ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO BÀI TẬP LỚN BỘ MÔN TRÍ TUỆ NHÂN TẠO
NĂM HỌC 2024 – 2025

NHÓM 9

Môn học	Trí tuệ nhân tạo
Lớp học phần	INT3401E 5
Thành viên	Trương Minh Việt - 22028314

HÀ NỘI - 2025

Mục lục

1 Phân chia công việc

Để đảm bảo sự tiến triển của dự án, công việc được phân chia rõ ràng giữa các thành viên trong nhóm. Các nhiệm vụ chính bao gồm:

Họ và tên	**MSSV**	**Đóng góp**	**Công việc**
Trương Minh Việt (Nhóm trưởng)	22028314	100%	Lên ý tưởng, tìm hiểu công nghệ triển khai chung để làm ứng dụng. Làm tài liệu, slide, thuyết trình, quay video demo sản phẩm.

Bảng 1: Phân chia công việc trong nhóm

2 Giới thiệu

Trí tuệ nhân tạo (AI) được định nghĩa là khả năng trí tuệ được thực hiện bởi hệ thống máy tính. AI phân biệt chính nó với các hình thức học tập ở động vật và con người. Một cách cụ thể, quá trình phát triển trí tuệ nhân tạo khởi đầu từ việc lập trình các hành vi. Trong lĩnh vực trò chơi điện tử, trí tuệ nhân tạo thường đóng vai trò thiết yếu trong tương tác với người dùng, điển hình là một thực thể đối kháng với người chơi. Tồn tại các kịch bản mà AI được thiết kế để hỗ trợ, cũng như các trường hợp AI thực hiện đồng thời cả vai trò đối kháng và hỗ trợ người dùng. Trong phạm vi nghiên cứu AI, có sự hiện diện của nhiều thuật toán truyền thống bên cạnh các thuật toán học máy. Đối tượng nghiên cứu của dự án này là trò chơi cờ vua. Ngôn ngữ lập trình Python đã được ứng dụng để phát triển trò chơi cờ vua tích hợp trí tuệ nhân tạo trong khuôn khổ dự án. Python là một ngôn ngữ lập trình được đánh giá cao về tính tinh gọn và hiệu quả, luôn hướng đến mục tiêu đơn giản hóa và tối ưu hóa quy trình phát triển.

3 Tổng quan tài liệu

3.1 Trí Tuệ Nhân Tạo Trong Game (Game AI) là gì?

Trí tuệ nhân tạo trong game (Game AI) chủ yếu tập trung vào các tác vụ mà một thực thể cần thực hiện trong một tình huống cụ thể. Trong các tài liệu truyền thống về AI, điều này được gọi là một "tác nhân thông minh" (smart agent), và các tác nhân này thường là nhân vật trong game, nhưng cũng có thể là phương tiện, robot, hoặc trong một số trường hợp, là các khái niệm trừu tượng đại diện cho toàn bộ một nhóm thực thể. Sau cùng, chính thực thể đó quan sát môi trường, đưa ra quyết định tương ứng và hành động dựa trên các quyết định đó. Quá trình này đôi khi được gọi là chu trình nhận thức / tư duy / hành động.

- **Nhận thức (Sense):** Tác nhân phát hiện – hoặc được thông báo về – các yếu tố trong môi trường có thể ảnh hưởng đến hành vi của nó (ví dụ: các mối đe dọa gần đó, các điểm

ưa thích cần điều tra).

- **Tư duy (Think):** Tác nhân đưa ra quyết định về hành động cần thực hiện để phản hồi (ví dụ: xem xét liệu có an toàn để thực hiện một nước đi có giá trị thắng cao hơn hay không).
- **Hành động (Act):** Tác nhân thực hiện các hành động để đưa quyết định trước đó vào thực tế (ví dụ: bắt đầu di chuyển theo một lộ trình về phía kẻ thù hoặc về phía vật phẩm, v.v.).

Trong các vấn đề AI của thế giới thực, đặc biệt là những vấn đề liên quan đến việc thu thập thông tin tại bất kỳ thời điểm nào, trọng tâm thường đặt nặng vào phần "nhận thức" của chu trình này. Điều này thường được thực hiện thông qua một số dạng học máy, vốn đặc biệt phù hợp để thu thập lượng lớn dữ liệu nhiễu trong thực tế và trích xuất cũng như hiểu thông tin ngữ nghĩa.

Các trò chơi điện tử có tính độc đáo ở chỗ chúng thường không yêu cầu một hệ thống phức tạp để trích xuất thông tin này. Kết quả là, phần "nhận thức" của chu trình thường đơn giản hơn nhiều, và sự phức tạp đến từ việc áp dụng các khía cạnh "tư duy" và "hành động".

3.2 Những Hạn Chế Trong Phát Triển Trí Tuệ Nhân Tạo Game

Nhìn chung, trí tuệ nhân tạo trong game (Gaming AI) có một số giới hạn nhất định, bao gồm:

- Thông thường, các hệ thống này không được "huấn luyện trước" như các thuật toán học máy. Việc tạo ra một mạng nơ-ron trong quá trình phát triển và học theo cách tốt nhất là không thực tế. Trò chơi này nhìn chung mang tính giải trí và thử thách hơn là "tối ưu".
- Các tác nhân có thể được huấn luyện để tìm kiếm phương pháp tiếp cận tốt nhất cho một người, nhưng các nhà thiết kế không thực sự mong muốn điều đó. Quá trình này được thực hiện trong thời gian thực. Điều này có nghĩa là thuật toán không thể độc chiếm việc sử dụng bộ xử lý trong một thời gian dài để đưa ra quyết định trong bối cảnh này.
- Hầu hết các trò chơi chỉ có từ 16ms đến 33ms để thực hiện toàn bộ quá trình xử lý cho khung hình đồ họa tiếp theo, do đó, ngay cả một quyết định ngắn 10ms cũng là quá dài.
- Ít nhất một phần của hệ thống không được mã hóa cứng (hard-coded), và trong trường hợp cơ sở dữ liệu, lý tưởng nhất là nó cho phép những người không phải lập trình viên thực hiện các điều chỉnh nhanh chóng hơn.

Với những điều này, chúng ta có thể xem xét một phương pháp tiếp cận AI rất đơn giản để xử lý hiệu quả toàn bộ chu trình nhận thức / tư duy / hành động.

3.3 Giới thiệu về Cờ Vua

Cờ vua là một trò chơi có thông tin hoàn hảo, nơi mọi chi tiết đều được công khai giữa hai người chơi, khiến nó trở thành môi trường lý tưởng để thử nghiệm các thuật toán trí tuệ nhân

tạo. Ban đầu, sức mạnh tính toán và khả năng xử lý brute-force (duyệt toàn bộ) của máy tính được xem là ưu thế chính. Mặc dù bàn cờ chỉ có 64 ô và tối đa 32 quân cờ, số lượng thế cờ hợp lệ rất lớn; tuy nhiên, máy tính có khả năng xử lý các nước đi nhanh và chính xác hơn con người. Trong quá khứ, hạn chế về bộ nhớ và năng lực xử lý đã giới hạn các chương trình cờ vua. Cùng với sự phát triển về tốc độ máy tính và kỹ thuật lập trình, đặc biệt là việc ứng dụng cây trò chơi Minimax, các chương trình cờ vua đã trở nên mạnh mẽ và thông minh hơn đáng kể. Thuật toán Minimax được coi là một trong những thuật toán AI nền tảng và phổ biến nhất trong lĩnh vực lập trình game đối kháng theo lượt.

4 Các Thuật Toán Ra Quyết Định

4.1 Thuật Toán Tìm Kiếm

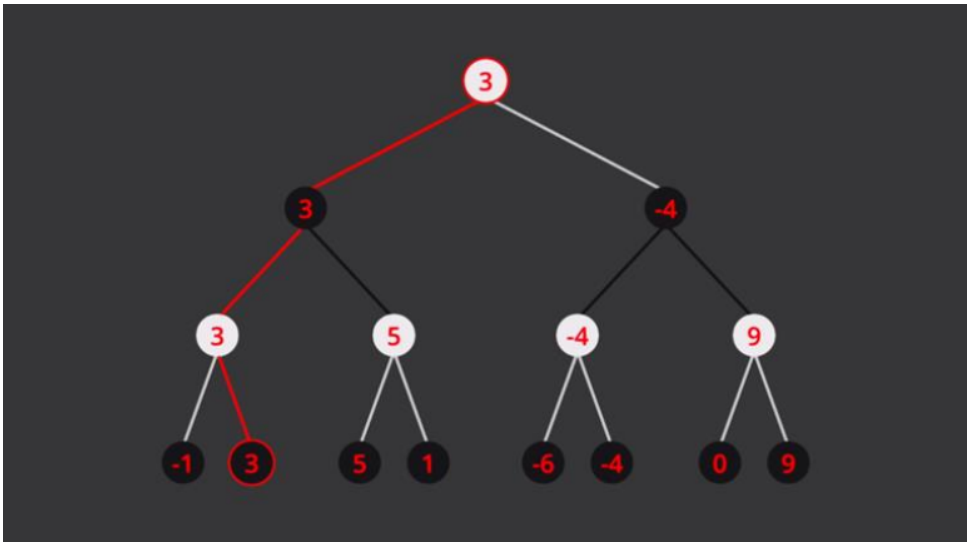
Động cơ cờ vua sử dụng hai thuật toán tìm kiếm cơ bản để ra quyết định: Minimax và Alpha-Beta Pruning.

4.1.1 Thuật Toán tìm kiếm Minimax

Minimax là thuật toán đệ quy được sử dụng để xác định nước đi tối ưu trong trò chơi có tổng bằng không như cờ vua. Cách triển khai tuân theo các nguyên tắc sau:

- Thuật toán khám phá cây trò chơi đến độ sâu đã xác định trước (được định nghĩa trong hằng số `DEPTH`)
- Ở mỗi cấp độ, nó luân phiên giữa quan điểm của người chơi tối đa hóa và tối thiểu hóa
- Tại các nút lá (độ sâu=0), vị trí được đánh giá bằng hàm đánh giá
- Thuật toán chọn các nước đi tối đa hóa điểm đánh giá cho AI và tối thiểu hóa điểm cho đối thủ

Sau đây là ví dụ về một cây tìm kiếm như vậy của thuật toán minimax. Chúng ta có thể thấy rằng thuật toán tìm thấy một đường đi có khả năng chiến thắng trò chơi cao nhất



Hình 1

Tầng lá (Leaf Nodes) Các giá trị tại các nút lá là: $(-1, 3), (5, 1), (-6, -4), (0, 9)$

Tầng trên (Nút trắng - tầng MAX) Tại tầng này, mỗi nút sẽ chọn giá trị lớn nhất từ các nút con của nó:

- Nút có con $(-1, 3)$ sẽ chọn: $\max(-1, 3) = 3$
- Nút có con $(5, 1)$ sẽ chọn: $\max(5, 1) = 5$
- Nút có con $(-6, -4)$ sẽ chọn: $\max(-6, -4) = -4$
- Nút có con $(0, 9)$ sẽ chọn: $\max(0, 9) = 9$

Vậy, các giá trị tại tầng này sau khi MAX lựa chọn là: $(3, 5, -4, 9)$.

Tầng tiếp theo (Nút đen - tầng MIN) Tại tầng này, mỗi nút sẽ chọn giá trị nhỏ nhất từ các nút con của nó (là các giá trị đã được chọn bởi tầng MAX ở trên):

- Nút đen bên trái có con $(3, 5)$ sẽ chọn: $\min(3, 5) = 3$
- Nút đen bên phải có con $(-4, 9)$ sẽ chọn: $\min(-4, 9) = -4$

Vậy, các giá trị tại tầng này sau khi MIN lựa chọn là: $(3, -4)$.

Nút gốc (Nút đỏ - tầng MAX) Nút gốc sẽ chọn giá trị lớn nhất từ các nút con của nó (là các giá trị đã được chọn bởi tầng MIN ở trên):

- Nút gốc có con $(3, -4)$ sẽ chọn: $\max(3, -4) = 3$

Áp dụng trong dự án

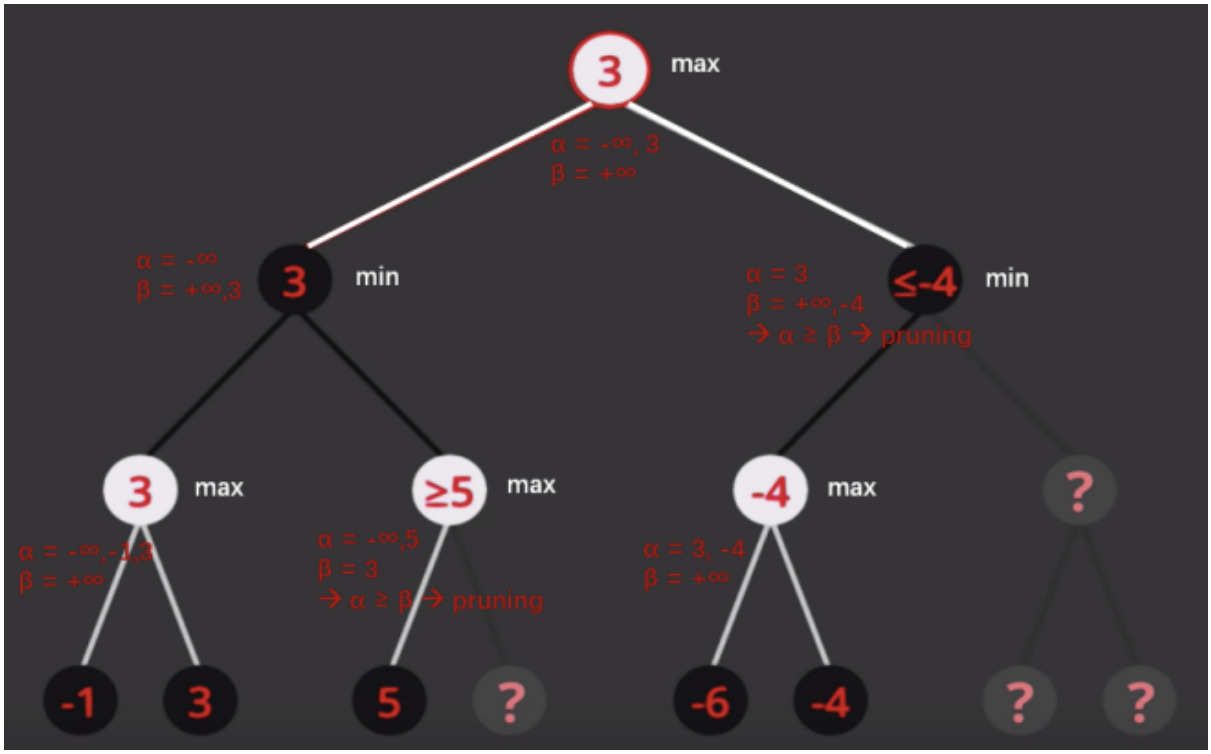
```python
def __MiniMax(self, gs: GameState, depth, maximizingPlayer):

    """Return a move """
    if depth == 0:
        self.total_nodes_leaf += 1
        return self.evaluation(gs)

    moves = gs.getValidMoves()
    if maximizingPlayer:
        maxEval = - math.inf
        for move in moves:
            gs.makeMove(move)
            self.total_node += 1
            eval_score = self.__MiniMax(gs, depth - 1, False)
            gs.undoMove()
            if eval_score > maxEval:
                maxEval = eval_score
                if depth == DEPTH:
                    self.bestMove = move
        return maxEval
    else:
        minEval = math.inf
        for move in moves:
            gs.makeMove(move)
            self.total_node += 1
            eval_score = self.__MiniMax(gs, depth - 1, True)
            gs.undoMove()
            if eval_score < minEval:
                minEval = eval_score
                if depth == DEPTH:
                    self.bestMove = move
        return minEval
```

Hình 2

4.1.2 Alpha-Beta Pruning

Để cải thiện hiệu quả, động cơ chủ yếu sử dụng Alpha-Beta Pruning, giúp giảm đáng kể số lượng nút được đánh giá so với Minimax tiêu chuẩn:

- Alpha đại diện cho điểm số tối thiểu mà người chơi tối đa hóa được đảm bảo
- Beta đại diện cho điểm số tối đa mà người chơi tối thiểu hóa được đảm bảo
- Khi một nước đi được phát hiện là tệ hơn nước đi đã kiểm tra trước đó, cây con sẽ được "cắt tỉa" (không khám phá thêm)
- Thuật toán theo dõi các thống kê như tổng số nút được khám phá, số lần cắt tỉa nhánh và số nút lá



Hình 3

1. **Tại Nút Gốc (MAX):** Khởi tạo $\alpha = -\infty$, $\beta = +\infty$.
   - **Nhánh Con Trái (MIN):**
     - Nút MIN này nhận $\alpha = -\infty, \beta = +\infty$.
     - Nút Cháu MAX (bên trái của MIN) duyệt các con $(-1, 3)$, kết quả là 3. Nút MIN cập nhật giá trị tạm thời là 3 và $\beta = 3$.
     - Nút Cháu MAX (bên phải của MIN) bắt đầu duyệt. Nó nhận $\alpha = -\infty$ (từ gốc) và $\beta = 3$ (từ cha MIN của nó).
       - Khi duyệt thấy con đầu tiên là 5, giá trị α cục bộ của nút MAX này trở thành 5.
       - **CẮT TỈA LẦN 1:** Tại nút MAX này, điều kiện $\alpha(5) \geq \beta(3)$ được thỏa mãn. Nhánh con '?' còn lại của nút MAX này bị cắt tỉa.
     - Nút MIN (nhánh con trái của gốc) có giá trị là 3. (Vì $\min(3, \text{kết quả từ nhánh bị cắt tỉa}) = 3$).
     - Nút MIN này trả về 3 cho Nút Gốc.
   - **Nút Gốc (MAX) cập nhật:** Giá trị tạm thời là 3. α của Nút Gốc trở thành 3.
2. **Nút Gốc (MAX) tiếp tục:**
   - **Nhánh Con Phải (MIN):**
     - Nút MIN này nhận $\alpha = 3$ (từ Nút Gốc) và $\beta = +\infty$.
     - Nút Cháu MAX (bên trái của MIN này) duyệt các con $(-6, -4)$, kết quả là -4.

- Nút MIN (nhánh con phải của gốc) cập nhật giá trị tạm thời là -4 và $\beta = -4$.
- **CẮT TỈA LẦN 2:** Tại nút MIN này, điều kiện $\beta(-4) \leq \alpha(3)$ được thỏa mãn. Toàn bộ nhánh con còn lại của nút MIN này (nút MAX mờ với các dấu '?') bị cắt tỉa.
- Nút MIN này trả về -4 cho Nút Gốc.

3. **Nút Gốc (MAX) quyết định:**
   - Nó đã có giá trị 3 từ nhánh trái và nhận -4 từ nhánh phải.
   - Giá trị cuối cùng là $\max(3, -4) = 3$.

4.2 Đánh Giá Vị Trí

4.2.1 Đánh Giá Truyền Thống

Đánh giá truyền thống bao gồm:

1. **Điểm Quân Cờ**: Mỗi loại quân có giá trị được gán (ví dụ: Hậu=900, Xe=500)
2. **Điểm Vị Trí Quân Cờ**: Bảng vị trí cụ thể cho mỗi loại quân phản ánh lợi thế vị trí
   - Bảng khác nhau cho vị trí vua trong giữa ván và cuối ván
   - Bảng đối xứng cho quân trắng và quân đen

Đánh giá kết hợp cân bằng quân cờ và cân nhắc vị trí:

```
evaluation = materialScore + positionScore
```

Động cơ có thể phát hiện kịch bản cuối ván và điều chỉnh đánh giá phù hợp, đặc biệt là đối với vị trí vua.

4.3 Mạng Thần Kinh để Cập Nhật Đánh Giá (NNUE)

4.3.1 Kiến Trúc NNUE

Động cơ triển khai kiến trúc NNUE (Neural Network for Updating Evaluations) với:

- Lớp đầu vào: 768 neuron (12 loại quân $\times$ 64 ô)
- Lớp ẩn: 256 neuron với hàm kích hoạt ReLU
- Lớp đầu ra: 1 neuron (điểm đánh giá)

Các đặc trưng đầu vào sử dụng sơ đồ mã hóa one-hot, trong đó mỗi loại quân trên mỗi ô kích hoạt một neuron cụ thể.

```
def forward(features):
    hidden = features × fc1_weights + fc1_bias
    hidden_output = ReLU(hidden)
    output = hidden_output × fc2_weights + fc2_bias
    return output
```

4.3.2 Huấn Luyện NNUE

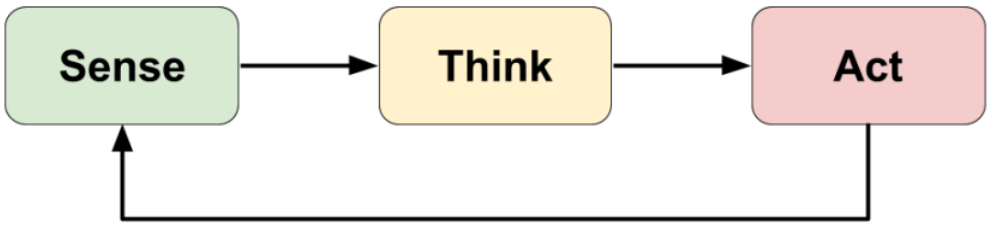
Mô hình NNUE được huấn luyện bằng học có giám sát từ các bản ghi ván cờ PGN:

1. **Chuẩn bị dữ liệu**:
    - Các ván cờ được phân tích từ file PGN với hơn 20000 ván đấu có elo cao (> 2200)
    - Vị trí bàn cờ được chuyển đổi thành vector đặc trưng
    - Nhãn được lấy từ kết quả ván đấu (1.0 cho trắng thắng, 0.0 cho hòa, -1.0 cho đen thắng)
    - Chiến lược lấy mẫu chọn vị trí từ các giai đoạn khác nhau của ván đấu
2. **Quá trình huấn luyện**:
    - Lan truyền ngược với tối ưu hóa gradient descent
    - Hàm mất mát Mean Squared Error (MSE)
    - Xử lý theo batch để cải thiện độ ổn định khi huấn luyện
    - Giảm tốc độ học tập (hệ số nhân 0.9 mỗi epoch)
    - Trọng số mô hình được lưu để sử dụng trong tương lai
3. **Chi tiết triển khai**:
    - Giao diện dòng lệnh để cấu hình huấn luyện
    - Hỗ trợ huấn luyện tăng dần trên các mô hình hiện có
    - Tham số có thể cấu hình (epochs, batch size, learning rate)

Đánh giá NNUE cung cấp phương pháp đánh giá vị trí tinh tế hơn so với phương pháp truyền thống, nắm bắt các mẫu vị trí tinh tế được học từ các ván đấu của bậc thầy thay vì chỉ dựa vào các phương pháp đánh giá được thiết kế thủ công.

5 Kiến Trúc AI

AI chạy theo Chu trình sau



Hình 4

5.1 SENSE (Cảm Nhận)

5.1.1 Biểu diễn trạng thái bàn cờ

- Bàn cờ được biểu diễn bằng mảng 2D (8×8) trong lớp `GameState`
- Mỗi ô chứa chuỗi 2 ký tự: màu quân (`'w'`, `'b'`) và loại quân (`'p'`, `'R'`, `'N'`, `'B'`, `'Q'`, `'K'`)
- Theo dõi vị trí vua qua `kingLocation` để tối ưu việc kiểm tra chiếu
- Lưu trữ lịch sử thế cờ trong `position` để phát hiện lặp ba

5.1.2 Nhận biết quy tắc đặc biệt

- Phát hiện tình huống chiếu thông qua phương thức `()`
- Theo dõi quyền nhập thành bằng lớp `CastleRights`
- Xử lý các nước đi đặc biệt: nhập thành, bắt tốt qua đường (en passant), phong tốt

5.1.3 Biểu diễn đặc trưng cho NNUE

- Chuyển đổi bàn cờ thành vector đặc trưng 768 chiều (12 loại quân × 64 ô)
- Vector one-hot cho mỗi loại quân và vị trí của nó trên bàn cờ
- Cung cấp đầu vào hiệu quả cho mạng neural đánh giá vị trí

5.2 THINK (Suy Nghĩ)

5.2.1 Thuật toán tìm kiếm

- **Minimax**: Thuật toán cơ bản xem xét tất cả các nước đi có thể
  Tìm nước đi tốt nhất cho người chơi hiện tại hoặc đối thủ
- **Alpha-Beta Pruning**: Cải tiến Minimax, loại bỏ các nhánh không cần thiết
  Cắt tỉa các nhánh không có khả năng tốt hơn kết quả hiện tại
- Theo dõi hiệu suất: Đếm tổng số node duyệt, node lá, số nhánh bị cắt tỉa

5.2.2 Phương pháp đánh giá vị trí

Đánh giá bằng NNUE (Neural Network Update Efficiently)

- Mạng neural hai lớp với ReLU:

```
def forward(self, features):
# Layer 1  v i  ReLU
hidden = np.dot(features, self.fc1_weights) + self.fc1_bias
hidden_output = self.relu(hidden)
# Layer 2 (output)
output = np.dot(hidden_output, self.fc2_weights) + self.fc2_bias
return output[0]
```

5.2.3 Huấn luyện mô hình NNUE

- Xử lý dữ liệu từ file PGN qua lớp `PGNProcessor`
- Trích xuất các vị trí từ ván cờ thật làm dữ liệu huấn luyện
- Thực hiện huấn luyện với backpropagation
- Lưu và tải mô hình đã huấn luyện từ file

5.3 ACT (Hành Động)

5.3.1 Sinh nước đi hợp lệ

- `getValidMoves()` sinh tất cả các nước đi hợp lệ
- Xử lý riêng các nước đi khi bị chiếu
- Các hàm chuyên biệt cho từng loại quân:

```
self.getFunctionMove = {'p': self.getPawnMoves, 'R': self._getRookMoves,
'N': self.getKnightMoves, 'B': self._getBishopMoves,
'Q': self.getQueenMoves, 'K': self._getKingMoves}
```

5.3.2 Lựa chọn nước đi tối ưu

- AI cập nhật `self.bestMove` trong quá trình tìm kiếm
- Nước đi tối ưu được chọn dựa trên kết quả đánh giá sâu nhất (DEPTH)
- Alpha-Beta đảm bảo cắt tỉa các nhánh tìm kiếm kém hiệu quả

5.3.3 Thực thi nước đi

- Phương thức `makeMove()` thực hiện nước đi đã chọn
- Xử lý các tình huống đặc biệt (nhập thành, en passant, phong tốt)
- Cập nhật thế cờ sau mỗi nước đi
- Lưu trữ lịch sử nước đi để hỗ trợ việc hoàn tác trong quá trình tìm kiếm

6 Xây Dựng Sản Phẩm

6.1 Các Công Nghệ Sử Dụng

Để xây dựng hệ thống trò chơi cờ vua có tích hợp AI, dự án đã sử dụng các công nghệ sau:

- **Python**: Ngôn ngữ lập trình chính cho toàn bộ dự án, dễ dàng tích hợp các thư viện và mô-đun mở rộng
- **Pygame**: Framework game 2D cho việc hiển thị giao diện, xử lý sự kiện và âm thanh
- **Pygame_gui**: Thư viện xây dựng giao diện người dùng, hỗ trợ tạo các panel, cửa sổ, nút bấm và text box
- **Pygame_menu**: Thư viện tạo menu game với các theme và widget có thể tùy chỉnh
- **NumPy**: Thư viện xử lý số học, được sử dụng cho các tính toán trong mô hình NNUE và thuật toán AI
- **Multiprocessing**: Module xử lý đa tiến trình, cho phép thuật toán tìm kiếm AI chạy trong tiến trình riêng, tránh làm đơ giao diện người dùng
- **Pickle**: Module lưu trữ và tải dữ liệu cấu trúc Python, được sử dụng để lưu và tải mô hình NNUE đã huấn luyện

6.2 Thiết Kế Giao Diện

Giao diện của trò chơi được thiết kế theo hướng đơn giản, dễ sử dụng và thẩm mỹ:

1. **Màn hình Menu**:
    - Hiển thị tiêu đề game và nền tùy chỉnh
    - Các nút điều hướng: Play (PvP), Play with AI (PvC), Comp vs Comp (CvC) và Quit
    - Theme sử dụng màu sắc hài hòa với độ trong suốt phù hợp
2. **Bàn Cờ**:
    - Kích thước chuẩn 8x8 với màu ô xen kẽ
    - Quân cờ được hiển thị bằng hình ảnh với kích thước tỷ lệ phù hợp
    - Đánh dấu các ô được chọn và các nước đi hợp lệ với màu khác
    - Hiển thị nước đi mới nhất bằng màu đánh dấu
3. **Panel Thông Tin**:
    - Hiển thị lịch sử các nước đi bằng ký hiệu cờ vua chuẩn

Hình 5

- Hiển thị lượt đi hiện tại (Trắng/Đen)
- Hiển thị số lượng nước đi hợp lệ
- Hiển thị trạng thái chiếu (Check)

4. **Thông Báo Kết Thúc**:
   - Hiển thị hình ảnh thông báo bên thắng cuộc khi kết thúc ván đấu
   - Hiệu ứng trong suốt để không che khuất hoàn toàn bàn cờ

6.3 Các Chức Năng Trong Dự Án

6.3.1 Chế Độ Người vs Máy

Chế độ này cho phép người dùng thách đấu với AI:

- **Cấu hình**: Người chơi mặc định sử dụng quân Trắng (đi trước), AI sử dụng quân Đen
- **Cơ chế AI**:
  - AI sử dụng thuật toán Alpha-Beta Pruning với độ sâu tìm kiếm được cấu hình trong hằng số DEPTH
  - Cơ chế bảo vệ thời gian: AI có giới hạn thời gian suy nghĩ (mặc định 10 giây)
  - Cơ chế fallback: Nếu tìm kiếm chính vượt thời gian, hệ thống tự động chuyển sang tìm kiếm nông hơn (REDUCED_DEPTH)

  - Khi cả hai mức tìm kiếm đều thất bại, AI sẽ chọn nước đi ngẫu nhiên từ các nước hợp lệ

- **Tương tác**:
  - Người chơi chọn quân và di chuyển bằng chuột
  - Hệ thống đánh dấu các nước đi hợp lệ khi chọn quân
  - Hỗ trợ phím tắt: R để reset ván đấu, Z để hoàn tác nước đi

- **Trạng thái kết thúc ván đấu**:
  - Chiếu hết (Checkmate): Hiển thị bên thắng
  - Hòa cờ do bế tắc (Stalemate): Không bên nào có nước đi hợp lệ nhưng không bị chiếu
  - Hòa cờ do lặp lại (Threefold Repetition): Cùng một thế cờ xuất hiện 3 lần

6.3.2 Chế Độ Máy vs Máy

Chế độ này cho phép hai AI đối đầu với nhau:

- **Cấu hình**: Hai engine AI riêng biệt, một điều khiển quân Trắng và một điều khiển quân Đen

- **Tự động hóa**:
  - Trò chơi tự động tiến hành mà không cần sự can thiệp của người dùng
  - Có độ trễ giữa các nước đi (mặc định 0.5 giây) để người dùng có thể theo dõi

- **Cơ chế bảo vệ**:
  - Cả hai AI đều có cơ chế giới hạn thời gian tìm kiếm
  - Tương tự chế độ người vs máy, mỗi AI có cơ chế fallback khi vượt quá thời gian
  - Phát hiện và xử lý các trường hợp đặc biệt (hòa do lặp lại, hết nước đi)

- **Tương tác**:
  - Người dùng có thể dừng/reset trận đấu bằng phím R
  - Có thể hoàn tác nước đi gần nhất bằng phím Z để xem lại thế cờ trước đó
  - Panel thông tin hiển thị đầy đủ lịch sử nước đi và trạng thái

- **Phân tích**:
  - Hệ thống ghi nhận và hiển thị thời gian suy nghĩ của mỗi AI
  - Theo dõi các trạng thái đặc biệt và điều kiện kết thúc ván đấu

Các chế độ chơi này kết hợp với nhau tạo thành một hệ thống cờ vua hoàn chỉnh, cho phép người dùng thực hành, tìm hiểu và phân tích các ván cờ tự động với nhiều cấp độ tương tác khác nhau.

7 Kết luận

Qua quá trình nghiên cứu và phát triển, đề tài đã đạt được những kết quả đáng khích lệ. Hệ thống này đã được xây dựng với nhiều tính năng và công nghệ hiện đại, cụ thể:

7.1 Kết quả đạt được

- **Thuật toán tìm kiếm hiệu quả**: Đã triển khai thành công thuật toán Alpha Beta Pruning, nâng cao hiệu suất tìm kiếm nước đi so với Minimax thông thường, cho phép AI tìm kiếm sâu hơn trong thời gian giới hạn.
- **Mô hình đánh giá tiên tiến**: Hệ thống đã tích hợp mô hình NNUE (Efficiently Updatable Neural Network) để đánh giá trạng thái bàn cờ, mang lại khả năng đánh giá chính xác hơn so với các phương pháp đánh giá tĩnh truyền thống.
- **Giao diện thân thiện với người dùng**: Giao diện đồ họa được phát triển bằng Pygame, trực quan và dễ sử dụng với nhiều chế độ chơi khác nhau: Người với Người, Người với Máy, và Máy với Máy.
- **Tính năng phong phú**: Hệ thống hỗ trợ đầy đủ các quy tắc cờ vua, bao gồm các nước đi đặc biệt như nhập thành, bắt tốt qua đường, phong cấp, và nhận diện các trường hợp hòa do lặp ba lần.
- **Khả năng mở rộng**: Kiến trúc mô-đun cho phép dễ dàng tích hợp các thuật toán mới hoặc cải tiến các thành phần hiện có.

7.2 Hạn chế và hướng phát triển

- **Cải thiện mô hình NNUE**: Cần thu thập thêm dữ liệu và huấn luyện mô hình NNUE trên tập dữ liệu lớn hơn để nâng cao hiệu suất đánh giá vị trí.
- **Tối ưu hóa tìm kiếm**: Có thể áp dụng thêm các kỹ thuật như Iterative Deepening, Transposition Table, Quiescence Search để cải thiện thêm hiệu suất tìm kiếm.

Tóm lại, hệ thống AI chơi cờ vua đã được phát triển thành công với khả năng chơi ở mức độ tốt, kết hợp các kỹ thuật tìm kiếm truyền thống và phương pháp học máy hiện đại. Dự án này không chỉ có giá trị học thuật mà còn là nền tảng tốt để tiếp tục phát triển các hệ thống AI game phức tạp khác trong tương lai.

8 REFERENCES

1. https://www.aihorizon.com/essays/chessai/intro.htm
2. https://www.researchgate.net/publication/220814778_The_Role_of_Chess_in_Artificial_Intelligence_Research

3. https://www.semanticscholar.org/paper/The-Role-of-Chess-in-Artificial-Intelligence-Lew
fac0dd639b47514c89efa0393c3b6c29a1d408e6

4. https://www.researchgate.net/publication/319390201_APPLYING_ALPHA-BETA_ALGORITHM_
IN_A_CHESS_ENGINE

5. https://urmm-csci.github.io/senior-seminar/seminars/spring2017/marckel.pdf