**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**BÁO CÁO PROJECT I**

**XÂY DỰNG CHESS BOT ĐỂ ĐẤU VỚI NGƯỜI CHƠI BẰNG AI**

**TRẦN HỮU HUY – 20183557 – IT1-02-K63**

huy.th183557@sis.hust.edu.vn

**Ngành Khoa học Máy tính**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | TS. Nguyễn Tuấn Dũng |
| **Bộ môn:** | Project 1 |
| **Viện:** | Công nghệ thông tin và Truyền thông |

**HÀ NỘI, 12/2020**

Giáo viên hướng dẫn

Ký và ghi rõ họ tên

**Lời cảm ơn**

Trong thời gian làm bài tập lớn môn Project I, em đã nhận được rất nhiều sự giúp đỡ, đóng góp ý kiến của thầy và các bạn.

Em xin gửi lời cảm ơn chân thành đến TS. Nguyễn Tuấn Dũng, giảng viên Viện Công nghệ thông tin và truyền thông, trường Đại học Bách khoa Hà Nội, người trực tiếp hướng dẫn em hoàn thành bài tập lớn môn Project I.

Em cũng xin cảm ơn các thầy cô, bạn bè đã luôn tạo điều kiện quan tâm, giúp đỡ để em hoàn thành học phần này.

Em rất vinh dự nếu nhận được sự góp ý của thầy cô và các bạn để hoàn thiện đề tài này hơn.

Em xin chân thành cảm ơn!

**Tóm tắt nội dung**

* Vấn đề cần thực hiện: xây dựng Chess Bot để đấu với người chơi bằng AI
* Phương pháp thực hiện: Deep learning và phương pháp Minmax+Cắt tỉa Alpha-Beta
* Ngôn ngữ lập trình:
* Python (sử dụng PyTorch để build model Deep Learning, thư viện PyGame để tạo UI chơi game, Flask để deploy demo lên web)
* HTML, CSS, JavaScript để deploy lên web kết hợp với Flask
* Kết quả: Chess Bot có thể dễ dàng đánh bại những người chơi ở mức độ rất kinh nghiệm đổ lại
* Tính thực tế: Chương trình đơn giản có thể giúp các bạn học sinh, sinh viên rèn luyện tư duy chơi cờ
* Định hướng phát triển: Tìm cách cải tiến mới và tạo thêm Bot chơi các bộ môn khác như cờ tướng, cờ vây, v.v…

Sinh viên thực hiện

Ký và ghi rõ họ tên

Trần Hữu Huy

**MỤC LỤC**

[CHƯƠNG 1. PHÁT BIỂU VÀ GIẢI QUYẾT BÀI TOÁN 1](#_Toc60267668)

[1.1 Phát biểu bài toán 1](#_Toc60267669)

[1.2 Phương pháp thực hiện 1](#_Toc60267670)

[1.2.1 Deep Learning 1](#_Toc60267671)

[1.2.2 MinMax kết hợp với cắt tỉa Alpha-Beta 4](#_Toc60267672)

[1.2.3 Giao diện cho người chơi 6](#_Toc60267673)

[CHƯƠNG 2. KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ 7](#_Toc60267674)

[2.1 Deep Learning 7](#_Toc60267675)

[2.2 Minmax kết hợp cắt tỉa Alpha-Beta 8](#_Toc60267676)

[TÀI LIỆU THAM KHẢO 8](#_Toc60267677)

**DANH MỤC HÌNH ẢNH**

[Hình 1. Mã hóa bàn cờ về vecto 1](#_Toc60267678)

[Hình 2. Model Siamese 2](#_Toc60267679)

[Hình 3. Đầu vào model để quyết định nước cờ tối ưu cho máy 2](#_Toc60267680)

[Hình 4. Model Auto-Encoder 3](#_Toc60267681)

[Hình 5. Data gốc để train 3](#_Toc60267682)

[Hình 6. Parse data gốc về vecto, lưu dạng file .npy 4](#_Toc60267683)

[Hinh 7. Trọng số quân Tốt 4](#_Toc60267684)

[Hình 8. Trọng số quân Mã 5](#_Toc60267685)

[Hình 9. Trọng số quân Tượng 5](#_Toc60267686)

[Hình 10. Trọng số quân Xe 5](#_Toc60267687)

[Hình 11. Trọng số quân Hậu 6](#_Toc60267688)

[Hình 12. Trọng số quân Vua 6](#_Toc60267689)

[Hình 13. Giao diện PyGame 6](#_Toc60267690)

[Hình 14. Giao diện deploy, có thêm chức năng máy đấu máy 7](#_Toc60267691)

[Hình 15. Test loss model Auto-Encoder (MSE Loss) sau 200 epochs train 7](#_Toc60267692)

[Hình 16. Test loss model Siamese (BCE Loss) sau 200 epochs train 8](#_Toc60267693)

# PHÁT BIỂU VÀ GIẢI QUYẾT BÀI TOÁN

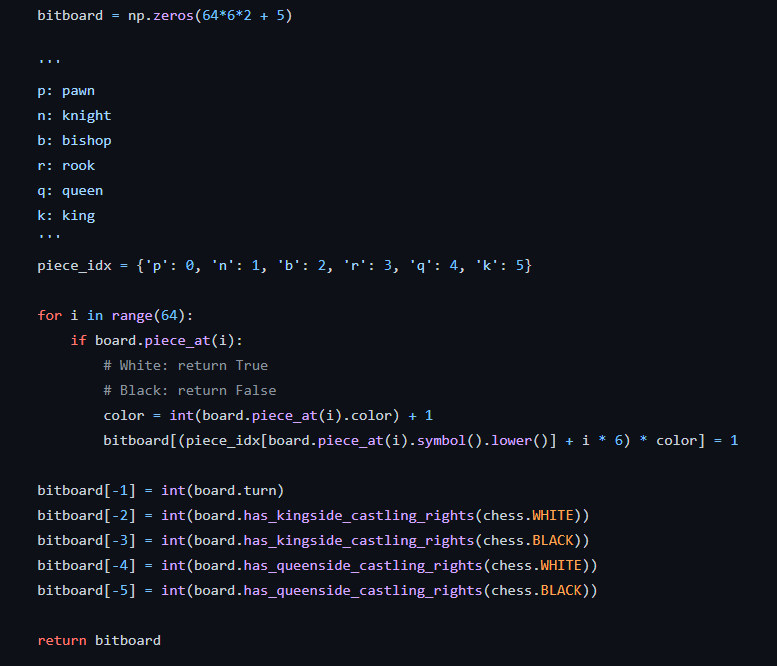
## Phát biểu bài toán

* Tạo Bot đơn giản đấu cờ vua với người chơi, có nhiều cấp độ dễ, trung bình, khó
* Tạo Bot với 2 hướng tiếp cận: Deep Learning và phương pháp MinMax kết hợp với cắt tỉa Alpha-Beta
* Với hướng tiếp cận Deep Learning: build model theo 1 bài báo đã public, bộ data để train model là bộ data đã train trong bài báo
* Với hướng tiếp cận sử dụng phương pháp MinMax kết hợp với cắt tỉa Alpha-Beta: đã được sử dụng nhiều, build them để đấu với Deep Learning phục vụ cho quá trình đánh giá

## Phương pháp thực hiện

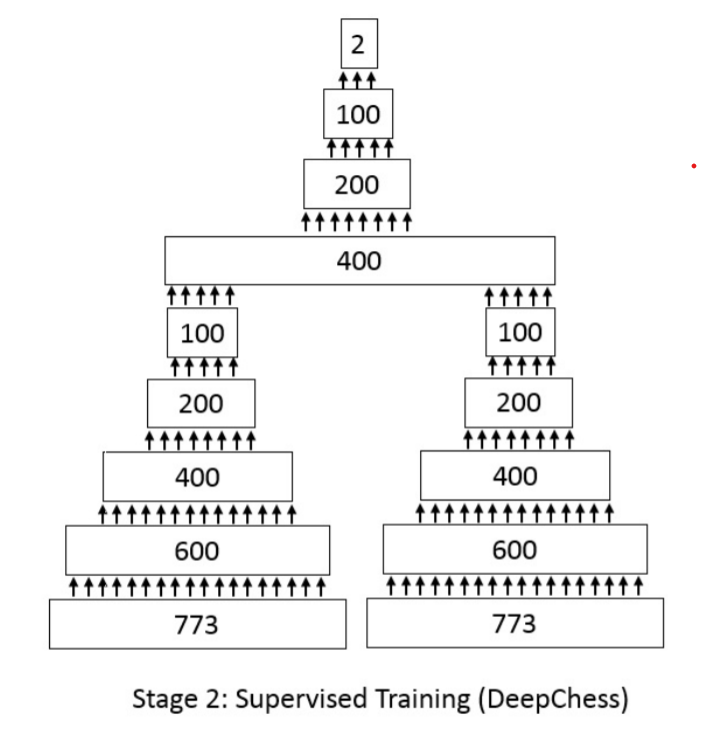
### Deep Learning

* Đầu vào: trạng thái của bàn cờ được mã hóa thành 1 vecto **773** chiều
* Giải thích: Bàn cờ kích thước 8\*8 = **64**, trên bàn cờ có **6** loại quân cờ (Tốt, Mã, Tượng, Xe, Hậu, Vua), **2** bên trắng đen đấu với nhau, **4** trạng thái nhập thành, **1** giá trị thể hiện lượt đi (0 là trắng đi, 1 là đen đi)
* 64 \* 6 \* 2 + 4 + 1 = 773



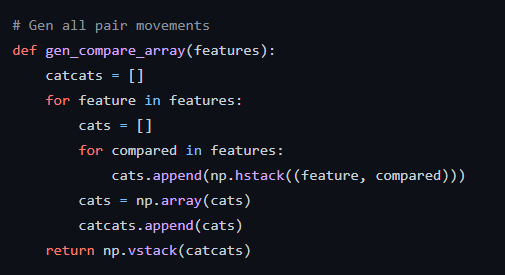
Hình 1. Mã hóa bàn cờ về vecto

* Đầu ra: vecto 2 chiều có 2 giá trị lần lượt là tỉ lệ thắng với thế cờ đầu vào của quân trắng và quân đen
* Kiến trúc chính của model:



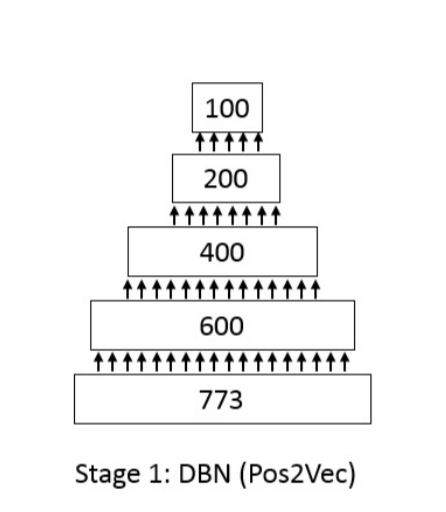
Hình 2. Model Siamese

* Model Neural Network cơ bản, kết hợp 2 nhánh Encoder chập lại vào thành 1 nhánh để train model Siamese (1 nhánh khi train chỉ đưa dữ liệu là trạng thái ván cờ của những ván quân trắng thắng, nhánh còn lại khi train chỉ đưa dữ liệu là trạng thái ván cờ của những ván quân đen thắng). Khi chạy thì đưa vecto trạng thái hiện tại của ván cờ (sau khi người chơi đi) stack với vecto trạng thái có thể có của ván cờ (những nước có thể đi của quân đen) và chọn thế cờ nào có tỉ lệ thắng cho quân đen nhất để máy đi.



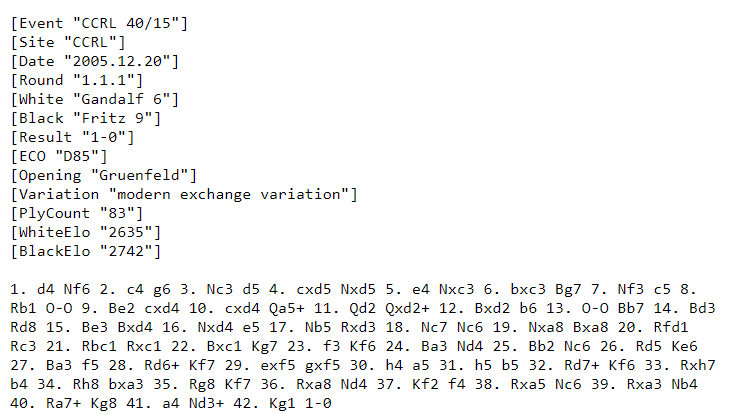
Hình 3. Đầu vào model để quyết định nước cờ tối ưu cho máy

* Model Auto-Encoder: Encoder vector 773 chiều về 100 chiều (quá trình train giống AE bình thường) mục đích giảm chiều vecto và giữ lại những thuộc tính quan trọng nhất

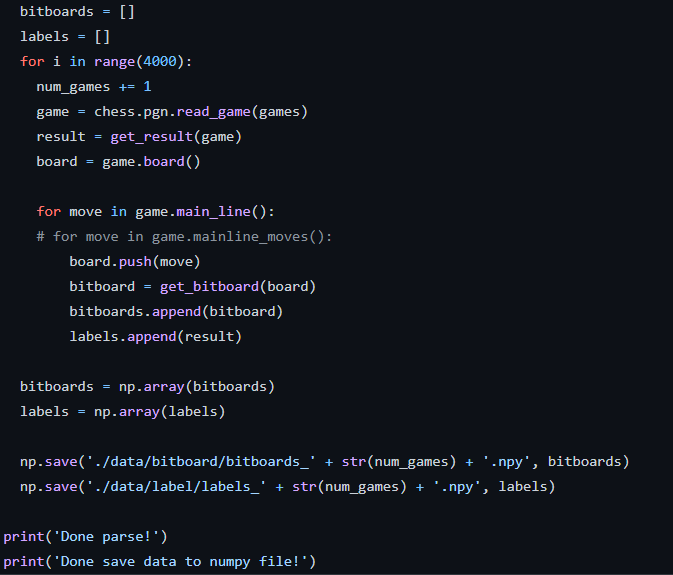


Hình 4. Model Auto-Encoder

* Data processing: data là tập các bản ghi các trận cờ đã đấu bởi các tuyển thủ, sử dụng thư viện python-chess để đọc bản ghi cờ và đưa về dạng vecto qua 1 vài thao tác



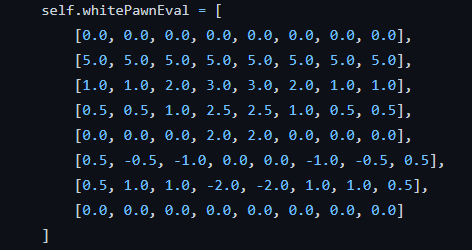
Hình 5. Data gốc để train



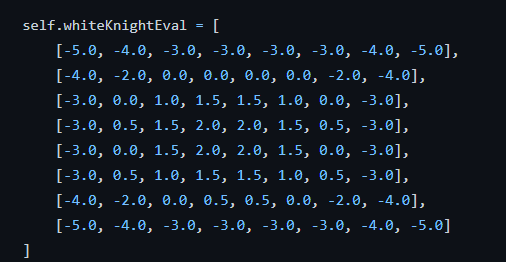
Hình 6. Parse data gốc về vecto, lưu dạng file .npy

### MinMax kết hợp với cắt tỉa Alpha-Beta

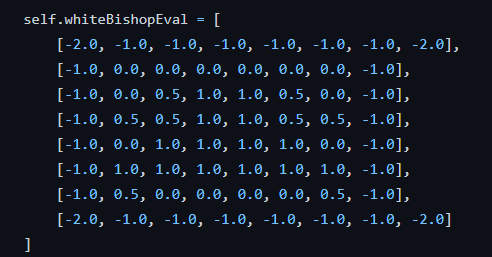
* Độ sâu càng cao khả năng thi đấu càng tốt nhưng đưa ra quyết định chậm
* Đánh giá các nước đi dựa trên 2 yếu tố chính:
* Trọng số quân cờ: Tốt 10 điểm, Mã 30 điểm, Tượng 30 điểm, Xe 50 điểm, Hậu 90 điểm, Vua 900 điểm
* Giá trị các vị trí trên bàn cờ: ví dụ tượng và mã mạnh nhất khi ở giữa bàn cờ nên có trọng số dương ở vị trí đó



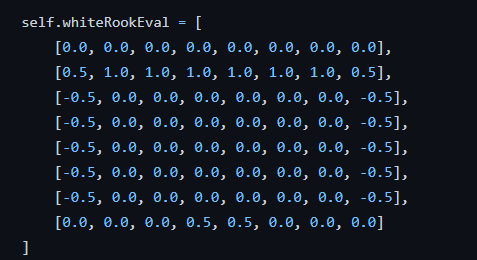
Hinh 7. Trọng số quân Tốt



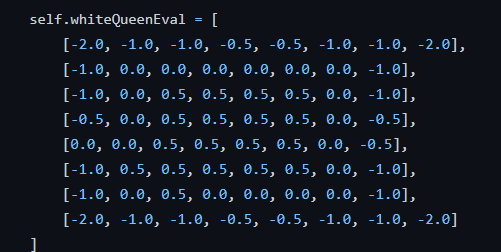
Hình 8. Trọng số quân Mã



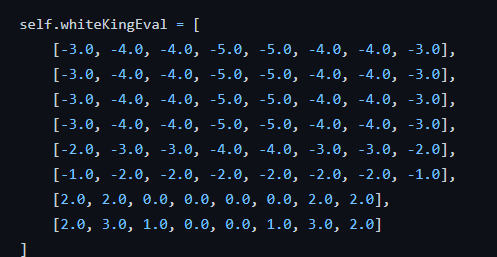
Hình 9. Trọng số quân Tượng



Hình 10. Trọng số quân Xe



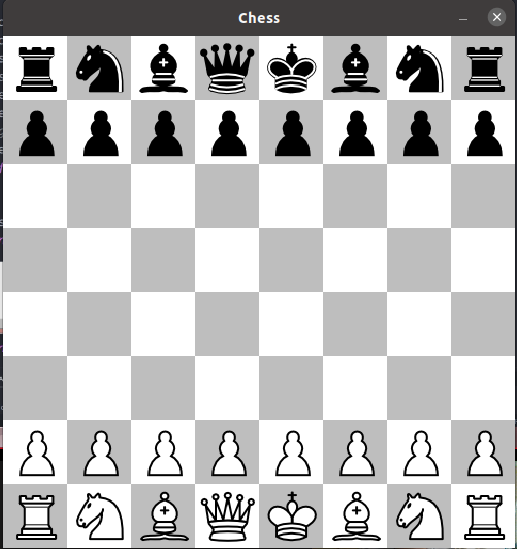
Hình 11. Trọng số quân Hậu



Hình 12. Trọng số quân Vua

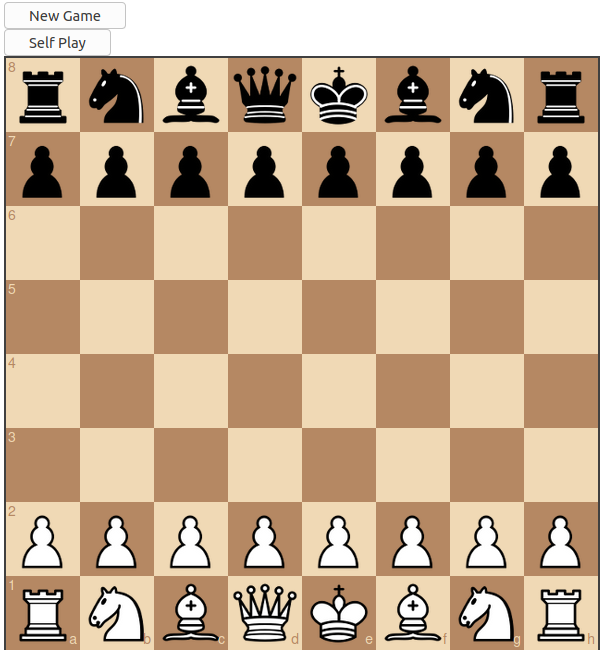
### Giao diện cho người chơi

#### Dùng thư viện PyGame



Hình 13. Giao diện PyGame

#### Giao diện deploy

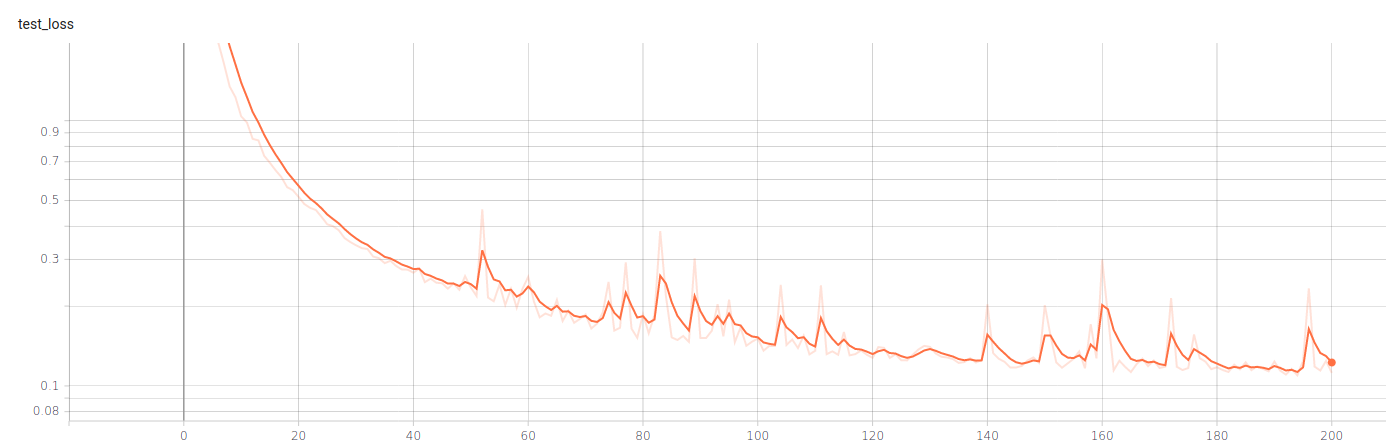


Hình 14. Giao diện deploy, có thêm chức năng máy đấu máy

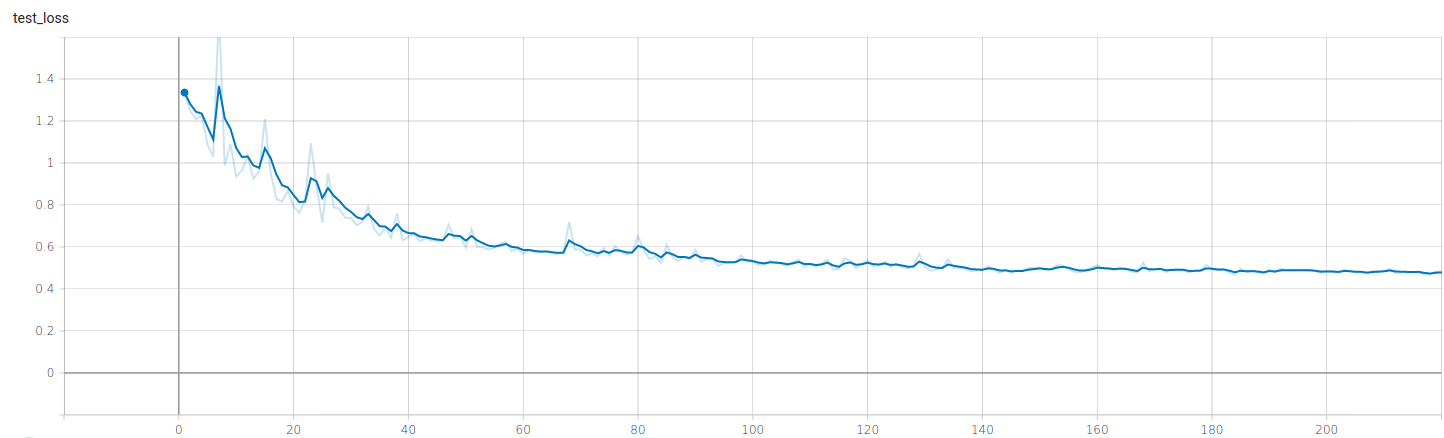
# KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ

## Deep Learning

* Đưa ra kết quả nhanh (dưới 1s) có thể thắng những người chơi non kinh nghiệm, nhưng do tài nguyên hạn chế (không train được hết bộ data) nên khả năng thi đấu vẫn còn kém so với phương pháp MinMax cắt tỉa (thua từ độ sâu 4 trở đi)
* Loss:



Hình . Test loss model Auto-Encoder (MSE Loss) sau 200 epochs train



Hình . Test loss model Siamese (BCE Loss) sau 200 epochs train

## Minmax kết hợp cắt tỉa Alpha-Beta

* Từ độ sâu 3 trở xuống đưa ra quyết định nhanh (dưới 6 giây) có thể thắng người mới chơi nhưng thua deep learning
* Từ độ sâu 4 trở đi thắng áp đảo deep learning và có thể thắng người chơi giỏi cờ vua (thử nghiệm với bạn bè và với em)

Source Code: <https://github.com/tranhuuhuy297/deep_chess>

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | David O.E., Metanyahu N.S., Wolf L. (2016) DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess. In: Villa A., Masulli P., Pons Rivero A. (eds) Artificial Neural Networks and Machine Leaning - ICANN 2016. ICANN 2016. Lecture Notes in Computer Science, vol 9887. Springer, Cham. <https://doi.org/10.1007/978-3-319-44781-0_11> |

[2] <https://github.com/dangeng/DeepChess>

[3] <https://en.wikipedia.org/wiki/Minimax>