chess pieces (Ravi et al., 2020).

6 Ethical Considerations

With the development of this project, several ethical considerations must be addressed if the project is released commercially. Firstly, users may use the model in competitive settings (either online or in-person), hindering competitive integrity. Professional players could use the model to analyze a given board position and identify the strongest possible move, giving them a strategic advantage over their opponents. These moves would not accurately reflect the user's understanding of the game, and thus cheating prevention measures must be set in place. Currently, phones are banned from being kept on professional players' bodies during competitive chess matches, suggesting that this should not be a significant issue, as safety measures are already in place. Nonetheless, this potential ethical issue should be considered if the model is released publicly.

Even in casual settings, the implementation of the model may cause an over-reliance on AI in chess games, preventing players from improving. Using this model too often could diminish the user's ability to develop their decision-making skills, limiting their growth, game sense, and ability to learn from mistakes. Recreational chess players often struggle to understand engine-recommended moves because the engine evaluates positions several moves ahead, making its suggestions seem unclear or unintuitive. Furthermore, recreational players may feel discouraged and unmotivated when facing opponents who appear to be of similar skill level but are actually relying on engine-generated moves, creating an unfair advantage.

Ultimately, while the model provides the user with a powerful tool to analyze chess positions and improve gameplay, its use must be carefully monitored to uphold competitive integrity and the spirit of learning. Creating clear guidelines and safeguards to prevent misuse in both professional and casual settings is essential for addressing these ethical concerns to benefit the chess community without undermining its core values.

7 Project Plan

We will have weekly meetings every Wednesday at 9 PM. These meetings will be through Discord calls and messages throughout the week will be sent via Discord chats. We chose this communication method as it would ensure the highest possibility of quick responses from the team, due to our usage of the application. During our meetings, we will first discuss updates from the previous week, followed by a collaborative work session where members ask each other questions for their tasks. Meetings are concluded with next steps for the following week, and any final comments a member wants to add. With this structure, we can ensure that each member is kept up to date with the works of the other members and they can use their help during meetings if needed. Though currently meetings will occur every Wednesday, as we get deeper into the project, we will call more frequent meetings as needed to ensure deadlines are met. Lastly, we will use Github to ensure that our code is kept safe and that we do not overwrite each other's work and all written reports will be stored in a shared google drive.

The following table outlines task distribution for Milestone 1 (Project Proposal):

Table 1: Milestone 1: Project Proposal Tasks and Deadlines

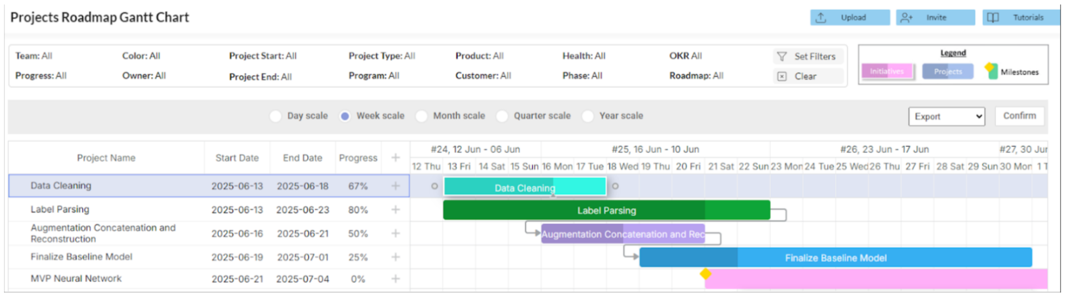
Member	Task	Deadline
Luthira Abeykoon	Data Processing	June 11th
	Baseline Model	June 11th
	Illustrations (final)	June 11th
Ved Patel	Introduction	June 11th
	Illustration (overview)	June 11th
	LaTeX/formatting	June 13th
Darshan Kasundra	Background	June 11th
	Risk Factors	June 11th
Gawtham Senthilvelan	Architecture	June 11th
	Project Planning	June 11th
	Ethical Considerations	June 11th
Everyone	Editing	June 13th
	Proofreading	June 13th

Looking ahead to Milestone 2 (Project Progress Report), we have assigned tasks as shown below:

Table 2: Milestone 2: Progress Report Tasks and Deadlines

Member	Task	Deadline
Luthira Abeykoon	Data Cleaning	June 18th
Darshan Kasundra	Augmentation Concatenation and Reconstruction	June 21st
Ved Patel	Label Parsing	June 23rd
	Minimum Viable Product Neural Network	July 4th
Gawtham Senthilvelan	Finalize Baseline Model	July 1st
Everyone	Progress Report Writeup	July 11th

For project management, we will use Mach AI, a user-friendly gantt chart website that allows us to visually determine project progress. The gantt chart will be updated by a different member each week, during our weekly meetings.



Team: All
Color: All
Project Start: All
Project Type: All
Product: All
Health: All
OKR All
Progress: All
Owner: All
Project End: All
Program: All
Customer: All
Phase: All
Roadmap: All

Day scale [✓ Week scale] Month scale Quarter scale Year scale

Project Name	Start Date	End Date	Progress
Data Cleaning	2025-06-13	2025-06-18	67%
Label Parsing	2025-06-13	2025-06-23	80%
Augmentation Concatenation and Reconstruction	2025-06-16	2025-06-21	50%
Finalize Baseline Model	2025-06-19	2025-07-01	25%
MVP Neural Network	2025-06-21	2025-07-04	0%

Figure 3: Project planning for Milestone 2.

8 Risk Register

To proactively address potential challenges, we have identified several key risks that may arise throughout development. These include both team-related and technical obstacles such as unexpected training times, data limitations, and model bias. For each risk, we have evaluated its likelihood and proposed concrete mitigation strategies to ensure the project remains on track.

Table 3: Risk Register

RISK	**LIKELIHOOD**	**SOLUTION**
Team member drops course	5% - Each member has completed 2nd year ECE and is interested in taking machine learning courses.	(1) Remaining work will be evenly distributed within the rest of the team members. (2) Internal deadlines will be expedited as there are less members to complete the tasks.
Training the model takes longer than expected	40% - Image processing of an entire chess board and pieces might be computationally heavy therefore is a possibility of a longer training time.	(1) Campus computers have a faster GPU than personal computers and therefore will allow us to train the model faster. (2) Scale down the size of the training data allowing us to train our model on a smaller portion first.
Inaccurate piece detection due to lighting/angle issues	60% - Each photo might vary in light intensity and angle from which the photo was taken making the model prone to errors.	(1) Collect data with various lighting intensity and different angles, exposing our model to many different photos.
Bias of model towards a specific chess board	55% - Model may perform better when photo contains a specific type of chess board	(1) Use multiple different data sets containing various types of chess boards to avoid bias towards a specific one.

9 Link to Github Repository

Here is the link to the team's Github repository: https://github.com/Luthiraa/CVChess

REFERENCES

ChessWatch. Chess events dynamics 2020–24. https://chesswatch.com/news/chess-events-dynamics-2020-24, 2024. Accessed: 2025-06-12.

Joost de Winter et al. Chess recognition dataset (chessred). https://data.4tu.nl/datasets/99b5c721-280b-450b-b058-b2900b69a90f/2, 2022. Accessed: 2025-06-12.

Jacky Ding et al. Chessvision: Chess board and piece recognition. https://web.stanford.edu/class/cs231a/prev_projects_2016/CS_231A_Final_Report.pdf, 2016. Stanford CS231A Final Report.

Spyridon Masouris and Jan C. van Gemert. End-to-end chess recognition. https://arxiv.org/abs/2310.04086, 2023. arXiv preprint arXiv:2310.04086.

Felix Notnil. Fenify-3d dataset for chessboard recognition. https://github.com/notnil/fenify-3D, 2022. Accessed: 2025-06-12.

Ankit Ravi et al. Livechess2fen: Deep learning model for generating chess notation from board images. https://ar5iv.labs.arxiv.org/html/2012.06858, 2020. Accessed: 2025-06-12.

Cristian Rizo-Ramirez. Chess-cv: Chess board detection and piece classification. https://github.com/Rizo-R/chess-cv, 2021. GitHub repository.

Nikola Stojanovic et al. Chessboard and chess piece recognition with the support of neural networks. https://arxiv.org/abs/1708.03898, 2017. arXiv preprint arXiv:1708.03898.

TheFamousRat. Synthetic chess board images. https://www.kaggle.com/datasets/thefamousrat/synthetic-chess-board-images, 2023. Accessed: 2025-06-12.

Victor Wu. Efficient chess vision – a computer vision application. https://cs231n.stanford.edu/reports/2022/pdfs/123.pdf, 2022. Stanford CS231n Final Project.