Université Paris Cité

Chess as Sequential Data in a Chess Match Outcome Prediction Using Deep Learning with Dynamic representation

Guillaume Dufau, Samy Méguéni, Logan Servant, Aïssatou Signate
Graduate School Computer Science and Mathematics
Université de Paris Cité
Paris, France

Abstract

The arrival of Big Data allowed the collection of high volumes of a variety of data in which valuable information and knowledge can be found. In the current era of Artificial Intelligence, data mining, as well as machine learning techniques, are developed to support advanced knowledge discovery from this huge quantity of data.
An important source of big data can be game data. In the case of chess, it is one of the most played games in the world, gathering millions of people, and creating numerous interactions, therefore one of the most studied games. Thus, a huge amount of data is produced and represented in various formats, making their exploration and analysis faster, easier, and more efficient.
In 2021, Rafal Drezewski and Grzegorz Wator published a paper describing how they developed a model that determines the outcome of a game, through two statics data representations (algebraic and bitmap): they obtained an accuracy of 69%.
In this paper, we present an alternative model to determine the outcome of a game with a dynamic representation. We had to experiment on the Bitmap and Algebraic representations first, to have reference values specific to our dataset and comparable to the results of the dynamic representation.
We obtained an accuracy for the bitmap of 53% for the Content Model and 58% for the Context Match Fusion. The algebraic representation gave us accuracies of 51% and 55% for the respective models. Eventually, our dynamic representation showed better results than algebraic results but remains inferior to the bitmap representation: 52% and 56% for the respective models.

1. Introduction

How can a simple board game cross the centuries and remain one of the most popular games? The intellectual effort it requires. The sensation it provides. The skills it mobilizes. There is a multitude of possible answers, and all of them are found on a board of 64 squares. Throughout the centuries, chess game has always generated significant enthusiasm, and Computer Scientists are no exception. During the 90s, one of the goals of early computer scientists was to create a chess-playing machine.
Finally in 1997 after several attempts, the IBM supercomputer Deep Blue defeated the former World Champion Garry Kasparov. The Chess game became the first discipline in which a machine succeeded in beating a World Champion.
Now, chess AI has completely overtaken human beings. To quantify this, the highest Elo rating ever reached by a chess player is 2882 (Magnus Carlsen – 2014). Stockfish, one of the most popular chess AI, achieves an Elo rating of 3544 (December 2021). And the newest one, AlphaZero, was estimated by Wesley So 1[1] as a 3700-4000 Elo rating.
Unlike checkers (draughts in British English) which is weakly solved, chess game has not yet been solved. We are not able to fully determine the outcome of a game.
Three outcomes are possible: Win, Lose or Draw. And despite what one might think, draw is as frequent as the other two outcomes.
During the 2021 World Championship between the World Champion Magnus Carlsen and the challenger Ian Nepomniachtchi, it was not until

[1] Ranked No 2 in February 2017

the 7th game that the 2 players settled with a victory for Magnus.
In theory, white has a non-negligible advantage from the start of the game because of its initiative, and computers have confirmed this to us. Because of this advantage, black doesn't hesitate to grab the draw when they can do so. It's for this reason that the draw outcome is so frequent.

In 2021, Rafal Drezewski and Grzegorz Wator published a paper2[2] about considering chess game data as sequential data. In their paper, through two statics data representations (algebraic and bitmap) and using models applicable to sequential data, they were able to develop a model to determine the outcome of a game with an accuracy of 69%. They also concluded that the fact of not weighting the values of the pieces (bitmap representation) made it possible to obtain slightly more satisfactory results than by using the classical algebraic weighting. This paper aims to complete their work and answer the following question:
What about dynamics representation?

Chess is a dynamic game where position must constantly be reassessed and where the weight of the pieces fluctuates depending on the situation. Therefore, it seems interesting to vary the values of the pieces using certain chess theories.

2. Related work

The main paper which we will begin with is Chess as Sequential Data in a Chess Match Outcome Prediction Using Deep Learning with Various Chessboard Representations by Rafal Drezewski and Grzegorz Wator.
In this paper, the authors assumed that the moves played in a chess game can be interpreted as sequential data.

[2] Chess as Sequential Data in a Chess Match Outcome Prediction Using Deep Learning with Various Chessboard Representations - 2021

Therefore, they built 2 models based mainly on LSTM3[3]:

- The content model: consists of recursive LSTM layers on the game data (moves played during the game).
- The context match fusion model: combines the content model and the data describing the game (Elo ranking mainly).

In addition to these 2 models, their goal was to compare 2 types of representation:

- Bitmap: assigning a static value of 1 to all pieces
- Algebraic: assigning a static value relative to the power of the piece (ex: 1 for Pawn, 9 for Queen, etc…).

An interesting finding was to see that the model with the bitmap representation had better accuracy than the one with the algebraic representation, which may seem surprising since the algebraic representation carries more information than the bitmap representation.
The result of the bitmap representation gave an accuracy of 69% which is better than most existing models and than a random choice which would be 33%.
We will use these models in our approach and try to improve the results.
We were also interested in the related works analyzed in their article:

<u>Online Prediction of Chess Match Result</u>
Scientists from the university located in the United Arab Emirates have proposed a solution that predicts the outcome of the game based on the moves made by players, meaning it considers the dynamism of the game. The chess database is a collection of past chess games. From this database, they extract and select features to generate training data, which will be used to train a classification model. Finally, when a new game is played, the probabilities of win/loss/draw are predicted after every move using the trained classifier. The proposed system dynamically updates the winning (or drawing) probabilities after each move of the game.
Their solution is based on extracting five features from each move—Piece, Column, Row, Check, and Capture. Below is a brief

[3] Long Short-Term Memory

explanation of what each of these features means:

- Piece — the type of piece. This property has six possible values P, R, N, Q, K, representing pawn, rook, knight, bishop, queen, and king respectively.
- Column — chessboard rank, possible values are a–h.
- Row — chessboard file, possible values are 1–8.
- Capture — whether the opponent's piece has been captured or not, possible values are 0, 1.

Their success rate prediction was 66%.

Chess game result prediction system

A Chess Game Result Prediction System was proposed by a group of scientists from the University of Stanford. Their project was to train a classifier with the World Chess Federation (FIDE) rating system using a training dataset of a recent eleven-year period, which ranges from the year 2000 to 2011, and then use their system to predict the outcome of chess games played by the same players in the following half-year. The authors proposed their player ranking system, which could be better than the widely used ELO system. Based on the designed system, they predicted the results of games played by the same players. As an indicator of whether this approach is good, they used a comparison between the predicted result and the actual result of the match. Their success rate of the prediction was 55.64%.

We also studied other articles to see the existing approaches specific to certain phases of the game and by intellectual curiosity.

Improving Opening Book Performance Through Modeling of Chess Opponents:

The best way to win a war is to gather information beforehand and use them to your advantage. A method has thus been developed to help computer chess players analyze their opponent's record and detect which type of openings they have played or have been played against them to play optimally from the start and create a precious momentum for the rest of any game.

A New Approach to Draw Detection by Move Repetition in Computer Chess Programming:

Drawing in a game happens very often, and is a strategy that computers do not usually consider when playing chess. The authors have implemented a new method allowing to detect when piece movement becomes redundant and starts showing hints of a draw. The algorithm tries to upgrade the usual matrix-based representation of the chessboard to a string-based representation to detect faster and more efficiently drawing positions.

Data Compression in chess positions:

Former methods of encoding legal chess positions could do so in 136 bits. The objective of this article is to build an algorithm that encodes such problems in less than 136 bits. Said algorithm discusses the ability to determine chess positions in an average of 70 yes/no questions, meaning 70 bits on average.

Deep learning investigation for chess player attention prediction using eye-tracking and game data:

In Art and life in general, we tend to focus on what is the most important. The article tries to implement a weighting algorithm based on where a professional chess player looks when playing to determine the most important pieces at a given moment: the most looked at piece is one of, if not the most important piece on the chessboard.

Comparing Typical opening move choices made by humans and chess engines:

Based on Shannon's very first paper about chess computer players, this article tries to compare opening books between both types of players (computer and human), using Shannon's entropy and other statistical tests to improve the quality of opening books.

A Machine Learning System for Supporting Advanced Knowledge Discovery from Chess Game Data:

In this paper, the authors presented a machine learning system, called "haystack" with supervised as well as unsupervised learning components to analyze large data sets of chess games (in portable game notation PGN, which records the moves in SAN and positions

of pieces in FEN). The unsupervised learning module groups similar chess games together: based on the similarity distance function. Whereas the supervised learning module builds a classifier to predict common chess board configuration and moves for victory. Their evaluation results showed that chess can be a great testing ground for machine learning and data mining techniques for big data analytics.

An Overview of Machine Learning in Computer Chess:
This paper collected various works that had been done on learning in chess with both man and machine approaches (up to 1986): rote learning, induction, and advice taking.
According to S.Skiena, the only real path to progress is based on the development of pattern-based programs reflecting the behavior of human masters. He then proposed new perspectives to the problem with an improvement of heuristic evaluation functions for example.

Chess AI with Different Behavioral Tendencies and Its Application:
In this article, researchers discuss different methods to make Chess AI better. These methods are based on a deeper experience of the board. By varying the weights of the evaluation functions throughout the game, this article is close to our experimentation which tends to consider different phases of the game: early, mid, and end game.

Knowledge Discovery in Chess Databases: A Research Proposal:
Throughout this article, the researchers tried to show how useful chess databases can be for data mining. An interesting point related to our work is that they consider moves as a sequence (as we do in our experiment). To this, they add that not all information is necessarily relevant and that the position of the pieces on the board may not necessarily be more useful than knowing if the piece is present or not.

DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess:
This article presents a deep neural network capable of playing at a Grandmaster level. It is close to our experimentation because it does not use any a priori knowledge about chess or its rules, it simply uses a huge database of several million games just like us. The playing style of DeepChess is close to a real player, more offensive than other computers.

3. Problem formulation

In addition to defining a relevant model, Rafal Drezewski and Grzegorz Wator showed that bitmap notation was better than algebraic. The difference between these two representations is the weightings assigned to each piece (Figure 1).
With the two representations, a position is presented as a vector of 12 elements. Each element is an 8x8 matrix representing specific types of pieces on the board: [White Pawn, White Knight, White Bishop, White Rook, White Queen, White King, Black Pawn, …].
In the case of the bitmap representation, regardless of the position, all the pieces have a value of 1. Unlike in the case of algebraic representation, each piece has a value relative to its power:

- Pawn: 1
- Knight & Bishop: 3
- Rook: 5
- Queen: 9
- King (optional): 10

A multitude of other variations exists but this is the most frequent one.
Although presenting adequate orders of magnitude, this weighting doesn't consider certain essential aspects of a chess game. Chess is a dynamic game where the value of the pieces varies with each move played.
In some situations, having more points should not be considered an advantage: position takes precedence over material imbalance.
On several occasions, the former world champion Garry Kasparov didn't hesitate to sacrifice his queen to gain a positional advantage. A position can be greatly advantageous even with a lower number of points. It's in such situations that algebraic notation can wrongly assume the results. And

in this way, it also explains why not weighting the parts can give better results.

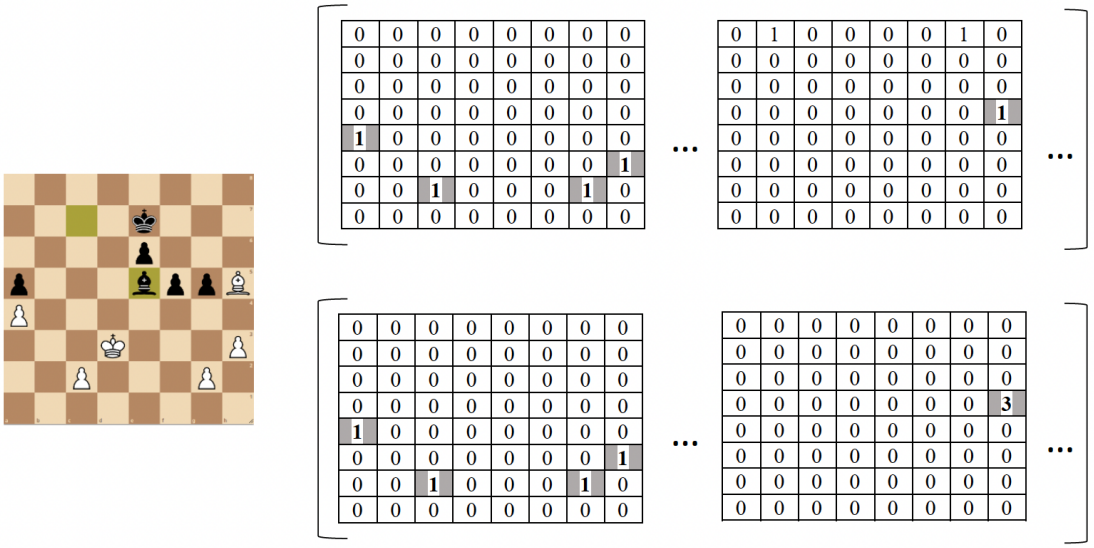

Figure 1: Bitmap & Algebraic representations

Let's take a few examples to illustrate this (King has been removed):

Figure 2: Random chess position 1

In this position, the white player seems to have a big advantage over the black player. In algebraic representation, they would have 3*1+3 = 6 against 4*1 = 4.
But the bishop is completely inactive. He can't cross the board, being blocked by black pawns. It would therefore not be fair to give it a value of 3. Its real value would be 1, at most. In this position, the bitmap representation is more adequate by attenuating the bishop's value.
If you change a bit the position and put the bishop in c6, it becomes the complete opposite where the bishop would have a value of at least 3. So, the algebraic representation becomes more adequate.

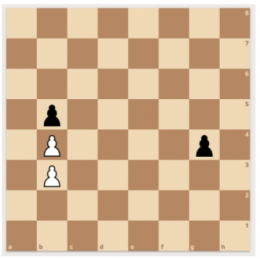

Figure 3: Random chess position 2

In this example, the 2 representations would result in a tie for the 2 players. But it's not: the 2 white pawns are called "doubled" pawns, one hindering the advance of the other. The single pawn alone blocks the 2 pawns, so it should have a value equal to the 2 white pawns.
And the second black pawn is called a "passed" pawn. Nothing can stop it and prevent it from becoming a Queen. Its value should be drastically higher than that of a classic pawn.
In conclusion, in this position, black has an overwhelming advantage and this position shouldn't be considered equal as the bitmap and algebraic representations suggest.

Hence the need to vary the value of pieces according to the position to combine the advantages of these 2 representations.

4. Solution

4.1. Dynamic Representation

There are many theories about pieces value' variations. It's not an exact science and it requires making many assumptions.

In 1999, Hans Berliner[4] defined value adjustments according to the positions of pieces and their status on the board. For example, Bishop, Rooks, and Queens gain up to 10 percent more value in open positions and lose up to 20 percent in closed positions.

He also defined precise values for pawns as well as coefficients to be applied to them according to their status.

It's easier to be exhaustive with respect to the status of pawns than with other major other pieces. So, our representation will be based on pawn variations.

Rank	a & h file	b & g file	c & f file	d & e file
2	0.90	0.95	1.05	1.10
3	0.90	0.95	1.05	1.15
4	0.90	0.95	1.10	1.20
5	0.97	1.03	1.17	1.27
6	1.06	1.12	1.25	1.40

Figure 4: Value of non-passed pawn in the opening

Rank	a & h file	b & g file	c & f file	d & e file
2	1.20	1.05	0.95	0.90
3	1.20	1.05	0.95	0.90
4	1.25	1.10	1.00	0.95
5	1.33	1.17	1.07	1.00
6	1.45	1.29	1.16	1.05

Figure 5: Value of non-passed pawn in the endgame

Figures 4 & 5 assign values to pawns according to their positions on the board (rows and columns) and the stage of the game.

Rank	Isolated	Connected	Passed	Passed & connected
4	1.05	1.15	1.30	1.55
5	1.30	1.35	1.55	2.3
6	2.1	2.1	2.1	3.5

Figure 6: Value of pawn advances (multiplier of base amount)

[4] The System: A World Champion's Approach to Chess - 1999

Indeed, a chess game is divided into 3 parts:

- The Opening: Theoretical part, it's the initial stage of a chess game.
- The Middlegame: Begins when both players have completed the development of most of their pieces
- The Endgame: When few pieces are left on the board

The line between these 3 parts is not clear but some experts have tried to define this line:

- Opening to middlegame: At the end of the theoretical part of the opening.
- Middlegame to endgame:
  - Speelman: Each player has 13 or fewer points (Excluding King).
  - Miney: 4 or fewer pieces on the board (excluding King and Pawn).
  - Fine: Position without Queen.
  - Flear: Each player has at most one piece (other than King and Pawn).

We will start from these theories to define our representation and we will add a multiplier (Figure 6) depending on the pawn's status throughout the game.

In 2020, the latest AlphaZero AI has determined a new weight for pieces. Classic algebraic representation is interesting but maybe a bit updated, AlphaZero representation looks more appropriate.

Other pieces will benefit from the values assigned by AlphaZero, theoretically the best static values:

- Knight: 3.05
- Bishop: 3.33
- Rook: 5.63
- Queen: 9.5
- King: 10

	Bitmap	Algebraic	Dynamic
b3	1	1	1.05
b4	1	1	1.155
b5	1	1	1.155
g4	1	1	2.015

Figure 7: Comparison between representations on figure 3

Figure 7 compares the three representations related to figure 3.

As said before, the algebraic and bitmap representations don't identify the winning player in this position while Black has a significant advantage:

- Bitmap: 2 for both.
- Algebraic: 2 for both.
- Dynamic: 3.17 for Black against 2.205 for White.

So, in theory, dynamic representation produces results that are more representative of the position. It shows that Black has a significant advantage with one more point. The difference is not greater because the King can play an important role depending on his position.

4.2. Model

As said before, we will use the same model as the main article: the Content model and the Context Match Fusion model.

The content model is defined as follows:

- Input layer with input size: 768.
- LSTM: 512 cells, tanh activation function, 0.2 dropout.
- LSTM: 256 cells, tanh activation function, 0.2 dropout.
- LSTM: 128 cells, tanh activation function, 0.2 dropout.
- Dense layer: 32 cells, ReLU activation function.
- Classifying layer: 3 cells, softmax activation function.

Additionally:

- Adam optimizer: learning_rate =0.000001, beta_1 = 0.9, beta_2 = 0.99, amsgrad = False.
- Early Stopping mechanism: to monitor the value of val_loss. Stop the training process when the model becomes worse after 5 epochs in a row to prevent overfitting.

And the Context Match Fusion model:

The Context Match Fusion model combines the model presented above, responsible for processing the content of the game (the moves sequence), and the part describing the chess match data (player names and ELOs).

The structure responsible for processing the context is defined exactly as follows:

- Input layer with input size: 4.
- Dense layer: 4 cells, activation function ReLU, dropout: 0.2.
- Dense layer: 16 cells, activation function ReLU, dropout: 0.2.
- Dense layer: 12 cells, activation function ReLU, dropout: 0.2.
- Dense layer: 8 cells, activation function ReLU.
- Flatten layer.

The architecture of the network processing the game flow is the same as defined in the Content Model.

Outputs of both models are concatenated and a classifier is built as follows:

- The layer connecting the outputs from the DBN and LSTM networks.
- Dense layer: 64 cells, ReLU activation function.
- Dense layer: 3 cells, softmax activation function.

Additionally:

- Adam optimizer: learning_rate =0.001, beta_1 = 0.9, beta_2 = 0.99, amsgrad = False.
- Early Stopping mechanism.

5. Experiments

5.1. Hardware and Dataset

Our main hardware is composed of an Intel Core i7-8700K CPU at 3.70GHz and an NVIDIA GeForce GTX 1080GB with a total RAM of 32 GB.

To carry out this experiment, we used a database in PGN format. It's a file format specific to the chess game and is very widely used in this context (Figure 9). It provides all the information needed to understand the game: opposing players, player ranking, result, opening used, and list of moves.

Chess has the advantage of having very large amounts of data available.

For this experiment, given the capacity of our hardware, and to be in line with the results obtained in Rafal Drezewski's and Grzegorz Wator's paper, we worked on 30.000 games that took place between 2017 and 2018.

We split the dataset as follows:

- Training set: 60%, 6454 games won by white, 5653 won by black, and 5893 draws.

- Test set: 20%, 2163 games won by white, 1905 won by black, and 1932 draws.
- Validation set: 20%, 2200 games won by white, 1880 won by black, and 1920 draws.

The balance of the distribution between white wins, black wins, and draws is very important, this avoids distorting the model.

To identify the end of the opening as announced in the previous part, we also used a database listing almost all existing openings (1755 openings and variants) associating the code ECO5 [5]with the corresponding sequence of moves. This allows us to start the midgame at the end of this sequence, and therefore use the values specific to this stage of the game in the dynamic representation.

We have converted each position of each game into a dynamic representation (Figure 8) by modifying the values according to the status of the pawns during the different stages of the game.

We, therefore, converted the 30000 PGN games into 30000 vectors each containing 80 vectors

(equivalent to 40 moves for each player), and each of these vectors contains twelve 8x8 matrices as mentioned above. So, if a game has more than 40 moves, it ends up cut at the 40th move of the black player, and if it has less than 40 moves, the remaining moves are filled by matrices made up of 0.

As in the main paper, we also built the bitmap representation to be able to compare our results to previous ones.

Insofar as we don't have the same hardware to train our models (120000 vs 30000), we will start from the results obtained by their representation on our dataset so that the comparison is more reliable.

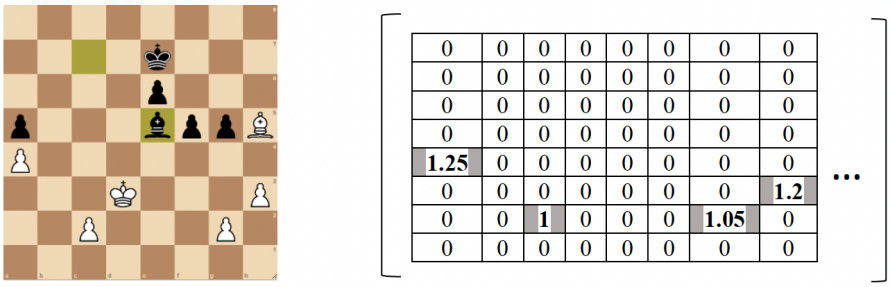

Figure 8: Dynamic representation

```
[Event "St Stefan/Belgrade m"]
[Site "Belgrade"]
[Date "1992.??.??"]
[Round "1"]
[White "Fischer, Robert James"]
[Black "Spassky, Boris V"]
[Result "1-0"]
[WhiteElo "2785"]
[BlackElo "2560"]
[ECO "C95"]

1.e4 e5 2.Nf3 Nc6 3.Bb5 a6 4.Ba4 Nf6 5.O-O Be7 6.Re1 b5 7.Bb3 d6 8.c3 O-O
9.h3 Nb8 10.d4 Nbd7 11.Nbd2 Bb7 12.Bc2 Re8 13.Nf1 Bf8 14.Ng3 g6 15.Bg5 h6
16.Bd2 Bg7 17.a4 c5 18.d5 c4 19.b4 Nh7 20.Be3 h5 21.Qd2 Rf8 22.Ra3 Ndf6 23.Rea1 Qd7
24.R1a2 Rfc8 25.Qc1 Bf8 26.Qa1 Qe8 27.Nf1 Be7 28.N1d2 Kg7 29.Nb1 Nxe4 30.Bxe4 f5
31.Bc2 Bxd5 32.axb5 axb5 33.Ra7 Kf6 34.Nbd2 Rxa7 35.Rxa7 Ra8 36.g4 hxg4 37.hxg4 Rxa7
38.Qxa7 f4 39.Bxf4 exf4 40.Nh4 Bf7 41.Qd4+ Ke6 42.Nf5 Bf8 43.Qxf4 Kd7 44.Nd4 Qe1+
45.Kg2 Bd5+ 46.Be4 Bxe4+ 47.Nxe4 Be7 48.Nxb5 Nf8 49.Nbxd6 Ne6 50.Qe5  1-0
```

Figure 9: PGN format example

[5] Identifier to recognize chess opening/variant.

5.2. Results

First, we had to carry out the experiment on the Bitmap and Algebraic representations in order to have reference values specific to our dataset and comparable to the results of the dynamic representation. Working on a smaller dataset than the reference article, our results were obviously lower.

We obtained an accuracy for the bitmap of 53% for the Content Model and 58% for the Context Match Fusion.

The algebraic representation gave us accuracies of 51% and 55% for the respective models.

For the dynamic representation, we have therefore produced 4 representations, each based on a theory making it possible to define the passage to the endgame (as described in 4.1).

These 4 representations were compared during the content model, and the best one was kept for the context match fusion.

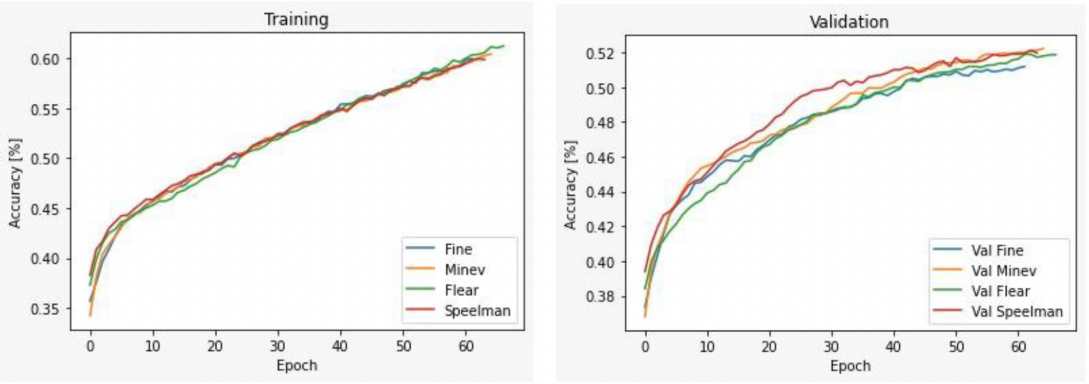

Figure 10: CM accuracy with training set and validation set for Dynamic Representation between the 4 endgame theories

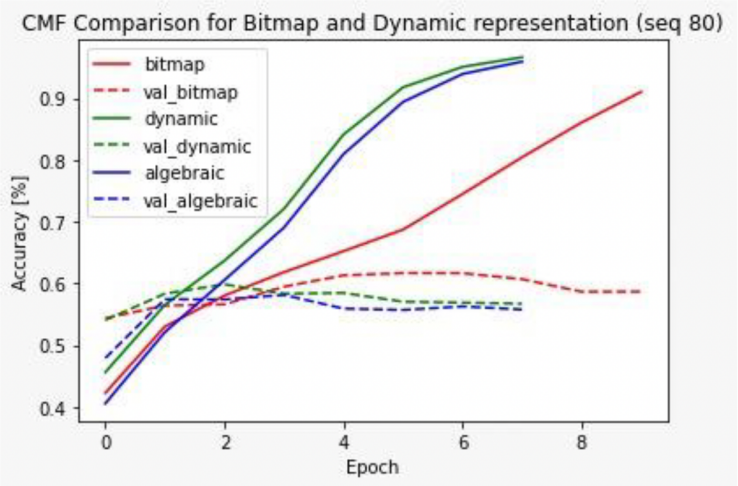

Figure 11: CMF accuracy comparison between Bitmap, Algebraic and Dynamic.

Method	CM Accuracy (%)	CMF Accuracy (%)
Bitmap	52.77	57.92
Algebraic	50.90	55.43
Dynamic Minev	50.70	
Dynamic Speelman	51.73	
Dynamic Fine	50.95	
Dynamic Flear	51.77	55.70

Figure 12: Results comparison between the three representations

For the content model, the majority gave better results than the algebraic representation (almost 52% with Flear's theory), but slightly weaker than the result of the Bitmap representation.

We applied the context match fusion with the Flear theory, giving us an accuracy of 51.77% which is slightly better than the algebraic representation, but again lower than the results of the bitmap representation (Figure 11).

The dynamic representation carries less biased information than the algebraic representation, which allows it to have greater precision, but which nevertheless remains lower than the bitmap representation (Figure 12).

5. Conclusion

To overcome the shortcomings of the static Bitmap and Algebraic representations, we have implemented a dynamic representation that causes the value of the pawns to fluctuate according to their position and their status (passed, connected, isolated, or not passed) in order to reduce the bias that certain positions may cause.

The Dynamic representation was therefore found to be better than the algebraic representation, but less successful than the bitmap representation.

By testing four endgame theories, we were able to show that for this kind of treatment, that of Flear is the most appropriate, namely by considering as an endgame a position where each player has at most one piece (other than King and Pawn).

This result could be greatly improved by taking into account the fluctuations of the major pieces (Queen, Rook, Bishop, and Knight) stated in the work of Hans Berliner.

References

Brown, James A.; Cuzzocrea, Alfredo; Kresta, Michael; Kristjanson, Korbin D.L.; Leung, Carson K.; Tebinka, Timothy W. (2017). [IEEE 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA) - Cancun, Mexico (2017.12.18-2017.12.21)] 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA) - A Machine Learning Tool for Supporting Advanced Knowledge Discovery from Chess Game Data. , (), 649–654. doi:10.1109/ICMLA.2017.00-87

Skiena, Steven S. (1986). An Overview of Machine Learning in Computer Chess. ICGA Journal, 9(1), 20–28. doi:10.3233/ICG-1986-9103

Masud, M. M., Al-Shehhi, A., Al-Shamsi, E., Al-Hassani, S., Al-Hamoudi, A., & Khan, L. (2015). Online Prediction of Chess Match Result. Lecture Notes in Computer Science, 525–537. doi:10.1007/978-3-319-18038-0_41

Yuanqi Hu and Ruimin Lv. 2021. Chess AI with Different Behavioral Tendencies and Its Application. In 2021 2nd International Conference on Artificial Intelligence and Information Systems (ICAIIS 2021). Association for Computing Machinery, New York, NY, USA, Article 265, 1–5. DOI:https://doi.org/10.1145/3469213.3470698

Johannes F¨urnkranz. Knowledge Discovery in Chess Databases: A Research Proposal. Technical Report OEFAI-TR-97-33, Austrian Research Institute for Artificial Intelligence, 1997.

David, O., Netanyahu, N.S., & Wolf, L. (2016). DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess. ArXiv, abs/1711.09667.

Vuckovic, V., & Vidanovic, D. (2004). A New Approach to Draw Detection by Move Repetition in Computer Chess Programming. ArXiv, cs.AI/0406038. Balkenhol. (1994). Data Compression in Encoding Chess Positions. ICGA Journal, 17(3), 132–140. https://doi.org/10.3233/icg-1994-17303

Steven Walczak. 1996. Improving opening book performance through modeling of chess opponents. In Proceedings of the 1996 ACM 24th annual conference on Computer science (CSC '96). Association for Computing Machinery, New York, NY, USA, 53–57. DOI:https://doi.org/10.1145/228329.228334

Louedec, J. L., Guntz, T., Crowley, J. L., & Vaufreydaz, D. (2019). Deep learning investigation for chess player attention prediction using eye-tracking and game data. Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications - ETRA ’19. doi:10.1145/3314111.3319827

Rafał Dreżewski, Grzegorz Wątor, Chess as Sequential Data in a Chess Match Outcome Prediction Using Deep Learning with Various Chessboard Representations, Procedia Computer Science, Volume 192, 2021, Pages 1760-1769, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2021.08.180.

Levene, M., & Bar-Ilan, J. (2007). Comparing Typical Opening Move Choices Made by Humans and Chess Engines. The Computer Journal, 50(5), 567–573. doi:10.1093/comjnl/bxm025