OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Cheat-Detection in Online Chess Using Convolutional Neural Networks and Engine Evaluations

by

Kai Mc Glacken

Research Supervisor: Dr. Frank Glavin

MSc in Software Engineering and Database Technologies

School of Computer Science,
College of Science and Engineering
University of Galway

Date: September 2023

Table of Contents

List of Figures

Abstract

This study uses a combination of convolutional neural networks and statistics to predict whether a player has cheated in a game of chess. Online chess is one of easiest games in which users can cheat, largely due to easily available chess-playing software that can be accessed through various chess websites. The model developed in this study classifies games into three categories, depending on whether the player with the white pieces cheated, the player with the black pieces cheated, or if neither player cheated. Over 12,000 online chess games were gathered, involving either normal games between regular players, or games between a computer and a regular player. The moves played in these games were then compared to moves chosen by a chess engine in the same position to calculate a similarity score between the engine moves and those played within the game. These scores were gathered for each game in the dataset and used to determine whether there is a threshold score for identifying cheaters. While individually these scores did not yield significant results, when combined with the convolutional neural network this provided greater classification accuracy.

Chapter 1: Introduction

1.1 Overview and Motivation

The interplay between Artificial Intelligence (AI) and the game of chess has captivated both researchers and game enthusiasts for decades. Since the early 1950s, chess has frequently been used as a benchmark for advancements in AI research, with the creation of a computer program capable of playing the game to a proficient level being pursued by many. In more recent years, these chess programs have advanced to the stage where they easily outperform even the world's top players. With this type of software being readily available to any person with a smartphone or computer, this has introduced new challenges for researchers in this domain, such as upholding general ethics and fair-play policies to counteract cheating. The strength of the current top chess-playing programs has necessitated innovative solutions to ensure the integrity of the game in competitive spaces, such as in online games or in live tournaments.

This study aims to use a combination of convolutional neural networks and statistics to determine whether a player has cheated in a game of chess. Due to easily available chess-playing engines that can be accessed through various chess websites, there are many opportunities for players to cheat within games. The model discussed in this study seeks to classify games into three categories, depending on whether the player with the white pieces cheated, the player with the black pieces cheated, or if no cheating was involved.

1.2 Research Objectives

Throughout this research, we investigate the current anti-cheating measures in play in online games and over the board, with a particular focus on methods that can detect cheating solely through the moves that are played in any given game. Based on this problem, the aim of this research is to expand upon and investigate the efficacy neural networks and statistics-based approaches in detecting cheating in online chess games. The classification model we aim to develop will potentially provide greater insight into how useful these current approaches are and how they may be improved upon.

1.3 Contributions of this Thesis

While many of the top chess websites employ their own software tools for evaluating games for instances of cheating, many do not reveal the full set of data, metrics, and type of tracking that they use. This may be considered reasonable, as a fully open-source software may show weaknesses in the system which people may then exploit to develop more refined cheating methods, which can then go undetected. However, as with any software, there are debates as to whether making it open source makes it more secure than closed source alternatives. Open-source software provides greater transparency to its users and other developers, allowing them to evaluate its efficacy and make suggestions for changes that can improve the system. Compared to the proprietary cheat-detection software kept hidden by these sites, there are far fewer peer-reviewed papers in this area discussing or experimenting with various cheat-detection measures in chess. Of the few available papers on this topic, many are lacking in detail or do not consider different aspects of the game which may affect accuracy, such as game length, opening moves and endgames, player rating and time-controls. Additionally, while some of these studies focus either on statistical methods or neural networks as a means of cheat detection, none investigate employing a combination of methods to increase classification accuracy. This research aims to address some of these shortcomings and provide greater insight into what aspects of cheat-detection may be improved upon.

1.4 Thesis Structure

This study is divided into a further four chapters. In chapter 2, we provide an overview of the most current research in cheat-detection and prevention in chess, identifying the most prevalent methods both in online play and over the board. We briefly examine the advancement of AI chess programs and how this has impacted both players and organisations, before highlighting the most relevant obstacles that remain in cheat-detection. In chapter 3, we discuss how the relevant data for the current study was collected and processed, as well as the methodology that was employed for the development of our own cheat-detection model. Specifically, we outline how the different game statistics or evaluations were calculated and the architecture of the convolutional neural network we use. In chapter 4, we examine the results obtained using the neural network and the statistics-based approaches, as well as a combination of these methods. In the final chapter, the implications of the current research are discussed and what potential practical applications the developed model may have. Here, we also examine the limitations of the model and how this may be improved upon in future research in the area.

Chapter 2: Literature Review

2.1 Impact of Artificial Intelligence on the Game of Chess

Chess has been held as an example of progress in Artificial Intelligence (AI) for decades. In 1950, Claude Shannon published one of the first papers describing a program that could theoretically be capable of playing chess (Shannon, 1950). The development of programs that could play games became a major research area in this relatively new field and building a chess program that could compete with the worlds' best players became one of the 'grand challenges' for AI research (Schaeffer, 2002). Since then, there have been huge advances in chess-playing software, or 'chess engines', from IBMs Deep Blue in 1997 to the recent development of more powerful programs such as AlphaZero[1] in 2017. Illiescu (2020), highlights some of the key differences between these earlier chess engines. These relied heavily on brute force analysis, searching millions of positions per second, whereas the more recent ones rely on the more efficient method of reinforcement learning. This more advanced method closely imitates the human learning process by trying to maximise set rewards in each environment. The rewards in this case, are those moves within a game of chess which lead to a victory. Crucially, instead of feeding the program data in the form of millions of games and positions, AlphaZero learnt to select moves and evaluate positions only using data gathered by playing against itself. Through this method, the AlphaZero engine can '*learn to optimize decisions in any scenario without changes or guidance'* and proved to be ground-breaking in the area of chess-playing software, easily beating the world's top human players (Sadler and Regan, 2019). The AlphaZero engine beat the strongest chess engine at the time, Stockfish, with this self-learning process which only took a few hours to train.

In 2019, an algorithm that far surpassed the capabilities of even AlphaZero was released. This new algorithm developed by DeepMind, named MuZero, learnt to play the game better than any previous engine without initially even being given the rules of the game. According to Schrittwieser et al. (2020), this algorithm has a maximum of 1 million chess games saved in the buffer, with 3000 games played in parallel. Using Google's cloud infrastructure, DeepMind

[1] https://www.deepmind.com/blog/alphazero-shedding-new-light-on-chess-shogi-and-go

could use thousands of tensor processing unit chips specifically designed for neural network calculations. Following this, even open-source engines such as Leela Chess Zero[2] began to reach the same level as MuZero (Illiescu, 2020) (Somers, 2018).

While it was thought that artificial intelligence would be used as a learning tool when it came to chess, Illiescu (2020) claims that there is now more of a demand to employ AI to identify violations of fair play and players who are performing higher than they should be given their game history. For example, the authors report that for the European Online Chess Championship in May 2020, about 2% of players ended up being disqualified for cheating using a chess engine. At most major chess tournaments, a metal detector is usually enough to prevent cheating in live games. However, cheat detection in online games has proven to be much more difficult because each player is playing from their own home. While there has been a major increase in the accuracy of cheat-detection software they remain unable to determine cheaters with 100% accuracy. Illiescu further highlights the need to improve the accuracy of cheat-detection methods to maintain the integrity of the game.

According to Yan and Randall (2009), online gaming in general makes up a significant proportion of recreational Internet usage and there is a large amount of money involved in game development, game hosting sites and sponsorship among certain top players. Barnes and Hernandez-Castro (2015), highlight the monetary consequences of cheating in chess, across both professional and amateur levels, pointing out that increases in the number of cheating cases leads to the game losing its integrity which will in turn dissuade players from taking part in certain tournaments.

2.2 Anti-Cheating Measures at Live Events

In a recent article by Zaksaitė (2022), the current state of anti-cheating measures being implemented at live chess tournaments was examined. Such measures are dealt with by the Fair

[2] https://www.newyorker.com/science/elements/how-the-artificial-intelligence-program-alphazero-mastered-its-games

Play Commission (FPC) of the International Chess Federation (FIDE). The FPC have defined cheating in chess tournaments and other events as '*the deliberate use of electronic devices or other sources of information or advice during a game'* (*Anti-Cheating Regulations,* 2018) Throughout the article, Zaksaitė identifies the preventative measures currently in place at tournaments, the key challenges to applying these measures and the trade-off between cheat-prevention and safeguarding the organisation of the event. The measures currently in place at most major events include metal detectors, space separation between players and spectators, applications to detect mobile phones and computers, a list of prohibited items and an authorisation policy.

The metal detectors used are reasonably efficient in that they do not require each player to be scanned individually. However, this efficiency is often lost as they can be extremely sensitive to even the smallest bits of metal, reacting to parts in a players' shoes, belts and even credit cards in wallets, among other items. Each of these signals requires a further check of that player by one of the Fair Play Officers which causes delay. In prestigious tournaments where players are required to follow a formal dress code, these delays are further exacerbated, as shoes, belts and other items need to be placed into separate boxes. This further impacts the scheduling of the event and has a negative impact on the game, particularly for blitz and rapid events where there are tighter time controls for each player. For these events, even missing the first minute of the game would be costly.

As well as metal detectors, linear and non-linear scanners are used to detect devices that players may be concealing. The linear scanners are used to detect materials made of metal and the non-linear scanner are used to detect connections from smart cheating, such as earpieces. Non-linear scanners are useful in identifying parts of certain materials that cannot be seen by the naked eye. However, it is important to note that such scanners require people with the knowledge and experience needed to read the data they provide. This is an additional cost and an inconvenience that any chess tournaments, especially smaller ones, might not be able to afford. There are also methods that can be used to avoid detection from such scanners, the most common being to wrap the phone in tinfoil to deflect any signals. Another is to use a special type of mobile phone which is small and made almost completely of plastic. It is therefore important to make use of both scanners to detect devices, which would incur an additional cost. Cheaper solutions may involve the use of specialised apps to detect phones or other devices. The most common of these is the 'Bluetooth Finder' which can locate Bluetooth devices using the received signal strength indication (RSSI). Any device within range will be displayed within the app along with the name of the device and its MAC ID.

Separation of players and spectators is another preventative measure in place at tournaments. The risks involved with player and spectator interaction can be viewers attempting to hand over smartphones or other devices to players, or even using pre-arranged signals to communicate moves. Because of this, any interaction is best avoided. Set areas are designated for both players and for spectators. The organisers and arbiters for the event look to prevent any information on games from leaving the playing area. Separate areas are also often provided for bathrooms, smoking areas and refreshment areas. Particular attention is paid to bathroom areas, as these provide several opportunities for cheating given the privacy they require. Event organisers even take the design of the toilet into account, preferring toilets that don't have a cistern and non-movable panels on ceilings where phones might easily be hidden. Of the disciplinary cases brought before the FIDE's Ethics and Disciplinary Commission, several of these were for mobile phone use in bathroom areas. Fair Play Officers are instructed to pay close attention to this area at all times precisely for this reason.

Authorisation policies are frequently implemented at major tournaments to allow certain journalists and sponsors into the playing area with electronic devices. According to the FIDE Laws of Chess Article 12.8, '*unless authorised by the arbiter, it is forbidden for anybody to use a mobile phone or any kind of communication device in the playing venue or any contiguous area designated by the arbiter'*. At major events, this involves issuing a Match Card along with the Player Accreditation to distinguish between players and non-players. Once a player finishes their games, the Match Card should be handed over to the Arbiter to differentiate the players who have finished their game from those whose games are still running and other non-players. Authorisation can only be granted to certain journalists and sponsors to carry electronic devices by the Chief Arbiter in consultation with the Fair Play team in a tournament. However, according to Zaksaitė (2022), the Fair Play team and arbiters at these events have great difficulty monitoring signs of fair-play violations when there are more than fifty accredited non-players inside the playing area.

While there are a number of these measures in place at most tournaments, they cannot fully guarantee that all cheating will be prevented, and they come at the cost of impacting the course of the event and distracting players from their game. The noise from metal detectors and scanners would be distracting to most players during a game, which is why all players would have to be scanned and the detectors covered before any play could begin. Alternatively, these scanners would have to be in a separate part of the venue where they are out of earshot. Random testing can also negatively impact the performance of players, especially if they call for the player to be tested while they are in the middle of a match. Generally, less intrusive measures are carried out before a game begins but players can be asked to submit to a frisk search during games if there is a reasonable suspicion that they are cheating. This can be extremely disbalancing and distracting for both players and would preferably be avoided. Given that it is nearly impossible to completely prevent players consulting chess engines, there should be a greater focus on cheat-detection software as opposed to implementing various inefficient preventative measures.

2.3 Cheating and Online Chess Games

There has been a dramatic rise in the popularity of online chess in the past few years, with the biggest contributing factor possibly being the COVID-19 pandemic. The number of people being forced to isolate at home led to greater interest in the hobby and a growth in the number of users of online chess platforms. The largest chess platform, Chess.com, had 1.5 million new subscribers in April 2020. This was more than double the 670,000 new users who subscribed in January 2020. However, with this rise in popularity came an equal rise in the number of cheating cases in online games. Chess.com stated in October 2022 that they have banned over 550,000 accounts for cheating and that they predicted this figure to reach 600,000 by the end of 2022 and over a million accounts by early 2024. The following figure shows the closure rates of account accused of cheating on the chess.com website from January 2021 to September 2022.

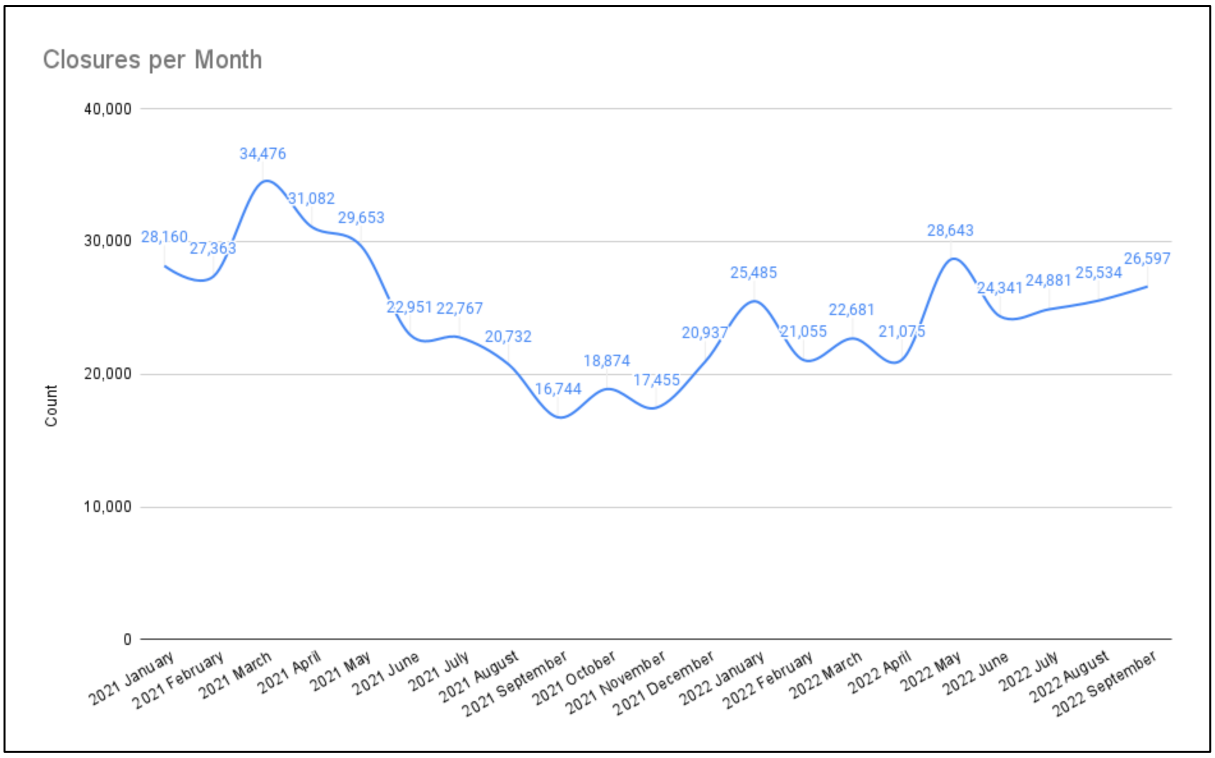


Figure 1: Account closure rates per month on Chess.com (Chess.com, 2022)

According to the article, the website invests hundreds of thousands each year to tackle cheating issues. This involves maintaining a fair-play team of analysts, researchers, grandmasters, and developers to improve and maintain their detection systems. They claim to take multiple behavioural factors and other data into consideration before closing the account of a potential cheater, but more specific details are confidential as the site believes this would make it easier for users to avoid detection (*About Online Chess Cheating, 2022*).

2.4 Current Cheat-Detection Software and Methods

Chess engines provide an easy means of cheating in online chess, even for complete beginners. All that is required is to input the opponents moves into a corresponding game running against an engine in a separate tab and then to take whatever move the engine plays in response to that. This in turn will create suspicion and distrust among players who will feel that they can never know for certain whether they are playing a cheater. Because of this, many of the top chess sites have developed their own cheat detection tools to identify these players. One of the few publicly available systems is that developed by Lichess.com, one of the most popular chess platforms.

There has been limited research done on the topic of engine use as a means of cheat detection in chess. One of the first significant studies on the issue was conducted by Regan et al. in 2009, in which they looked to calculate the rating of a player by inferring Bayesian methods to make a distribution of player ratings based on the moves that were played in games as opposed to basing it off the outcome of the game. This paper does not specifically deal with cheat-detection methods in chess, but it does highlight the fact that there is a strong correlation between the rating of a player and the strength of the moves they play, as would be expected. In a later 2011 paper, Regan developed upon this idea of an intrinsic player rating by comparing a human players' moves to an engine's evaluations on high depth. This method built upon the theory proposed by Haworth in 2007, in which he suggested to analyse the probability of a player choosing a move based on their own skill level. Based on this idea, Regan and Haworth published a paper in 2010 in which they suggested developing an imperfect chess engine using Bayesian methods.

These initial cheat-detection techniques proposed by Professor Kenneth Regan were based on predictive analysis. His research sought to determine whether a player's move selection is statistically consistent with the move section of similarly rated players in previous games when compared to the move selection of a chess engine. This was done by using the players' rating both before and after a tournament as well as their performance during the tournament. Given that chess engines are rated much higher than the best human chess player's, a player who surpasses the expected level of accuracy for their ranking would be more likely to have benefitted from some type of assistance. While this remains one of the leading approaches to cheat-detection today, it does need to be cautiously applied, especially when measuring the performance of very young players, as it is not unusual for players under a certain age to see rapid improvement in their games. There would also be difficulty in applying this method in online chess games, where new users are all given the same starting ELO, usually 1200 or 1500 depending on the website. If this new user was actually a very experienced player it would be reasonable to expect them to play very accurately against players far below their skill level and see a rapid rise in their ELO rating.

This approach of using the chess engine analysis to determine cheating was questioned by Barnes and Hernandez-Castro in 2015. They stated that using an engine in such a way could only prove the accuracy of the moves that were played and would be unable to prove whether there was any cheating in the games. Despite this, Regan and his colleagues were able to demonstrate that there was a correlation between the strength of the moves played by a person and their ‘ELO rating’. In chess games, the skill level of different players is measured using an ‘ELO system’, which was invented by Dr. Arpad Elo in 1978. Using this system, players can gain or lose points based off the results of the games they play. The number of points that are won or lost depends on the rating of their opponent, with a win against a stronger opponent yielding more points and a win against a weaker opponent yielding fewer points. Similarly, a loss to a weaker opponent will lose you more points than a loss to a stronger opponent.

In Barnes et al. (2015), they build upon the work of Regan and discuss the limitations of engine analysis for cheat detection in chess. By examining a large collection of chess games where they could be confident that no cheating took place, they sought to highlight the difficulties associated with identifying engine-use purely from analysing the moves within a given game. They found that there remains a high risk of classifying games as false positives for cheating and that it is not possible to base any assumptions of cheating using the moves of a single chess game. Furthermore, they maintain that it is not possible to provide definitive scores to measure the similarity between a certain game and a particular chess engine, given that these values will inevitably differ across different engine depths when multi-threading evaluation is employed.

Barnes and Hernandez-Castro (2015), employ a slightly different method of measuring similarity to chess-engines, especially in their approach to opening moves. For game data, they made use of the ChessBase cbmega database(ChessBase, 2014) for games played within the 19th and 20th century. For games played between 1997 and 2013 they made us of Mark Crowther’s TWIC archive. They used Stockfish v3.0 as their analysis engine. Each position in the games was passed to this engine to be evaluated to the chosen depth. Once the evaluations were retrieved, they were written to an XML file. The chess engine was set to run in MultiPV mode, meaning it would return the top *n* number of moves, in this case the five best moves in each position. If a move played in a game was not considered to be one of the top five moves by the engine, this was caught and an evaluation would be forced. The evaluation was expressed either in terms of an integer value for the move (expressed as a positive or negative number of centipawns) or as ‘mate in N’, where N is the number of moves required to force a checkmate and win the game. This approach differs to that employed by Regan who adopts an exhaustive search approach for each position and eliminates branches of analysis that have an evaluation of 300 centipawns or higher. A ‘centipawn’ is the unit of measure in chess used to measure advantage, with a single centipawn being equal to 1/100 of a pawn (Stoiljkovikj, Bratko, and Guid, 2015). In chess, pieces are given a value, with pawns being worth 1 point, bishops and knights worth 3 points, rooks worth 5 points and the queen worth 9 points. Because the king cannot be captured, it is not given a number value. There is no use for these values in the game but they are essential for chess engines to evaluate positions (Hoque, 2021). For his cheat-detection analysis, Regan uses a ‘move matching percentage’ (MM) as a metric. This is the percentage of moves made by a player that match those picked by an engine, with a 100% match meaning that that person is consulting an engine. However, an MM score of 90% or even 80% can still mean that the player is using an engine. To determine whether this is the case, Regan makes use of rating-related statistical analysis. He maintained that it is possible to pick out outliers in rating improvement that might be because of engine use by examining the players past performance and previous ratings.

Barnes and Hernandez-Castro (2015), make use of an almost identical metric, though with some slight differences. They use a figure between 0 and 1, which they refer to as the ‘coincidence value’ (CV), to represent the portion of non-book moves that were played that were also chosen by the engine. A ‘book-move’ in this case, refers to commonly known moves that are frequently played and memorised for the opening phase of a game of chess. A ‘non-book move’ on the other hand, refers to the unique moves that have rarely, or never, been seen in a previous game. The CV value is different to MM in that it is a percentage of played moves having the same evaluation as the engine, whereas MM is the proportion of moves that are the same as the moves chosen by the engine. The reasoning behind choosing such a metric is that judging one move to be superior to another when the evaluation score for both moves are identical is a seemingly random decision made by chess engines as the choice of move can often vary from one run of the engine to another.

Barnes and Hernandez-Castro stated that both the MM and CV metrics are too basic to be able to identify engine-moves chosen with detection avoidance in mind. This is because more capable cheaters will know not to choose the top-engines move, but an equally winning third or fourth ranked move. This will allow them to bypass a naïve detection metric based off rank while also being able to play strong moves. To counteract this, Regan developed a second metric known as the average error (AE). This refers to the average difference in the evaluation of the move played and the best move chosen by the engine. This figure is also expressed in centipawns. Should a potential cheater be missed by either a low MM or CV score, a very low AE score can be a sign of more sophisticated engine use. While the AE metric is also used by Barnes and Hernandez-Castro in their research, they highlight that the variance in evaluations for mating sequences cannot be shown without difficulty in the form they are given by an engine. This can occur in a number of cases, such as the engine finds a mating sequence but the move played does not lead to a forced mate, when the engine finds a mating sequence shorter than that which was played, or when the engine gives a non-mating move but a worse move played leads to a forced mate. As a result, it was decided that such cases would be left out so that the study could rely on a complete numerical analysis. The one exception to this was that alternative forced mates of the same length using different moves were scored as a difference of zero. However, the researchers mention that it should be possible to assign a centipawn equivalent score to each of the other cases.

To compare moves made by a player within a chess game to those chosen by an engine in the same position, a distinction must be made between the opening moves, also known as 'book-moves', and those moves chosen in unique positions found in the middle and end game phases of chess. Opening moves frequently involve the repetition of moves from games that have been played in the past, as they have been proven to be effective strategies. These sequences of moves can often be twenty moves in length or more and are often memorised by players, especially at higher level games. They are categorised under 3-letter 'ECO-codes.' Such moves are not considered when comparing player moves to engine moves given that they are so commonly played and provide little additional information.

Regan adopts a methodology whereby the first 8 moves of each game are considered book-moves. However, if analysing a particular game in more detail, the point at which a new move is played that strays from the opening theory will mark the point at which analysis of the game should begin. In comparison, Barnes and Hernandez-Castro adopt a more comprehensive approach to classifying the opening phase of the games they have gathered. Due to the wide number of games, they chose across different time periods, they found it essential to distinguish between early games where opening sequences were less refined and later games where they became more developed. They developed a historical database of board positions from all the games they used which added up to a total of around 87 million different positions. For each unique chess position in this database, the date of this game is added to the table as the first known game with that position. When adding a new game position to the database, should this game pre-date any previous entry in the table, then the new date replaces the old entry. The overall ECO code was then found for each game analysed as part of the study. By comparing the positions to the corresponding table in the database, they were able to see how many moves of the game could be considered book-moves at the time that it was played. Only those positions which were seen before the date the game was played on were used to calculate the length of the number of book-moves for that game. From there, this depth of book-moves was added to the game as a tag labelled ‘BookDepth’. This was in turn passed to the analyser the researcher developed to compare the players moves to the engine moves. The disadvantage of this method is that games with different ECO scores but which end up in the same position as seen in previous games will not be compared. This can arise from games which have a different sequence of moves but wind up with the same position at different points in the game. While a database such as this is likely not historically accurate nor a fully comprehensive collection of all chess games, the authors believe that employing a method such as this to be a more accurate approach than assuming the same random cut-off point for every game. They also stress that this should give a more accurate measure of the CV of similar games that were played during different periods.

According to Favian et al. (2021), currently, the most common approaches to cheat detection in chess are statistics-based approaches and neural-networks based approaches. In their own research, they use a neural network to analyse playing patterns of cheaters and non-cheaters. They made use of online game data gathered from Lichess’s public API. Users were then categorised into groups labelled ‘cheater’ and ‘non-cheater’ by using a public Python application called ‘Cheat-Net.’ This application provides a list of users on Lichess and whether they have cheated in games. Using 152 user accounts, they randomly selected 2500 games for both the cheater and non-cheater groups. However, using data provided by this application seems problematic, as the authors would be relying on the accuracy of the Lichess cheat-detection system to classify games. This would likely mean that some games which are wrongly labelled as ‘non-cheat games’ could end up in the non-cheat dataset and games wrongly labelled ‘cheat games’ could be put in the cheat dataset. The game data was further split into two parts, which the authors refer to as ‘analysed’ and ‘unanalysed.’ The unanalysed games were those that contained only the board positions from games, whereas the analysed games are the same games that contain evaluation scores from the Stockfish chess engine and the remaining clock time after each move. Doing so allowed them to compare the results of the neural networks they had trained using either analysed or unanalysed games to see which group is better able to detect cheating. The collected games were converted into a multidimensional array with the shape m × 80 × 14 × 8 × 8, where m was the number of games. The second dimension is the number moves in each game which the authors limited to a maximum of 80 moves. For games with less than 80 moves, the arrays were filled with 0 for the remaining moves. The third dimension represents the different piece types in chess, along with two more arrays to represent a combination of all white pieces and all black pieces. The 8 x 8 dimensions represent the chessboard with 64 squares, where the value 1 was used to indicate that there was a piece present in that specific square and 0 to indicate that no piece was present there. Four different neural network models were assessed in the experiment. Two of these were densely connected neural networks (DCNN) and two were convolutional neural networks (CNN). The 5,000 games from both the 'analysed' and unanalysed' groups were randomly shuffled and 80% of the games were used for training and 20% were used for testing. The authors state that they applied a SoftMax activation function on the output layer that has three neurons, with the first corresponding to a classification of non-cheat games, the second to games where the white player cheats, and the third to games where the black player cheats. However, they do not clearly state what proportion of the 2,500 games they have classified as cheating involved games where white cheated and games where black cheated. The CNN with analysed data was shown to have the highest validation score of 57.5% whereas the DCNN model which used the unanalysed data had the lowest score with 48.75%. Of the remaining two models, the CNN trained with the unanalysed data scored 53.75% and the DCNN trained with the analysed data scored 56.25%. Overall, the CNN models performed better than the DCNN models regardless of what data was inputted. It was noted that this could be due to the characteristics of the input, as the board positions gives the CNN models an advantage because they are better at analysing patterns of piece placement. Even when additional variables were added to the analysed input data the DCNN underperformed in comparison. However, the authors do not provide any evaluation metrics other than accuracy. Given that half of the games are labelled as cheat games and half as non-cheat games this gives only slightly higher accuracy than if we were to classify all games as cheat games. The results also showed that analysed data helped improve the score of all the models. Furthermore, the number of games used seems to be quite low for training a convolutional neural network. For example, in a study by Oshri and Khanwala (2016), in which they sought to predict moves in chess, they trained a convolutional neural network using 20,000 games with over 245,000 moves in total. Similarly, Panchal et al. (2021), while also focusing on chess move prediction as opposed to cheat detection, made use of over 1,500,000 board positions to train a convolutional neural network. However, perhaps the biggest concern with the data used by Favian et al. (2021), is that it contains 50% cheaters, which would not be representative of a real-world sample of games from one of these websites.

A similar study by Dalloul and Bechthold (n.d) also sought to identify the use of engines in chess games using neural networks. The games which were used as input data for the neural network were gathered from the Free Internet Chess Server (FICS) games database (*FICS Games Database,* n.d). A total of 8,141 games were initially split into three categories. The first category consisted of approximately 3,400 games between humans, the second category consisted of 1,100 games between humans and AI, and the third group consisted of 3,600 games between AI. All the human players in these games had ELO ratings between 1800 and 1999, which is a subset of very strong players but quite far below the level of an international master (above 2400 ELO) or grandmaster (above 2500 ELO). The collected games were filtered to remove games which had less than a total of 40 moves (20 moves each for both black and white players). A multidimensional array (8 x 8 x 6) was created for each board position reached in each game, with 8 x 8 representing the rows and columns of the chess board and 6 representing each of the different piece types. Unlike in the study by Favian et al. (2021), where 12 dimensions were used to represent the configuration of the different piece types and an additional 2 dimensions were used to shows the configuration of all of the black and white pieces, the authors used a '1' at a certain index position to represent a white piece was present, '-1' to represent a black piece, and '0' to show that the square was unoccupied. Each game could then be represented by an 8 × 8 × 6 × 40 matrix where there are 40 total moves in a game. The games were labelled as either 00, 01, 10, or 11, representing human vs human, human vs AI, AI vs human, and AI vs AI respectively. The first digit of these labels represents the nature of the player with the white pieces while the second digit represent the nature of the player with the black pieces. The dataset was split into a training set made up of 80% of the games from each category and the remaining 20% of each category was used for the validation set. After ensuring that the neural network was functioning correctly, the dataset was expanded to 8,000 games for each class in the training set, 999 games for each class in the validation set, and 1001 games for each class in the test set. The neural network model was started with a set of fully connected hidden layers. To implement this, the original matrices were permuted into matrices with dimensions 6 x 40 x 8 x 8. These matrices were then flattened, which created 15,360 features. A cross-entropy loss function with L2 regularization was applied. This gave the training data the dimensions m x 6 x 40 x 8 x 8, where m is the number of games in the training set. A ReLU activation function was used between hidden layers and a Softmax activation function was used in the final layer to classify the games. The Adam optimizer with the default parameters was used as the optimization function. After implementing a fully connected version of the network, a convolutional network was created. Two 3D filters of size 5 x 3 x 3 were used, with the first dimension corresponding to the number of moves being assessed at a time, and the 3 x 3 referring to the region of the board being assessed within the filter. The authors believe that a sequence of five moves would be most useful in classification while a 3 x 3 region of the board would capture many of the interactions between pieces on the board given that knights, pawns and kings are can only affect squares within these regions. A stride with dimensions 2 x 1 x 1 was implemented to prevent too much overlap between 5 move sequences while also still identifying the most relevant patterns in the groupings of moves. The first convolutional layer began with 6 filters with subsequent layers increasing to 64 and then to 128 filters to learn more complex features. The model achieved an overall accuracy of 79.3% on the test dataset which indicates a reasonably good performance given that the majority classifier accuracy would be expected to be approximately 25%. The classification accuracy for each of the aforementioned classes, was 77.9%, 77.8%, 79.6% and 82.5% respectively. These results would indicate that a convolutional neural network approach to cheat detection may be quite promising.

2.5 Conclusion

The current literature on the topic suggests that, while there are certainly many cheat-prevention measures in place at chess tournaments, the implementation of these measures is both costly and time-consuming and can negatively impact player performance and spectatorship. Despite this, there are still numerous incidents of cheating being reported at major tournaments each year. However, the cheat-detection methods discussed have shown promising results. If it were possible to implement a cheat-detection system that can reliably determine whether a player has cheated simply by analysing the moves played within a game, this would eliminate the need for any tedious cheat-prevention measures at live events and improve the experience of online chess as well.

Chapter 3: Methodology

This research was conducted with several steps as shown in Figure 2. below. The required game data was gathered from online chess games on the Lichess website. The data was then categorised into ‘cheat’ games and ‘non-cheat’ games and certain games were removed from the dataset. Steps were also taken to convert the games from the downloaded PGN format into game objects and to remove the opening sequence of moves from each game. All the collected games were analysed using the Stockfish 15 chess engine and these evaluations were used to calculate the Coincidence Value (CV) and Average Error (AE) of each game. To train the neural network, additional steps were taken to convert the board positions in each game to NumPy arrays so they could be used as input.

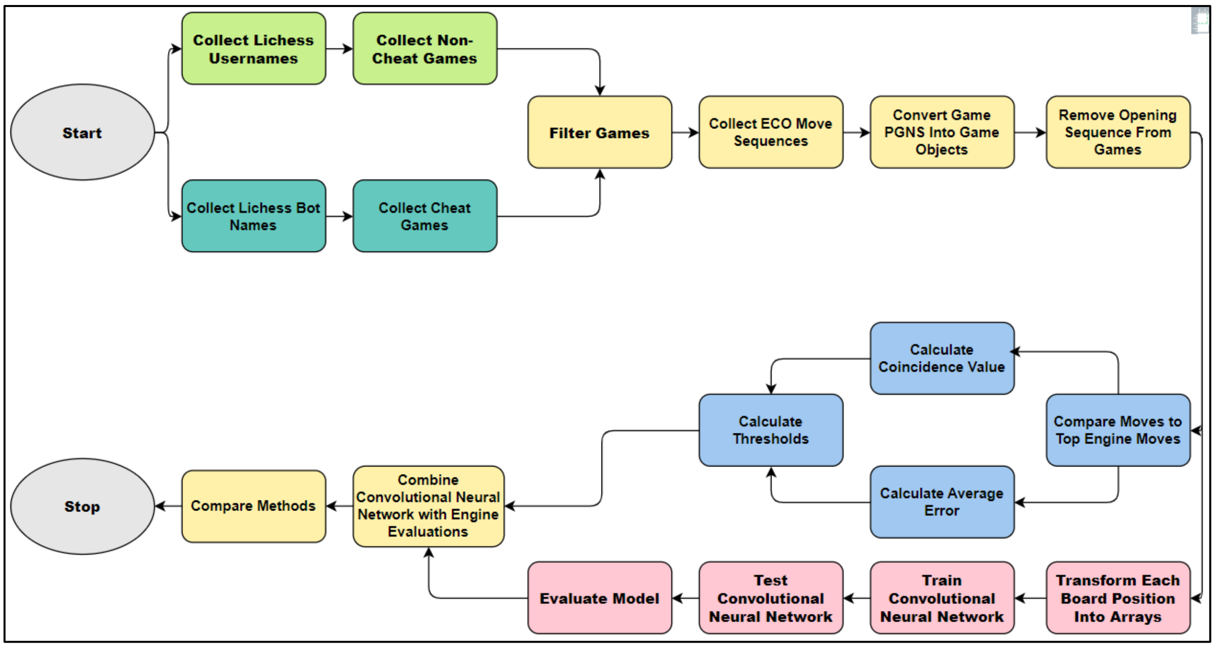

Start
Collect Lichess Usernames
Collect Non-Cheat Games
Collect Lichess Bot Names
Collect Cheat Games
Filter Games
Collect ECO Move Sequences
Convert Game PGNS Into Game Objects
Remove Opening Sequence From Games
Compare Moves to Top Engine Moves
Calculate Coincidence Value
Calculate Average Error
Calculate Thresholds
Transform Each Board Position Into Arrays
Train Convolutional Neural Network
Test Convolutional Neural Network
Evaluate Model
Combine Convolutional Neural Network with Engine Evaluations
Compare Methods
Stop

Figure 2: Overview of Methodology

3.1 Data Collection

All games were collected from the Lichess website which is a free and open-source internet chess server. The Lichess API provides a wide range of endpoints for downloading games and other data from users.

3.1.1. Collecting Lichess Usernames

For games which could be classified as ‘non-cheat’ games, regular games which were played between humans were gathered from the website. The Lichess website does not provide a list of usernames which one can iterate over to gather games from. Therefore, it was necessary to first gather all the games played by five users of varying strengths and then to collect the usernames of all the users that they have played against in each of these games.

3.1.2. Collecting Non-Cheat Games

After filtering this list of usernames to only include unique values, this could be iterated over to gather a single game from each corresponding user. Only standard variant chess games were gathered from each user with time controls ranging from ultra bullet (one minute per side) to classical (over 30 minutes per side). Other chess variants such as Chess960[3] and longer time controls were not gathered.

3.1.3. Collecting Lichess Bot Names

For data which could be classified as ‘cheat’ games, games which were played between users and various AI accounts on the platform were gathered. These games are used to simulate games where people make use of a chess engine to cheat. The Lichess website provides a page with a list of Bot accounts which one can gather games from.

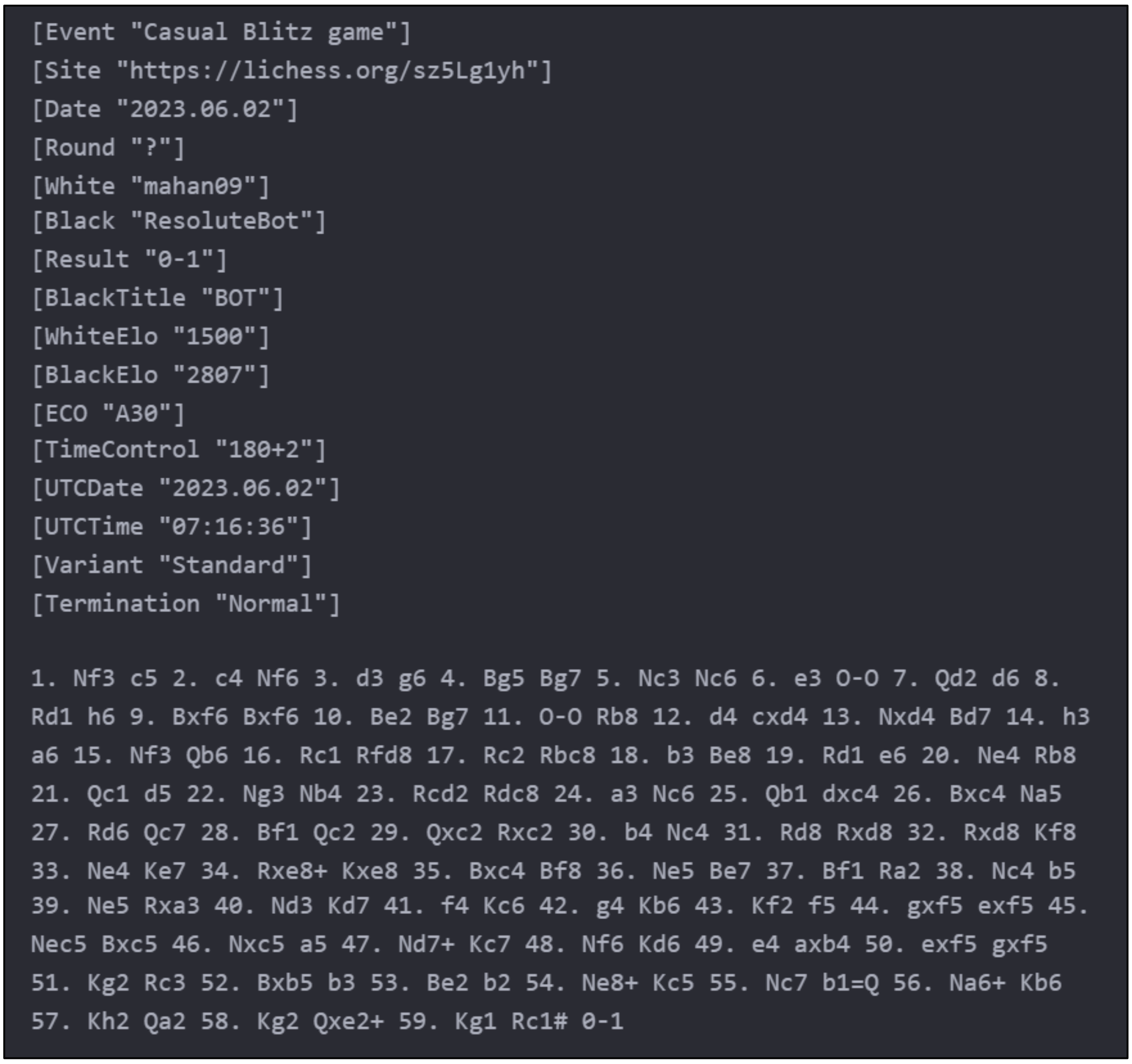
[3] A variant of chess in which the pieces are placed semi-randomly on the first and last rows of the board while the position of the pawns remains the same.

3.1.4. Collecting Cheat Games

A variety of different AI accounts with ELO ratings ranging from 2100 to 3200 were used to gather these games. As with the non-cheat games, only standard variant chess games with time controls between ultra bullet and classical were gathered.

3.1.5. Portable Game Notation

All the collected games are in a PGN (Portable Game Notation) format as shown in the example in Figure. 3. This is a standard format for recording chess games which can also be processed by computers. It gives a variety of different information on the game played such as the names of those playing, the time control of the game played, and most importantly, the sequence of moves that were played.



```
[Event "Casual Blitz game"]
[Site "https://lichess.org/sz5Lg1yh"]
[Date "2023.06.02"]
[Round "?"]
[White "mahan09"]
[Black "ResoluteBot"]
[Result "0-1"]
[BlackTitle "BOT"]
[WhiteElo "1500"]
[BlackElo "2807"]
[ECO "A30"]
[TimeControl "180+2"]
[UTCDate "2023.06.02"]
[UTCTime "07:16:36"]
[Variant "Standard"]
[Termination "Normal"]

1. Nf3 c5 2. c4 Nf6 3. d3 g6 4. Bg5 Bg7 5. Nc3 Nc6 6. e3 O-O 7. Qd2 d6 8.
Rd1 h6 9. Bxf6 Bxf6 10. Be2 Bg7 11. O-O Rb8 12. d4 cxd4 13. Nxd4 Bd7 14. h3
a6 15. Nf3 Qb6 16. Rc1 Rfd8 17. Rc2 Rbc8 18. b3 Be8 19. Rd1 e6 20. Ne4 Rb8
21. Qc1 d5 22. Ng3 Nb4 23. Rcd2 Rdc8 24. a3 Nc6 25. Qb1 dxc4 26. Bxc4 Na5
27. Rd6 Qc7 28. Bf1 Qc2 29. Qxc2 Rxc2 30. b4 Nc4 31. Rd8 Rxd8 32. Rxd8 Kf8
33. Ne4 Ke7 34. Rxe8+ Kxe8 35. Bxc4 Bf8 36. Ne5 Be7 37. Bf1 Ra2 38. Nc4 b5
39. Ne5 Rxa3 40. Nd3 Kd7 41. f4 Kc6 42. g4 Kb6 43. Kf2 f5 44. gxf5 exf5 45.
Nec5 Bxc5 46. Nxc5 a5 47. Nd7+ Kc7 48. Nf6 Kd6 49. e4 axb4 50. exf5 gxf5
51. Kg2 Rc3 52. Bxb5 b3 53. Be2 b2 54. Ne8+ Kc5 55. Nc7 b1=Q 56. Na6+ Kb6
57. Kh2 Qa2 58. Kg2 Qxe2+ 59. Kg1 Rc1# 0-1
```

Figure 3: An image of an example game in PGN format from the original uncleaned dataset.

3.2 Data Processing

3.2.1. Filtering Games

The data was cleaned using an open-source command-line program called pgn-extract, developed by David Barnes[4]. This program allows one to manipulate and format chess games which are stored in a PGN format. The collected games were filtered to only include games with more than 40 moves each for both the black and white players. This is because games

[4] https://www.cs.kent.ac.uk/people/staff/djb/pgn-extract/

with fewer moves become more difficult to classify as there is less variance between the positions in each game. As well as that, games which were played from a set position were also removed so only standard chess games played from the beginning were included. For the non-cheat group, any games where users played against an AI were removed. The games in the cheat group were divided into games where white was the AI and where black was the AI. There were also several games where an AI was playing against another AI which were also removed.

3.2.2. Collecting Opening Move Sequences

A list of the most common chess openings and their move sequences are organised using a classification system known as the Encyclopaedia of Chess Openings (ECO). This list is publicly available and can be downloaded from various chess websites (*Scid Code*, n.d). After doing this, every position in each of these opening sequences was converted to a FEN (Forsyth-Edward Notation) so that they could be compared to each game in our datasets (*Forsyth-Edwards Notation*, 2022). This is the standard notation for describing the position reached during a game of chess. For example, the starting position of a chess game as shown in Figure. 4. has the following FEN : rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1.

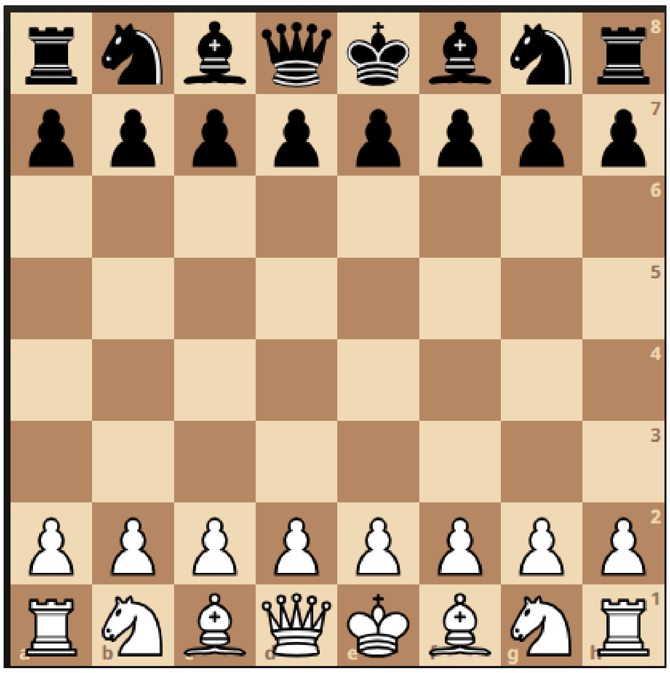


Figure 4: Picture of the starting position of a game of chess corresponding to the following FEN description: rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1.

3.2.3. *Converting PGNs to Game Objects*

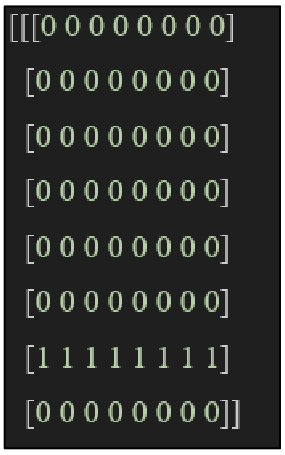
Once all the undesirable games were removed from each dataset, each game was converted from a PGN format into a game object using the Python chess library. This library allows one to iterate over each move in the game and to create board positions from the move sequences provided in the PGN file.

3.2.4. *Removing Opening Sequences from Games*

By iterating over each move in every game, the board position reached after each move could be compared to every board position from the ECO openings list which was converted to a FEN format. Once a board position was reached that was not recorded in the ECO openings list, the next 30 moves would be recorded for both black and white players. Essentially, this would leave the middle 30 moves for each game and remove the opening sequences and endgame portion of each game. The opening moves of a game of chess are frequently memorised by players and contain very few inaccuracies . Therefore, if someone were to play a long and commonly known opening sequence of moves, this would result in them having a more accurate game but would not be indicative of whether or not they were cheating. Similarly, the endgame often involves several forcing moves and endgame sequences that can be memorised like in the opening. Therefore, we would expect higher accuracy even from human players in these stages of the game. The middlegame, in comparison, generally has many more unique and complicated positions that can't be memorised.

3.2.5. Transforming Board Positions into Arrays

To further prepare the datasets for the neural network, each board position reached in a game following the opening was converted to a 60 x 14 x 8 x 8 NumPy array, where 60 is the number of moves that will be taken from each game. The second dimension will represent the different types of pieces for each player as well as a combination of all pieces for both sides. This means six board positions for each type of piece as well as one board position with every piece for each side. The 8x 8 dimensions represent each square of the chessboard with the value 1 being used to show that there is a piece occupying that square and a 0 being used to show that no piece is occupying that square. For example, Figure. 5. Shows an image of the starting position of a game looking only at the section representing the white pawns from the 14 x 8 x 8 matrix.



```
[[[0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0]
  [1 1 1 1 1 1 1 1]
  [0 0 0 0 0 0 0 0]]
```

Figure 5: White pawn segment of the 14 x 8 × 8 matrix corresponding to the starting position of a game of chess.

3.3 Methods

3.3.1. 3D Convolutional Neural Network

Labels were created for each category of game with 0 meaning no cheats, 1 meaning white cheats, and 2 meaning black cheats. There were 10,028 games used for 'non-cheat' data, 1275 games for 'white cheats' data, and 1311 games for 'black cheats' data. The data and labels were shuffled in the same order using random indices. A stratified five-fold cross-validation technique was applied to split the data into multiple subsets. This ensures that each fold contains a similar distribution of classes as the original dataset helps prevent any single fold from being biased towards a specific class or having an insufficient representation of certain classes. This is particularly important with imbalanced datasets such as the one we are using. Within each fold the data is further split into a training set (X_train and y_train) and a test set (X_test and y_test) using an approximately 80-20 split.

For the neural network the keras and tensorflow Python libraries were used to create a sequential model. The first layer was a 3D convolutional layer with 32 filters and a kernel size of 3 x 3 x 3. It also uses the ReLU activation function. This was followed by a 3D max pooling layer to downsample the feature maps obtained from the previous layer. This pooling layer has a pool size of 2 x 2 x 2 and it uses max pooling, which means that it will take the maximum value within that region. Another 3D convolutional layer is added after this with the same kernel size and activation function as the previous convolutional layer but with 64 filters. This is followed by another 3D max pooling layer with the same parameters as the previous pooling layer but using 'same' padding. This adds padding to the input so that the output has the same length as the original input. A flatten layer is then added to convert the multi-dimensional output from the previous layers into a 1D vector and to connect the convolutional layers to the fully connected layers. Next, a dense layer with 128 neurons and a ReLU activation function is added. A dropout layer is included with a dropout rate of 0.5 to help prevent overfitting. The final layer is a dense layer which has a SoftMax activation for multi-class classification. It has 3 neurons to represent the 3 different output classes. The model is compiled with a categorical cross-entropy loss function, the Adam optimizer, and the accuracy metric. It is then trained using a batch size of 32, 100 epochs, and a validation split of 0.2 (20% of the training data used for validation). Early stopping is also included so that the training is stopped once the model is no longer improving. The validation loss is monitored after each epoch and the training is stopped after five consecutive epochs where this does not improve. This further helps to prevent overfitting and saves time.

The model is trained and tested five times, with each iteration using a different fold as the test set and the remaining four folds as the training set. To evaluate the model with each iteration, we calculate the accuracy, precision, recall and F1 scores and create a confusion matrix. After the fifth iteration, the evaluation metrics from testing the model on each fold are averaged to obtain a better estimate of the model's performance.

3.3.2. Calculating the Coincidence Value

The coincidence value is the proportion of top moves, excluding the opening moves, that were played in a game. To calculate this, the Stockfish 15 chess engine was used to find the top three moves for each position. For all the evaluations, the Stockfish engine was run at a depth of 20 with a minimum thinking time of 20 and a hash value of 8192. If the move that was played in the game matched one of these top three moves, a 'True' value was added to a corresponding array for each player, or a 'False' value if it does not match. This was calculated for the 30 moves following the opening sequence for each player. The proportion of 'True' values for a game is the coincidence value and a higher value should indicate that the player likely used an engine. Once this value is calculated for each player in each game across each category of games, it should be possible to determine whether there is a threshold for the coincidence value which can clearly indicate cheating. This will be done using a logistic regression model to perform binary classification to first determine whether the white player cheated or not and then whether the black player cheated or not. The probability threshold for determining whether a player cheated will be set at 0.5, with probabilities below this meaning the player will not be classified as a cheater and a threshold above this meaning the player will be classified as a cheater. This will also be assessed using stratified five-fold cross-validation, with accuracy, precision, recall and F1 scores being measured at each fold.

3.3.3. Calculating the Average Error

The average error involves finding the difference in centipawns between the top move for a given position and the move that was played in the game. To calculate this, the Stockfish 15 chess engine was used to find the centipawn value of the move that was played in that position in the game and value of the top move. The same parameters were used as when calculating the coincidence values, with the Stockfish engine using a depth of 20, a minimum thinking time of 20, and a hash value of 8192. This centipawn difference between the top move and played move was calculated for the 30 moves for both black and white players following the opening sequence and added to a list for each of the players. Following this, the average error was calculated for both players by adding all the differences in each list and dividing by the number of moves within the list. The program used does not compare moves where the value is given as 'Mate in n', where n is the number of moves left to checkmate. That is because there is no specific centipawn value given to checkmate, as it costs the player the game. In such cases, no move is added to the list tracking the difference in centipawn values for each player. It should also be noted that Stockfish can give different evaluations of the same position in different runs of the program. This is because Stockfish is an AlphaBeta algorithm that includes pruning, reduction and extension techniques that are state dependent. This means the results of the search of a subtree depends on the state with which it entered the subtree. Starting with hash table entries that influence move ordering leads to a different result for a search of same depth (Bijl and Tiet, 2019). Because of this, when the centipawn value of the top move and the played move are calculated, if the centipawn value of the played move is then greater than the top move, we simply input the centipawn difference as 0. The reason for this is that the difference in centipawn evaluation between the two moves is low enough that one move may be considered better than the other depending on different runs of the program. The idea behind calculating this average error is that a lower score is meant to indicate that a player may have cheated. It is also meant to be more useful for identifying more intelligent forms of cheating, where players are consulting an engine but not choosing the top move each time to avoid detection. A player may still be choosing winning moves suggested by an engine but if it was only a top 4 or 5 ranked move it would not be detected by the coincidence value. As with the coincidence value, once the average error is calculated for each player in each game across all categories, it should be possible to determine whether there is a threshold value for the average error which can clearly indicate cheating. Again, this will be done using a logistic regression model to perform binary classification to first determine whether the white player cheated or not and then whether the black player cheated or not. The probability threshold for determining whether a player cheated will also be set at 0.5, with probabilities below this meaning the player will not be classified as a cheater and a threshold above this meaning the player will be classified as a cheater. This will also be assessed using stratified five-fold cross-validation, with accuracy, precision, recall and F1 scores being measured at each fold.

3.3.4. Combining 3D Convolutional Neural Network and Engine Evaluations

This final method involves combining the 3D Convolutional Neural Network with the coincidence values and average error scores retrieved using the Stockfish engine evaluation. The matrix data will be passed to convolutional layers followed by 3D max pooling layers as described previously, while the engine evaluations will be passed to fully connected Dense layers. These two pathways will then be merged and a SoftMax activation layer will be added to predict the final class. As before, the model will use the Adam optimizer as well as the categorical loss function. Again, a stratified five-fold cross-validation technique will be used to evaluate the precision, accuracy, recall and F1 scores at each fold.

Chapter 4: Results and Analysis

In this chapter, we present the result of the experiments as outlined in the previous chapter.

4.1 Results of the 3D Convolutional Neural Network

The following results were obtained using only the 60 x 14 x 8 x 8 NumPy arrays being passed to the 3D CNN as described in sections 3.2.5 and 3.3.1 of the previous chapter.

Using just the game positions passed to the 3D CNN resulted in an accuracy of 85.4%, a precision score of 83.9%, a recall score of 85.4%, and an F1 score of 83.9%. These were the average scores obtained across all five folds from the five-fold stratified cross-validation. All these evaluation metrics seem to be within a similar range, which would suggest that the model is providing a balanced performance.

The following confusion matrix was also calculated using the sum of the predictions across all five folds to show how each different type of game was classified:

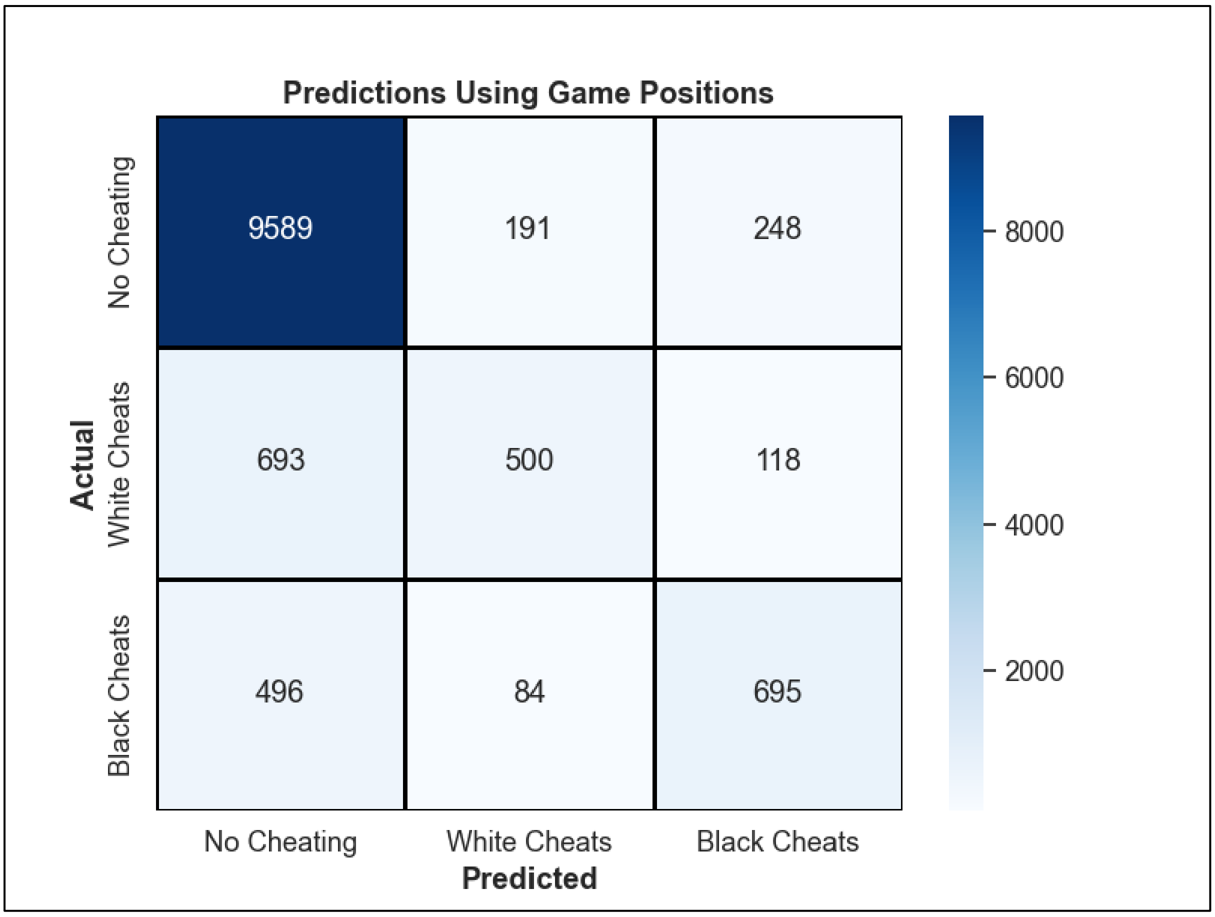


Figure 6: Confusion matrix showing the predictions made by the 3D CNN model using just the game positions.

The first row represents all the non-cheat games, the second row represents the games where white cheated, and the last row represents the games where black cheated. The first column represents the number of games from each row that were classified as non-cheat, the second column the number of games that were classified as white cheats, and the third column the number of games classified as black cheats. The diagonal elements of the matrix represent the true positive cases for each class. This shows that approximately 95.6% of the non-cheat games were correctly classified compared with just 39.2% of the games where white cheated and 53% of the games where black cheated. If we were to randomly assign games to each class, we would expect approximately 33.3% of all games to be correctly classified.

Based on accuracy score of the model, we can say that the total number of correct predictions is quite high. However, most of the correct predictions appear to be due to the large number of games being classified as non-cheat which is the class with the most data points. The percentage of correctly classified white cheats games is much lower, with most of them instead being classified as non-cheat. While most of the games where black cheated were correctly classified as such, this was closely followed by predictions for non-cheat games.

4.2 Results of Calculating the Coincidence Value

In this section, we examine the difference in coincidence values across the different groups of games. As discussed in section 3.3.2, the coincidence value is the proportion of moves within a game that match one of the top three moves chosen by an engine for each position. The following histogram shows the distribution of the coincidence values across games where there was no cheating and games where the white player cheated:

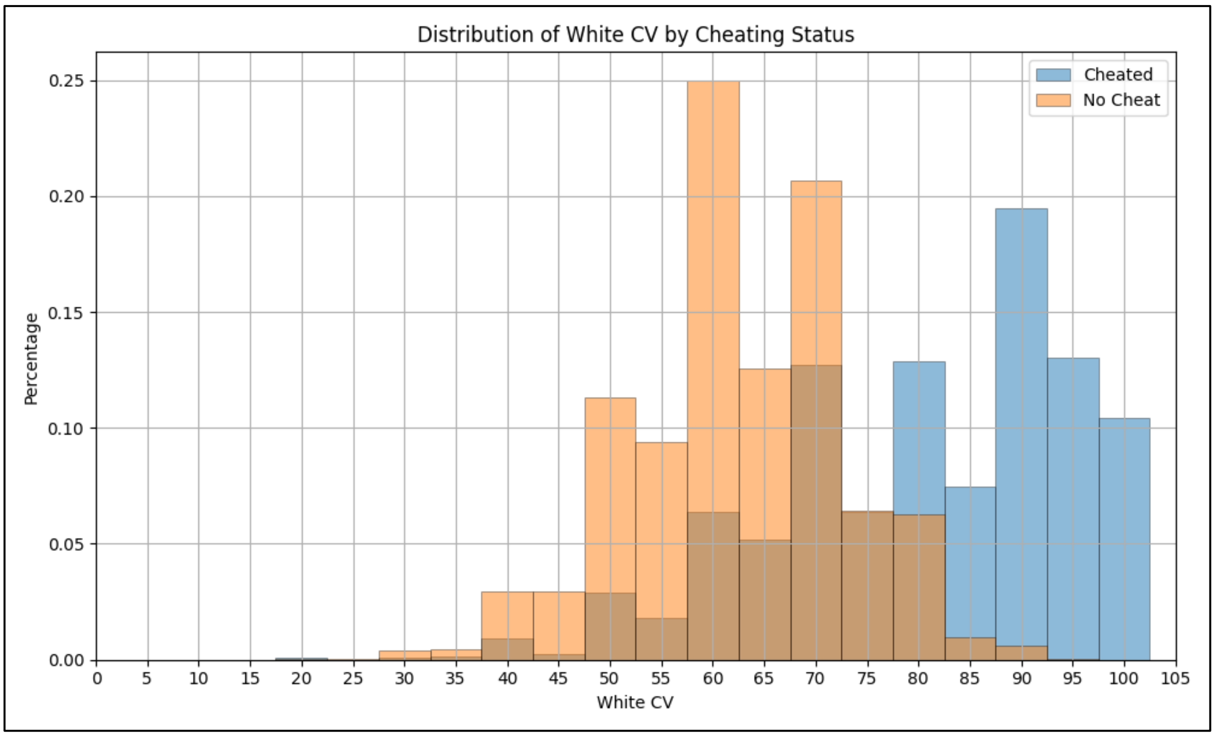


Figure 7: Graph showing the difference in distribution of CV scores across games where white cheated and no player cheated.

Each bar in the above graph displays the percentage of games in each group that scored within the range of the coincidence value shown on the x-axis. The blue bars are used to show the distribution of games that involved white cheating whereas the orange bars show the distribution of the none-cheat games. As can be seen from the graph, the highest proportion of cheat games have a coincidence value in the range of 85-90%. The highest proportion of non-cheat games have a coincidence value in the range of 55-60%. There are also no none-cheat games with a CV equal to 100%, whereas there are 133 games where white cheated that have a CV equal to 100%. The graph shows a higher distribution of CV scores for the games where white cheated than games where there were no cheaters.

A separate graph was used to show the difference in distribution of black CV scores between games where there were no cheaters and games where the black player cheated:

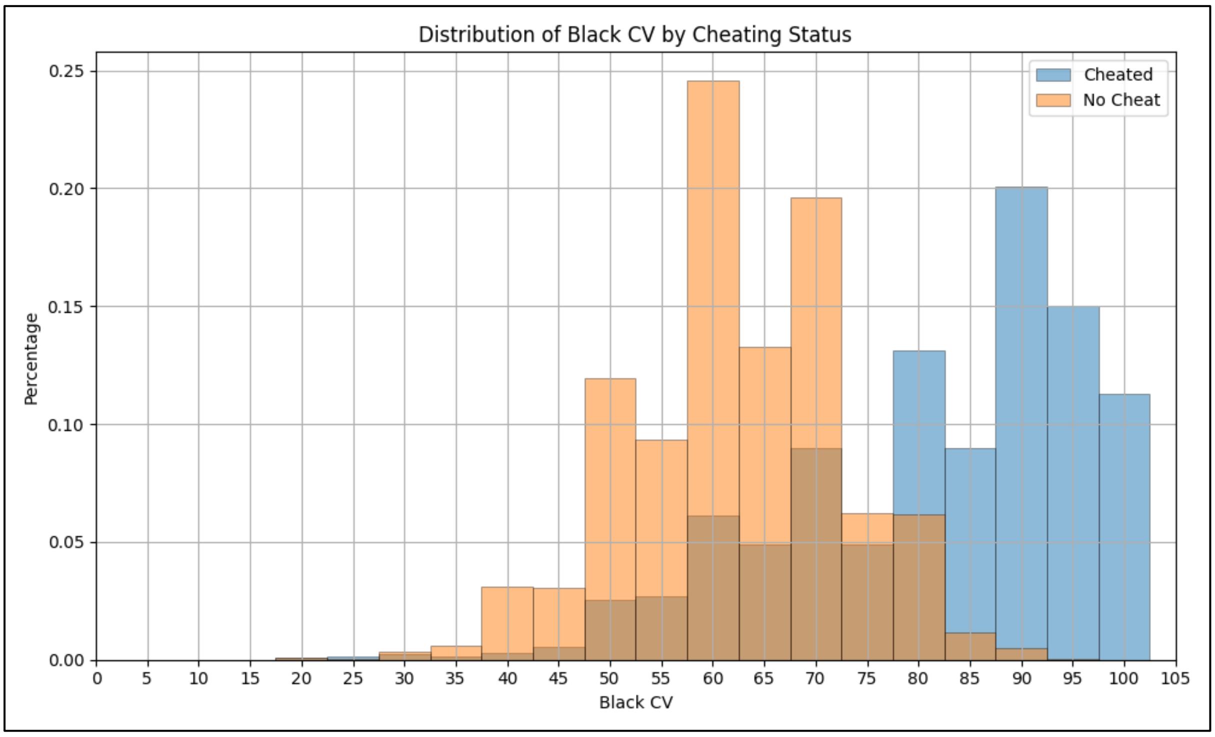


Figure 8: Graph showing the difference in distribution of CV scores across games where black cheated and no player cheated.

Like the games where white cheated, the highest proportion of the games where black cheated have the black player score a coincidence value in the range of 85-90%. There are also 148 games where black cheated that give the black player a CV equal to 100%. Based off both distributions, cheating is generally associated with a higher CV score than not cheating. However, there is no clear threshold score that can be seen from these distributions that would allow one to classify games as cheat or non-cheat.

A logistic regression model was used to analyse the CV of each player in each game to determine if there was a threshold score to correctly classify most players as cheating or not cheating. This was first done to analyse the CV of players who cheated using the white pieces and players who didn't cheat using the white pieces. A threshold of 87.3% was calculated using all the games in these categories, with a value below this meant to indicate that a player likely didn't cheat and a value above this meant to indicate that the player likely cheated. This is shown in the following graph of the model which shows the regression line and the distribution of the data points, with 1 used to indicate whether the player cheated:

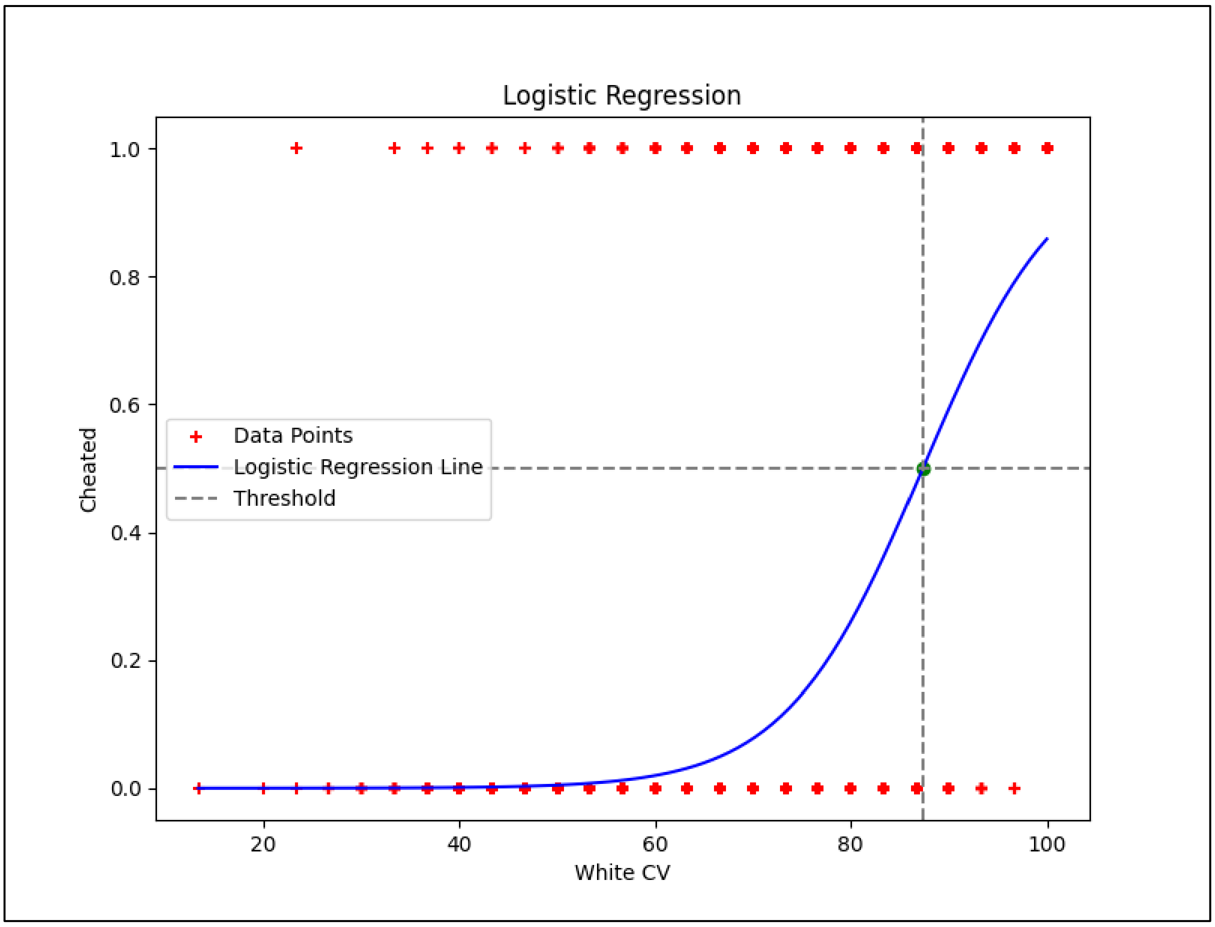


Figure 9: Graph showing the distribution of CV scores between games where white cheated and games with no cheater and the logistic regression curve used to determine a threshold for cheat-detection.

The model achieved an accuracy score of 93.2%, a recall score of 42.9%, a precision score of 89.5%, and an F1 score of 57.9%. These were the average scores calculated across all five folds from the five-fold stratified cross-validation. Based off these evaluation metrics we can see that relying on CV scores alone would not be a reliable method of classifying games. Given that approximately 88.7% of the games used to train this logistic regression model are non-cheat games, the accuracy score gotten would not be much of an improvement over the accuracy we would get if we were simply to classify each game as non-cheat. The low recall score indicates that this model performs poorly when identifying actual positive cases of cheating. The F1 score of 57.9% also suggests that this model only performs slightly better than random chance. If we were to apply this threshold to the games we used to train the model, with games scoring a CV below the threshold being classified as non-cheat and games scoring equal to or above the CV being classified as cheat, this would result in 728 (57%) of the games where white cheated being classified as non-cheat and 63 (4.9%) of the non-cheat game being classified as games where white cheated. This means that most games which involve white cheating would go uncaught while a significant number of games with no cheaters would be labelled as games where white cheated.

The same model was used to evaluate the CV of players who cheated using the black pieces and players who didn't cheat using the black pieces. A similar threshold of 87% was calculated using the games in these categories. We would expect the threshold for both black and white players to be roughly the same but slightly lower for black as we see here. This is because the black player is generally at a slight disadvantage and playing defensively for the majority of moves in most games. However, the similarity in the thresholds does suggest that this is an accurate representation of the distribution of CV scores. The following graph shows the regression line and distribution of the data points:

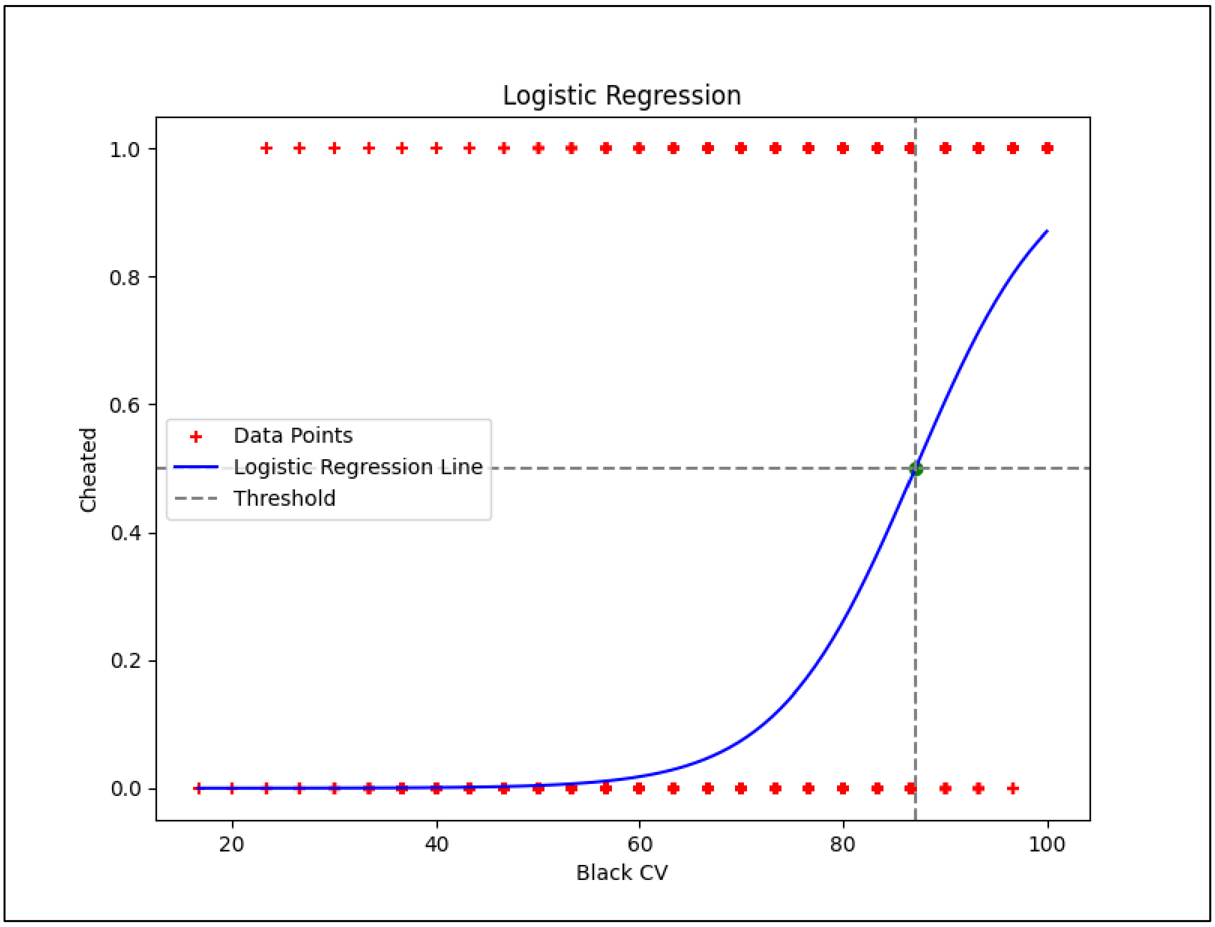


Figure 10: Graph showing the distribution of CV scores between games where black cheated and games with no cheater and the logistic regression curve used to determine a threshold for cheat-detection.

Using the CV for the black players, the model achieved an accuracy score of 93.3%, a recall score of 46.3%, a precision score of 91.7%, and an F1 score of 61.5%. These were the average scores calculated across all five folds from the five-fold stratified cross-validation. The accuracy is very consistent with that scored for the model using the games where white cheated. Again, because approximately 88.4% of the games used in this case were non-cheat games, this accuracy score is not much of an improvement over an accuracy score that would be obtained by labelling all games as non-cheat. The recall score for this model is slightly higher than for the previous model but would still indicate that it performs poorly when identifying actual positive cases of cheating. The F1 score suggests that the model is slightly better at determining whether black cheated or not based off CV scores compared to games where white cheated but that it is overall not very reliable. If we were to apply this threshold to the games we used to train the model, with games scoring a CV below the threshold being classified as non-cheat and games scoring equal to or above the CV being classified as cheat, this would result in 703 (53.6%) of the games where black cheated being classified as non-cheat and 54 (4.1%) of the non-cheat game being classified as games where black cheated. Again, this would mean that a high number of games where the black player cheats would go unnoticed while also falsely labelling a significant number of players with the black pieces as cheaters.

4.3 Results of Calculating the Average Error

In this section, we examine the difference in average error across the different groups of games. As discussed in section 3.3.3, the average error is the difference in centipawns between the top move for a given position and the move that was played in the game.

The following histogram shows the distribution of the AE values across games where there was no cheating and games where the white player cheated:

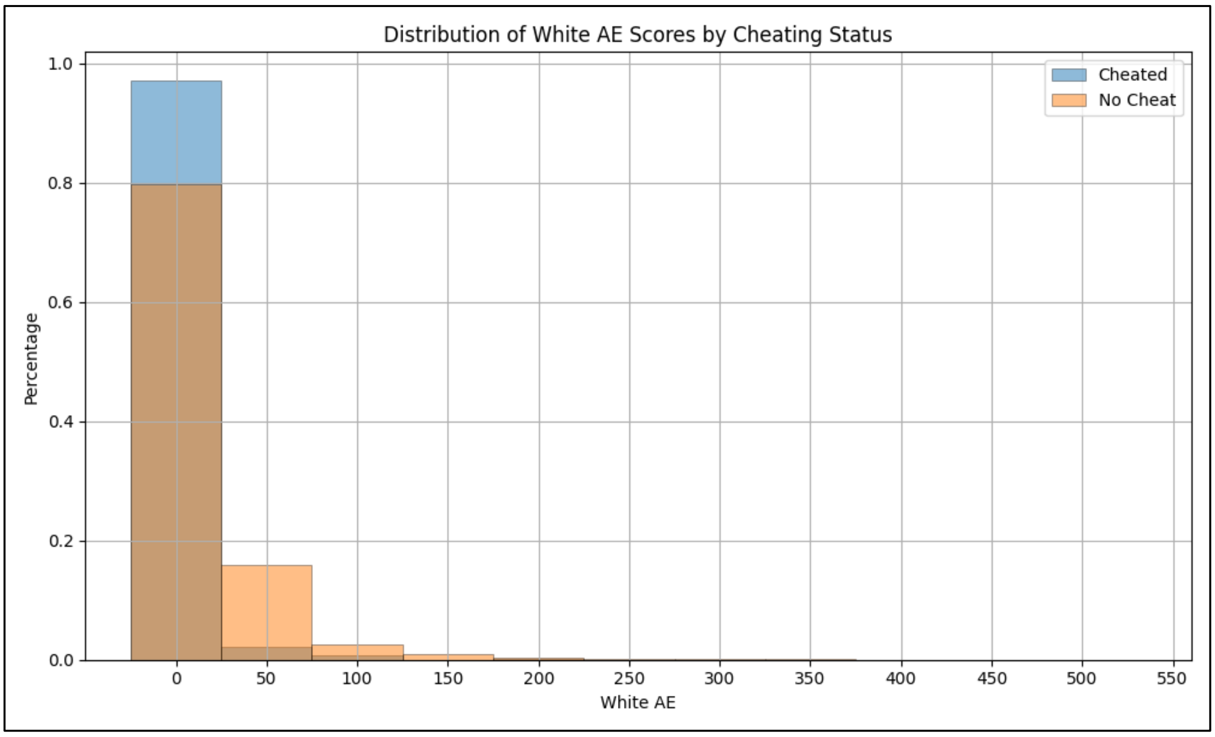


Figure 11: Graph showing the difference in distribution of AE scores across games where white cheated and no player cheated.

Most of the games for both groups had the white player score an AE between 0 and 50. Approximately 97% of the games where white cheated and 79.7% of games where white didn't cheat were within this range. There were ten games from the group of games where white cheated that scored an AE above 100, whereas there were 439 games from the non-cheat group that scored an AE over 100. The graph shows that there is very little variance in AE scores for both groups with 4493 (44.8%) of the non-cheat games and 405 (31.7%) of the white cheats games scoring an average error of 1 or less. While the graph does show that the higher scores for AE are mostly associated with cheating there is not enough of a spread in the data to determine an AE threshold for cheating.

A separate graph was used to show the difference in distribution of black AE scores between games where there were no cheaters and games where the black player cheated:

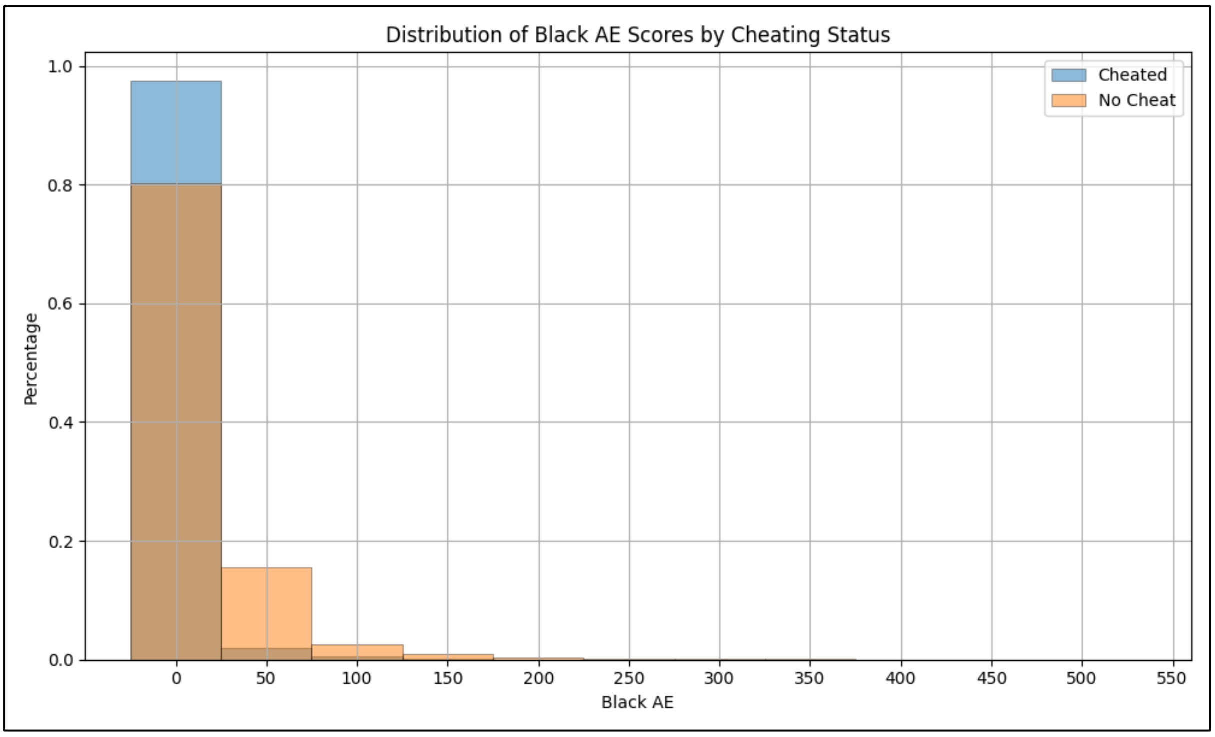


Figure 12: Graph showing the difference in distribution of AE scores across games where black cheated and no player cheated.

Like the games where white cheated, most of the games where black cheated had the black player score an AE between 0 and 50. Approximately 97.4% of the games where black cheated were within this range. There were seven games from the group of games where black cheated that scored an AE above 100. As with the games involving white cheating, this graph shows that there is very little variance in AE scores with 451 (34.4%) of the black cheats games scoring an average error of 1 or less. While the graph again shows that the higher scores for AE are mostly associated with cheating there is not enough of a spread in the data to determine an AE threshold for cheating.

As might be expected given the distribution of both the black and white AE scores, there were no conclusive results using the logistic regression model to predict cheating using the AE scores of players. For both black and white players, there were no cases where a game was predicted to have a player cheating. This resulted in an average accuracy score of 88.7% for classifying games using the white players AE and a score of 0 across all the other evaluation metrics; precision, recall, and F1 scores. Comparing the AE of the black players resulted in an average accuracy of 88.4% and a score of 0 across all the other evaluation metrics. In both cases, these accuracy scores are meaningless given that these simply represent the proportion of non-cheat games used for each model.

4.4 Results of Combining the 3D Convolutional Neural Network and Engine Evaluations

In this section we present the results of combining the 3D CNN with the AE and CV scores of each player. The model was tested using a combination of both scores and combinations with each individual score.

4.4.1 Results of Combining the 3D Convolutional Neural Network and Coincidence Values

Combining the 3D CNN with the CV resulted in an average accuracy of 90.1%, a precision score of 89.4%, a recall score of 90.1%, and an F1 score of 89.4%. These were the average score obtained across all five folds. The following confusion matrix was also calculated to show how each different type of game was classified:

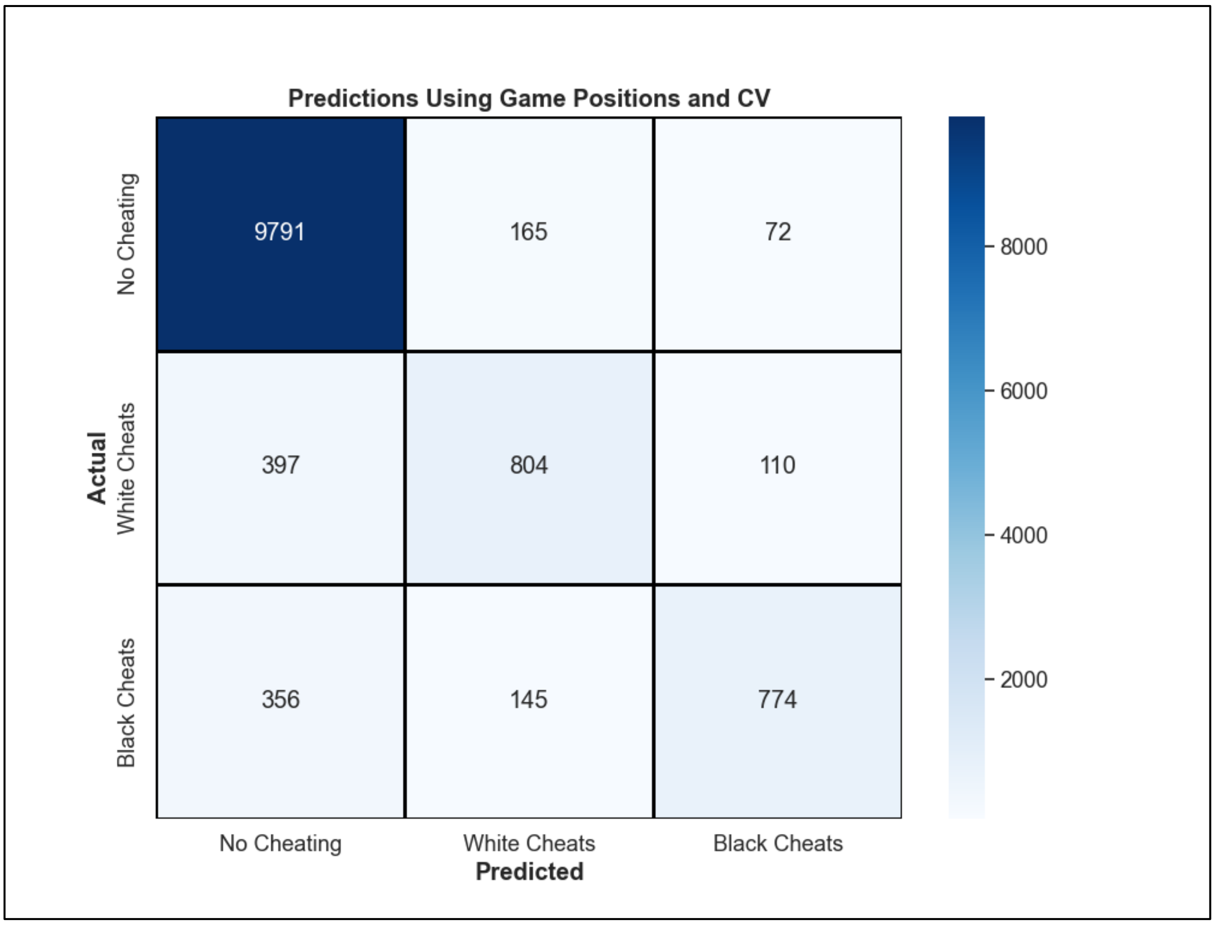


Figure 13: Confusion matrix showing the predictions made by the 3D CNN model using the game positions and CV scores.

These evaluation metrics were the highest of the various combinations. Compared to the previous CNN model which relied solely on data from the different board positions of each game, we can see a significant increase across all metrics. This is also reflected in the confusion matrix where we can see that a much higher proportion of the games where white and black players cheated were also labelled as such. Approximately 97.6% of non-cheat games, 63% of the games where white cheated and 59% of the games where black cheated were correctly classified. This shows a significant improvement in the ability to classify the games where white and black cheated, with little positive change in the ability to predict games where there was no cheating. It is important to note as well that for both the groups where black and white cheated, the next highest predicted class was for no cheating. This further suggests that there is some bias in the model towards labelling games as non-cheat. Overall, it would make sense that combining the CV scores would improve the model given the distribution of the scores across the different groups we saw in the previous section.

4.4.2 Results of Combining the 3D Convolutional Neural Network and Average Error

Combining the 3D CNN with the AE scores resulted in an average accuracy of 87.2%, a precision score of 86.2%, a recall score of 87.2%, and an F1 score of 85.8%. These were the average scores obtained across all five folds. The following confusion matrix was also calculated to show how each different type of game was classified:

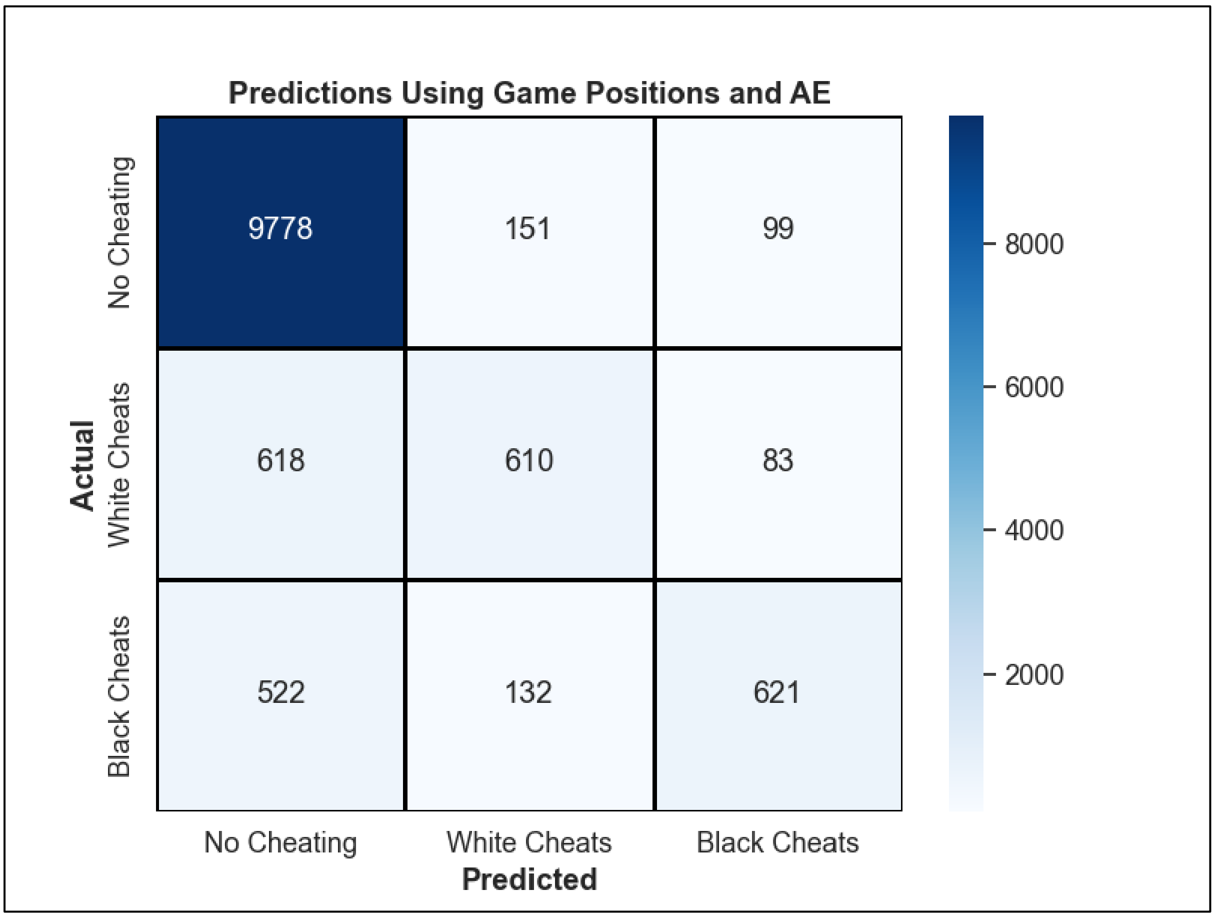


Figure 14: Confusion matrix showing the predictions made by the 3D CNN model using the game positions and AE scores.

These evaluation metrics show a slight improvement over those scored just using data from the different board positions of each game but not as much compared to when the model is combined with CV scores. This is consistent with the results we saw in section 3.3 where it was shown that the AE score is unreliable for predicting cheating. However, it was shown that for the small proportion of games that did have a higher AE score, these were more likely to be non-cheat games. This might explain the small improvement in the evaluation metrics we see here over the model which just relies on board positions. Approximately 97.5% of non-cheat games, 46.5% of the games where white cheated and 48.7% of the games where black cheated were correctly classified. This still shows an improvement in the ability to classify the games where white cheated compared to the model which just used board positions. Overall, it performs worse than the previous model which makes use of both the board positions and the CV. The model predicts most games involving white cheating to be non-cheat games. This is also the second most likely class to be predicted for games that involve black cheating, again highlighting that there may be a bias towards predicting the most common class.

4.4.3 Results of Combining the 3D Convolutional Neural Network, Coincidence Value and Average Error

Combining the 3D CNN with both the AE scores and the CV resulted in an average accuracy of 89.6%, a precision score of 89%, a recall score of 89.6%, and an F1 score of 88.9%. These were the average score obtained across all five folds. The following confusion matrix was also calculated to show how each different type of game was classified:

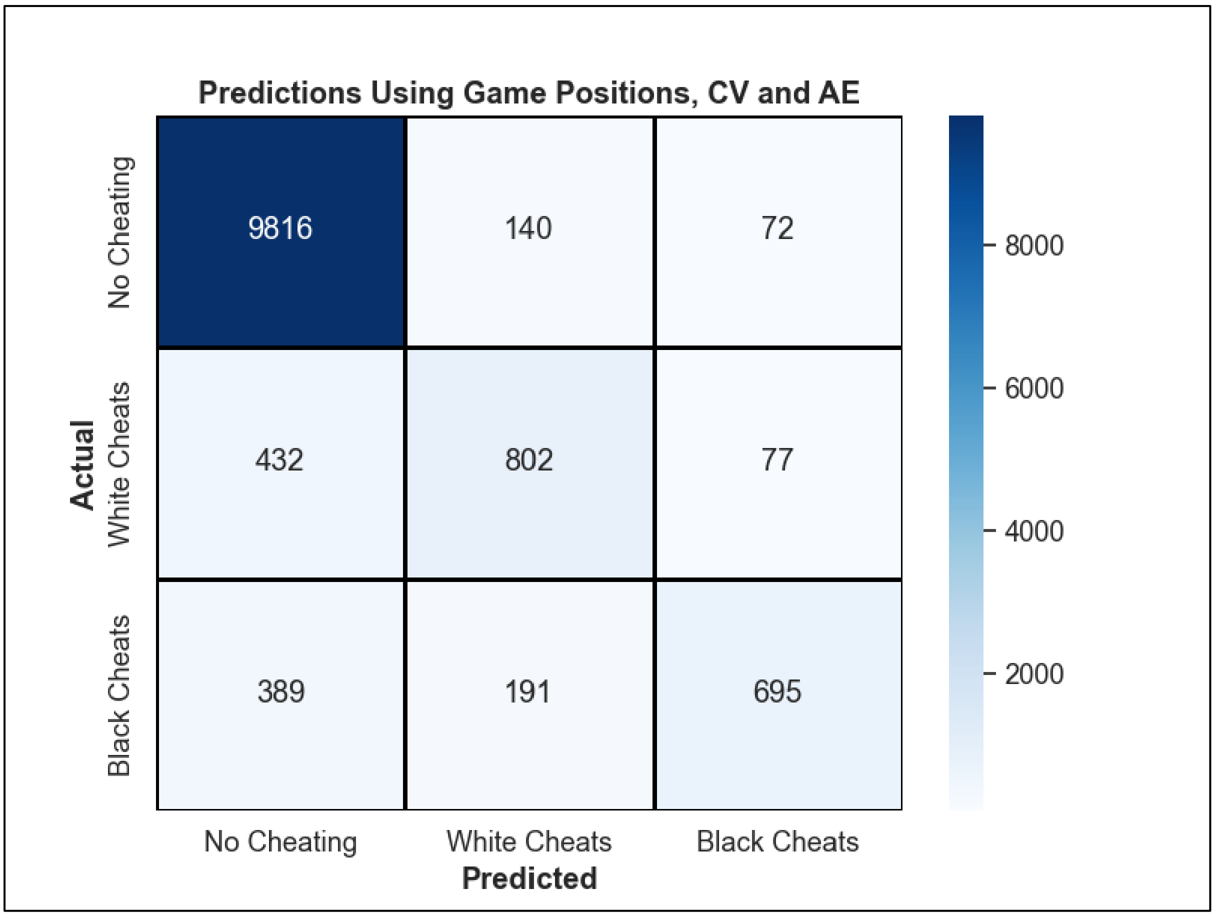


Figure 15: Confusion matrix showing the predictions made by the 3D CNN model using the game positions, CV scores and AE scores.

Approximately 97.8% of non-cheat games, 62.9% of the games where white cheated and 53% of the games where black cheated were correctly classified. This shows an improvement over the model which just relied on board positions. The model performs nearly as well as the model that uses board positions and CV with only a slightly worse ability to identify games where black cheated. Again, for both classes that involved cheating, the next highest prediction was for non-cheat games.

4.5 Conclusion

In this chapter, we presented an analysis of the results obtained from the various experiments conducted to detect chess engines. First, we examined the outcome of using a 3D CNN to classify games based only on the positions that arose within the game. For this method, the accuracy, precision, recall and F1 scores indicated that the model gave a balanced performance. However, there were more difficulties classifying games that involve cheating than games that involved no cheating. By analysing the distribution of CV scores, we could see that, while there was a noticeable correlation between higher CV scores and cheating, establishing a reliable threshold for cheat-detection was not possible. Similarly, the AE scores had a slight, albeit much weaker, correlation with cheating. This metric proved to be the worst indication of cheating, being unable to predict any of the cheating cases when tested. The combination of the 3D CNN and engine evaluations was shown to have an improved prediction performance, with a combination of the CV scores shown to be the most accurate. However, it was highlighted that there was a certain bias towards predicting the most common class (non-cheat games) across each of the different combinations. Overall, the results emphasise the complexity of engine detection in chess games. While a combination of methods showed a modest performance, a reliably high accuracy in classifying instances of cheating was not attained.

Chapter 5: Discussion, Conclusions and Future Work

5.1 Discussion

The results outlined in the previous chapter indicate that, while not fully reliable, a combination of engine analysis and convolutional neural networks can be useful in identifying cheaters in chess. However, relying solely on engine evaluations from individual games was shown to be ineffective.

While higher CV scores were generally correlated with cheating, relying only on this score was not useful for determining which games involved cheating. This result was somewhat expected and is consistent with the findings of Barnes and Hernandez-Castro in 2015, in which they discussed the limitations of engine analysis for cheat prediction. While cheating would result in a higher proportion of the top moves being played, highly proficient players, especially grandmasters, can also be expected to play a significant proportion of top engine moves throughout a game. This was reflected in the results, where a significant number of the non-cheat games had both black and white players with CV scores above 90%. It is also interesting to note that none of the no-cheat games had a perfect CV score of 100% for either the black or white players. This demonstrated that there is some merit in using engine evaluations to catch the most rudimentary types of cheating, in which the player relies fully on an engine and makes no attempts to disguise this by including non-engine moves or lower-ranking engine moves.

Contrary to the hypothesised association between low AE scores and cheating, it was found that there was very little variance in this score regardless of whether a player was classified as cheating or not. When examined closely, it was found that a large proportion of the players in each game scored an AE of 1 or less, regardless of whether they belonged to the cheating group or not. This is likely due to how the AE was calculated, with moves evaluated as ‘mate in N’ being omitted from the calculation and moves with little difference between the played move and top engine move being evaluated as zero. While more of the non-cheat games tended to have higher AE scores, there was no way to determine a significant cheating threshold using this score.

When used individually, neither of these engine evaluations could be used to classify games consistently and accurately. Similarly, relying purely on neural networks to determine differences in gameplay patterns to detect cheating, while more accurate, could not be fully relied upon. It was noted that there appeared to be a bias towards predicting the most common class in the dataset, which in this case was the non-cheat group. This could be remedied by including a higher proportion of cheat games in the dataset, but these games proved to be quite difficult to source. While there are several recorded games online of people playing against chess engines, there are far fewer games in which an average chess player can reach 40 moves against a very strong engine without first being checkmated. This was the length of game required for inclusion in our current dataset and could suggest that there is an over-representation of very strong players, or even cheaters, in the games which should involve an AI chess bot playing a regular human player. There may also be a tendency for the model in the current study to simply classify games in which white or black wins as cheat games. This is because only games played by engines in which the engine won were included in the dataset. It was also considerably more difficult to classify the cheat games as either white cheats or black cheats. This might suggest that there is a more significant difference between the patterns of games that involve some type of cheating than those that don't, whereas it is harder to discern which side is cheating from move patterns alone.

The most accurate results were obtained when using a combination of the engine evaluations and the 3D CNN, with the combination of the CV scores with the neural network being the most accurate. This is to be expected, given that these were the two most reliable individual methods for predicting cheating. There was a slight improvement seen when combining AE scores with the neural network, despite its limited value in cheat prediction when used by itself.

5.2 Future Work

Future research in this area would benefit from analysing a larger dataset, especially games for the cheat groups, as these were harder to source and underrepresented in the current study. As suggested by Dalloul and Bechthold (n.d), including a fourth class of games in which two chess engines play each other could also be beneficial, as it would limit any bias in classifying the winning side as a cheater.

Further improvements could also be made by splitting players into different groups based on ELO rating. Doing so, may make it possible to determine if there is a different CV or AE threshold that can be used to determine cheating based on the player's rating. This would mean weaker players would be expected to have a lower CV threshold and stronger players expected to have a higher CV threshold. The opposite would be true for the AE thresholds. Differences in player rating might also be considered, as stronger players would generally find it much easier to play accurately against weaker players as they tend to make more mistakes.

The engine evaluations for the games in this study were obtained using only the Stockfish 15 chess engine at a set depth. It should be noted that different chess engines may give different evaluations for the same positions depending on what depth they are run at. Future research in this area may consider using a variety of different engines and experimenting with different depths to increase the cheat-detection accuracy of models.

Finally, it should be highlighted that the current research only analysed the middle 60 moves of each game in the dataset. Chess games can be much longer, with certain games consisting of more than 100 moves. Future research may consider using endgame tables to simply cut moves from a game once they have reached a solved endgame position. These are databases that contain precalculated and exhaustive analysis of common endgame positions that are known to lead to a win or a draw for a particular side given perfect play. While it was not made possible to compare games of varying length in the current study, it may also be beneficial to do so in future research on the topic.

5.3 Conclusion

At the outset, we sought to investigate whether the use of chess engines could be detected based only on the moves that were played in any given game. Doing so would greatly simplify the ability of chess websites and live tournament organisers to maintain fair play policies and the overall integrity of the game. The current research has shown that using a combination of engine analysis and neural networks is a promising means of engine-detection. The results indicate that there is a particular pattern to how chess engines play the game, and this can be recognised by convolutional neural networks. However, it remains impossible to ascertain which games involve cheating based purely off moves played within a single match. As it stands, these methods can only suggest the likelihood of an engine being involved in a game of chess. Further research in this area is required before any such methods could be used as conclusive proof of cheating.

References

About Online Chess Cheating (2022) chess.com, available: https://www.chess.com/article/view/online-chess-cheating [accessed 04 Mar 2023]

Anti-Cheating Regulations (2018), fide.com, available: https://fpc.fide.com/wp-content/uploads/2020/05/Anti-Cheating-Regulations-2018.pdf [accessed 01 Mar 2023]

Bijl, P. and Tiet, A.P. (2021) Exploring modern chess engine architectures. *Victoria University, Melbourne*.

Barnes, D.J. and Hernandez-Castro, J. (2015) 'On the limits of engine analysis for cheating detection in chess'. *Computers & Security*, *48*, pp.58-73.

Elo, A.E., 1978. *The rating of chessplayers, past and present*. Arco Pub.

Dalloul, J. and Bechthold, M. (n.d), 'Chess Agent Prediction using Neural Networks'. available: https://cs230.stanford.edu/projects_spring_2021/reports/66.pdf

Di Fatta, G., Haworth, G.M. and Regan, K.W. (2009) 'Skill rating by Bayesian inference', *2009 IEEE Symposium on Computational Intelligence and Data Mining* (pp. 89-94). IEEE.

FICS Games Database (n.d) ficsgames.org, available: https://www.ficsgames.org/ [accessed 08 Sep 2023]

Crowther, M. (n.d) *The week in chess, Front Page for the week in chess*. available: https://theweekinchess.com/twic [accessed 12 Jun 2023].

ChessBase (2014) *Simply the best: Mega database 2015, Chess News*. available: https://en.chessbase.com/post/simply-the-best-mega-database-2015 [accessed 12 Jun 2023]

realnihal (2023) *Chess-AI-with-TensorFlow,* available: https://github.com/realnihal/Chess-AI-with-TensorFlow/blob/main/Chess_AI.ipynb [accessed 05 Sep 2023]

Forsyth-Edwards Notation, (2022) gambiter.com, available: https://gambiter.com/chess/computer/fen.html, [accessed 21 Aug 2023]

Haworth, G. (2007) 'Gentlemen, stop your engines!', *ICGA Journal*, *30*(3), pp.150-156.

Haworth, G., Regan, K. and Di Fatta, G. (2010) 'Performance and prediction: Bayesian modelling of fallible choice in chess'. *Advances in Computer Games: 12th International Conference, ACG 2009, Pamplona Spain, May 11-13, 2009. Revised Papers 12* (pp. 99-110). Springer Berlin Heidelberg.

Hoque, M. (2021) *Classification of Chess Games: An Exploration of Classifiers for Anomaly Detection in Chess*. Minnesota State University, Mankato.

Iliescu, D.M.D. (2020) 'The impact of artificial intelligence on the chess world'. *JMIR serious games*, *8*(4), p.e24049.

Oshri, B. and Khandwala, N. (2016) 'Predicting moves in chess using convolutional neural networks.'

Panchal, H., Mishra, S. and Shrivastava, V. (2021) 'Chess Moves Prediction using Deep Learning Neural Networks'. *2021 International Conference on Advances in Computing and Communications (ICACC)* (pp. 1-6). IEEE.

Patria, R., Favian, S., Caturdewa, A. and Suhartono, D. (2021) 'Cheat detection on online chess games using convolutional and dense neural network'. *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)* (pp. 389-395). IEEE.

Regan, K. and Haworth, G. (2011) ‘Intrinsic chess ratings’, *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 25, No. 1, pp. 834-839).

Sadler, M. and Regan, N. (2019) Game changer. *AlphaZero’s Groundbreaking Chess Strategies and the Promise of AI. Alkmaar. The Netherlands. New in Chess*.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T. and Lillicrap, T. (2020) Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, *588*(7839), pp.604-609.

Scid Code (n.d) sourceforge.net, available: https://sourceforge.net/p/scid/code/ci/master/tree/scid.eco [accessed 03 Jul 2023]

Shannon, C.E. (1950) XXII. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *41*(314), pp.256-275.

Somers, J. (2018) ‘How the Artificial-Intelligence Program AlphaZero Mastered Its Games’. available: https://www.newyorker.com/science/elements/how-the-artificial-intelligence-program-alphazero-mastered-its-games [accessed 12 Jun 2023]

Stoiljkovikj, S., Bratko, I., Guid, M. and UNI, F. (2015) ‘A computational model for estimating the difficulty of chess problems’. In *Proceedings of the third annual conference on advances in cognitive systems ACS* (p. 7).

Why can Stockfish give different evaluations at the same depth? (2019) chess.stackexchange.com, available: https://chess.stackexchange.com/questions/25278/why-can-stockfish-give-different-evaluations-at-the-same-depth [accessed 30th June 2023]

Zaksaitė, S. (2022) Anti-cheating protection measures in chess: current state of play. *Crime Prevention and Community Safety*, *24*(3), pp.255-265.

Appendix

The code of this project is available at

https://github.com/KaiMcGlacken/KaiMcGlacken_Thesis.git.