DEGREE PROJECT IN COMPUTER ENGINEERING,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2017

An Investigative Study about Application of Supervised Learning for Making Predictions in Chess

HARRY VUONG

SYLWESTER LILJEGREN

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

An Investigative Study about Application of Supervised Learning for Making Predictions in Chess

Harry Vuong
Sylwester Liljegren

Degree Project in Computer Science, First Cycle (DD142X)
Date: June 2, 2017
Supervisor: Roberto Guanciale
Examiner: Örjan Ekeberg
Swedish title: *En utredande studie kring tillämpning av supervised learning för att göra förutsägelser i schack*
School of Computer Science and Communication

Abstract

Supervised learning is not as popular as reinforcement learning in chess programming due to its inability to achieve as high prediction accuracies as reinforcement learning. However, through extensive search by the authors, there seems to be a few numbers of research conducted that focus on applying supervised learning into chess. Therefore, this study investigates how supervised learning could be used to make predictions in chess so that an empirical understanding of supervised learning using both logistic regression and convolutional neural networks is provided. Both the machine learning algorithms will be tested and compared to the prediction accuracies acquired by reinforcement learning through other studies (it will not be implemented in this study). The prediction task was to predict the position from which the next chess piece moves in a chess game.

It has been concluded from this study that convolutional neural networks are better at predicting than logistic regression, but had higher tendencies to suffer from overfitting compared to logistic regression. When comparing these two supervised learning algorithms to reinforcement learning, supervised learning algorithms do not achieve as high prediction accuracies as reinforcement learning in general, but could be used as heuristics in various programming contexts in the future.

Future research should investigate regularization techniques to overcome with overfitting tendencies in both machine learning algorithms and investigate how data representations may affect the prediction accuracy of respective machine learning algorithm.

Sammanfattning

Supervised learning är inte lika populär som reinforcement learning i schackprogrammering på grund av dess oförmåga att uppnå samma höga förutsägelsenoggrannhet som reinforcement learning. Genom omfattande sökning av författarna verkar det dock finnas några få undersökningar som fokuserar på att tillämpa supervised learning i schack. Därför undersöker den här studien hur supervised learning kan användas för att göra förutsägelser i schack för att erhålla vidare kunskap om supervised learning med hjälp av både logistic regression och convolutional neural networks inom schackprogrammering. Båda maskininlärningsalgoritmerna kommer att testas och jämföras med de förutsägelsenoggrannheter som erhållits av reinforcement learning genom andra studier (det kommer inte att implementeras i denna studie). Förutsägelserna, som görs i denna studie, var att förutsäga den position, från vilken nästa schackpjäs kommer att röra sig.

Slutsatserna från denna studie är att convolutional neural networks är bättre på att förutsäga än logistic regression, men hade högre tendenser att drabbas av overfitting jämfört med logistic regression. När man jämför dessa två ”supervised learning”-algoritmer med reinforcement learning visar resultaten att ”supervised learning”-algoritmerna inte uppnår lika höga förutsägelsenoggrannheter som reinforcement learning i allmänhet, men kan användas som heuristiker i diverse programmeringssammanhang i framtiden.

Vidare forskning bör vara att undersöka regularization-tekniker för att överbrygga problemen med overfitting-tendenser i båda maskininlärningsalgoritmerna och undersöka hur datarepresentationer kan påverka förutsägelsenoggrannheten hos respektive maskininlärningsalgoritm.

Table of contents

1 Introduction *6*

1.1 Purpose and problem statement **7**

1.2 Assumptions and constraints **7**

1.2.1 Assumptions about training data 7

1.2.2 Prediction task 8

1.2.3 Technical constraints 9

2 Background *10*

2.1 Chess **10**

2.2 Supervised learning **11**

2.2.1 Logistic regression 12

2.2.2 Artificial neural network 13

2.2.2.1 Convolutional neural network 13

2.3 Overfitting **14**

2.4 Regularizations **14**

2.5 Cross-validation **15**

2.6 Related work **16**

2.6.1 Supervised learning 16

2.6.1.1 Logistic regression 16

2.6.1.2 Convolutional neural networks 17

2.6.2 Reinforcement learning 17

3 Method *18*

3.1 Choice of learning algorithms **18**

3.2 Dataset and the format of data samples **18**

3.3 Metrics **20**

3.4 The structure of the study **21**

3.5 Software **22**

4 Results *23*

4.1 Fraction of datasets between the training data and test data 23

4.2 Move rounds in chess parties 24

4.3 Number of games used from the dataset 25

4.4 Execution time of initializing, training, and testing machine learning models altogether 26

5 Discussion *27*

5.1 Interpretation of results **27**

5.2 Analysis of results **28**

5.3 Future research **30**

6 Concluding remarks *31*

7 References *32*

Appendix *34*

Appendix A – Git repository of this study **34**

Appendix B – Further readings **34**

Reinforcement learning 34

Temporal-Difference learning 34

PGN 35

ELO rating 36

1 Introduction

There are some general flaws with strict algorithmic approaches such as minimax search, alpha-beta pruning and domain-knowledge based evaluations in chess games. One of those flaws is that the strict algorithms are short-sighted [1]. Such algorithms/strategies are lacking in intuition against their opponents when deciding the move to be done in a current board state to increase chances of winning. Therefore, implementing learning algorithms for the chess computers could be an alternative approach to make the chess computers more capable of competing against the best chess players in the world [1].

Two learning paradigms have been present since the introduction of learning algorithms in the chess programming field and these are supervised learning and reinforcement learning (the latter one explained in Appendix B). It has been observed, through extensive research by the authors of this study, that supervised learning is less popular due to its inability to achieve higher accuracies for predicting chess moves compared to reinforcement learning. It could be due to extrinsic details such as data representation or certain properties in chess datasets that are more disadvantageous for supervised learning than for reinforcement learning. However, these are just speculations, since there does not seem to exist studies that explains why reinforcement learning and supervised learning perform differently when being applied for predicting moves in chess.

Today, there are numerous studies that focus on applying reinforcement learning in chess, as opposed to supervised learning where there are less studies that put focus on application of said learning paradigm in chess. The only studies that could be found after extensive search in the internet that focus heavily on supervised learning in chess, were *''Classifying Chess Positions''* by Christopher De Sa in 2012 using logistic regression [2], and *''Predicting Moves In Chess Using Convolutional Neural Networks''* by Barak Oshri and Nishith Khandwala in 2015 using convolutional neural networks [3].

This study, on basis of the studies, intends to investigate the two supervised learning algorithms, logistic regression and convolutional neural network and provide an empirical understanding of each supervised learning method. Because of the lack of studies with supervised learning in chess, this empirical understanding should contribute to increase the common knowledge of supervised learning in chess.

1.1 Purpose and problem statement

The purpose of this study is to compare two academically documented supervised learning algorithms to each other with respect to accuracy and execution time, from which it is possible to conclude the best applications of these machine learning algorithms within chess engines. The problem statement (or the question) of this study is as follows:

- *How do the logistic regression and the convolutional neural network perform in terms of prediction accuracy and execution time in chess?*

1.2 Assumptions and constraints

Assumptions and constraints on the study will be presented in this subsection. The assumptions apply to what is being assumed about the training data and the constraints applies to what is being predicted in the study.

1.2.1 Assumptions about training data

It is assumed that players in general engaged the chess games seriously in the sense that moves being done by the respective players were done with consideration and was not poorly executed because of not being serious. Despite this, there likely are moves made by players that did not play seriously or even did not intend to win certain chess games. However, the frequency of such moves is assumed to be low and thus negligible.

1.2.2 Prediction task

The prediction task of the machine learning algorithms is to predict the position that the next chess piece is being moved from. The prediction task will however *not* predict the new position that the chess piece is being moved to. The reason for this is due to the limitations on the provided implementations/libraries and thus specializations of all the provided implementations/libraries for each machine learning algorithm were not possible. However, despite this simplification of the prediction task, knowing what coordinate the next pieces are to be moved from is still an important detail to consider if one wants to have the complete information about a whole move that has been done in chess. This information is used within the sophisticated chess engines, such as Giraffe invented by Matthew Lai in 2015 [4].

Furthermore, the predictions being made are not goal-oriented: the predictions only concern with the typical moves that are done by any human being in certain board states without investigating the long-term outcome consequences of a chess game (i.e. prediction task is related to simply predicting behavior in chess games). Investigating the long-term effects of the consequences requires more information about the move that is predicted in this study and therefore will be excluded from this study.

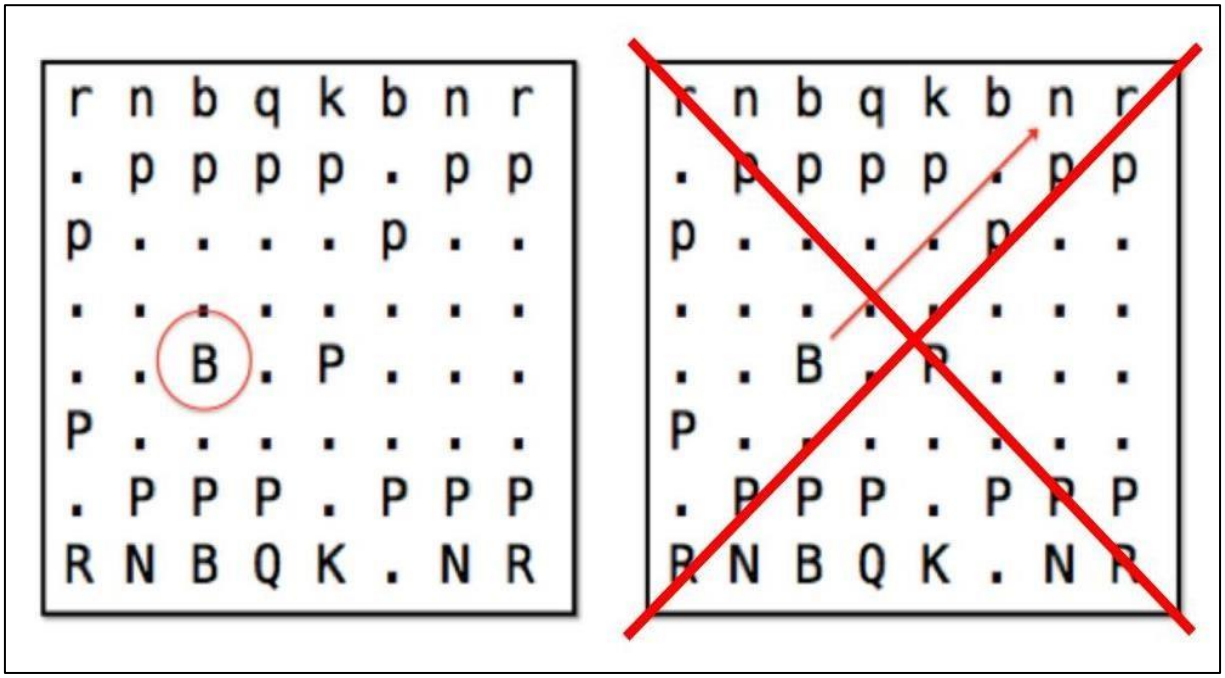

Figure 1: Left) Predicting the start position of the next piece that is to be moved next, which is the prediction task. Right) Predicting where the same piece is to be moved, which is not part of the prediction task. Source: Oshri/Khandwala [3]*, red cross excluded)*

1.2.3 Technical constraints

The technical constraints being made in this study are mainly derived from the work of Oshri/Khandwala regarding application of convolutional neural networks in chess [3] that are applied on the chess games from the dataset being used in this study. These constraints apply to the parameter settings for the implementation of convolutional neural network and the chosen topology of the convolutional neural network. In the study of Oshri/Khandwala, a three-layered convolutional neural network was chosen since the highest accuracies were obtained with this kind of topology. Other technical constraints regarding the implementation of the convolutional neural network in this study such as weight initialization, types of layers being used and which activation functions were all chosen accordingly to the study of Oshri/Khandwala to maximize the prediction accuracy of the convolutional neural network.

Furthermore, implementations based on reinforcement learning will not be implemented in this study. The reason for this is due to lack of libraries and that the few libraries that were found were hard to understand and did not have documentations. However, reinforcement learning remains relevant to this study in the sense that it could be interesting to compare with supervised learning, since the most successful chess engines use reinforcement learning.

Finally, due to impossibilities to improve the time duration of the data parsing and the processing in the testing program because of obstacles with memory management on the software, this study relied on re-parsing and re-processing data all over again before training each model and then obtaining test results. This, however, lead to higher time consumption during each program execution and this detail limited both the choice of testing methods and the sizes of the data.

2 Background

This section begins with introducing the chess game. When introducing chess, its rules and basics are explained for the reader only so that the reader may understand the game characteristics of the chess game. Furthermore, supervised learning will be introduced as a concept and the specific supervised learning algorithms being used in this study will be explained. Further concepts/ideas to be introduced later will be overfitting, regularization and cross-validation in general. This section finishes with brining up related works that have been conducted in chess programming.

2.1 Chess

Chess is a two-player board game, where the board has the dimension 8x8 and each player has an assigned set of pieces to move with during the game (which are colored either white or black). The goal of the game is to checkmate the opponent, i.e. to capture the king of the opponent [5]. Other outcomes than win or loss is also possible, like stalemate, meaning that the chess game is drawn between the players. The white player always begins with making the first move. Each player has initially following pieces:

- **Pawns (x6)** – A pawn is a piece that can go only one step further in the forward direction. A pawn can capture another piece if that piece stands one step from the pawn in the same direction that the pawn is heading. When the pawn has reached the last row in their own direction, they are promoted and become any other piece, which will be listed further here.
- **Rooks (x2)** – A rook is a piece that can go either all the way in the vertical direction or in a horizontal direction relative to the position they are moving from. They can capture any enemy piece if that enemy piece is on the same row or column as the rook.
- **Knights (x2)** – A knight is a piece that can jump two steps in one direction and one step in either of the horizontal directions after jumping two steps. All the steps of the knight remind of a "L". Knights can only capture if the enemy piece is on a square, to which the knight legally can jump.
- **Bishops (x2)** – A bishop is a piece that can move only diagonally on a chess board. It works just like a rook with the only difference that the bishops only move diagonally in all directions rather than either horizontally or vertically like the rook.
- **Queen (x1)** – A queen is a piece that can go either diagonally, horizontally and vertically in all directions. The queen could be described as a combination of the properties that a bishop and a rook possess.
- **King (x1)** – A king is a piece that can go one step only in all directions, whether they are diagonal, horizontal and vertical. A king cannot be captured and whenever the king is checked by the opponent, the player is obliged to prevent the capture of the king by either moving its own king to somewhere else, blocking the check with another of its pieces or capture the piece that checks the king. If the king can be captured no matter what move the player-to-move do by the player's opponent, then the same player have been check-mated and lost the chess game.

The initial chess board state usually looks like this:

Figure 2: The initial board state of a chess game. Source: www.chess.com [5]

The number of legal chess board configurations has been, through a review of the literature in chess, observed to be a controversial topic because different researchers have used different methods to count the number of legal chess board configurations in chess. However, in 2012 Eric Holcomb has concluded in his study [6] that there are approximately $4*10^{40}$ legal chess board configurations in chess. This fact will be used later in Discussion.

2.2 Supervised learning

Supervised learning is about finding a function that describes an amount of inputs and corresponding outputs as well as possible [7]. This function could then be used to predict the output of a first-seen input based on those inputs and corresponding outputs, on which the function was trained.

Formulated mathematically: assume that we have N samples that are on the form (X, Y), where $X = [x_1\ x_2 \dots x_p]$ and x_i is a feature and p number of features in each vector (this is the same for Y but the number of features may not necessarily be p as for X). Then X is the input vector and Y the corresponding output vector (or possibly some scalar output value). The goal with supervised learning is to find a function $f: X \rightarrow Y$ that describes the input-output relation for each of the N samples as best as possible. This function will then be used for predicting output of first-seen input that is passed to the mentioned function. The function f is sometimes rather referred to as model.

NOTE: The input X is not limited to only being a vector. It could also be either a scalar value or a multidimensional vector, depending on the type and settings of a certain machine algorithm.

2.2.1 Logistic regression

Logistic regression (Logit or LogReg) is the discrete version of the linear regression, i.e. the outputs from a model are discrete rather than continuous. Provided a number of samples on the form (X, y) (where the input vector X is described as mentioned in this section and y some discrete value), logistic regression means that a model is trained on a dataset by estimating the most appropriate weight vector $W_y = [w_{y,1}\ w_{y,2}\ \dots\ w_{y,p}]$, including the constant coefficient $w_{y,0}$, (usually estimated with Maximum-Likelihood) in accordance to a class label y and use it to determine the value of $\Pr(Y = y \mid X, W_y)$ by calculating the logistic function expression [8], which is as follows:

$$\Pr(Y = y \mid X, W_y) = \frac{e^{w_{y,0} + W_y X}}{1 + e^{w_{y,0} + W_y X}}$$

Provided weight vectors for each class label, the prediction is done by selecting the class label that maximizes the logistic function expression, i.e. $L(X) = \max_{y \in Y} \Pr(Y = y \mid X, W_y)$.

NOTE: Usually, when using the term logistic regression, it often refers to the binary logistic regression, where an input data is assigned to either of two class labels. Meanwhile, many authors stress the term multinomial logistic regression when talking about logistic regression where number of classes are more than two. In this study, when logistic regression is mentioned, it will only refer to the multinomial logistic regression (since more than two class labels are dealt in this study) rather than the binary logistic regression.

2.2.2 Artificial neural network

An artificial neural network (usually abbreviated ANN) is a machine learning algorithm that historically is inspired by the cognitive structure of a human brain and the information processing that is involved in human brains. Artificial neural networks are applied in various contexts such as in finances [9].

An artificial neural network consists usually of an input layer, ≥1 (1 or more) hidden layers and an output layer. Each layer in turn consists of neurons (alternatively called nodes or units). A neuron from a layer is connected to neurons in the upcoming layer and the connection between them are associated with a numeric weight value. The output h_i from i:th neuron in a hidden layer is

$$h_i = \sigma\left(\sum_{j=1}^{N} V_{ij}x_j + T_i^{hid}\right)$$

where σ is an activation function, N number of input neurons, V_{ij} the weights, x_j input to the input neurons and T_i^{hid} the threshold terms of the hidden layers. The activation function is needed both to introduce nonlinearity and to bound the value of the neuron so that the artificial neural network is not paralyzed by divergent neurons.

2.2.2.1 Convolutional neural network

Convolutional neural network (also called ConvNet or CNN) is a type of an artificial neural network that is comprised by one or several convolutional layers, which in turn is followed by several fully connected layers as in a traditional multilayered neural network. A convolutional neural network takes a three-dimensional input and the task of the convolutional layers is to transform an input space into a smaller region of the input space through feature mapping either one time or several times. The size of the region is usually determined by the pool size, i.e. the size of the regions being transformed into. After performing these convolutions through convolutional layers, the region of the input space will undergo the same calculation procedure as in a traditional artificial neural network [10]. The figure down below show how convolutional neural networks operate on technical basis when e.g. classifying images. The convolutional neural network has had great successes within several application fields such as face recognition.

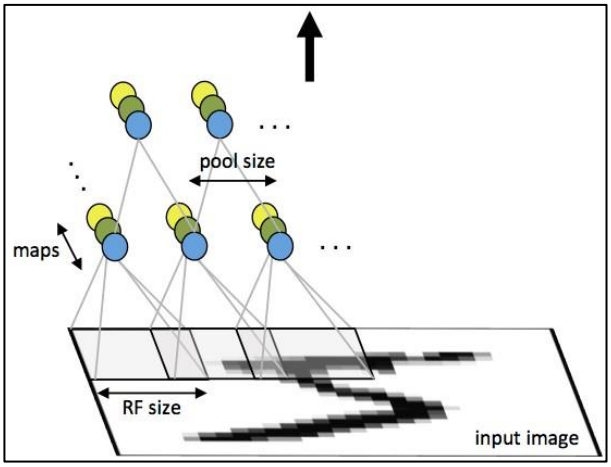

Figure 3: How a CNN typically works when e.g. predicting the digit of an image. Source: Stanford University [10]

2.3 Overfitting

Overfitting around a machine learning algorithm occurs when a machine learning algorithm generalizes the data poorly. It means that the trained machine algorithm performs well at predicting on known data, but significantly worse on unknown data [8]. The most common reasons for overfitting is either that the machine learning algorithm is too complex for the prediction task, that the data used for training is not representative (too many irregular data samples, which are called outliers) or simply there is not enough amount of data samples the machine learning algorithm learns from. Another possible explanation for the occurrence of overfitting would be that there are too many data features in the data samples that make all the predictions more dependent of the properties of the data samples. The occurrence of overfitting implies that the true prediction accuracy of a machine learning algorithm may not be as high as suggested if the train accuracy is much higher than the test accuracy.

To prevent overfitting for a machine learning algorithm, one could increase the number of data samples to decrease the variance in the predictions being made. However, when working with limited amount of data samples and increase of that amount is not possible or impractical due to certain reasons, one could apply regularization on the machine learning algorithm. The idea of regularization will be explained in the next subsection.

2.4 Regularizations

Regularization is a process of solving problems due to ill-posed problems or overfitting in machine learning. In practice, it usually means that the parameters of a model are tuned by using e.g. penalty terms to decrease the variance of the model [8]. Decreasing the variance will imply increasing bias, but the same model will in this scenario be able to better generalize data no matter of its belonging to the training data, upon which the model is trained. This will result that the model, despite predicting on the training data worse, will be better at predicting on non-training data.

2.5 Cross-validation

Cross-validation is a model validation technique. Its purpose is to assess how well the results produced by the models generalizes to independent datasets [8]. There are several methods of cross-validation. The key part of cross-validation is to divide a dataset into training data and test data. The training data is used to train the model and the test data is used to assess how well the trained model generalizes to data that it has not seen before. Another name for test data is validation data. How the test and training data is partitioned and how many times it is partitioned depends on the method. The most common test method is k-fold cross-validation.

k-fold cross-validation partitions the data sample into test data and training data. It does this by dividing the original dataset by k. The k:th partition is used as test data and the other $k-1$ partitions are used as training data. This process is then iterated k times and each time the partition of the original dataset for the test data and training data remains the same but the training data and test data varies. k is usually chosen to be either 5 or 10 [8]. The process of k-fold cross-validation can be illustrated in the following picture:

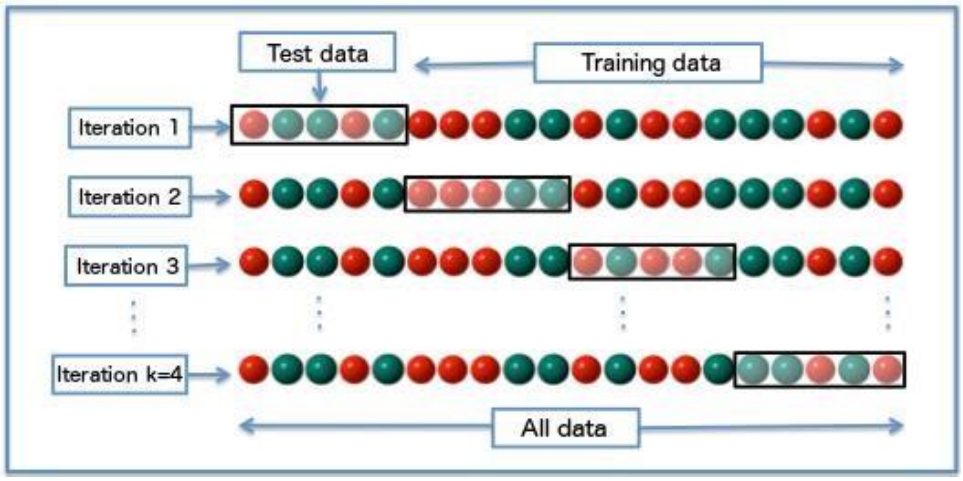

Figure 4: 4-fold cross-validation. Source: Wikipedia [11]

The results can be averaged as a measure of estimating how well the model perform with independent datasets.

The holdout method is the simplest form of cross-validation [12]. The holdout method partitions the original dataset into test data and training data in the same way as k-fold, except that the method does not iterate and only runs once. Due to this, in comparison to k-fold cross-validation, holdout runs faster in general, which is beneficial when performing cross-validation on greater datasets if the model is slow [13].

2.6 Related work

In this subsection, all the related work that concerns with application of supervised learning in chess will be brought up and explained to the reader to highlight how the supervised learning algorithms have been used in earlier work conducted by other authors. Also work that is related to application of reinforcement learning is also brought up and discussed so that the context around the study is clarified.

2.6.1 Supervised learning

All work related specifically to supervised learning is brought up here. This is to deepen in the specific algorithms that this study investigates and give the background to the application of each necessary supervised learning algorithm that is used within this study.

2.6.1.1 Logistic regression

In chess programming, the application of logistic regression was researched by Christopher De Sa in 2012 [2]. The dataset used in the De Sa's study only contained games played between humans. In that study, 500,000 training samples and 500,000 test samples were used to perform a cross-validation test with logistic regression, where a prediction accuracy of approximately 70 % on training data and test data was obtained. The prediction task in the mentioned research could be defined as quoted by the author itself:

"(...) Formally, we can express this problem as a ML problem as follows: our content $x^{(i)}$ is a legal chess position from a game played by humans, and our annotation $y^{(i)}$ is the outcome of that game (a win or loss by the player- to-move). We are trying to predict the expected value of the outcome of the game given the position."

The logistic regression algorithm in Christopher De Sa's work was implemented by receiving 8x8x6x2 matrices as inputs (8x8 indicating board position, 6 pieces and 2 color) and outputting a value telling the probability of winning the game when selecting some chess position to move to. The parameters of the logistic regression model were computed using the Newton-Raphson method. Because it dealt with large data amounts, it used sparse matrices whenever possible and avoided the step to inverse a Hessian matrix by using the conjugate gradient method on the linear system $Hs = g$ (where H is the logistic Hessian, s computed step and g the logistic gradient). For further details about the implementation of the logistic regression, please check out [2].

2.6.1.2 Convolutional neural networks

In 2015, Barak Oshri and Nishith Khandwala (former Stanford University students) implemented a convolutional neural network that was used for predicting moves in chess [3]. The convolutional neural network was implemented by using three layers and the ReLU (rectified linear unit) activation function. The convolutional neural network receives inputs with a dimension of 8x8x6 (which will be explained in *3 Method*) and then outputs 8x8 matrix, where each element indicates the probability. This probability could symbolically mean different things depending on what is predicted (this will be brought up in the next coming paragraph). The prediction is done by selecting the position that has the highest probability. For further technical details, please check out [3].

To predict a whole move, the authors have used seven copies of the same convolutional neural network to predict different details that concerned with a move in chess. They assumed that a move was composed by the position the next piece moves from and the position, to which the same pieces moves. One of those copies predicted the position, from which the next piece is most likely to move from. Then one of those six remaining copies was selected depending on what kind of piece probably was moved (since different pieces can move differently) and thus, predicted the new position of the piece being moved.

For training the CNN algorithm, they used 245,000 moves made by players with an ELO rating (explained in Appendix B) higher than 2,000 from FICS. The test data had a size of 20 % of the training data when they performed cross-validation. According to their study, they have received an accuracy of 38.3 % with the same prediction task as specified for this study.

2.6.2 Reinforcement learning

The latest study about application of reinforcement learning in chess is about a chess engine called Giraffe that utilizes deep reinforcement learning [4]. Giraffe uses Temporal-Difference learning (explained in Appendix B) along with neural networks for storing values related to chess play strategy. The prediction task of Giraffe was to predict the best possible move that could be done in a board state, i.e. predict the move that increases the chance of winning the chess game. According to the same study, the best accuracy of Giraffe obtained was 46 % for all different type of moves. Further details about Giraffe were that it was trained on five million games after 72 hours and that it played on the same level as a FIDE International Master, with ELO rating of approximately 2,200-2,500. This is the only academically documented accuracy of a chess engine from a study concerning Temporal-Difference learning.

Commercial chess engines predict better than 46 %, since e.g. KnightCap that makes use of Temporal-Difference learning as well, reaches a rating of slightly above 2,500 in ELO rating [14]. This rating is above the FIDE International Master rating. Therefore, the prediction accuracy could be assumed to be higher than 46 % among the best commercial chess engines that use Temporal-Difference learning in some way.

3 Method

The format of the study will be described in this section. This section intends to give an overview of all the technical details that explains how the study is conducted with respect to the chess game datasets, how to interpret the results of the study as well as what software will be used in this study.

3.1 Choice of learning algorithms

Logistic regression was used and tested with the motivation that:

1) There were no other choices of supervised learning algorithms being documented academically.
2) It could be by common interest to examine how a simple machine learning algorithm like logistic regression works for more complex prediction tasks.

However, when performing the study, the implementation of the logistic regression will be the one from SciKit-Learn since no complete implementation that corresponds to the one that Christopher De Sa used was available as open source that was compatible with the requirements of being used within this study.

As far as it concerns convolutional neural networks, due to the interest of elaborating on the CNN in chess further with respect to other aspects such as execution time and overfitting tendencies, while there were no other studies besides this one about supervised learning in chess (except the one about logistic regression), CNN was chosen to be applied within this study. Worth notifying: There are several different topologies that describe a convolutional neural network with varying parameters. However, for this study, the topology and parameters around the convolutional neural network are determined by the research work done by Barak Oshri and Nishith Khandwala in how convolutional neural networks can be used for predicting the moves of chess pieces.

3.2 Dataset and the format of data samples

The database of the FICS (Free Internet Chess Server) has chess games recorded from 1999-2017, accumulating data containing chess games played in each month during a year. The dataset used for the study is all the chess games played in December 2016 that have been recorded by FICS (Free Internet Chess Server) [15] from chess games that were played on their website. Only a month of chess games played was chosen due to the limitations on the software being used, which will be discussed in a latter subsection of this section (due to inefficient parsing, the PGN (explained in Appendix B) file content had to be shortened by approximately half to parse it). This dataset is acquired from the FICS website as a PGN file, where it has been filtered to retrieve standard games with no constraints on ELO ratings and no move times (how much seconds has elapsed before making a move on FICS by the player) included. Worth noting is that the choice of dataset consisting of 1 month of chess games, is believed to not be significant for this study, since the discrepancies between prediction accuracies for different datasets are not great. For further information, please check out *3.4 The structure of the study*.

Each input to the machine learning algorithms will be a chess board state that is modelled as an 8x8x6 matrix, where 6 denotes a certain chess piece and 8x8 the number of squares on the chess board. Each feature in this matrix will be denoted with one of the discrete values in $\{-1,0,1\}$, where -1 denotes a square occupied by the enemy, 0 an empty square and 1 a square occupied by yourself. This

representation could easily be applied to both the black players and the white players by changing the sign of the values in the input. An example of such input is illustrated in figure X below:

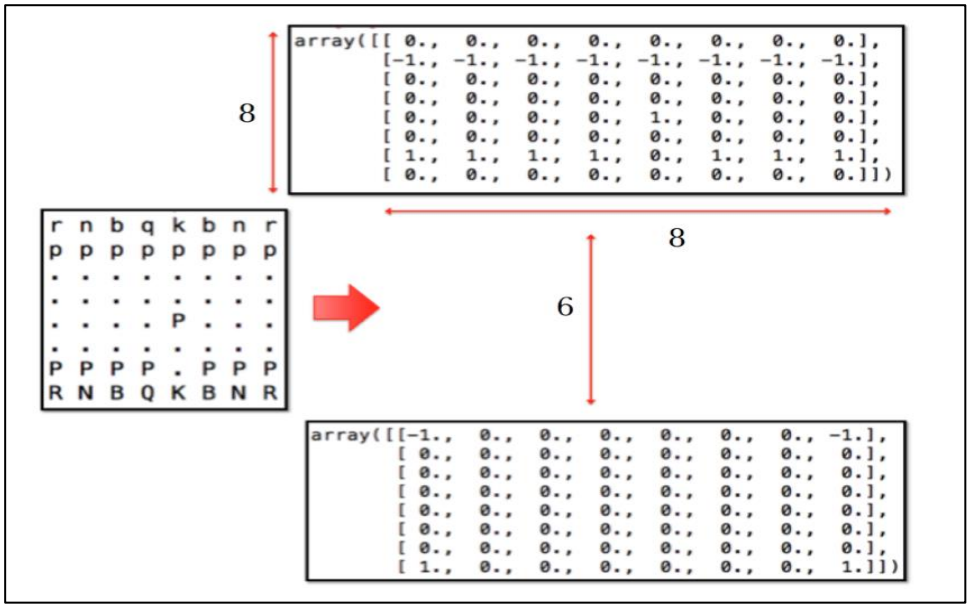

Figure 5: How a chess board is converted to the specified input. Source: Oshri/Khandwala [3]

The idea of this data representation was inspired by the work of Oshri and Khandwala regarding applying CNN in chess [3]. The reason for the choice of this representation was because the implementation of CNN was rigorously based on receiving inputs with this representation. Any changes to the representation would lead to major changes in the implementation of the CNN, which could be too complicated to deal with. In long term, this could lead to unexpected behavior of the model due to lack of deep understanding of the implementation.

In the case of logistic regression, the same data representation used for the CNN model was used for logistic regression without greater changes. The only requirement for using this data representation for the logistic regression was that the data representation gets flattened, i.e. transformed from the 6x8x8 matrix into a 384-dimensional vector.

Furthermore, the output as function of the input described in the previous paragraph is a discrete value $y \in [0,63]$, where y is the position index in the chess board from which the next piece is to be moved. This output format applies to both the CNN model and the logistic regression model.

When one would like to predict the position index based on simple random guessing, the accuracy of randomly predicting the correct position index on the chess board would be 1/16=0.0625 and when approaching to the latter phases of a chess game, fewer pieces will likely be left and thus higher probability to select the correct position, from which the next piece is about to be moved. To know the probability of picking randomly the correct start position at round i, it was estimated by following formula:

$$\Pr(i \mid X_i) = \frac{\sum_{x \in X_i} \frac{1}{|x|}}{|X_i|}$$

where X_i a set of board states at move round i, $x \in X_i$ a board state and $|x|$ the number of pieces left that could be moved by the player in its turn in a board state. The board states are retrieved from the dataset specified in the beginning of this section.

This number could be used as a lower threshold for comparison, to see if each machine learning algorithm perform better than this threshold in terms of accuracy. Otherwise, the practical use of the machine learning algorithms will be rendered trivial.

3.3 Metrics

For this study, the following metrics will be used in the numerical study:

- **Accuracy** - The accuracy of a machine learning algorithm will indicate how well the machine learning algorithm predicts on arbitrary input. The range of accuracy will be $[0,1]$, where 0 indicates that the machine learning algorithm do not predict correctly for any input in a dataset and 1 indicates that the machine learning algorithm always predicts correctly for all inputs in a dataset. Provided N samples and $N_{correct}$ number of samples being predicted correctly by some machine learning algorithm, the accuracy A is calculated through the formula $A = N_{correct}/N$.

- **Degree of overfitting** - To which degree a machine learning algorithm will overfit indicates whether the same machine learning algorithm makes any difference on the training data, which it is trained on, and other data (not included among other samples in training data) when predicting. Mathematically, the degree of overfitting d for a machine learning algorithm provided a dataset is to be defined as $d = |A_{test} - A_{train}|$, where A_{test} is the accuracy of the machine learning algorithm on test data and A_{train} the prediction accuracy on training data. When the degree of overfitting d increases, the more machine learning algorithm tends to overfit its own training data in relation to other non-training data.

- **Execution time** - For this study, the execution time will be a sum of the time of initializing each machine learning algorithm models, the time of training each machine learning model and the time of performing validation test for respective machine learning model. The reason is because one of the implementations both trained and tested its accuracy simultaneously and since these two processes could not be separated from a point of implementation view, the execution time measured for each machine learning model will instead measure the total amount of time required to both initialize models, train the models and test the models.

3.4 The structure of the study

For the study, there are three parameters that will be used and varied when all the machine learning algorithms are being tested. The parameters are as following:

- **Number of games** - how many games out of those provided in the dataset are to be used. The number of games will range between 1,000 and 10,000 games and will be gradually increased by 1,000 each time. 10,000 games were chosen due to limitations on the memory of the computers being used. The default value here will be 5,000 games when separately varying the other two parameters, move round and fraction. *NOTE*: This is not the actual number of samples. Samples in this study are each input-output pair binding whose format is specified in the beginning of this section. To get an estimation of how many samples n_{moves} were dealt at some occasion, then use the following equation: $n_{moves} = 2 * n_{rounds} * n_{games}$, where n_{rounds} is the number of move rounds and n_{games} is the number of games.
- **Move round** - The move round parameter specifies how many rounds to include maximally when building the models, starting from round 1 to the number the move round specify. For instance, if the move round is 4 that means that the models will be trained with respect to round 1, 2, 3 and 4 in a chess game. Each move round consists of the move that the white player and the black player made at that round. However, in this study, the models will be trained starting with 4 move rounds and gradually increasing the move round parameter with 4 until 40 is reached. This means that the move rounds will go from 4, 8, 12, … up to 40. The maximal boundary 40 was chosen since statistically, the average number of move rounds in a chess game is 40 move rounds [16]. The default value which will be used when separately varying the other two parameters, number of games and fraction, will be 20, since it is approximately the middle phase of a chess game.
- **Fraction** - provided all the moves acquired from the data set, the share of all moves to be used as training data. The fraction will range between 0.1 and 0.9 and will be gradually increased with 0.1 each time. The default value here is 0.5.

The first task was to extract the moves from the dataset as PGN file and providing the three parameters, number of games, move round and fraction. After parsing the PGN file and retrieving both the inputs and associated outputs as specified in *3.2 Dataset and the format of data samples*, the machine learning models were trained and tested before outputting the accuracies and execution time for respective machine learning model.

In this study, holdout was performed instead of k-fold cross-validation. The main reason to this is due to high time consumption caused by details such as parsing data, processing, initializing, training and testing, which made holdout a better candidate as it in general runs faster than k-fold cross validation.

When the machine learning models are examined through its accuracies as function of e.g. number of games with its specified range and gradual increase, the other parameters will have their default values as specified earlier in this subsection. This is to reduce possible side effects from other parameter settings affecting the true picture of accuracies in each machine learning model. This applies to all other parameters.

There is one thing to note: the logistic regression model does not contain any randomness when being trained, which means that for given settings accordingly to the parameters above, the accuracies of the logistic regression model will be deterministic. Meanwhile, the CNN model does rely on some randomness being involved in its calculations when being trained, which implies that the acquired accuracies after running the tests with the CNN model once are not deterministic. To get a good picture

of how the randomness may affect the accuracies of the CNN model, 5 example program executions will be run and an arithmetic average value will be estimated on each accuracy for the CNN model. Furthermore, the standard deviation value will be derived from the previous calculations to construct a confidence interval with 95 % as confidence level, which is to be illustrated in the results section. The same technique is also to be used when it comes to estimating the execution time of each machine learning algorithm given a determined setting since the execution times are also not deterministic and hence should be estimated just like the accuracies of the CNN model.

3.5 Software

The programming within this study was done in a Python 2.7 IDE (for this study, Spyder). This Python 2.7 IDE was installed by installing Anaconda platform with prebuilt libraries for scientific computation in Python programming like NumPy, SciKit-Learn and other standardized libraries for scientific computation.

Logistic regression was implemented by using SciKit-Learn library that implemented a logistic regression for general purpose and in this study specialized for predicting pieces to be moved in chess games.

The CNN implementation was retrieved as open-source [3] related to the work done by Oshri and Khandwala. In addition, not only the whole CNN implementation was retrieved, but also tools for parsing the PGN files into inputs and outputs in the specified data representation as described in 3.2 were used.

Further details about the code project and usage of this code project could be acquired by following the link to the Git repository related to this study in Appendix A.

4 Results

The results that were gathered from the tests for each machine learning algorithm will be presented. This section is ordered in the following fashion: the fraction between the training- and test data, the move rounds in the chess parties and the number of games used from the dataset for each machine learning algorithm when describing the accuracies as function of each of these parameters separately.

4.1 Fraction of datasets between the training data and test data

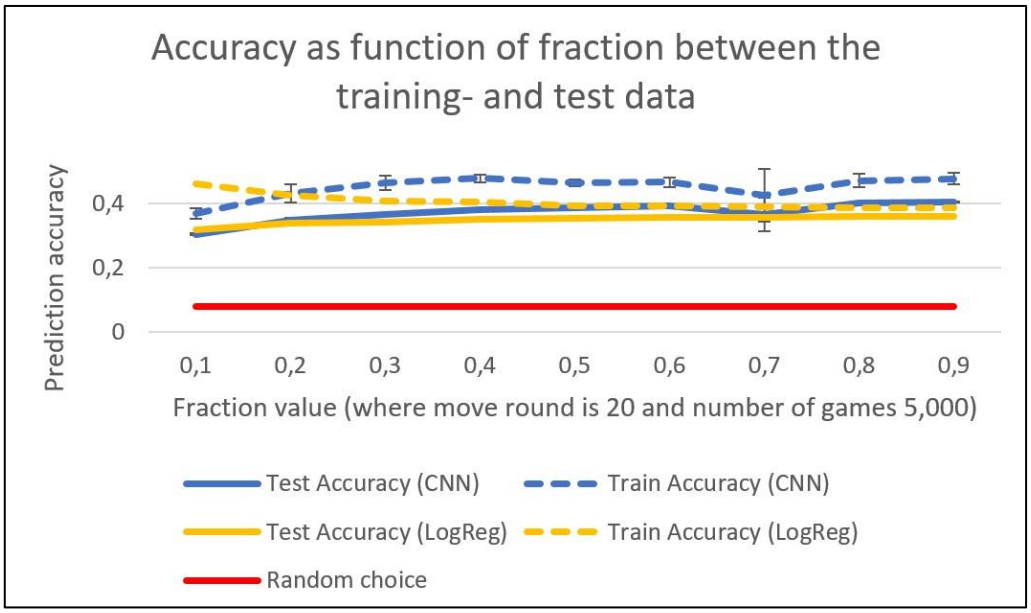

Figure 6: Accuracy based on the fraction being made between the training data and test data

As seen in the above chart, the test accuracies of the CNN model are stable for all fraction values. However, the train accuracy of the CNN model is not as stable as the corresponding test accuracies, since the confidence interval (vertical black lines with ends) of the train accuracy is significantly greater than the confidence interval of the test accuracy for all different fraction values. Furthermore, it is also possible to observe, by inspecting the chart above, that the difference between both the accuracies, the training and the test, of the CNN model along the x-axis is relatively constant and have almost the same shape when inspecting the chart from a visual perspective. The sharp decrease done between 0.6 and 0.7 in fraction value accordingly to the chart will be subject to discussion in the discussion section.

When it comes to the logistic regression model, there are visible differences between the train- and test accuracy when the fraction value is low, such as 0.1 in the chart above. When the fraction value increases, the differences between these accuracies decreases significantly, while the test accuracy is increasing and the train accuracy decreasing, and to a point where both accuracies approach each specific value when considering fraction values bigger than 0.8.

As the fraction value gradually increases so does the accuracy. As seen in the above chart, the test accuracy of CNN model increases slightly higher than the corresponding test accuracy of the logistic regression model.

4.2 Move rounds in chess parties

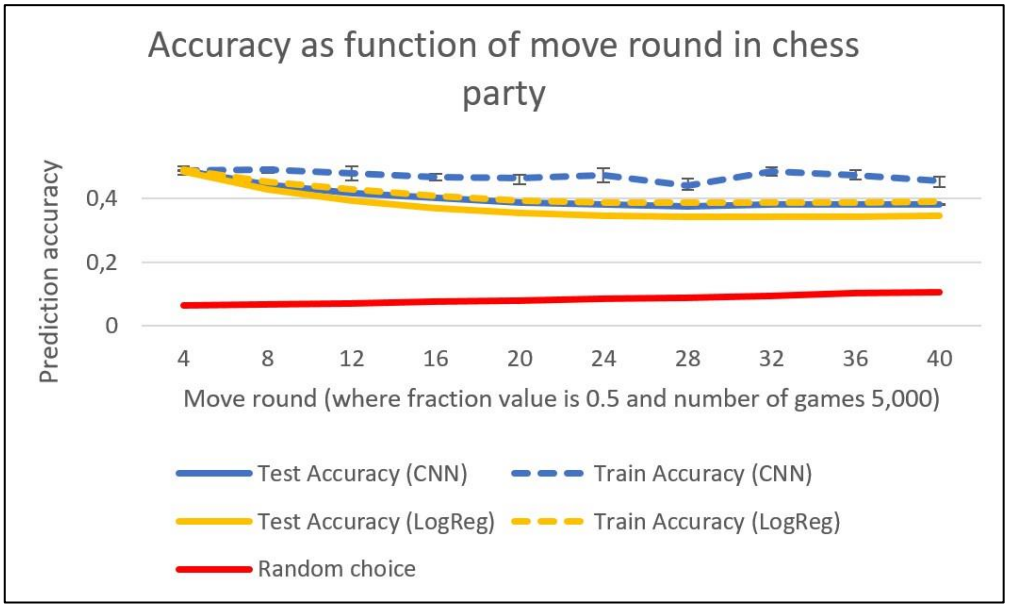

Figure 7: Accuracy based on the move round in each chess party being analyzed

The patterns for both models, when learning with more latter moves and making predictions based on it, are quite similar: Both predicts very well when only learning from the first four rounds, where both predict as nearly as 50 % with minimal differences between their test accuracy and their train accuracy. As the move rounds increases, both the models predict worse and increase the differences between the test- and train accuracy for respective model. Despite that the overall patterns are identical, there are some details that are different. The first detail is that the difference between the test- and train accuracy is greater for the CNN model in comparison to the corresponding difference between the test- and train accuracy for the logistic regression model, which the chart above illustrates. The second detail is that there is some sharp decrease for the train accuracy of the CNN model between the move rounds 24 and 28 before, again sharply, the train accuracy increases between move rounds 24 and 28, while the curves of both accuracies of the logistic regression remain soft along the x-axis in the chart. This phenomenon will be explained in upcoming discussion section.

Just like in the previous subsection, the confidence interval of the train accuracies is significantly greater than the corresponding confidence interval of the test accuracies of the CNN model.

4.3 Number of games used from the dataset

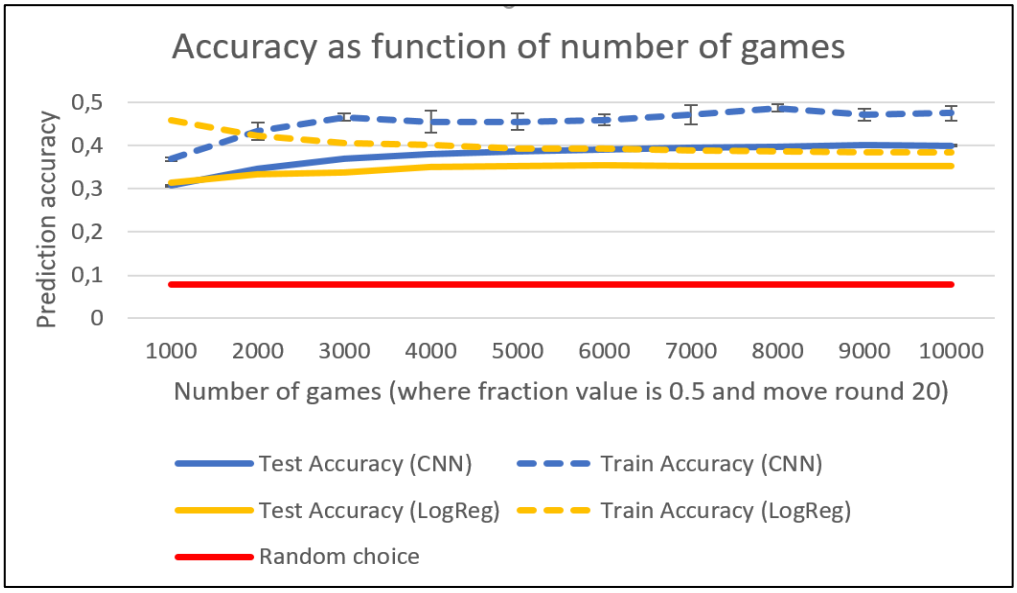

Figure 8: Accuracy based on the number of games being used

When it comes to the CNN model, the test accuracy steadily increases before it becomes more insignificant as long as the number of games increase (probably approaching some limit value), while the train accuracy interchangeably increases and decreases at some occasions as illustrated in the chart above. Between 1,000 and 3,000 games, the train accuracy of the CNN model increased steadily before a sudden decrease occur between 3,000 and 4,000 games. Between 4,000 games and 8,000 games, the train accuracy of the CNN model increases before it decreases between 8,000 games and 9,000 games.

The logistic regression model has a different pattern: when the number of games is small (such as 1,000 games as shown in the chart above), the difference between the train- and test accuracy is great. However, when the number of games is increased, the difference between the accuracies decreases, where the test accuracy increases between 1,000 and 6,000 games slightly and then maintains its level in practice after 6,000 games, while the corresponding train accuracy decreases as long as the number of games is increased.

4.4 Execution time of initializing, training, and testing machine learning models altogether

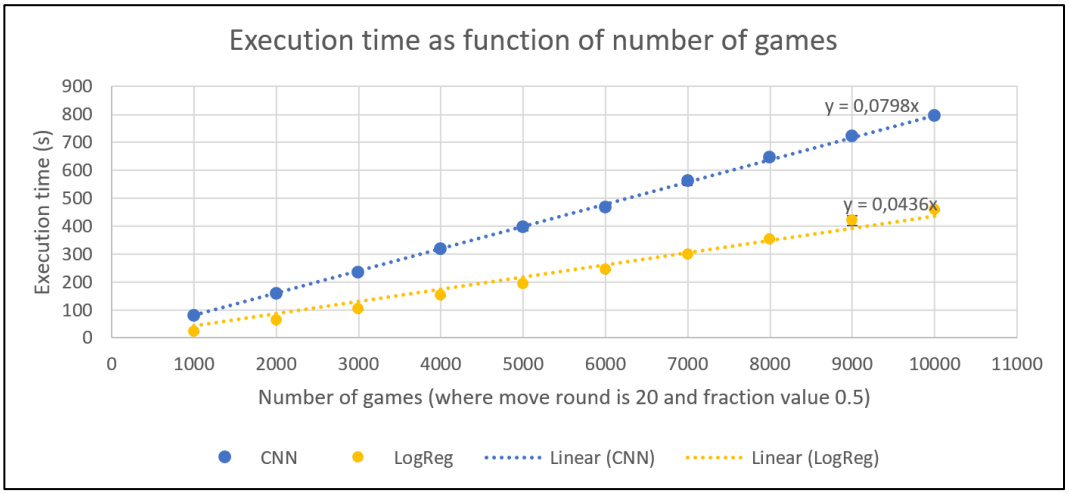

Figure 9: Execution time based on the number of games being used

As the chart above illustrate, it takes longer both for CNN to be initialized, trained and tested compared to the logistic regression and as the number of games increases, the more time the CNN model needs to be trained. This could be shown by comparing the slopes of each model's trend lines as shown in the chart. The slopes of the trend lines indicate that the execution time of CNN model increases 0.0798/0.0436≈1.83, i.e. almost two times faster as the corresponding execution time of the logistic regression when the number of games is increased. This can also be interpreted as two times slower for the execution time of the CNN model compared to the logistic regression model.

As further observed, the confidence interval in the chart for each execution time are small (visually, they are barely to be seen) and it suggests that the confidence interval for all execution times being inspected in the chart are small, showing that the runtimes were not varying to a high degree.

However, there were limitations in how much time all testing was done prior to the parsing stage and the processing stage as well. According to obtained estimations and provided the software and dataset that were used for this study, it took in average approx. 33 seconds to accomplish the parsing stage (i.e. moving the data from the PGN file into the Python IDE). At the processing stage (where parts of the data are selected with respect to number of move rounds to examine maximally in each chess game and the number of chess games and, finally, converting the selected data into an input-output format as specified in section *3 Method*), the time for this stage in seconds could be approximately described by the multilinear function $T(n, m) = 0.59n + 0.005m$ (this estimation can be found on the Excel sheet uploaded in the Git repository of this study. Access the Git repository by clicking on the link in Appendix A), where n is the number of move rounds and m the number of games used. This clarifies the time conditions that were dealt with during the study regarding the processing stage.

5 Discussion

In this section, the results gathered and presented in the previous section will be interpreted, analyzed and put in relation to a general context regarding chess programming today and in the future. This section is to give details and deeper explanations that has led to the conclusions in this study. The interpretation of the result will explain the different phenomenon that came up among the results by fundamental machine learning terminology to clarify the whole picture of the results gathered in this study. Afterwards, the results will be analyzed and reflected before putting these results into a greater context of chess programming.

5.1 Interpretation of results

As mentioned at some points in the previous section, there were different phenomenon that occurred that could not be explicitly explained only by looking at the charts that were presented. The following phenomena have risen in the previous section, for which an explanation, by the help of fundamental machine learning terminology from the background, will be provided:

- **Sudden decrease in both the train accuracy and the test accuracy of the CNN model** - In the first chart in the results section, a visually sharp decrease occurred for the test accuracy and the train accuracy in CNN model at the same time. A possible explanation is that one of the data points led to both poor train accuracy and poor test accuracy, which in turn is a consequence of gradient calculations that caused the weights in the CNN model to converge to some local minima and remained there. This is a frequent problem for neural networks in general [17]. This could also explain the great confidence interval both the train accuracy and test accuracy had that could be observed in the chart. It had to do with the way the weights in the CNN model were initialized in an unfortunate way that has led to convergence to some local minima when being iteratively updated. However, it could also be seen that when running the same program on the same conditions (i.e. same parameter settings) several times, this problem does not occur for the CNN model in other iterations.

- **Increasing differences between the train accuracy and the test accuracy for each machine learning algorithm along move round** - It has been noted that when moving forward along the maximal number of move rounds in the chess games that the differences between the train accuracy and the test accuracy are growing for each machine learning algorithm. This phenomenon is caused by the fact that the latter board states in the chess games are to be considered as outliers compared to the early move rounds that contained many similar board states (because not so many moves have been done) and thus could not predict well on these board states. When it comes to the board states in later move rounds of each chess game, that are in most cases completely different to the early common board states, both the machine learning algorithms will not predict as well as for the early move rounds, which the chart regarding accuracy as function of move rounds has clearly shown.

5.2 Analysis of results

The prediction task in Christopher De Sa's work is easier than the prediction task specified in this study. A reason for this is that in a board state, many legal chess positions could lead to a win in long term and thus, the correct prediction that a chess position amongst others (especially in the beginning phase) lead to either win or loss is easier to be made. The only outcomes for each prediction in the work of Christopher De Sa was either that a legal chess position could lead to a win or loss, which corresponds to 1/2 in probability if a legal chess position was to be predicted randomly. The probability that the position index in this study is correctly predicted (if only the legal moves are dealt) is 1/16 if being predicted randomly. It was believed that the logistic regression would not be able to compete with CNN with respect to prediction accuracy, since the prediction task in this study is more complex than the one in Christopher De Sa's work. It was therefore unexpected that the logistic regression model gained accuracy levels that could be comparable to the corresponding accuracies of the CNN model, since it was believed before the study that the logistic regression would not be able to compete with the CNN in terms of prediction accuracy. It is the CNN model that tends to mostly overfit its own data since the differences between both the accuracies are greater for the CNN model than the corresponding differences for the logistic regression in general. However, the overfitting that occur to both the machine learning algorithms (especially CNN) makes it hard to tell what the true prediction accuracies of each machine learning algorithm could be. Therefore, the true prediction accuracies of both the models could possibly be either more similar or more different than what the charts in *Results* show.

In this study, the CNN model did nearly perform as well as indicated in the study by Oshri and Khandwala, since the accuracies obtained in this study was in overall like those accuracies that were obtained by the authors. Therefore, the accuracy levels of the CNN model within this study were retained in relation to the same accuracies obtained by the authors in their work with convolutional neural networks in chess. However, the problems with potential overfitting may cause long term problems (such as overrated prediction accuracy) when using the CNN model more extensively. This potential of overfitting could be attributed to the fact that almost no regularizations were applied to the CNN, as stated explicitly by Oshri and Khandwala in their study.

During the study, one interesting detail was observed: when both the machine learning algorithms did not take into consideration the past moves being made in each chess game, the roles between them were opposite. The logistic regression predicted more properly than the CNN generally, but suffered more from overfitting compared to the CNN model. However, the overall prediction performance of each machine learning model in this study was decreased significantly (maximal prediction accuracy was below 30 % for both machine learning models). From this observation, it was concluded that to achieve better prediction accuracy for a machine learning model, past moves in the same chess games need to be considered because it is believed that it could give both machine learning models more information about how the different properties of each chess board state could deduce the start position of the next piece to be moved. The fewer machine learning models know about these properties of a chess board state, the more unstable the predictions will be.

It is difficult to state which of the machine learning algorithms truly is the better one with respect to prediction accuracies, mainly due to overfitting tendencies that contribute to a more inaccurate picture of the ‘‘true’’ prediction accuracy of each machine learning algorithm. However, the first chart presented in the *Results* reveals that when only using a limited amount of training data for predicting on greater amounts of non-training data, both the CNN model and the logistic regression model

correctly predicted almost equally, but the logistic regression model suffered to much larger extent by overfitting, especially when the fraction value was 0.1.

The reason for investigating the prediction accuracies as function of the fraction value was to observe how each machine learning algorithm behave when only using a limited amount of data samples and still must cope with predicting on unknown data, when the amount of unknown data is significantly larger when compared to the training data being used. This is believed to be the most realistic scenario when developing chess engines with learning algorithms due to different implementation constraints such as time consumption and memory management as well on the software being used. According to Eric Holcomb, there are approx. $4*10^{40}$ legal board states in chess [6]. Hence, the property to predict properly on arbitrary data using limited amount of training data is important to any machine learning algorithm. In this aspect, the CNN model outperforms the logistic regression model since the CNN model does not overfit as the logistic regression model when being trained on small training data.

Given the gathered results from this study, both the CNN model and the logistic regression model are better choices than random selection in most of the cases. However, compared to contemporary chess engines, these supervised learning algorithms do not perform as well as e.g. Giraffe or other chess engines using reinforcement learning such as KnightCap, despite simplified prediction task. Therefore, chess engines that are based entirely upon one of the machine learning algorithms used in this study will not work. It could also be said that the CNN implementation is slower than e.g. Giraffe with its 5,000,000*0.0798/3,600≈111 hours duration to be initialized and trained compared to Giraffe with its 72 hours when being trained on 5,000,000 games. However, a possible source of error in this claim would be software/hardware conditions, during which the training is done. It is not revealed in the study of Giraffe what software/hardware was used for training. Additionally, all the execution times being estimated in this study are based on running the tests on one computer. The estimations could give other results on another computer with different software/hardware conditions. Furthermore, it is still unknown if the specified topology of the CNN is the optimal one. Further research about the topology of CNN in chess must be conducted to figure out the best settings for the CNN in chess. When it comes to the logistic regression: although it predicts worse than Giraffe on simpler prediction task, it is faster with 61 hours (since 5,000,000*0.0436/3,600≈61) compared to Giraffe's 72 hours. This could indicate that logistic regression could be applied as heuristic in the meaning that it could speed up series of calculations within complex predictive systems or in other ways that could lead to overall speed improvements of greater predictive systems.

Ultimately, it is believed that the CNN has better prospects than logistic regression, since there is still much more to research about the CNN compared to the logistic regression. Even though the same thing could be said about the logistic regression, that it also could undergo more thorough research by primarily implementing a more complex algorithm, the nature of logistic regression is less complex than CNN. It is therefore believed that CNN possibly has higher potential to reach for better accuracies in comparison to the logistic regression.

5.3 Future research

This study with its empirical studies of the supervised learning algorithms relied on parameter settings and/or topologies that were proven to either be the most efficient (such as in the case of the CNN implementation) or simply were default (such as in the case of the logistic regression, whose implementation was imported from SciKit-Learn and used it in its default mode). The results gathered within this study could have looked differently if the performance requirements on the used software were not too constraining.

Therefore, it is strongly recommended to try out different settings and/or topologies on each machine learning algorithm introduced in this study and find out more efficient variants of each machine learning algorithm that could lead to better prediction accuracy in overall. This also includes the testing methodology around this study. If more time is given one could e.g. apply k-fold cross-validation instead of the holdout method that has been applied in this study to verify the results from this study with more robust testing methodologies and see if the results hold and, consequently, conclusions as well. The code project associated with this study can be acquired by checking out Appendix A, where a link to the Git repository is provided to access the code project.

As noted in the previous subsection, there are some problems with overfitting, especially with the CNN model. It is thus recommended to investigate sophisticated regularization techniques especially suited for the CNN model. Also, it is recommended to elaborate on corresponding regularization techniques for the logistic regression model to be able to generalize better. Elaboration on the topology of the CNN in chess programming would be welcomed as this may give more insight in how to improve the prediction of neural networks in general and come closer to a more sophisticated topology that could give much more stable and accurate predictions than e.g. the convolutional neural networks that have been applied in this study.

One should investigate further how different data representations could affect the prediction accuracy of an arbitrary machine learning algorithm in the chess programming field to gain more knowledge on how data should be presented to model the chess games as best as possible. In this study, the data representation was taken from the same one that was present in the study about CNN in chess, but it was not elaborated on how the results would look like if another data representation was to be used instead.

6 Concluding remarks

After the logistic regression and convolutional neural network were compared to existing chess engines, the main conclusion was that they do not perform well in terms of accuracy and time execution of the training. However, it is possible that these mentioned supervised learning algorithms, especially logistic regression, could be used as heuristics in combination with other complex machine learning algorithms to speed up calculations when doing predictions in different contexts. On the other side, the CNN does have better prospects than logistic regression in achieving higher accuracies if its properties would be more extensively researched.

As mentioned in the previous section, it is encouraged to elaborate further on how different data representations could affect different machine learning algorithms when used for predicting in chess games and consider other settings that were not considered in this study. It was suggested to investigate possible improvements that could have been done on each machine learning algorithm, such as implementing new regularization techniques to solve problems related to overfitting. Finally, it is also suggested that a more robust cross-validation than holdout method is performed to test the results and conclusions in this study.

7 References

[1] J. Schaeffer, Long-range planning in computer chess, New York: ACM, 1983.

[2] C. De Sa, "Classifying Chess Positions," 14 December 2012. [Online]. Available: http://cs229.stanford.edu/proj2012/DeSa-ClassifyingChessPositions.pdf. [Accessed 2 May 2017].

[3] B. Oshri and N. Khandwala, "Predicting Moves In Chess Using Convolutional Neural Networks," 2015. [Online]. Available: http://cs231n.stanford.edu/reports/2015/pdfs/ConvChess.pdf. [Accessed 2 May 2017].

[4] M. Lai, "Giraffe: Using Deep Reinforcement Learning to Play Chess," 14 September 2015. [Online]. Available: https://arxiv.org/pdf/1509.01549.pdf. [Accessed 2 May 2017].

[5] Chess.com, "How To Play Chess: Rules and Basics," [Online]. Available: https://www.chess.com/learn-how-to-play-chess. [Accessed 1 June 2017].

[6] E. Holcomb, "So How Many Chess Board Positions Are There?," 16 January 2012. [Online]. Available: http://www.nwchess.com/articles/misc/Chess_Board_Positions_article.pdf. [Accessed 12 May 2017].

[7] S. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," 16 July 2007. [Online]. Available: http://www.informatica.si/index.php/informatica/article/viewFile/148/140. [Accessed 2 May 2017].

[8] G. James, D. Witten, T. Hastie and R. Tibshirani, An Introduction to Statistical Learning with Applications in R, New York: Springer, 2013.

[9] S.-C. Wang, Artificial Neural Network, New York: Kluwer Academic Publishers, 2003.

[10] Stanford University, "Convolutional Neural Network," [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/. [Accessed 12 May 2017].

[11] Wikipedia, "Cross-validation (Statistics)," 17 April 2017. [Online]. Available: https://en.wikipedia.org/wiki/Cross-validation_(statistics). [Accessed 12 May 2017].

[12] J. Schneider, "Cross Validation," 7 February 1997. [Online]. Available: https://www.cs.cmu.edu/~schneide/tut5/node42.html. [Accessed 5 May 2017].

[13] S. Yadav and S. Shukla, "Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification," IEEE, 2016. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7544814. [Accessed 2 June 2017].

[14] J. Baxter, A. Tridgell and L. Weaver, "Learning To Play Chess Using Temporal Differences," 19 January 2001. [Online]. Available:

https://www.cs.princeton.edu/courses/archive/fall06/cos402/papers/chess-RL.pdf. [Accessed 2 May 2017].

[15] FICS, "FICS Games Database - Download," [Online]. Available: http://ficsgames.org/download.html. [Accessed 2 May 2017].

[16] ChessGames, "Statistics Page," [Online]. Available: http://www.chessgames.com/chessstats.html. [Accessed 2 May 2017].

[17] D. Kriesel, "A Brief Introduction to Neural Networks," 2005. [Online]. Available: http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-1col-dkrieselcom.pdf. [Accessed 2 May 2017].

[18] Wikipedia, "Reinforcement learning," 15 April 2017. [Online]. Available: https://en.wikipedia.org/wiki/Reinforcement_learning. [Accessed 2 May 2017].

[19] M. Martin, "Reinforcement Learning: Searching for optimal policies," 2011. [Online]. Available: http://www.lsi.upc.es/~mmartin/Ag4-4x.pdf. [Accessed 2 May 2017].

[20] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," 2012. [Online]. Available: http://people.inf.elte.hu/lorincz/Files/RL_2006/SuttonBook.pdf. [Accessed 5 May 2017].

[21] Saremba, "Standard: Portable Game Notation and Implementation Guide," [Online]. Available: http://www.saremba.de/chessgml/standards/pgn/pgn-complete.htm#c8.1. [Accessed 2 May 2017].

[22] Wikipedia, "Portable Game Notation," 21 April 2017. [Online]. Available: https://en.wikipedia.org/wiki/Portable_Game_Notation. [Accessed 2 June 2017].

[23] Wikipedia, "Elo rating system," 19 April 2017. [Online]. Available: https://en.wikipedia.org/wiki/Elo_rating_system. [Accessed 9 May 2017].

[24] Wikipedia, "Chess rating system: Elo rating system," 13 February 2017. [Online]. Available: https://en.wikipedia.org/wiki/Chess_rating_system#Elo_rating_system. [Accessed 9 May 2017].

Appendix

Here will other materials that were not crucial for this study, but still published to give the reader a possibility to deepen in some of the theories being provided in the study and check out the code project that this study relied upon for obtaining the results being introduced in this study. Furthermore, the reader could also check the numbers that are underlying all the charts in this study by checking out the corresponding Excel file with both numbers and visual charts included alongside with the code project.

Appendix A – Git repository of this study

The link to the Git repository that is related to this study could be found precisely down below:

https://gits-15.sys.kth.se/syllil/Degree-Project---Supervised-Learning-Chess

Appendix B – Further readings

This subsection contains further material for reading regarding some terms that were used in this study, but still did not have any vital impact on the results and the conclusions derived from the obtained results.

Reinforcement learning

The input data is provided in the same way as for the supervised learning. However, the output associated with each input is unknown. Instead, there is a reward value that the machine learning algorithm tend to maximize for each action in a certain state and later works as an indicator for which action is to be taken in that state, given a policy. Formulated mathematically: the principle of the reinforcement learning could be modelled by two so called different value functions under a policy π. The first one is a state-value function, i.e. $V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\}$, and the second one is an action-value function $Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\}$, where $R_t = \sum_{k=0}^{inf} \gamma^k r_{t+k+1}$, γ the discount factor and r_t the reward value at time step t [18] [19]. The objective of the reinforcement learning is to take an action in a state that maximizes the reward given a policy and the current value functions as defined recently.

In the context of chess programming, the most known machine learning algorithm utilizing reinforcement learning is Temporal-Difference Learning, which will be covered in the upcoming section.

Temporal-Difference learning

Temporal-Difference learning is a machine learning algorithm that is based on a combination of Monte Carlo algorithm (can learn from data without modelling dynamics of any environment) and dynamic programming (estimations is learned on earlier estimations without knowing the outcome). In Temporal-Difference learning, it is about to get an estimate v of state-value v^{π} given a policy π for nonterminal states S_t at time step t. This estimate therefore will be used in estimating v of v^{π} at time step $t+1$ and so forth. The mathematical notation for describing this iterative estimate evaluation is $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$, where $V(S_t)$ is the state value of nonterminal states S_t, α some constant and other symbols with the same definition as previously stated. To implement Temporal-Difference learning, it is usual in this context to apply TD(λ)-algorithm which is a learning algorithm invented by Richard S. Sutton [20].

PGN

Portable game notation, abbreviated as PGN, is a computer format that consists of plain text [21] and is used to easily read and record chess games and scores. Files with the file extension ".pgn" conforms to the standard given by PGN. Given below is an example of a PGN file:

```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 {This opening is called the Ruy Lopez.}
4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3 O-O 9. h3 Nb8 10. d4 Nbd7
11. c4 c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15. Nb1 h6 16. Bh4 c5 17. dxe5
Nxe4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21. Nc4 Nxc4 22. Bxc4 Nb6
23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nxf7 Rxe1+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5
hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. f3 Bc8 34. Kf2 Bf5
35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5 40. Rd6 Kc5 41. Ra6
Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```

Figure 8: Chess party described with PGN syntax. Source: English Wikipedia article on PGN

The content of a PGN file is divided into two separate parts. The first part, consisting of characters enclosed in [], is called tag pairs. The tag pairs consist of a tag name in plain text followed by a value, which is enclosed in quotation mark as "", and acts as the value for that tag pair. When archiving PGN files, 7 tag pairs must be provided and in the right order as shown in the picture above. More tag pairs can be added in addition to the 7 mandatory ones mentioned in the picture. Some of these tag pairs can be time control (moves per second) or time (when the game started). For more information look at [21].

The second part of a PGN file is the "movetext", which are the moves made by both players during the game. The moves are recorded as standard algebraic notations (SAN), which is based on a system of coordinates, describing the distinct positions on a chessboard. The coordinates are denoted with a lowercase letter followed by a number as in figure 9 below:

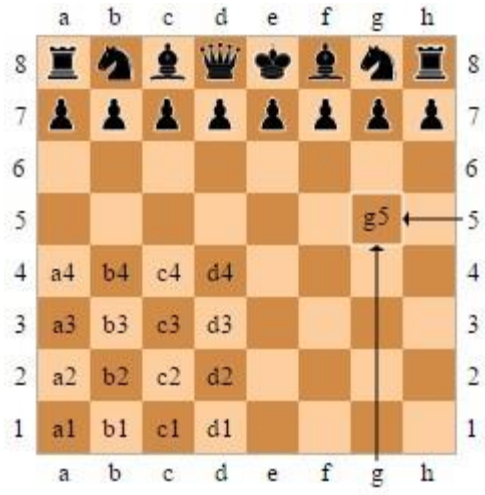

Figure 10: Chessboard notation. Source: English Wikipedia article on PGN [22]

The horizontal line is referred to as file and the vertical is rank. The SAN consists of notations for the chess pieces and the coordinates on the chessboard as can be seen in figure 9.

The movetext consists of a move number indicator followed by what chess piece is used and what position on the chessboard the chess piece is moved to for both players. The move number indicator is a number representing the number of a round and is followed by a dot. For instance, it can be deduced that the total number of rounds in figure 8 were 43. The different notations for the chess pieces are uppercase letters N (Knight), R (Rock), B (Bishop), Q (Queen) and K (King). Pawns do not have any letters.

Reading for instance the first move in figure 8 as 1. e4 e5 means that the white player moved a pawn to e4 and that the black player moved a pawn to e5.

Making a capture with a piece is denoted as x and is appended before the destination for the piece, so Rxe4 means that Rock captured a piece on e4. When pawns make a capture, the file that the pawn came from is also used, so gxh5 means that a paw on the g-file captured a piece on h5.

Sometimes it's necessary to provide more information to the move when ambiguity can arise due to two or more identical pieces being able to go to the same position. An example of this is when two knights on g1 and d2 can move to f3, so the notation would be either Ngf3 or Ndf3, where g and f in both notations corresponds to the position that the chess pieces moved from.

If a pawn is promoted when making it to the last rank, the promoted piece will be appended to the end of the destination position. For instance, when a pawn is promoted to the queen on e8, e8Q is written.

King castling is denoted as uppercase O letters O-O and queen castling by O-O-O. A check move corresponds to a + sign and is appended at the end of the move that made the check. A double check, which is a checkmate, is denoted by # and is also appended to the end of the move that made it. At the end of the game 1-0 indicates that white won, 0-1 that black won and ½-½ a draw.

ELO rating

The ELO rating is a measurement to rank a player, no matter what sport is in question (e.g. football, basket, chess etcetera), based on previous game results the player has undergone. The ELO rating was invented and called after Arpad Elo, who was a Hungarian-born American physics professor [23]. All ELO ratings specified in this study are estimated accordingly to the FIDE, who has its calculation techniques to determine the rank of a chess player. Other organizations such as ICC, USCF and PCA have their own implementation of calculating ELO rating [24]. However, the FIDE standard of ELO rating is the most common one and thus, all ELO ratings specified in this study are ELO ratings accordingly to FIDE.