**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**….…oOo…….**



**BÁO CÁO: TRÍ TUỆ NHÂN TẠO**

**Đề tài: Ứng dụng cờ vua tích hợp**

**module chess AI ứng dụng Reinforcement learning và Alpha-Beta pruning**

Môn học: Trí tuệ nhân tạo

(2425II\_INT3401E\_5)

Giảng viên: TS. Nguyễn Văn Vinh

ThS. Đỗ Minh Đức

Sinh viên thực hiện: Trần Sĩ Việt – 22026187

Mục lục

**I.Lời mở đầu2**

II. Giao diện người dùng và cấu trúc mã nguồn 3-5

III.Module chessAI6

**1.Các thành phần chính6-13**

2.Thuật toán tìm kiếm Minimax kết hơp Alpha-Beta Pruning14-15

3.Quy trình huấn luyện AI16-27

IV.Lời kết28

**I.Lời mở đầu**

* Module chess AI ứng dụng Reinforcement learning và Alpha-Beta pruning được phát triển dựa trên ngôn ngữ lập trình Python tích hợp neural network và thuật toán tìm kiếm minimax kết hợp với cắt tỉa alpha beta để đánh giá trạng thái bàn cờ và đưa ra nước đi thích hợp
* Giới hạn thời gian ra nước đi của module trên là <2s và elo hiện tại vẫn bị kẹt ở ~1499,92 elo
* Module trên được tích hợp vào chương trình python cung cấp giao diện đánh cờ cho phép chọn đối thủ, độ khó(chỉ cho stockfish engine chess em lấy trên mạng) và chế độ huấn luyện AI.

**II. Giao diện người dùng và cấu trúc mã nguồn**

1.Mô hình độ họa

Kiến trúc độ họa hiển thị:

* ‘Mode\_selection’: Chọn chế độ chơi:

A screenshot of a game mode

AI-generated content may be incorrect.

+AI vs AI

A screenshot of a computer game

AI-generated content may be incorrect.

+ Human vs AI

* AI\_selection:Chọn AI đối thủ (Stockfish hoặc Module AI được train(ChessDove))

A screenshot of a computer

AI-generated content may be incorrect.

* Elo\_selection: Chọn mức Elo (chỉ với Stockfish)

A screenshot of a computer

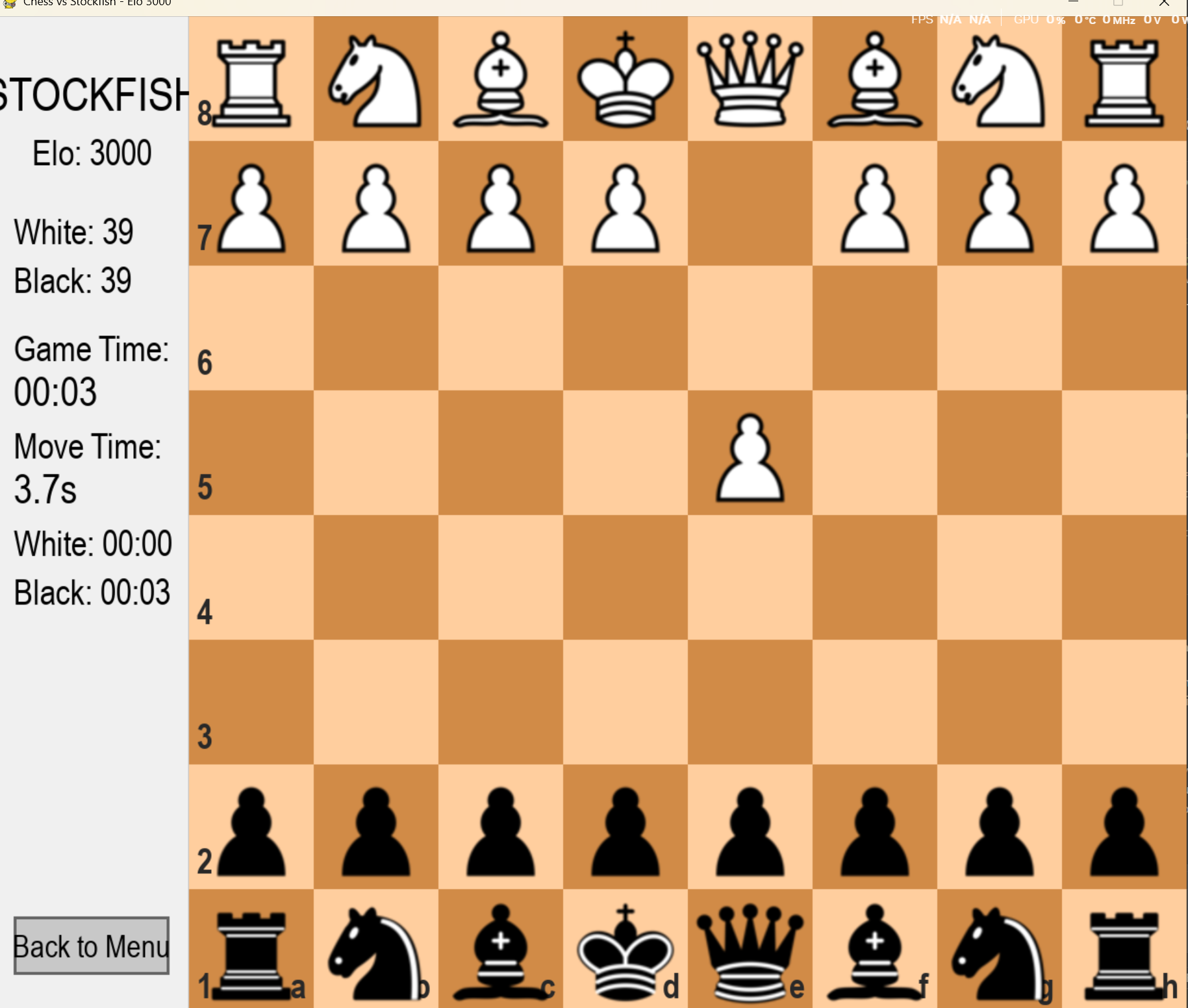
AI-generated content may be incorrect.

* Side\_selection:Chọn quân trắng và quân đen

A screenshot of a computer

AI-generated content may be incorrect.

* Giao diện đánh cờ giữa người dùng và AI:



2.Cấu trúc mã nguồn

Chess project/

├── main.py   # Điểm vào chính, quản lý menu và điều hướng

├── graphic.py  # Xử lý độ họa chung và các hàm vẽ

├── gui.py         # Giao diện khi người chơi đấu với Stockfish

├── guix.py       # Giao diện khi người chơi đấu với module AI

├── chess\_logic.py  # Logic cờ vua

├── chess\_clock.py  # Đồng hồ đếm thời gian

├── assets/         # Thư mục chứa ảnh ô bàn cờ, quân cờ,…

│   ├── board.png

│   ├── white\_pawn.png

│   ├── black\_pawn.png

│   └── ... (other chess piece images)

├── stockfish/         # Stockfish chess engine download trên mạng

│   └── stockfish.exe

└── ChessDove/     # Thư mục chứa mã nguồn AI tự phát triển

    ├── ChessDove.py # Module AI chính

├── ComputingAI.py # Đánh giá vị trí các quân cờ

├── play\_vs\_stockfish.py # Đấu cùng stockfish để lấy dữ liệu huấn luyện

├── train\_evaluator.py # Huấn luyện mạng neural

├── evaluator.py # Neural network

├── GamePlay.py # Self-play và học tăng cường

├── watch\_ai\_battle.py # Giao diện xem AI đấu với nhau

└── evaluate\_elo.py # Đánh giá điểm Elo của AI

**III.Module chess AI (chessdove)**

1. Các thành phần chính
2. Logic AI

* ChessDove.py: file chứa logic chính của AI bao gồm:

+ Biểu diễn bàn cờ dưới dạng tensor 8x8:Quân trắng từ +1(tốt) đến +6 (vua), quân đen từ -1 (tốt) đến -6 (vua), ô trống=0A screenshot of a computer program

AI-generated content may be incorrect.

+ Class ChessDove:

* Hàm \_\_init\_\_(self, depth=3, use\_nn=True):
  + Khởi tạo AI với độ sâu tìm kiếm là 3 và flag sử dụng mạng neural
  + Tải mô hình đánh giá từ file best\_eval.pt
  + Khởi tạo bảng giá trị quân cờ ban đầu và bảng vị trí
* Hàm get\_move(self, board, time\_limit=2.0):

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

* Đưa ra nước đi tốt nhất (best move) trong 2 s
* Sử dụng tìm kiếm sâu dần (interative deepening)
* Kiểm tra các nước có thể dẫn tới chiếu hết
* Hàm minimax(self, board, depth, alpha, beta, maximizing, start\_time, time\_limit)

A screen shot of a computer program

AI-generated content may be incorrect.

* Thuật toán tìm kiếm minimax kết hợp với alpha beta pruning (nói chi tiết bên dưới)
* Tích hợp một số biến để kiểm soát thời gian
* Hàm evaluate\_position(self, board):



A computer screen shot of text

AI-generated content may be incorrect.

* Kết hợp đánh giá từ neural network và đánh giá thủ công để đánh giá vị trí các quân cờ dựa trên điểm(score)
* Xử lý đặc biệt cho trạng thái kết thúc (chiếu hết, hòa)
* Phạt khi vua di chuyển sớm và thưởng khi nhập thành

+ Các hàm tương thích ngược

* get\_legalmoves, make\_move, is\_game\_over: dành cho codebase cũ sử dụng tensor
* choose\_best\_move:wrapper để kết nối với API cũ
* change turn(): đổi lượt toàn cục
* ComputingAI.py: Cung cấp công cụ tính toán, hằng số để đánh giá vị trí trạng thái bàn cờ, các quân cờ

+ Hằng số và bảng dữ liệu:

A computer screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Các bảng giá trị này khuyến khích :
* Tốt tiến lên phía trước và chiếm trung tâm
* Mã và tượng kiểm soát trung tâm
* Xe kiểm soát các cột mở
* Vua an toàn ở hang đầu và góc bàn cờ trong giai đoạn đầu và giữa ván
* Hậu kiểm soát bàn cờ từ xa và ép góc trong một số tình huống

+ Hàm chuyển đổi định dạng

* board\_to\_ tensor(broad): Chuyển đổi đối tượng (vị trí các quân cờ) từ board thành tensor 8x8
* tensor\_to\_board(tensor): Chuyển đổi dữ liệu tensor thành vị trí các quân cờ trên board của python-chess

+ Đánh giá vị trí :

* Hàm evaluate\_conventional(board, piece\_values, piece\_position\_scores)
* Tính toán giá trị điểm dựa trên vị trí từng quân cờ
* Đánh giá dựa trên mobility( số lượng nước đi hợp lệ )
* Xử lý một số trường hợp đặc biệt như nhập thành và phong cấp
* Hàm material\_diff(board): Tính chênh lệch vật chất giữa trắng và đen

1. Neural network

+ Code:

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A computer screen shot of text

AI-generated content may be incorrect.**

* Neural network “EvaluatorNet “ được thiết kế để đánh giá trạng thái bàn cờ vua dạng tensor (8x8) và trả về giá trị trong khoảng [-1,1] biểu thị cho kết quả trận đầu(-1 là đen thắng, 1 là trắng thắng, 0 là cân bằng)
* Mạng sử dụng PyTorch và xử lý tensor để biểu diễn bàn cờ

+ Kiến trúc mạng

* Mạng gồm hai phần chính các lớp tích chập (convolutional layers) và các lớp kết nối đầy đủ (fully connected layers) với độ sâu tìm kiếm mặc định là 3 nước đi để có thể dùng trong thuật toán tìm kiếm minimax kết hợp alpha beta pruning với độ sâu tương tự.
* Lớp tích chập :

. conv1: lớp tích chập đầu tiên nhận đầu vào là một tensor (1 kênh, 8x8) và tạo ra 16 kênh đặc trưng (feature maps) với kernel 3x3, padding=1 để giữ kích thước 8x8

. conv2: lớp tích chập thứ hai, nhận 16 kênh từ conv1, tạo ra 32 kênh đặc trưng, cũng với kích thước 8x8

. Hàm ReLU được áp dung sau mỗi lớp tích chập để them tính phi tuyến.

* Lớp kết nối đầy đủ:

. fc1: Nhận tensor được làm phẳng từ lớp tích chập (kích thước 32x8x8=2048) và giảm xuống 128 đơn vị, với hàm kích hoạt ReLU

. fc2: Giảm từ 128 đơn vị xuống 1 đơn vị, tạo ra giá trị đầu ra.

. Hàm tanh được áp dụng giới hạn đầu ra trong khoảng [-1,1].

2. Thuật toán tìm kiếm Minimax kết hợp Alpha beta Pruning

- Thuật toán Minimax là một phương pháp tìm kiếm dựa trên cây trò chơi (game tree) để xác định nước đi tốt nhất trong các game 2 player như cờ vua. Trong đoạn code bên trên, thuật toán này được tích hợp, cải tiến bằng cắt tỉa Alpha-Beta để giảm số lượng nút cần đánh giá và có thêm kiểm soát thời gian để đảm bảo AI phản hồi về trong giới hạn thời gian cho phép

+ Giá trị tốt nhất (best\_ score): Điểm số đánh giá bàn cờ .

+ Nước đi hay nhất (best \_more):Nước đi đem lại kết quả tốt nhất

Cấu trúc hàm:

” def minimax(self, board, depth, alpha, beta, maximizing, start\_time, time\_limit)”

+ Tham số :

* self: chỉ ChessDove AI chứa hàm này, có phương thức evaluate\_position(dùng mạng nơ-ron EvaluatorNet để đánh giá).
* board: Bàn cờ từ thư viện python-chess , lưu trạng thái hiện tại và cung cấp danh sách nước đi hợp lệ (board.legal\_moves).
* depth: Độ sâu tìm kiếm còn lại (số nước đi tối đa để xem xét)
* alpha: Giá trị tốt nhất mà bên Max (trắng) chắc chắn đạt được (ban đầu là -vô cực).
* beta: Giá trị thấp nhất mà bên Min (đen) chắc chắn đạt được (ban đầu là +vô cực).
* maximizing: Cờ boolean, True nếu là lượt của bên Max (trắng), False nếu là lượt của bên Min (đen).
* start\_time: Thời điểm bắt đầu tìm kiếm (dùng để kiểm soát thời gian).
* time\_limit: Giới hạn thời gian (giây) cho toàn bộ quá trình tìm kiếm.

+ Đầu ra :

* best\_score: Điểm số tốt nhất từ góc nhìn của bên đang chơi.
* best\_move: Nước đi dẫn đến điểm số tốt nhất (hoặc None nếu không có nước đi hợp lệ hoặc hết thời gian).
* Cách đánh giá giá trị nước đi:

+ Đánh giá bàn cờ dựa trên hàm evaluate\_position

* Đầu vào: Bàn cờ(board) được mã hóa thành tensor kích thước 8x8,mã hóa vị trí các quân cờ thông qua mạng neural và trả về kết quả [-1,1] biểu thị lợi thế đang nghiêng về bên nào

. Gần 1ưu thế về bên trắng.

. Gần -1 ưu thế về bên đen

. Gần 0 cân bằng cho hai phe

* Kết quả : điểm số từ evaluate\_position được dùng cho các nút lá (khi depth ==0), khi trò chơi kết thúc hoặc khi hết thời gian lượt đi
* Cách Minimax chọn nước đi

+ Game tree: bao gồm các nút/lá là trạng thái các nhánh là nước đi ở mỗi mức độ sâu, nó mô phỏng các nước đi hợp lệ và đánh giá giá trị bàn cờ ở các nút/lá thông qua hàm evaluate\_postion.

+ Bên Max(trắng): chọn nước đi dẫn đến điểm số cao nhất

+ Bên Min(đen): chọn nước đi dẫn đến điểm số thấp nhất

+ Cắt tỉa Alpha-beta: Giảm số lượng nhánh cần đánh giá bằng cách loại bỏ các nhánh không thể cải thiện kết quả:

* Nếu một nhánh của Max cho điểm số thấp hơn giá trị beta,Min sẽ không chọn nhánh đó🡪 bỏ qua các nhánh tiếp theo
* Tương tự với một nhánh của Min, nếu cho điểm số cao hơn alpha, Max sẽ không chọn nhánh đó và bỏ qua các nhánh tiếp theo

+ Sau khi duyêt game tree đến độ sâu depth (cụ thể ở đây là 3) (hoặc dừng sớm do hết thời gian /trò chơi kết thúc) ,thuật toán trả về nước đi(best\_move) dẫn đến trạng thái tốt nhất từ góc nhìn của bên đang chơi .

1. Quy trình huấn luyện AI

* Module AI ChessDove được phát triển thông qua một quy trình huấn luyện nhiều giai đoạn khác nhau kết hợp giữa học tăng cường reinforcement learning và học có giám sát (supervised learning).
* Sơ đồ quy trình huấn luyện:

A screen shot of a diagram

AI-generated content may be incorrect.

* Khởi tạo mô hình : Giai đoạn đầu tiên tập trung vào việc tạo ra một mô hình đánh giá ban đầu, sử dụng mạng nơ-ron EvaluatorNet. Do chưa có dữ liệu thực tế, hệ thống sinh ra dữ liệu tổng hợp, bao gồm các bàn cờ ngẫu nhiên với các quân cờ được đặt ở các vị trí ngẫu nhiên. Giá trị đánh giá cho mỗi bàn cờ được tính dựa trên hàm material\_evaluation, có thể là sự chênh lệch về giá trị quân cờ giữa hai bên (ví dụ: tốt = 1, mã = 3, v.v.). Quá trình này giúp mô hình ban đầu học cách đánh giá các vị trí cơ bản, tạo nền tảng cho các giai đoạn sau

A screen shot of a computer program

AI-generated content may be incorrect.

+ Tạo dữ liệu huấn luyện ban đầu

A screen shot of a computer program

AI-generated content may be incorrect.

+ Huấn luyện khởi tạo:A computer screen shot of a program

AI-generated content may be incorrect.

* Kiến trúc mạng neural:

+ Mạng Đánh Giá Vị Trí (Evaluation Network)(bên trên)

+ Mạng Policy cho Reinforcement Learning(trong Gameplay.py): Một mạng nơ-ron policy được thiết kế để chọn nước đi, với hai kiểu thiết kế: mạng thông thường và mạng "aggressive" (tấn công), nhằm tạo sự đa dạng trong chiến lược. Mạng này nhận đầu vào là tensor biểu diễn bàn cờ và xuất ra xác suất cho các nước đi có thể

A computer screen shot of a program code

AI-generated content may be incorrect.

* Thu thập dữ liệu huấn luyện thông qua đấu với stockfish(play\_vs\_stockfish.py)

+ Hệ thống chơi một số lượng lớn trận đấu với Stockfish, được cấu hình với độ sâu tìm kiếm nhất định (ví dụ: 10 nước). Trong mỗi trận đấu, các vị trí bàn cờ được ghi lại, cùng với kết quả cuối cùng (thắng, thua, hoặc hòa). Giá trị cho mỗi vị trí được gán dựa trên kết quả, với hệ số chiết khấu (discount factor) để phản ánh tầm quan trọng của các vị trí gần kết thúc.A screen shot of a computer program

AI-generated content may be incorrect.

+ Chuẩn bị dữ liệu huấn luyện

A screen shot of a computer program

AI-generated content may be incorrect.

+ Huấn luyện neural network(Train\_evaluator.py):

* Dữ liệu thu thập được sử dụng để huấn luyện EvaluatorNet, với mục tiêu giảm sai số bình phương trung bình (MSE) giữa dự đoán và giá trị thực tế. Quá trình huấn luyện bao gồm chia dữ liệu thành tập huấn luyện và validation, huấn luyện qua nhiều epochs, và lưu mô hình có hiệu suất tốt nhất trên tập validation. Kết quả là mạng nơ-ron học được cách đánh giá các vị trí bàn cờ tương tự như Stockfish, cải thiện khả năng ra quyết định của AI.
* Cấu hình huấn luyện:

A black rectangle with white text

AI-generated content may be incorrect.

* Huấn luyện chi tiết:

A computer screen shot of a program code

AI-generated content may be incorrect.

* A screen shot of a computer screen

  AI-generated content may be incorrect.Tự học tăng cường qua selfplay:

+ Sau khi có mô hình đánh giá ban đầu, ChessDove AI tiến hành học tăng cường thông qua self-play, trong đó AI chơi cờ với chính mình để thu thập kinh nghiệm và cải thiện chiến lược.

+Quy trình selfplay:

A computer screen with many colorful text

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

Trong self-play, hai phiên bản AI (có thể là cùng một mạng hoặc khác nhau) chơi với nhau. Các trạng thái bàn cờ, nước đi, và phần thưởng được ghi lại, với phần thưởng dựa trên sự thay đổi về vật chất (material difference) hoặc kết quả trận đấu. Để tránh hòa do tính đối xứng, hệ thống áp dụng các kỹ thuật như thêm bias ngẫu nhiên, tăng epsilon trong epsilon-greedy (0.2), và sử dụng mạng khác biệt.

+ Cập nhật lại mạng policy: Sau mỗi trận đấu, mạng policy được cập nhật bằng thuật toán REINFORCE

A computer screen shot of a program

AI-generated content may be incorrect.

Tính toán discounted rewards và tối ưu hóa để tối đa hóa tổng phần thưởng mong đợi.

A screen shot of a computer code

AI-generated content may be incorrect.

* Đánh giá và tinh chỉnh: Giai đoạn cuối tập trung vào đánh giá hiệu suất của ChessDove AI thông qua các trận đấu với Stockfish và tính toán điểm Elo.

+ Đánh giá elo: AI chơi một số trận đấu (ví dụ: 30 trận) với Stockfish ở độ sâu nhất định, và kết quả (thắng, thua, hòa) được sử dụng để tính điểm Elo, so sánh với Elo ước lượng của Stockfish.

A computer screen shot of a program

AI-generated content may be incorrect.

+ Công thức tính điểm elo

A screen shot of a computer program

AI-generated content may be incorrect.

* Kết quả sau khi huấn luyện cho tới hiện tại:

+ Elo~1499.9

**IV. Lời kết :**

* Ứng dụng cung cấp giao diện người dùng thân thiện dễ dàng thao tác
* Tuy nhiên module AI do em làm còn nhiều thiếu xót :

+ Giai đoạn đầu ván: AI đã học được các khai cuộc chuẩn, chơi khá tốt

+Giai đoạn giữa ván: Khả năng nhận diện cấu trúc tốt, tìm kế hoạch tốt

+Giai đoạn cuối ván: Còn yếu, đặc biệt trong các tình huống hết quân phức tạp

+Phòng thủ: Khá tốt trong việc nhận diện và tránh các chiến thuật phổ biến

+Tấn công: Học được các mẫu tấn công cơ bản, nhưng còn thiếu tính sáng tạo khá nhát

+ Điểm elo thấp tăng rất chậm kể từ mốc 1400 do khả năng tự học còn chậm tốn thời gian

+ Mạng neural đơn giản

* ChessDove hiện đạt mức Elo 1499,93, tương đương với người chơi nghiệp dư trung bình, nhưng còn nhiều dư địa để cải thiện thông qua việc tối ưu hóa kiến trúc, thu thập thêm dữ liệu huấn luyện chất lượng cao, và tiếp tục tinh chỉnh thuật toán tìm kiếm.
* Đó là toàn bộ chương trình mà em đã cố gắng làm dù biết bản thân em chưa học được nhiều và còn yếu kém nhiều khoản cũng như thiếu xót nên mong được thầy góp ý chỉ bảo thêm ạ.