ChePT - Applying Deep Neural Transformer Models to Chess Move Prediction and Self-Commentary

Stanford CS224N
Mentor: Mandy Lu

Colton Swingle
Department of Computer Science
Stanford University
cswingle@stanford.edu

Henry Mellsop
Department of Computer Science
Stanford University
hmellsop@stanford.edu

Alex Langshur
Department of Computer Science
Stanford University
adl@stanford.edu

Abstract

Traditional chess engines are stateful; they observe a static board configuration, and then run inference to determine the best subsequent move. Additionally, more advanced neural engines rely on massive reinforcement learning frameworks and have no concept of explainability - moves are made that demonstrate extreme prowess, but oftentimes make little sense to humans watching the models perform. We propose fundamentally reimagining the concept of a chess engine, by casting the game of chess as a language problem. Our deep transformer architecture observes strings of Portable Game Notation (PGN) - a common string representation of chess games designed for maximum human understanding - and outputs strong predicted moves alongside an English commentary of what the model is trying to achieve. Our highest performing model uses just 9.6 million parameters, yet significantly outperforms existing transformer neural chess engines that use over 70 times the number of parameters. The approach yields a model that demonstrates strong understanding of the fundamental rules of chess, despite having no hard-coded states or transitions as a traditional reinforcement learning framework might require. The model is able to draw (stalemate) against Stockfish 13 - a state of the art traditional chess engine - and never makes illegal moves. Predicted commentary is insightful across the length of games, but suffers grammatically and generates numerous spelling mistakes - particularly in later game stages. Our results are an insight into the potential for natural language models to gain tractability on tasks traditionally reserved for reinforcement learning models, while additionally providing a degree of insight into the decisions mate. These findings significantly build on the work of Noever et al[1], Jahmtani et al., and Zang et al[3].

1 Introduction

In 1985, the famed Chess Grandmaster Garry Kasparov defeated IBM's Deep Thought and subsequently claimed that no computer program would ever best him. Only half a decade later, Kasparov was defeated by IBM's improved algorithm, Deep Blue. While FIDE scores - an international chess ELO score instituted by the World Chess Federation - aren't officially assigned to chess engines, it is estimated that the first installment of Deep Blue deserved a FIDE rating of around 2900. As a reference, the best FIDE rating ever achieved by a human player is 2882 (Magnus Carlsen in 2019).[8]

30 years later, the world's most advanced chess engine, Stockfish 13 (released in February 2021), is estimated to have a FIDE rating of approximately 3575. The history of artificial intelligence in chess is incredibly rich – Stockfish 13 is among a massive collection of engines that introduce novel algorithms and modelling approaches to far outperform human players.[9]

Yet, despite the inherent differences between these state-of-the-art chess engines that give meaning to the competitive landscape of algorithmic chess, they do in fact possess a few significant similarities. These engines often rely on massive depth-limited search trees, supported by human-tuned weightings of board configurations and pieces. Newer chess engines that employ deep learning, such as AlphaZero, are structured in similar ways, using Monte Carlo Tree Searches and neural network-based evaluation functions in order to achieve learnable depth limiting. These state-based models all subscribe to the Markov property[10]; they observe static board states and act only on this information. Moreover, none of these models provide any form of insight into why they choose particular moves.[9]

Our goal is to explore the use of end-to-end deep neural transformer models to predict the next move and associated chess commentary. As such, we present an unconventional alternative to traditional approaches - encoding the playing and commentating of Chess as a unified natural language task. By employing PGN - a human-centric and readable plain text format that denotes Chess games - we are able to frame the problem of chess move prediction and commentary as an end-to-end language problem. Rather than basing predictions off a discrete board state, we frame the game as a series of transitions; the model dynamically learns to rebuild the board using only the previous PGN. While doing so, unlike any other mainstream chess model, the system will not have access to a database of legal moves to make, or known state positions. Much like neural text generation - or a human writing their own commentated PGN - the model will have to predict subsequent moves and commentary character-by-character.

2 Related Work

With respect to analysing how generative language models perform at playing the game of chess, the literature is relatively limited. The first and only paper in this field was written by Noever et al.[1], and provides a high-level exploration of this task. The authors believed that instead of analysing chess games as state based, the problem could instead be reframed as transition based. Predictions could then be made by the model only looking at sequential data of game moves. With this consideration, they therefore decided to investigate whether new NLP techniques – techniques that are inherently designed to deal with sequential data – could be applied to turn-based strategy games in this manner. Thus, in this paper, they fine-tuned a pretrained GPT-2 model to play chess and analyzed the strength of its predictions.[1]

Additionally, the authors provide interesting ideas for training, analyzing, and extending upon their model. For training, the authors provide multiple methods of dataset aggregation and separation. Different methods such as mixing black and white games, black-win, white-win, and filtering by input game ELO contribute to different results. For analysis, the authors utilized Athena visualization, `python-chess` (a Stockfish-inspired chess library), and Google Colab to create a real-time visual interface to allow human testing of the model. Finally, the authors offer ideas for future work. Specifically, they believe that transformer models should be tested on other strategy-based games, as well as proposing the idea of testing a similar experiment in chess with different-sized models and more evaluation of model strategy.

In the commentary space, the literature is somewhat more standardized. There appears to be a single collected dataset that research papers use for training and testing.[2] The papers also all utilize human reviews to analyze the output's fluency, accuracy, insights, and overall interpretability, citing difficulties with standarised metrics such as BLEU (although BLEU is still reported).[3] Datasets are commonly split by "move quality", "comparisons", "planning", "descriptions", and "contexts". State of the art systems evaluate with a BLEU score of 0.14%.[3]. These papers suggest that future work may include utilizing models that can understand the entire game rather than just the most recent move, with an ultimate goal of combining play and commentary together.

As such, we seek to achieve the future work as suggested by these papers; we intend to experiment with better model evaluation and find more ideal model sizes, as suggested by Noever et al. Moreover, we intend to combine the move-prediction modelling with the commentary modelling, to imbue the model with a greater understanding of its own decision making, as suggested by Zang et al. and Jahmtani et al.

3 Approach

3.1 Move Prediction Baseline & Input/Output Behaviour

For move-prediction, we begin our approach by observing the baseline performance demonstrated by Noever et al., which employs a pretrained GPT-2 language model with 774M parameters. The inputs to the model are PGN strings in the format of the following, which encodes the first few moves of the 1992 game between Bobby Fischer and Boris Spassky. We then append the move prediction mask character (??).

```
1.  e4 e5 2.  Nf3 Nc6 3.  Bb5 a6 4.  Ba4 Nf6 5.  O-O Be7 6.  Re1 ??
```

The output of the model is a series of characters, terminated with a space, that represent (hopefully) valid algebraic notation. For example, if the model played optimally, it would generate the move `b5`. For our move generation model, we will mimic this input/output behaviour.

3.2 Combined Move Prediction + Commentary Input/Output Behaviour

For our commentary baseline results, we observe the work of Zang et al. and Jahmtani et al. However, we define our own - subtly different - input/output behaviour, owing to the fact that our model does both move predictiona and commentary generation - something that none of these papers achieve.

Once we extend our pretrained move prediction model to provide commentary, we employ a similar strategy to our pretraining inference input/output behaviour. As input, the model receives the prior move-only PGN string, concatenated with the move prediction mask character (??) and the commentary prediction mask character (?!). This allows the model to generate a complete move, and then subsequently generate any length of commentary – up to the defined block size.

```
1.  e4 e5 2.  Nf3 Nc6 3.  Bb5 a6 4.  Ba4 Nf6 5.  O-O Be7 6.  Re1 ??  ?!
```

3.3 Model Outline

Our hypothesis is that the complexity of chess semantics in relation to those of the unconstrained natural language problem (which GPT was designed to address) do not merit hundreds of millions of parameters. In a similar line of reasoning, a pretrained transformer model would need to unwind a significant fraction of the learned weightings from an unconstrained language model.

As such, we propose a single unified model, trained from scratch only on Chess-related tasks that will present next-move generation and prediction. We will pre-train the model in order to learn the rules of chess, fine-tune the model to improve on strategic inference, and finally present a further fine-tuning step to introduce the ability for the model to commentate on the actions that are taken. While we examined a number of separate model architectures, as we will expand upon in the Experiments section, our final transformer model consists of only 9.6M parameters.

3.4 Chess Next-Move Pretraining

In order to pretrain our next-move prediction model, we employ `milesh1`'s Kaggle dataset of 3.2 million games from players of mixed ELO[7]; we can use a mixed-ELO dataset as our pretraining objective is for the model to learn basic openings, basic strategy, and the fundamental rules of chess. More information on this dataset can be found in the experiments section. We structure our pretraining objective as next-character prediction on padded sequences of 512 characters. We use cross-entropy loss to evaluate each predicted character, and pass a single epoch across all 3.2 million training examples.

3.5 Chess Next-Move Finetuning

Our pretrained model performed strongly with respect to understanding the rules of Chess. We will elucidate on the exact evaluation details further in our Experiments section, but crucially the model would typically only attempt to make an illegal move 17% of the time over the course of a 28 move match, played against Stockfish 13[‡]. With a maximum of 3 move-resamples[†], this decreased to 3% illegal moves per game. However, while move legality was strong throughout the game and move quality was excellent in early game, move quality was far weaker in mid-to-late game.

We postulate that this is because the relative sparsity of data in chess games increases dramatically as the game goes on; there are relatively few legal moves to be made in the opening of a chess match, but by 15 moves in the number of legal moves is astronomical. When the model was being pretrained, it saw far more instances of early-game situations, where it was able to rote-memorise responses, and far fewer instances of a given late-game situation.

As such, we structured a three-layered finetuning approach. Firstly, we broke game data up into three sections: early game (defined as the first 8 moves of a match for each player), mid-game (defined as moves 9 through 16 for each player), and late-game (moves 17 onwards for each player). We then applied the same next-character prediction objective to each of these game phases, spending proportionally more training time on later phases. We trained early-game for only a single epoch (passing over the entire Kingbase dataset, which we will introduce in the Data section) of fine-tuning, but trained mid-game for 4 epochs, and late-game for 12. This ensured that late-game training received proportionally more attention, as it is more data-sparse. We selected the Kingbase dataset here as it is constrained to players of over 2000 ELO, which reserves the games to being generated by extremely competent players and helps to ensure move quality.

3.6 Chess Commentary Finetuning

For the commentary finetuning step, we used a chess commentary dataset that we will elucidate more on in our Experiments data section. The dataset is unnamed, as it required us building a web-crawler to extract around 300,000 commentated data points from the internet. After processing, we have kept 200,000 datapoints for training and 51,000 for testing. We then ran 6 epochs of finetuning. We inserted a masking character after the chess PGN and before the commentary, and another mask character after the commentary. We pad each datapoint to 512 characters. For the expected output, we replace the chess moves with padding also, such that the model only is penalized for incorrect commentary.

An example training data point (truncated down from 512 characters to fit on the page) may look like:

X: `d4 Nf6 c4 e5 dxe5 Ng4 Nf3 Nc6 Bf4 Bb4+ Nc3 Bxc3+ bxc3 Qe7 Qd5 O-O h3 Qa3 Qd2 ?!guarding the threat.?!■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■`

Y: `■■■■■■■■■■■■■■■■■■■■■■?!guarding the threat.?!■■■■■■■■■■■■■■■■■■■■■■■`

We chose to perform the commentary fine-tuning last, as the model will hypothetically be able to draw upon the latent game understanding that it developed throughout the prior pretaining and finetuning tasks in order to have the ability to better predict forthcoming moves, which may be able to be incorporated into the commentary prediction scheme.

4 Experiments

4.1 Data

4.1.1 Data Sourcing

For our models, we draw upon three separate datasets.

- Kingbase Chess Dataset. This dataset consists of 2.8 million games collected from players over 2000 ELO. (`"gpt-2-poetry/data/kingbase-ftfy.txt"`). This data is in plain PGN format.
- Kaggle Milesh1 Chess Dataset. This dataset consists of 35 million games collected from players of assorted ELO. (`"milesh1/35-million-chess-games"`). This data is in plain PGN format.
- Custom Commentary Dataset. Obtained utilising a custom-built web scraper adapted from Jhamtani, Harsh, et al.'s web scraper at https://github.com/harsh19/ChessCommentaryGeneration/tree/master/Data/crawler. We crawled 11,700 games collected from "gameknot.com" that were manually annotated by chess masters. The dataset is split into 298,000 examples.

[‡]Stockfish 13 is the state-of-the-art chess algorithm with an approximated FIDE rating of 3575.

[†]If the model plays an illegal move, it will be generate a new move. This process will be repeated three times until the model produces a legal move. If the model does not generate a legal move, the model resigns (forfeits)

4.1.2 Data Preprocessing

For the Kingbase and Kaggle datasets, we preprocess the data by removing move numbers, context of the games (player names, player ELO, game outcome string, etc.). In order to expedite training, we filter for post-processed strings with less than or equal to 512 characters, which is around 50 moves per player. We find that this prunes only the longest 2% of games from the database, which has a negligible impact on the size of our dataset. However, this choice will limit our bot from inference on games longer than around 70 total moves.

For the custom commentary dataset, the raw data is separated into .html files containing sequences of moves and the corresponding commentary. The PGN strings for related .hmtl files chronologically compile to build up a single chess game. Thus, in preprocessing, these moves are continually appended to build up the entire PGN string. The corresponding commentary is concatenated after the respective PGN string. This process is done separately to create a train and test file. Next, we post-processed by filtering to only keep strings with less than or equal to 512 characters–retaining over 98% of the data. We then remove nonsensical and short commentary (such as "? . ? ? ?"). Finally, we utilize the Python package `langdetect` to filter out any examples that are not in English. After these steps, we are left with 84% of the original data.

4.2 Evaluation Method

Given that the model performs two distinct tasks, it makes sense to analyse each of these tasks separately.

4.2.1 Chess Next-Move Evaluation

Our initial evaluation metric for next-move prediction was simply the percentage of illegal moves that the model attempts to make across games. This metric was selected as it helps us realise not only how well the model understands the rules of chess, but as the set of illegal moves varies by game configuration, it also allows us to gain insight into the model's ability to memorise game configurations based only on the set of transitions that it has previously observed.

Moreover, the model must not just become adept at making legal moves, but also demonstrate proficiency at executing a cogent strategy within legal constraints. This presented an interesting problem; how best do we analyse the proficiency of a model's movements in a game as complicated as Chess?

Therefore, we developed a custom scoring system to analyse model ability, based on move quality analysis provided by state-of-the-art Stockfish 13. We call this scoring system the *Model Proficiency Score*.

To calculate the Model Proficiency Score, consider a game of length n moves by each player. As such, $2n$ total moves are made throughout the game. Each model move is denoted as $m_i \quad i \in \{1, ..., n\}$. Additionally, we find the 'ideal' move at each point in the game by checking to see the move that Stockfish 13 predicts, and we denote this move as $s_i \quad i \in \{1, ..., n\}$.

Now, Stockfish 13 also provides an API that uses extra compute time to calculate a move-quality score based upon the current game state, $g_i \quad i \in \{1, ..., n\}$. This move-quality score is a positive or negative real number; the larger the number, the better the move. We can denote the move-quality score as $Qual(move, g_i) \in \mathcal{R}$ for any model or Stockfish 13 move. We therefore define the Model Proficiency Score for an entire game as follows:

$$MPS = \frac{\sum_{i=1}^{n} Qual(m_i, g_i) - Qual(s_i, g_i)}{\sum_{i=1}^{n} |Qual(s_i, g_i)|}$$

Essentially, the Model Proficiency Score for a game is the difference in quality between model moves and optimally predicted moves, then normalised by the quality of these optimal moves. A key fundamental assumption is that Stockfish 13 plays optimally, which is almost always true[11]. Therefore, in practice this metric becomes bounded $MPS \in [-1, 0]$ and therefore can be easily interpreted: for a given MPS, ChePT's moves are on average -MPS% worse than optimal.

We can then break MPS evaluations up by gamestate in a similiar way to how we evolve our finetuning objective over time; specifically, we can target early-game by analysing the first 8 moves from either side, mid-game by analysing moves 9 to moves 16 from each player, and late game by analysing moves 17 onwards. This gives us an insight into how the quality of the moves varies as the game is played.

Finally, to assess Next-Move Evaluation we also consider the number of wins, losses and stalemates (draws) that our model achieves against Stockfish 13.

4.2.2 Chess Commentary Evaluation

Evaluation of Chess Commentary provided to be relatively more difficult, given the massively unconstrained space that exists in Chess commentary data; some commentary gives formal chess analysis, other commentary contains jokes about the move played, and some commentary is just disparaging about the player's intellect. We will discuss this more in analysis, as it presents scope for future work.

Our first evaluation metric is calculating the BLEU scores of commentary predictions against our reference commentary. This evaluation metric is particularly limited, giving the sheer variation in commentary versus other tasks such as neural translation. Moreover, we only have one commentary entry for each given PGN string, which further limits the utility of the BLEU score. Despite this, it is still an interesting comparison metric to assess. Existing literature recommends using BLEU-4 score as the most popular metric in NLG tasks. However, Jhamtani et al. found that BLEU-4 was too restrictive[2] in the often short-form descriptions found in chess commentary. Therefore, we will report BLEU-2 alongside BLEU-4.

The second evaluation metric that we employ is hand-evaluation for a random set of extracted PGN strings and their predicted commentary. This is standardised in the commentary-evaluation literature, as per Zang et al. and Jahmtani et al.

4.3 Experimental Details

4.3.1 Next-Move Prediction Experiments

We began by utilizing the Python/Pytorch implementation of transformer architecture provided to us by CS224N, and heavily adapting it to our needs. We initially focused on the "next-move prediction" task, iteratively making design choices and analysing the following models:

1. **Baseline:** Vanilla model architecture with self-attention. 12,500,000 parameters; 4 layers; 8 heads; 256 hidden size; learning rate $1e^{-3}$; block size 1024. Adam Optimisation.
2. **Baseline-Large:** Vanilla model architecture with self-attention. 25,200,000 parameters; 32 layers; 8 heads; 256 hidden size; learning rate $1e^{-3}$; block size 1024. Adam Optimisation.
3. **Baseline-Small:** Vanilla model architecture with self-attention. 6,800,000 parameters; 16 layers; 8 heads; 256 hidden size; learning rate $1e^{-3}$; block size 512. Adam Optimisation.
4. **Many Heads:** Vanilla model architecture with self-attention. 2,400,000 parameters; 12 layers; 32 heads; 128 hidden size; learning rate $1e^{-3}$; block size 512. Adam Optimisation.

We originally settled on utilizing the "Many Heads" architecture as it was the fastest to train (with the least parameters) while producing the least number of invalid moves per game. We believe that this is due increasing the number of heads for attention–allowing the model to focus on more moves of a long chess PGN.

4.3.2 Commentary Experiments

After using this initial structure, we adapted the model for our second task of move commentary. We quickly found that the "Many Heads" model was struggling to learn commentary and tried a few new experiments. For each of the experiments below, we repeated the following process:

```
Pretrain Chess Moves -> Finetune Early-Game Chess Moves -> Finetune Mid-Game Chess Moves ->
                    Finetune Late-Game Chess Moves -> Finetune Commentary
```

For each finetune iteration, we utilized a lower learning rate ($1e^{-4}$ instead of $1e^{-3}$), as we noticed that the model loss was not converging well with the higher learning rate.

We ran the following experiments with the above protocol:

1. **Dual-Baseline:** Vanilla model architecture with self-attention. 12,800,000 parameters; 16 layers; 32 heads; 256 hidden size; learning rate $1e^{-3}$; block size 512. Adam Optimisation.
2. **Dual-Small:** Vanilla model architecture with self-attention. 6,400,000 parameters; 8 layers; 16 heads; 256 hidden size; learning rate $1e^{-3}$; block size 512. Adam Optimisation.
3. **ChePT Model:** Vanilla model architecture with self-attention. 9,600,000 parameters; 12 layers; 16 heads; 256 hidden size; learning rate $1e^{-3}$; block size 512. Adam Optimisation.

We settled on the final "ChePT Model" architecture. The "Dual-Baseline" model proved too large to train with a relatively small commentary dataset (200,000 datapoints). Similarly, the "Dual-Small" model likely did not have enough layers for two complicated tasks and failed to combine them–performing fine on either task independently, but poorly when trained to do both.

5 Results

Once again, we define early game (defined as the first 8 moves of a match for each player), mid-game (defined as moves 9 through 16 for each player), and late-game (moves 17 onwards for each player) in order to understand how statistics vary across full games.

5.1 Chess Next-Move Results

Below you may find our results for next-move prediction. All results are extracted from games played against Stockfish 13. MPS Denotes the Model Proficiency Score as explained previously.

Model	Games Played	Moves	First Illegal	Wins	Draws (Stalemates)	Losses	Illegal Move %
Noever et al. (Baseline)	X	**33**	X	X	X	X	10%
Pretrain	500	28	16	0	1	499	3.06%
Finetune Early	500	27	19	0	3	497	3.96%
Finetune Mid	500	29	22	0	4	496	2.31%
Finetune Late	500	32	**-**	0	**10**	**490**	**0%**

Table 1:
Next-Move Prediction Game Results

Model	Games Played	Full Game MPS	Early Game MPS	Mid Game MPS	Late Game MPS
Pretrain	500	-0.3095	-0.2896	-0.5098	-0.6109
Finetune Early	500	-0.2884	**-0.2102**	-0.3019	-0.5011
Finetune Mid	500	-0.2419	-0.2113	-0.2287	-0.4411
Finetune Late	500	**-0.2209**	-0.2139	**-0.2184**	**-0.3198**

Table 2:
Next-Move Prediction Model Proficiency Score

5.2 Chess Commentary Results

5.2.1 Automated Evaluation

For chess commentary, we again split the data into early/mid/late game, as delineated and defined above. From here, we find the following BLEU in Table 3, including reported full-game BLEU scores from our two baselines:

Model/Game Phase	BLEU-4 Score	BLEU-2 Score
Jhamtani et al. Baseline (Total Game Mean Average, Quality)	0.06	0.31
Zang et al. Baseline (Total Game Mean Average, Quality)	0.03	0.21
Our Model (Total Game Mean Average)	0.04	0.33
Our Model (First 4 Moves)	0.11	0.36
Our Model (Early Game)	0.08	0.28
Our Model (Mid Game)	0.03	0.18
Our Model (Late Game)	0.01	0.13

Table 3:
Model BLEU Scores

5.2.2 Manual Evaluation

For each of four game stages - first four moves, early game, mid game and late game, as defined above - we selected 50 examples to manually evaluate. While we do not have space to enumerate all of our analysis, here are a few choice examples (each taken from different games) that we feel are representative of how the model performs at various game phases.

These examples elucidate how the model is learning to understand Chess, as well as demonstrate how well (or, in some cases, poorly) it can articulate this. The first four moves, early game, mid game and late game are visualised on a Chess board on Figures 1 to 4 respectively.

First 4 Moves

```
Sample PGN: 1.  f4 e6 2.  g3 g5 3.  Nh3 g4
```

this PGN is visualised in Figure 1. The model predicts the commentary "white bird's first opening knight". This opening is indeed the little known 'Bird's Opening', and we see that white makes move Nh3 which is indeed a movement of the knight. Clearly, the model is demonstrating an understanding of common openings, the names of pieces, and the colour names of each side.

Early Game

```
Sample PGN: 1.  e4 e5 2.  Qh5 d6 3.  Bc4 Be6 4.  Bxe6 Qf6 5.  b4
```

Based on this PGN, the model predicts the commentary "this invaved my opponent, i got from the side i had a lunder of mind the second centre run.. that is the bishop would have been nothing or nevitable..". This example showcases some of the shortcomings of the model; namely that typos are frequent, and the commentary is not entirely cogent English.

However, it also demonstrates some fascinating insight by the model. As you can see in Figure 2, white has indeed invaded the opponent's side of the board unusually aggressively (and foolishly). The bishop was indeed the second run on the center of the board, and was a blunder - not a 'lunder'. By this, we mean that white has unnecessarily lost it's bishop when a winning play was easily available. Moreover, the commentary realises that black taking e6 is inevitable.

Mid Game

```
Sample PGN: 1.  d4 Nf6 2.  c4 g6 3.  Nc3 d5 4.  cxd5 Nxd5 5.  g3 Bg7 Bg2 6.  Nxc3 bxc3 7.  e5 e3
                                    8.  exd4 cxd4 9.  c5
```

For this example, the model predicts the commentary "white has taken a good center ..... ? ? ! ? ? ? ? ! also theu black position strong, black may advances the swap of b7, supporting.". White does indeed have strong central control, and from

Figure 1: Bird's Opening, accurately commentated by the model.

Figure 2: Highly aggressive play from white, with a blunder. Sentiment captured well by the model.

Figure 3: White has central control and the commentary-predicted move of b6 makes sense as black's next choice.

Figure 4: Commentary recognises forced mate from white's perspective.

white's perspective, it is entirely reasonable to expect that black may advance b7 to b6, in order to support the pawn at c5 - its most recent move.

However, it is also clear that the model is struggling grammatically; the sentence has a number of typos, and punctuation characters are seemingly randomly injected into the middle of the sentence.

Late Game

```
Sample PGN: 1.  d4 Nf6 2.  c4 g6 3.  Nc3 d5 4.  cxd5 Nxd5 5.  g3 Bg7 6.  Bg2 Nxc3 7.  bxc3 e5 8.
e3 exd4 9.  cxd4 c5 10.  Ba3 cxd4 11.  exd4 Qxd4 12.  Qxd4 f6 13.  Rd1 g5 14.  Qe4+ Kf7 15.  Rc1
                                Be6 16.  Rc8 Rxc8 17.  Bd6 Ke8
```

As you can see in figure 4, it is white's turn, and if white plays Qxe6, black must play Kd8, and then if white plays Qe7 we have checkmate. Therefore, if white plays Qxe6 and demonstrates a cogent strategy, white will win.

The commentary predicted from this game state is "good i have to take the bishop, it's all good, and then i would have mat". The commentary system recognises exactly what the white player must do in this circumstance, and commentates it. Again, the model clearly heavily suffers grammatically, which we will discuss further in analysis.

6 Analysis

6.1 Next-Move Prediction Analysis

Given that our MPS scores are always negative, it is clear that our model underperforms Stockfish 13. However, this is not at all disappointing, nor surprising. As we have discussed earlier, Stockfish 13 is state-of-the-art, and for all intents and purposes plays ideally. Moreover, we are heavily data limited here; we will never approach the performance of Stockfish using human-generated training data, as humans are significantly weaker than Stockfish.

Crucially, our move prediction massively outperforms the only other transformer-based chess engines that we encountered during our research. We make fewer illegal moves than Noever et al. - in the end, making zero illegal moves - and our final model makes moves that are on average just 20% worse than those predicted by SOTA (significantly better performance than Noever et al). It shows a fundamentally excellent understanding for the rules of Chess, and reflects well on our refined choice of finetuning objective (which Noever et al. did not employ).

We do notice a significant performance decrease throughout the games; early game performance is extremely strong, whereas by the late game, moves are far weaker. This raises questions about what the next-move prediction model is actually doing; is the model actually learning Chess, or is it learning to play rote responses to moves that it has seen before? Rote memorisation would surely fail later in the game, where certain game configurations have will never have been encountered in training.

Some factors do give us confidence that the model is truly learning; in `gui_inference.ipynb`, one can find a script that allows humans to play versus the trained model in a GUI. Playing absurd moves (moves so atrocious that the model is unlikely to have observed them in the high level games that it was trained on) through this interface does result in the model effectively capitalising on the mistake, showing that it does indeed understand strategy beyond playing rote response.

6.2 Commentary Analysis

While our BLEU scores are extremely weak, many of the conclusions that can be drawn by instead analysing the commentary manually while observing the board configuration are far more promising. BLEU is a challenging metric to use in this context, because we only have a single reference commentary to compare predictions to. The space of potential commentary is extremely large. For example, ChePT could choose to commentate on game context, the most recent move quality, the forthcoming strategy, etc.

Moreover, the BLEU score heavily penalises spelling mistakes - something that our model makes a large number of. This is very likely because we never pretrained the model on an English corpus. Moreover, the data in our commentary dataset is also typo-ridden, and often makes little grammatical sense itself.

When we look beyond the model's inconsistent grammar, however, we do find genuinely fascinating insights that it has into the state of the game, and into the how the game is likely to progress. The model consistently comments on pieces that it is actively defending, moves that it might choose to make in the future, and mistakes that it - or the opponent - may have made.

As such, it is clear that the model's fundamental understanding of Chess is unquestionable. The model falls short solely with it's knowledge of the English language.

In their papers, Zang et al. and Jhamtani et al. break commentary prediction down into five categories; Description, Quality, Comparison, Planning and Context. In contrast, we have a single commentary objective owing to our unified model and PGN-integrated commentary approach, and this makes comparison difficult. Our commentary integrates all of these above categories together, and allows the model to predict the commentary it is trained to deem most relevant - an amalgamation, and in practice a subset, of these categories.

6.3 Potential Improvements & Future Work

A clear avenue for improvement is addressing these noted issues with the English language modelling. We elected not to pre-train the model on an English corpus because we were concerned that it would compromise the model's Chess move prediction. We also had limited computing resources and time; especially given that we had consumed significant resources ensuring that move prediction exceeded that of Noever et al., and that with the same model we could at least provide basic commentary. With more time and computing resources, it would be extremely interesting to perform further experimentation on other pretraining tasks, and with larger models, in order to explore imbuing the model not just with a competency in playing chess, but also with the English language itself.

Moreover, collecting better commentated chess gameplay would significantly bolster our results. As discussed, our commentated dataset was compromised of disparate samples gathered through a web crawler, and quality between samples varied significantly. If we had access to a better, and standardised dataset, our model would likely be able to utilise its fundamental understanding of Chess to provide significantly better and more consistent commentary.

With respect to move prediction, while our engine is strong, it does begin to suffer towards the end of longer games. To improve this, we would require more computing resources so that we could train on games longer than 512 characters efficiently, but also increase our attention and embedding sizes to more effectively carry game state information throughout the model.

Finally, in relation to our last note in Commentary Analysis - discussing the different categories of commentary performance - one important avenue for future work would be providing a more comprehensive analysis on how our commentary performs across the categories defined by Jhamtani et al. and Zang et al. This would require a larger team of human volunteers to categorise samples of our commentary, and unfortunately given our dual goals of move prediction and commentary, we were unable to investigate this fully.

7 Conclusion

Our results here raise seriously interesting questions within the realm of model explainability. If our Chess commentary was improved, potentially through exploring the avenues suggested above, it is entirely conceivable that our Chess model (which is already more powerful than most amateur players) could possess the ability to fully and insightfully explain the decisions that it makes. Already, we see insights into this propensity, and with additional resources we could certainly further this process. This not only has implications for developing teaching tools for strategy games like Chess, but also more generally in the growing field of Deep Learning model interpretability and explainability.

Moreover, our strong results for move prediction in particular yield interesting observations for the potential role of deep sequence models in problem spaces traditionally reserved for reinforcement learning approaches. In this paper, we have demonstrated that sequence models originally designed for language may not necessarily be able to outperform state-of-the-art reinforcement learning systems in the context of strategy games like Chess, but they are able to perform extremely competently - certainly more competently than most humans. As such, it raises the question of whether these results might be more portable; in situations where state and actions spaces are hard to define, but data is prolific, it is certainly conceivable that a similar sequence modelling approach may hold considerable merit. Moreover, it may be indeed conceivable to develop a single sequence model that can competently play a number of sequential strategy games - this in particular would be fascinating to explore.

Finally, the benefits of combining the two models together are clear; compared to existing commentary models (such as those by Jahmtani et al. and Zang et al.), and existing transformer chess engines (such as Noever et al.) we significantly reduce the number of total model parameters, likely due to the fact that our model can share learnings that apply to developing chess and commentary insights across both tasks. Our model is the first to do this; Jahmtani et al. do not perform move prediction, and Zang et al. use an external engine to drive move prediction. Our model demonstrates significantly greater insight into Chess than Jahmtani et al., who focus more on literal commentary of the current gamestate, rather than showing true insight into the nuances of Chess.