university of groningen / faculty of science and engineering / artificial intelligence

A Few-Shot Embedding Learning Approach for Predicting the Behavior of Individual Chess Players

Bachelor's Project

Lennart August, s4800036
Supervisor: Steven Abreu
Second Supervisor: Dr. Herbert Jaeger

Abstract

This study presents a few-shot embedding learning approach to predict the behavior of individual chess players, based on only 100 games. Traditional models have relied on extensive datasets, often requiring thousands of games to achieve accurate move predictions. In contrast, our method leverages a limited number of games to generate dense vector representations, or embeddings, that capture a player's unique style. We trained a neural network to create these embeddings and used them to predict subsequent moves. Our results indicate that the embedding model performs well across various player sets and can accurately identify players even at scale among a great player population, picking out players with 84% accuracy from among 100k candidates. There are indications that including information on the clock situation during the game improves the embedding process, although our findings are inconclusive. Despite these limitations, our approach shows promise in making personalized chess training more accessible and highlights the potential for embedding learning in human-centered AI applications. Future work will aim to refine both the embedding and move prediction models and explore its application in other domains.

1 Introduction

Right now, machine learning models are surpassing human abilities in a growing number of previously difficult domains, such as imperfect information games (Perolat et al., 2022), visual tasks (Maslej et al., 2023), and common sense question answering (Xu et al., 2022). For humans to benefit and maybe also to survive competition with them in the labour market, finding beneficial ways of interaction and cooperation with these technologies is crucial (Organisation for Economic Co-operation and Development, 2023).

In chess, computer programs have long outperformed human decision-making. In 1997, Deep Blue became the first computer to beat a human world chess champion in a match (Campbell et al., 2002). The first match between a commercially available machine and the human world champion was won in 2006 [1]. Since then, progress on so-called chess engines has continued and has made several big steps forward. Humans no longer stand a chance. Perhaps the most notable in recent history was AlphaZero in 2017 (Silver et al., 2018), which introduced neural networks into the world of chess. Now, neural networks are dominating, playing an important role in all leading engines.

While these advancements highlight the dominance of AI in chess, they also leave a gap: the integration of these technologies in a way that directly complements human players. During this period of rapid progress, there has been limited focus on designing models to cooperate with or assist the learning process of human chess players (McIlroy-Young et al., 2020). McIlroy-Young et al. (2020) pioneered the development of human-centered chess models that learn to predict the behavior of players

[1] https://en.chessbase.com/post/kramnik-vs-deep-fritz-computer-wins-match-by-4-2, accessed: 04.07.2024

in a certain skill range as per an Elo rating system. The eventual aim is to build neural networks that can help teach chess and make game analyses.

To continue their work, they applied a meta-learned fine-tuning strategy to adapt one of their previously trained models to predict the moves of individual players (McIlroy-Young et al., 2020; Wang et al., 2020). This fine-tuning approach required a player to have played 5,000 games in a Blitz time format to have any positive impact on move-matching accuracy. A single Blitz game takes around 5 to 12 minutes, meaning a total time investment in the ballpark of a full month of playing time and up to 8 months for the greatest benefit for accuracy. Even now that the online chess community is very large and has seen a great increase in interest over the last years[2], players with this much activity are rare - about 1.3% of all players[3]. Still, the availability of public chess data has grown substantially over the past years. Each month, one of the two largest online chess websites, `lichess.org`, publishes all rated games[4] played on the platform [5]. This means that while there might still not be many individuals with an amount of games at the order of 5,000 Blitz games, there are more than 2 million players with at least 100 games. For the scope of this work, criteria are applied that limit the number of players to 105,110. These criteria are specified in Section 6.1. One requirement placed on the activity history is that they need to have played 1000 games. This criterion, along with others as the first/ last game dates, can likely be relaxed in future work.

The specific goal of this project will be to focus on capturing the behavior of a chess player in an embedding from a limited number of games and subsequently predicting behavior in novel situations. An embedding is a dense vector representation that captures the semantic relationships within data, like words or images, in a low-dimensional space (Wang et al., 2020; Bengio et al., 2013). In this case, the data refers to chess moves that, in a compressed form, could serve as a foundation for future individualized training and analysis tools. Currently, personalized chess training and lessons can only be offered by human chess coaches. Personalized models could make chess more accessible to many people who cannot afford an expensive coach. It should also be noted that the potential impact on competitive chess is great, as these methods could radically transform how strategies are formed in preparation to a game where the opponent is known (McIlroy-Young et al., 2022).

The usage of embeddings for identifying players based on how they played will be explored as well.

In summary, this thesis aims to investigate the potential of embedding learning to generate a representation of a chess player's style, which can 1) uniquely identfy them and 2) from which their behavior can be predicted.

RQ How accurate can a few-shot embedding learning approach be for identifying chess players at scale, and how does it compare to meta-learning when predicting the moves of individual chess players?

2 Related Work

2.1 Few-Shot Learning

Few-shot learning encompasses the set of machine learning problems, where a task $\mathcal{T}$ has to be learned with only a few labeled data points $x \in D_{train}$ (Wang et al., 2020).

One popular few-shot learning approach is fine-tuning, where an existing set of parameters θ_0 and some, optional, new, additional parameters θ_{new} are adapted to $\mathcal{T}$. For example, θ_{new} could be the classification layer at the end of a BERT network with parameters θ_0. The existing set of parameters θ_0 has been optimized for a task similar to $\mathcal{T}$ for which abundant data is available. Thus, θ_0 can be adapted to the new task $\mathcal{T}$ quickly. Transfer learning takes this a step further by assuming that θ_0 is already

[2]https://www.chess.com/article/view/why-is-chess-so-popular-right-now, accessed: 04.07.2024

[3]When looking at the time period 2018 to 2022 on `lichess.org`.

[4]A rated game influences a player's Elo rating on the platform.

[5]https://database.lichess.org/, accessed: 04.07.2024

optimal and only optimizing θ_{new} is sufficient (Wang et al., 2020).

Another approach to this is meta-learning, where a meta-learner accumulates generic information over multiple different tasks, which makes learning a new task easier, compensating for the small size of D_{train}. The meta-learner may refine some training parameters, such as learning rate or restrict the model hypothesis space $\mathcal{H}$. Some approaches also learn the optimizer, resulting in a model that can predict the parameter updates based on the loss on D_{train} (Wang et al., 2020).

Embedding Learning Embedding learning is the technique to solve few-shot learning that was chosen for this work. A model f_e projects samples $x \in \mathbb{R}^n$ into a lower-dimensional space $\mathcal{Z} \subseteq \mathbb{R}^m$ with $n > m$. In $\mathcal{Z}$, semantically similar samples are close by some distance metric and semantically dissimilar samples are far away from each other. Some popular metrics include l_1 and l_2 distances or cosine similarity (Wang et al., 2020). When projected into the smaller feature space $\mathcal{Z}$, the information in a sample is more dense or less important features have been filtered out. That is why classifying samples in $\mathcal{Z}$ typically requires a less complex model and therefore the model hypothesis space $\mathcal{H}$ can be much smaller, saving a lot of computation during the exploration of $\mathcal{H}$. The embedding model/ function $f_e : \mathbb{R}^n \mapsto \mathbb{R}^m$ is the crucial aspect of embedding learning and represents the general knowledge that is gathered from other similar tasks where training data is abundant. In essence, this makes embedding learning a variant of multitask learning, with the embedding model being shared between tasks. After pre-training, the embedding model can effectively function as an input encoding module and as such be integrated into another model.

In contrast to embedding layers for input encoding, embedding learning models are trained to optimize a loss function specifically designed to reach the goal of placing projected samples close to semantically related and far from unrelated samples. To this end, multiple loss functions can be used in embedding learning, such as various contrastive losses (Chopra et al., 2005), the triplet loss (Weinberger et al., 2005), or the generalized end-to-end loss (GE2E) (Wan et al., 2020). For this work, GE2E will be the method of choice, following McIlroy-Young et al. (2021).

2.2 Stylometry

Stylometry is the study of finding a set of linguistic features that uniquely identify an agent, i.e. their style (Savoy, 2020). As Savoy (2020) points out, the element of choice is important in this case. Only when there are multiple options that can be seen as viable, can the behavior be characterizing of the deciding agent. Take many such decisions, and a style emerges.

Some classic applications of stylometry include plagiarism detection (Alsallal et al., 2013), (forensic) handwriting recognition (Khayyat & Elrefaei, 2020; Chaski et al., 2013), or the authorship attribution of text (Chaski et al., 2013) and code (Caliskan-Islam et al., 2015). However, many other suitable domains of application can be found, such as face (Liu et al., 2017) and fingerprint recognition (Elsadai et al., 2022) or speaker verification (Zhang et al., 2016). Especially the field of speaker verification has brought forth many embedding learning approaches (Wan et al., 2020; Li et al., 2017).

Behavioral Stylometry Analyzing decision-making styles is what McIlroy-Young et al. (2021) call behavioral stylometry. It provides a framework to view identification tasks in behavior-defined situations, such as games, but also driving, cooking, etc. Thus, it encompasses most areas of application for reinforcement learning. This thesis will be concerned with the domain of chess in behavioral stylometry.

2.3 Chess Engines

The quest to design an approximately optimal artificial chess player has been pursued since there have been computers (Copeland, 2004). Humans cannot hold their ground against the results of this quest for a long time already. Silver et al. (2018) at DeepMind created AlphaZero, which has been adapted into the open-source Leela Chess Zero (Lc0) project[6]. AlphaZero and Lc0 use Monte Carlo Tree Search (MCTS), with a residual neural network (He et al., 2016) as the evaluation function at its heart. In the Lc0 project, there have also been successful experiments with a transformer architecture [7]. Another long-running open source chess engine project is Stockfish[8]. It has been running for a long time and is the leading chess engine as of July 2024[9]. They use efficiently updateable neural networks (NNUEs) (Klein, 2022a), which can be run on modern CPUs and achieve very high evaluation speeds that are beneficial for an alpha-beta search, as many chess engines take much of their playing strength from a fast and deep search.

2.4 Human-like Chess Engines

Although perfect play in chess has been studied extensively in the past, reasonably well-performing human-like chess models are relatively new in the field of machine learning (Klein, 2022b). In 2020, McIlroy-Young et al. published their work on Maia, a neural network architecture whose models would exhibit more human-like play, being able to correctly predict the human move of players with about 2000 Elo with 53% accuracy. This accuracy metric is the average success rate of correctly predicting the human move in many randomly selected positions and is therefore measured independently of the game stage or other factors. However, in the works by McIlroy-Young et al., the first 10 positions and positions where players had less than 30 seconds left on their clock are excluded. These restrictions do not apply to the move-matching accuracies given in this paper, in order to make them more representative of performance in a full game simulation.

[6]`https://lczero.org/`, accessed: 04.07.2024

[7]`https://lczero.org/blog/2023/01/2022-progress/`, accessed: 04.07.2024

[8]`https://stockfishchess.org/`, accessed: 04.07.2024

[9]`https://computerchess.org.uk/ccrl/4040/index.html`, accessed: 04.07.2024

The Maia architecture is based on Lc0, thus using a ResNet, but no MCTS is applied. A compounding work discovered that when combining Maia with KL-regularized MCTS, the prediction accuracy can be improved to 54.3% (Jacob et al., 2022). Furthermore, McIlroy-Young et al. developed a fine-tuning strategy that would allow for even more accurate predictions of individual play. For players with 1,000 games in the database, they achieved 49.7% accuracy, worse than the 52.7% achieved by a general Maia baseline model, trained on many players in the same Elo range. With 5,000 games, they would get to an average of 55.0% move matching accuracy, and up to 58.0% for players with a total of 40,000 games or more.

3 Methodology

3.1 Data

Lichess[10] is the second-largest online chess platform. It is an open source project that publishes all rated games that are played on the platform in an open database[11]. Since 2021, the website has a volume of around 100M games played per month, and the database contains approximately 5.3B games, as of February 2024. For this work, games in Blitz time format from January 2018 to December 2022 will be considered. The Blitz time format defines a time limit between 3 minutes initial time per player with no increment and 5 minutes initial time with 3 seconds increment. Note that these PGN (portable game notation) files contain clock information in the form of timestamps. This is relevant because these timestamps will be included in the input, as a proposed improvement over the work of McIlroy-Young et al. (2021). The players which are considered have to have played at least 1,000 Blitz games in the period from 2018 to 2022, a mean rating between 1,000 and 2,000 and a low rating variance of at most 75. More criteria are specified in the appendix. This leads to a selection of 105,110 players, of which the chronologically first 1,000 games are used each,

[10]`lichess.org`, accessed: 04.07.2024

[11]`database.lichess.org`, accessed: 04.07.2024

resulting in > 105M games. The players were split 80:20 into training and testing set.

To compile the players, games and subsequently the move encodings (as explained in the following subsection) from the Lichess game PGNs, an AMD EPYC 7763 CPU runs for 27h with 120GB of RAM available.

3.2 Model

This section specifies the model architectures used for the player embedding, i.e. stylometry, and move prediction tasks. The corresponding training objectives and procedures are laid out in Section 3.3.

3.2.1 Embedding Model

The embedding models are designed to map players from their games into a discriminating embedding space. This is a stylometry problem and is approached similarly to McIlroy-Young et al. (2021). Four model architectures are evaluated. For only two of them, clock information is included in the input features. The models for which it is excluded are referred to with *(no clock)*. Additionally, there is a small and a larger version of the models with or without those input features. These different sizes are referred to as *512d* or *768d*, alluding to the final size of the embedding that they produce.

Move Encoding A move is represented by either 34 or 42 channels of 8x8 binary matrices that contain all relevant information. The 42-channel representation has, in contrast to McIlroy-Young et al. (2021), an extra 8 channels for clock information. Table 6.1 in the appendix explains the variables encoded there in more detail. Furthermore, there is 26 channels for piece positions before and after the move and 8 channels for game state information (castling rights, the active player's side, the 50-move-count rule and filler border information). Only the 26 channels for piece positions contain sparse topological information and benefit from a corresponding CNN network architecture, as the information in all other channels is uniform. Nonetheless, the move representations are processed by a residual network (He et al., 2016), following the work by Silver et al. (2018). The CNN architecture is so important because of the many local patterns that a chess board contains. First, a convolutional block processes the input through a convolutional layer, batch normalization (BN) and a ReLU activation. Then, a series of squeeze-and-excitation (SE) residual blocks is applied. Each block consists of a convolutional block, an SE block and a residual connection (He et al., 2016; Ioffe & Szegedy, 2015; Hu et al., 2018). Finally, the output of the residual block stack is passed through a convolutional layer, BN and ReLU and projected by a Multilayer Perceptron (MLP) into a 320- or 768-dimensional move encoding (see Figure 3.1). The dimensionality is greater for the larger model.

Game-Level and Player-Level Aggregation To generate an encoding of a game, an encoder neural network of the Transformer architecture is used (Vaswani et al., 2017). The input series consists of the move encodings projected into 1,024 dimensions, with an added sine and cosine positional encoding. The transformer is 12 layers deep, with multi-head attention with 8 heads and 64 dimensions per head[12] and an MLP-layer of width 2,048. The final game embedding is obtained by averaging the transformer outputs over the series of moves, applying a layer normalization, linearly projecting the result into 512 or 768 dimensions (depending on model size) and finally computing the tanh. A visualization of the architecture can be seen in Figure 3.2. The resulting player vector is obtained by averaging multiple game vectors. For the purpose of player identification and move prediction, the player vector will be the average of 100 game vectors.

3.2.2 Stylometry and Player Identification Pipeline

The aggregated player vector can be used to identify individuals from a pool of potential

[12]In other words, query, key and value vectors have 64 dimensions.

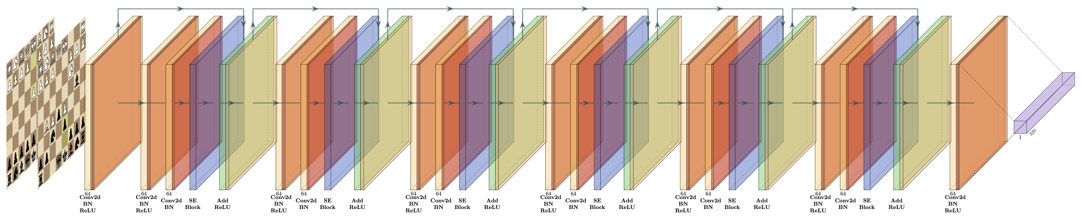

Figure 3.1: The Move Encoding Architecture (512d model)

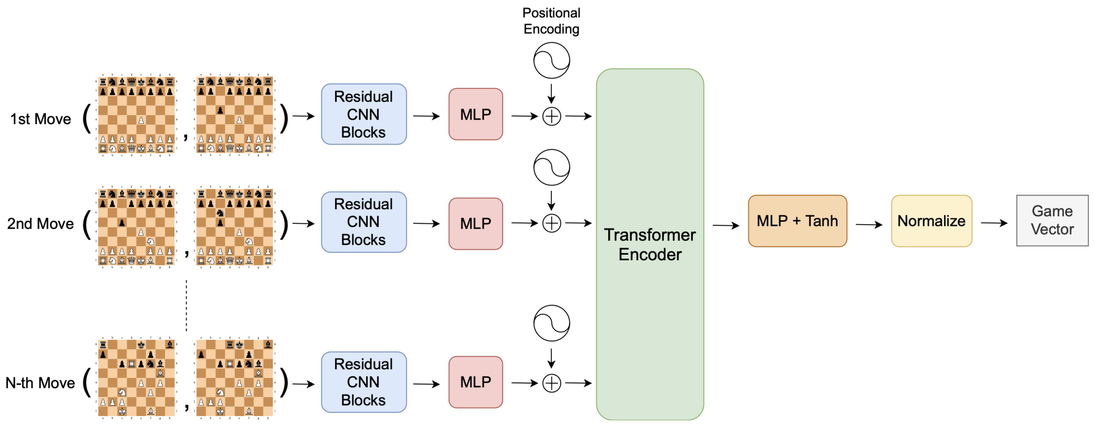

Figure 3.2: Overview of the Embedding Model Architecture. Figure was taken from McIlroy-Young et al. (2021)

candidates, or candidate pool. Given the computed player vectors of each player in the candidate pool, a new, unidentified player vector can be assigned by finding the closest match as per the cosine similarity metric. To evaluate the model's performance on this task, an evaluation pool of players is chosen from the candidate pool. 100 games are provided per player in the candidate pool. A set of 100 additional games is provided for each player in the evaluation pool. The model performance will be the average accuracy with which player vectors are assigned to the correct identity.

3.2.3 Move Prediction Model

For move prediction, the network architecture is adapted from Silver et al. (2017) (McIlroy-Young et al., 2020) and applied to multiple models with different (training) hyperparameters. Refer to Table 6.2 in the appendix for a full list of models and hyperparameters. The input consists of the board positions in the last 8 "ply" [13] up to the current position, i.e. 4 moves per side, including clock information but not opp_clock and opp_clock_percent, making a total of $\#ply \cdot (position + clock) + meta\_information = 8 \cdot (13 + 6) + 8 = 160$ 8x8 channels. To process this input, the same residual network architecture as specified for the embedding move encoding is applied, of which the output is flattened into N dimensions and merged with the player vector of M dimensions (either 512d or 768d). The player vector is generated from 100 games using the stylometry pipeline described in Section 3.2.2. The resulting, merged $N+M$-sized vector is fed through the final 1- to 3-layered MLP network. The output of the network is a move encoded in a policy map of 1,858 dimensions (see Figure 3.3). This policy map matches its indices to all theoretically possible moves and promo-

[13] A single action taken by one player is called a "ply."

tions. It is position-dependent to account for different castling rights (Silver et al., 2018).

3.3 Training Procedure

As has been shown in previous work, training an encoding network on the stylometry task alone can produce representations which contain useful information on player behavior (McIlroy-Young et al., 2021). To save on computational budget, the embedding network is therefore trained separately from the move prediction network.

3.3.1 Embedding Training

The full embedding model is trained end-to-end to minimize the GE2E loss with the SGD optimizer with momentum (Sutskever et al., 2013), starting with an initial learning rate of $3 \cdot 10^{-3}$ and momentum of 0.9. A learning rate schedule is set up that reduces the learning rate every 40k steps by a factor of 0.5. Each step computes the GE2E similarity matrix for $N = 15$ players, and $M = 30$ games per player. During training, the games are limited to a window of 32 moves length, for which the start point, i.e. position, is chosen randomly every time. The ith game of the jth player is embedded in a vector e_{ji}. For each player k, an average embedding, or centroid, c_k can be computed. The similarity matrix is defined as the scaled cosine similarities between each embedding vector e_{ji} and each centroid c_k ($1 \leq j, k \leq N$, and $1 \leq i \leq M$):

$$S_{ji,k} = w \cdot \cos(e_{ji}, c_k) + b, \tag{3.1}$$

where w and b are learnable parameters with $w > 0$. Based on this, a softmax loss is defined on each embedding vector e_{ji} as

$$L(e_{ji}) = -S_{ji,j} + \log \sum_{k=1}^{N} \exp S_{ji,k}. \tag{3.2}$$

The final GE2E loss L_G is the sum of all losses over the similarity matrix S ($1 \leq j \leq N, 1 \leq i \leq M$):

$$L_G(S) = \sum_{j,i} L(e_{ji}). \tag{3.3}$$

Equation 3.3 (Wan et al., 2020) is the training objective by which the entire embedding network is trained, including the move encoding residual network and the game-level aggregation transformer network (McIlroy-Young et al., 2021).

The model is trained over a total of 120k steps, which requires an A100 40GB GPU to run for 70h. This time span is about equal to the longest job run time permitted on the employed high performance cluster.

3.3.2 Prediction Training

The move prediction model is trained to minimize a cross entropy loss across the 1,858-dimensional output, with different optimizers being tested. The embedding Each training step contains 4,096 positions + move. The learning rate is initialized between 10^{-2} and 10^{-5} per model and is multiplied by a factor of 0.1 or 0.3 every 40k or 10k steps respectively. For the full training parameters for each model, see Table 6.3.

The models were trained on a V100 GPU for 10h to 70h. Training was stopped when reaching the maximum job run time or manually if the training curve showed no signs of improvement for more than 10k steps.

4 Results

4.1 Stylometry Results

This section compares the accuracy of various stylometry networks. The 512d model embeds games into 512 dimensions, and its residual network is comprised of 6 blocks with 64 filters in the convolutional layers. The 768d model is larger, embedding into 768 dimensions with a residual network of 10 blocks depth and the convolutional layers 80 filters wide.

As Table 4.1 shows, our models do not perform better on data from players that it was specifically trained on, i.e. seen players, compared to previously unseen ones. Furthermore, there is a significant performance gap between our models and the McIlroy-Young et al. model

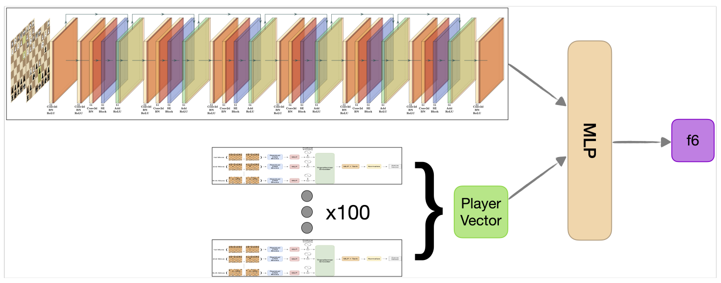

Figure 3.3: The Prediction Architecture. Position embedding and player embedding are passed through an MLP to arrive at the policy decision

|E|	Metric	With clock 512d	With clock 768d	No clock 512d	No clock 768d
|all| = 105,110	P@1	81.8%	**84.0%**	0.121%	0.219%
	P@5	88.3%	**88.6%**	0.366%	0.669%
|seen| = 84,088	P@1	81.7%	**83.9%**	0.119%	0.227%
	P@5	88.3%	**88.6%**	0.373%	0.666%
|unseen| = 21,022	P@1	81.8%	**84.1%**	0.122%	0.177%
	P@5	88.3%	**88.9%**	0.332%	0.659%

Table 4.1: Stylometry performance (P@1 and P@5 accuracy) on selected Lichess players with more than 1000 games. Reference and query sets consist of 100 games each. E stands for the evaluation pool, i.e. which players had to be identified. The candidate pool contained all 105,110 players. Seen players were part of the training set. Unseen players were part of the testing set. Random identification accuracy on this task is $9.51 \cdot 10^{-4}$%.

at a candidate pool size of 2844, as indicated in Table 4.2.

4.1.1 Scaling the candidate pool

Figure 4.1 shows that the identification accuracy does not have an inverse relationship to the cardinality of the candidate pool as one would expect. This contrast is shown in Figure 4.1 (c), where the simple inverse $1/|C|$ shrinks to 0 much quicker than even the curves for the models without clock information. Though the accuracy is decreasing with increasing $|C|$, this decrease is very gradual. For a more precise description of the relationship between identification accuracy and $|C|$, future will have to expand the candidate pool or train models with significantly smaller embedding sizes.

4.1.2 Analysis of Vector Embeddings

Clock Information The identification accuracy of the embedding models trained without clock information is about 400x smaller than that of those trained with clock information. This does not meet expectations. To try and understand this, Table 4.3 gives an idea of the relevance of the different clock features to the network's output. Especially the move_time and opp_clock features seem to have a large influence, reducing top-1 identification accuracy by 16.3% and 63.3% respectively. Further re-

|E|	$\|C\|$	768d (ours)	512d (ours)	McIlroy-Young et al.
2844	2844	91.8%	91.6%	**98.2%**

Table 4.2: Stylometry performance (P@1) comparisons on a candidate pool of 2844 players. For our 768d and 512d models, this is the average performance over 50 trials on a randomly selected subset of all Lichess players with more than 1000 games in our dataset. The McIlroy-Young et al. performance metric can be found in their paper (McIlroy-Young et al., 2021).

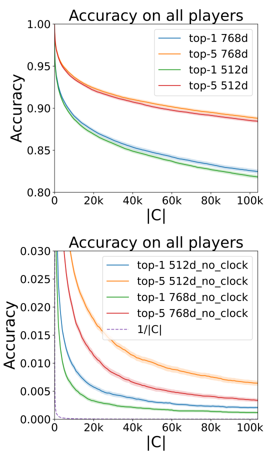

Figure 4.1: The plots show the identification accuracy for all players on an increasing candidate pool size, averaged over 50 random subsets per candidate pool size. The evaluation pool was limited to 2000, thus $|E| = \min(|C|, 2000)$. ***Note: For illustrative purposes, there are differing scales on the y-axis.***

search will be necessary to determine the reason for their large influence. Setting low_time to 0 slightly increased top-5 identification accuracy by 0.6%, which warrants further investigation as well.

Elo Strength The player embedding vectors generated by the 768d model could not be shown to contain information from which the player's playing strength could be inferred. To test this, two models were trained, a least squares linear regression and a two-layer MLP, with the MSE loss. Instead of uncovering any patterns in the data, both learn to predict the mean value of the training data.

Removed input metric	top-1	top-5
none	84.0%	88.6%
low_time	84.0%	89.4%
move_time	67.7%	78.8%
opp_clock	20.7%	31.5%
opp_clock_percent	81.3%	87.2%
post_clock_percent	82.0%	87.7%

Table 4.3: Top-1 and top-5 accuracy of 768d model on all players with the input layer for the corresponding aspect of the clock set to 0.

4.2 Move Prediction Results

For the personalized move prediction task, multiple models were trained. None of them performed well, compared to the 52.7% of the general Maia models developed by McIlroy-Young et al.. One significant difference is, that our models lack the second training task of evaluating how good a given position is (McIlroy-Young et al., 2020). The two models evaluated here are numbers 11 and 12 from Tables 6.2 and 6.3, referred to as AdamW and SGD respectively. The highest testing accuracy that was achieved is 2.6%. Figure 4.2 shows the progression of training and testing cross-entropy loss and accuracy during training for models 12 (SGD) and 11 (AdamW). It can be observed that the model in (a) and (c) converged very quickly and did not continue learning past the initial steps of training. By contrast, the model in (b) and (d) shows gradual improvement during training but does not reach a better testing loss/ accuracy.

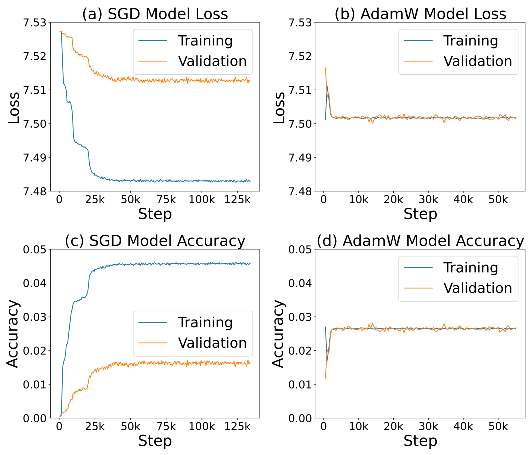

Figure 4.2: (a) & (b): Cross-entropy loss in relation to the number of training steps taken, of the (a) SGD- and (b) AdamW-optimized move prediction models. (c) & (d): Accuracy in relation to the number of training steps taken, of the (c) AdamW- and (d) SGD-optimized move prediction models.

5 Discussion

In this study, we explored the potential of an embedding learning approach to capture and predict the behavior of individual chess players. By focusing on a few-shot learning method, we aimed to address the limitations of previous models that required extensive data for accurate predictions. Our approach involved creating dense vector representations of players' styles from a limited number of games, which can be used for personalized training and analysis.

The results demonstrated that our model could effectively generate embeddings that encapsulate individual playing styles, achieving notable accuracy in identifying players from a large pool. However, the accuracy of our model did not show a significant improvement for players it was trained on compared to unseen players, while at the same time gaining performance from increasing model and embedding size. This indicates that further scaling up of the model and embedding size can result in performance gains.

Additionally, our analysis revealed that the applied identification method scales very well with the size of the candidate pool, as the accuracy decreases at a much slower rate than the rate at which pool grows. The scaling is presumably limited by the dimensionality of the embedding. While this was not specifically investigated in this study, research by McIlroy-Young et al. (2021) indicates the same conclusion. They found a model with 128 dimensions to plateau much quicker than one with 512 dimensions.

Despite the promising results, the impact of time information in the embeddings remains inconclusive, as the no clock models' performance should still be comparable to that of McIlroy-Young et al. (2021), highlighting an area for future research. Moreover, while the player embeddings could not be shown to capture information related to playing strength, this aspect warrants further investigation to enhance the model's utility. It is plausible that including Elo data could significantly improve move prediction accuracy, similar to training separate models per Elo range in the work by McIlroy-Young et al. (2021).

Overall, this work contributes to the ongoing efforts in human-centered AI by proposing a method that could make personalized chess training more accessible. The findings underscore the potential impact of embedding learning in creating AI systems that cooperate with human players, paving the way for more intuitive and effective training tools in chess and beyond. Future work will focus on refining the model, incorporating additional features, and exploring the broader applicability of embedding learning in other domains of human behavior prediction.

References

Alsallal, M., Iqbal, R., Amin, S., & James, A. (2013). Intrinsic Plagiarism Detection Using Latent Semantic Indexing and Stylometry. In *2013 Sixth International Conference on Developments in eSystems Engineering* (p. 145-150). doi: 10.1109/DeSE.2013.34

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Caliskan-Islam, A., Harang, R., Liu, A., Narayanan, A., Voss, C., Yamaguchi, F., & Greenstadt, R. (2015). De-Anonymizing Programmers via Code Stylometry. In *Proceedings of the 24th USENIX Conference on Security Symposium* (p. 255–270). USA: USENIX Association.

Campbell, M., Hoane Jr, A. J., & Hsu, F.-h. (2002). Deep Blue. *Artificial Intelligence*, *134*(1-2), 57–83. doi: https://doi.org/10.1016/S0004-3702(01)00129-1

Chaski, C., et al. (2013). Best Practices and Admissibility of Forensic Author Identification. *Journal of Law and Policy*, *21*(2), 5.

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (Vol. 1, p. 539-546 vol. 1). doi: 10.1109/CVPR.2005.202

Copeland, B. J. (2004, 09). Chapter 16: Chess. In *The Essential Turing* (p. 562-575). Oxford University Press. Retrieved from `https://doi.org/10.1093/oso/9780198250791.001.0001` doi: 10.1093/oso/9780198250791.001.0001

Elsadai, A., Adamović, S., Šarac, M., Saračević, M., & Kumar Sharma, S. (2022). New approach for fingerprint recognition based on stylometric features with blockchain and cancellable biometric aspects. *Multimedia Tools and Applications*, *81*(25), 36715–36733. Retrieved from `https://doi.org/10.1007/s11042-021-11581-w` doi: 10.1007/s11042-021-11581-w

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 770-778). doi: https://doi.org/10.1109/CVPR.2016.90

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-Excitation Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7132–7141). doi: https://doi.org/10.1109/CVPR.2018.00745

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Computing Research Repository*, *abs/1502.03167*. Retrieved from `http://arxiv.org/abs/1502.03167`

Jacob, A. P., Wu, D. J., Farina, G., Lerer, A., Hu, H., Bakhtin, A., ... Brown, N. (2022). Modeling strong and human-like gameplay with KL-regularized search. In *Proceedings of the 39th International Conference on Machine Learning* (pp. 9695–9728). PMLR. Retrieved from `https://proceedings.mlr.press/v162/jacob22a.html`

Khayyat, M., & Elrefaei, L. (2020, 09). A Deep Learning Based Prediction of Arabic Manuscripts Handwriting Style. *The International Arab Journal of Information Technology*, *17*, 702-712. doi: 10.34028/iajit/17/5/3

Klein, D. (2022a). Chapter 4.6: Efficiently updateable neural networks (NNUE). In *Neural Networks for Chess* (p. 191-216). arXiv. doi: https://doi.org/10.48550/arXiv.2209.01506

Klein, D. (2022b). Chapter 4: Modern AI Approaches - A Deep Dive. In *Neural Networks for Chess* (p. 123-126). arXiv. doi: https://doi.org/10.48550/arXiv.2209.01506

Li, C., Ma, X., Jiang, B., Li, X., Zhang, X., Liu, X., ... Zhu, Z. (2017). Deep Speaker: an End-to-End Neural Speaker Embedding System. *arXiv preprint arXiv:1705.02304*.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., & Song, L. (2017). SphereFace: Deep Hypersphere Embedding for Face Recognition. *Computing Research Repository*, *abs/1704.08063*. Retrieved from http://arxiv.org/abs/1704.08063

Maslej, N., Fattorini, L., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., ... Perrault, R. (2023). Chapter 2: Technical Performance. In *The AI Index 2023 Annual Report* (p. 92). AI Index Steering Committee, Institute for Human-Centered AI, Stanford University. Retrieved from https://aiindex.stanford.edu/ai-index-report-2023/ (Retrieved June 14, 2024)

McIlroy-Young, R., Wang, R., Sen, S., Kleinberg, J. M., & Anderson, A. (2020). Learning Personalized Models of Human Behavior in Chess. *Computing Research Repository*, *abs/2008.10086*. Retrieved from https://arxiv.org/abs/2008.10086

McIlroy-Young, R., Kleinberg, J., Sen, S., Barocas, S., & Anderson, A. (2022). Mimetic models: Ethical implications of AI that acts like you. In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society* (pp. 479–490). Association for Computing Machinery. doi: https://doi.org/10.1145/3514094.3534177

McIlroy-Young, R., Sen, S., Kleinberg, J. M., & Anderson, A. (2020). Aligning Superhuman AI with Human Behavior: Chess as a Model System. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (p. 1677–1687). Association for Computing Machinery. doi: https://doi.org/10.1145/3394486.3403219

McIlroy-Young, R., Wang, R., Sen, S., Kleinberg, J., & Anderson, A. (2021). Detecting individual decision-making style: Exploring behavioral stylometry in chess. *Advances in Neural Information Processing Systems*, *34*, 24482–24497.

Organisation for Economic Co-operation and Development, J. L. (2023). *Skill needs and policies in the age of artificial intelligence.* Retrieved from https://www.oecd-ilibrary.org/content/publication/08785bba-en doi: https://doi.org/https://doi.org/10.1787/08785bba-en

Perolat, J., De Vylder, B., Hennes, D., Tarassov, E., Strub, F., de Boer, V., ... others (2022). Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science*, *378*(6623), 990–996. doi: https://doi.org/10.1126/science.add4679

Savoy, J. (2020). Chapter 1: Introduction to Stylistic Models and Applications. In *Machine Learning Methods for Stylometry* (p. 3-18). Cham: Springer.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*(6419), 1140-1144. doi: https://doi.org/10.1126/science.aar6404

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... others (2017). Mastering the game of go without human knowledge. *Nature*, *550*(7676), 354–359.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* (p. III–1139–III–1147). Journal of Machine Learning Research.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wan, L., Wang, Q., Papir, A., & Moreno, I. L. (2020). Generalized End-to-End Loss for Speaker Verification. In *2018 IEEE International Conference on Acoustics, Speech* *and Signal Processing (ICASSP)* (pp. 4879–4883). doi: {https://doi.org/10.48550/arXiv.1710.10467}

Wang, Y., Yao, Q., Kwok, J. T., & Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. *Association for Computing Machinery Computing Surveys*, *53*(3), 1–34. doi: https://doi.org/10.1145/3386252

Weinberger, K. Q., Blitzer, J., & Saul, L. K. (2005). Distance metric learning for large margin nearest neighbor classification. In *Proceedings of the 18th International Conference on Neural Information Processing Systems* (p. 1473–1480). MIT Press.

Xu, Y., Zhu, C., Wang, S., Sun, S., Cheng, H., Liu, X., ... Huang, X. (2022). Human Parity on CommonsenseQA: Augmenting Self-Attention with External Attention. In L. D. Raedt (Ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22* (pp. 2762–2768). International Joint Conferences on Artificial Intelligence Organization. doi: https://doi.org/10.24963/ijcai.2022/383

Zhang, S.-X., Chen, Z., Zhao, Y., Li, J., & Gong, Y. (2016). End-to-End attention based text-dependent speaker verification. In *2016 IEEE Spoken Language Technology Workshop (SLT)* (p. 171-178). doi: 10.1109/SLT.2016.7846261

6 Appendix

6.1 Player Selection Criteria

The criteria to select a player to be included in the full (training + testing) dataset are as follows:

```
blitz_game_count > 1000
AND std_elo < 75
AND mean_elo < 2000
AND mean_elo > 1000
AND blitz_white_count / blitz_game_count
    < .55
AND blitz_white_count / blitz_game_count
    > 0.45
AND blitz_lost_count / blitz_game_count
    < .6
AND blitz_lost_count / blitz_game_count
    > 0.4
AND first_game before 2022.01.01
AND last_game after 2022.12.01
```

These criteria were also used by McIlroy-Young et al. (2020) to select players for their fine-tuning method. The limits on white/ black rates and win rates were used to filter out players that manipulate their match-ups through so-called colour cheating or the like. The games that count towards these statistics are rated games from the years 2018 to 2022.

6.2 Move Prediction Models

Variable name	Description
`pre_move_clock`	The player's total time remaining on their clock after the previous move in seconds.
`post_move_clock`	The player's total time remaining on their clock after the current move in seconds.
`pre_clock_percent`	The fraction of the player's time remaining on their clock after the previous move, divided by the total time given to the player as per the game's time control.
`post_clock_percent`	The fraction of the player's time remaining on their clock after the current move, divided by the total time given to the player as per the game's time control.
`opp_clock`	The opponent's total time remaining on their clock after their last move in seconds.
`opp_clock_percent`	The fraction of the opponent's time remaining on their clock after their last move, divided by the total time given as per the game's time control.
`move_time`	The time it took the player to make the current move in seconds.
`low_time`	A binary variable indicating if the player has less than 30 seconds remaining on their clock after the current move.

Table 6.1: Clock Information Variables

Model N°	# Residual blocks	# Convolutional channels (residual stack)	SE ratio	# Convolutional channels (layer before fc)	# FC layers	FC width	FC activation	# embedding dimensions
1	12	128	8	128	5	4096	ReLU	512
2	12	128	8	128	5	4096	ReLU	768
3	12	128	8	128	5	4096	ReLU	512
4	12	128	8	128	5	4096	Swish	512
5	12	128	8	128	5	4096	Swish	512
6	12	128	8	128	5	4096	Tanh	512
7	8	64	8	80	3	3072	ReLU	768
8	6	64	8	64	2	3072	ReLU	512
9	6	64	8	64	3	3072	Swish	512
10	6	64	8	80	3	3072	Tanh	512
11	6	64	8	80	3	3072	Tanh	512
12	6	64	8	64	3	3072	ReLU	512

Table 6.2: Move prediction architectures

Model N°	# steps	Train loss	Test loss	Optimizer	Momentum	lr (init)	Weight decay	Scheduler step size	Scheduler gamma	# players per batch
1	40k	7.48	7.51	SGD	0.9	0.01	0.0001	40k	0.1	40
2	27.5k	7.48	7.51	SGD	0.9	0.01	0.0001	40k	0.1	40
3	27.5k	7.51	7.51	AdamW		0.003	0.0001	40k	0.1	40
4	27.5k	7.51	7.51	Adam		0.0003	0	40k	0.1	40
5	47.5k	7.51	7.53	SGD	0.1	0.0003	0	10k	0.3	40
6	48k	7.49	7.51	SGD	0.9	0.1	0.0001	10k	0.3	40
7	54.5k	7.51	7.51	Adam		0.01	0	40k	0.1	32
8	72k	7.51	7.51	Adam		0.1	0	40k	0.1	32
9	92k	7.48	7.51	Adam		0.0001	0	40k	0.1	32
10	89k	7.52	7.52	AdamW		0.1	0.0001	40k	0.1	32
11	54.5k	7.5	7.5	AdamW		0.01	0.0001	10k	0.3	32
12	133.5k	7.48	7.51	SGD	0.1	0.001	0	40k	0.1	32

Table 6.3: Move prediction training parameters and results