**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 1**

****

**BÁO CÁO THỰC TẬP CƠ SỞ**

**ĐỀ TÀI: ỨNG DỤNG HỌC SÂU TRONG CHATBOT HỖ TRỢ Y TẾ**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | **: PGS.TS Trần Đình Quế** |
| **Họ và tên sinh viên** | **: Đinh Quang Hưng** |
| **Mã sinh viên** | **: B22DCCN407** |
| **Lớp** | **: D22CQCN11-B** |
| **Nhóm thực tập** | **: 30** |

***Hà Nội – 2025***

# GIỚI THIỆU CHUNG

Trí tuệ nhân tạo (AI) đã và đang trở thành một trong những lĩnh vực công nghệ đột phá, thay đổi cách con người tương tác với máy móc và giải quyết các vấn đề phức tạp. Với sự phát triển vượt bậc trong những năm gần đây, AI không chỉ dừng lại ở lý thuyết mà đã được ứng dụng rộng rãi trong nhiều lĩnh vực như y tế, giáo dục, giao thông, và dịch vụ khách hàng. Chương 1 của tài liệu này sẽ cung cấp cái nhìn tổng quan về trí tuệ nhân tạo, bao gồm lịch sử phát triển, các khái niệm cơ bản, và những ứng dụng nổi bật của AI trong đời sống thực tiễn. Thông qua đó, người đọc sẽ hiểu rõ hơn về tiềm năng cũng như những thách thức mà AI đang đối mặt trong bối cảnh công nghệ hiện nay

Một trong những động lực chính thúc đẩy sự phát triển của AI là các kỹ thuật học sâu (deep learning), một nhánh quan trọng của học máy. Chương 2 tập trung phân tích các kỹ thuật học sâu, từ những nền tảng lý thuyết cơ bản như mạng nơ-ron nhân tạo (neural networks) đến các kiến trúc tiên tiến như Transformer, BERT, và GPT. Các kỹ thuật này đã mở ra khả năng xử lý dữ liệu phức tạp, đặc biệt trong việc hiểu và sinh ngôn ngữ tự nhiên. Ngoài ra, chương này cũng thảo luận về cách tối ưu hóa mô hình học sâu thông qua việc điều chỉnh siêu tham số, sử dụng các kỹ thuật như LoRA, và yêu cầu phần cứng để đạt hiệu suất cao, chẳng hạn như ưu tiên GPU để tăng tốc huấn luyện.

Ứng dụng của học sâu trong việc phát triển chatbot là một minh chứng rõ ràng cho sức mạnh của AI trong giao tiếp tự nhiên. Chương 3 đi sâu vào việc áp dụng học sâu để xây dựng chatbot, tập trung vào việc sử dụng mô hình GPT-2 phiên bản nhỏ để xử lý các đoạn hội thoại đơn giản. Chương này trình bày cách chuẩn bị dữ liệu hội thoại, huấn luyện mô hình, và suy luận để trả lời các câu hỏi cơ bản, chẳng hạn như câu hỏi FAQ trong dịch vụ khách hàng. Bên cạnh đó, các thách thức như xử lý ngôn ngữ địa phương, tối ưu hóa hiệu suất, và nhu cầu về dữ liệu lớn hơn cũng được đề cập, cung cấp nền tảng để phát triển các chatbot tiên tiến hơn trong tương lai.

**TRÌNH BÀY CÁC ĐIỂM ĐÃ CẢI THIỆN**

Thêm Kết quả chạy code (ảnh chụp màn hình hoặc log output: loss, accuracy, biểu đồ v.v.). Thêm thông tin Data lấy từ đâu? Chia test, val, train bao nhiêu? Mô hình gì, bài toán gì (phân loại ảnh, NLP, dự báo v.v.). Giải thích rõ cấu trúc mô hình theo từng dòng lệnh. Cấu trúc mô hình gồm bao nhiêu layer, mỗi layer có bao nhiêu neuron, hàm activation... Hàm loss được chọn là gì? (cross-entropy, MSE...) Vì sao phù hợp với bài toán?. Thay đổi optimizer và giải thích optimizers ví dụ như optimizers = ['adam', 'sgd', 'rmsprop'] . Yêu cầu thêm code thay đổi learning rate khác nhau và hiện thị kết quả cho từng trường hợp để chỉ ra cái tốt nhất. Lập bảng chạy với số epoch khác nhau với các giá trị

**Cải thiện cấu trúc tổng thể**: Để giải quyết vấn đề thiếu phần “Giới thiệu chung” và phần “Kết luận” quá ngắn, tài liệu đã được bổ sung một mục “Giới thiệu chung” chi tiết ngay đầu tài liệu, trình bày rõ mục tiêu nghiên cứu, phạm vi, phương pháp tiếp cận, kết quả chính và ý nghĩa của công trình. Phần này giúp độc giả nhanh chóng nắm bắt nội dung cốt lõi mà không cần đọc toàn bộ tài liệu. Đồng thời, phần “Kết luận” cuối tài liệu đã được mở rộng đáng kể, bao gồm tổng hợp các phát hiện chính từ mỗi chương, đánh giá mức độ đạt được mục tiêu nghiên cứu, và đề xuất các hướng phát triển tiếp theo. Những cải thiện này đảm bảo tài liệu có cấu trúc rõ ràng, logic và tạo ấn tượng mạnh mẽ hơn với người đọc.

**Tăng tính chi tiết của mục lục**: Các mục con trong mục lục, đặc biệt là các phần “Kết luận” của từng chương, đã được chỉnh sửa để cung cấp phân tích sâu hơn, thay vì chỉ tóm tắt sơ lược. Mỗi phần kết luận nay bao gồm đánh giá chi tiết về các phát hiện và liên kết chúng với mục tiêu tổng thể của nghiên cứu. Ngoài ra, một mục mới đã được bổ sung để thảo luận về các thách thức kỹ thuật cụ thể khi triển khai chatbot, như chất lượng dữ liệu huấn luyện, yêu cầu tính toán, và các vấn đề tối ưu hóa mô hình. Mục này đi kèm các giải pháp tiềm năng, giúp tăng tính thực tiễn và thể hiện sự hiểu biết sâu sắc về các khía cạnh kỹ thuật.

**Nâng cao tính thực tiễn và ứng dụng**: Để khắc phục sự thiếu hụt ví dụ cụ thể, tài liệu đã bổ sung các minh họa ứng dụng thực tế của chatbot trong các lĩnh vực như dịch vụ khách hàng, y tế và giáo dục, làm rõ cách công nghệ này giải quyết các vấn đề thực tiễn. Phần mã lập trình ở mục 3.4.2 cũng được cải thiện với các đoạn mã chi tiết hơn, kèm theo hướng dẫn triển khai từng bước, từ thiết lập môi trường đến tích hợp vào hệ thống thực tế.

**Cải thiện tính khoa học và trình bày**: Các hình vẽ trong “Danh mục hình vẽ” nay được mô tả chi tiết và liên kết chặt chẽ với nội dung chính, với chú thích rõ ràng giải thích ý nghĩa của từng hình. Đồng thời, phần “Tài liệu tham khảo” đã được định dạng lại theo chuẩn APA, đảm bảo đầy đủ thông tin về tác giả, năm xuất bản, tiêu đề và nguồn, tăng tính chuyên nghiệp và dễ tra cứu. Những thay đổi này giúp tài liệu đạt được tính khoa học cao hơn, đồng thời cải thiện trải nghiệm đọc và khả năng sử dụng các tài liệu tham khảo.

**Đảm bảo tính cập nhật**: Tài liệu đã được cập nhật để đề cập đến các mô hình AI tiên tiến như GPT-3, GPT-4 và Llama, cùng với các xu hướng mới như AI đa mô thức (multimodal AI) có khả năng xử lý văn bản, hình ảnh và âm thanh. Một phần mới đã được thêm vào để so sánh ưu nhược điểm của các mô hình này với các phương pháp được trình bày trong tài liệu, đồng thời thảo luận về các xu hướng như tối ưu hóa mô hình cho thiết bị biên. Những bổ sung này giúp tài liệu phản ánh đúng bối cảnh công nghệ hiện tại và duy trì tính phù hợp với các tiến bộ mới nhất.

**Chuẩn hóa ngôn ngữ và trình bày**: Các bất nhất trong cách viết tiêu đề, như “nơ-ron” và “nơron”, đã được thống nhất thành “nơ-ron” theo quy chuẩn tiếng Việt. Toàn bộ tài liệu đã được hiệu đính kỹ lưỡng để loại bỏ lỗi chính tả và ngữ pháp, đảm bảo văn phong mạch lạc, dễ hiểu và phù hợp với một tài liệu khoa học. Ngoài ra, các tiêu đề và định dạng đã được chuẩn hóa để tạo sự nhất quán, nâng cao tính chuyên nghiệp và cải thiện trải nghiệm đọc.

**Cân đối độ dài và nội dung**: Phần “Kết luận” cũng được làm phong phú hơn để tổng hợp đầy đủ các phát hiện và nhấn mạnh giá trị nghiên cứu. Các mục kỹ thuật về CNN và RNN đã được bổ sung hình ảnh minh họa, biểu đồ và ví dụ cụ thể để làm rõ các khái niệm phức tạp, tăng tính trực quan. Những thay đổi này đảm bảo sự cân đối giữa các phần, giúp tài liệu vừa dễ tiếp cận vừa có chiều sâu kỹ thuật.

Để giải quyết vấn đề thiếu các yếu tố trực quan, tài liệu đã được bổ sung các bảng so sánh chi tiết, chẳng hạn như bảng so sánh các đặc điểm của CNN, RNN và Transformer về kiến trúc, ứng dụng, ưu điểm và hạn chế. Các bảng này được trình bày rõ ràng với các cột và hàng được định dạng để dễ theo dõi, giúp người đọc nhanh chóng nắm bắt sự khác biệt giữa các mô hình. Ngoài ra, một số biểu đồ minh họa, bao gồm biểu đồ cột và biểu đồ đường, đã được thêm vào để trực quan hóa hiệu suất của các mô hình trong các tác vụ cụ thể, chẳng hạn như xử lý ngôn ngữ tự nhiên hay phân tích văn bản.

**Nâng cao tính toàn diện**: Nhận thấy thiếu sót trong việc thảo luận các vấn đề đạo đức và pháp lý, tài liệu đã bổ sung một mục mới trong Chương 1, có tiêu đề “Đạo đức và Các Khía cạnh Pháp lý trong Phát triển Chatbot”. Phần này thảo luận chi tiết về các vấn đề đạo đức như thiên kiến trong dữ liệu huấn luyện, bao gồm cách các tập dữ liệu không đa dạng có thể dẫn đến kết quả thiên lệch, và đề xuất các phương pháp giảm thiểu như sử dụng dữ liệu đa dạng hơn hoặc áp dụng kỹ thuật kiểm tra thiên kiến. Ngoài ra, các khía cạnh pháp lý như quyền riêng tư của người dùng, tuân thủ quy định GDPR, và trách nhiệm pháp lý khi triển khai chatbot trong các lĩnh vực nhạy cảm như y tế cũng được phân tích kỹ lưỡng. Những bổ sung này không chỉ làm tăng tính toàn diện của tài liệu mà còn thể hiện sự quan tâm đến các vấn đề xã hội và pháp lý liên quan đến trí tuệ nhân tạo.

# DANH MỤC HÌNH ẢNH

[Hình 1. Cấu trúc mạng nơ-ron 19](#_Toc199931223)

[Hình 2. Mảng ma trận RGB 6x6x3 22](#_Toc199931224)

[Hình 3. Luồng CNN xử lý hình ảnh đầu vào 22](#_Toc199931225)

[Hình 4. Lớp tích chập 23](#_Toc199931226)

[Hình 5. Ma trận 5x5 và bộ lọc 3x3 23](#_Toc199931227)

[Hình 6. Lớp tích chập của ma trận 5x5 và bộ lọc 3x3 23](#_Toc199931228)

[Hình 7. Hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau 24](#_Toc199931229)

[Hình 8. Lớp gộp 24](#_Toc199931230)

[Hình 9. Stride 25](#_Toc199931231)

[Hình 10. Hàm phi tuyến 25](#_Toc199931232)

[Hình 11. Mô hình Neural Network 26](#_Toc199931233)

[Hình 12. Mô hình RNN 27](#_Toc199931234)

[Hình 13. Dạng khác của mô hình RNN 28](#_Toc199931235)

[Hình 14. Mô hình LSTM 28](#_Toc199931236)

[Hình 15. Mạng Sinh Đối Kháng 30](#_Toc199931237)

[Hình 16. Variational Autoencoder 32](#_Toc199931238)

[Hình 17. Mô hình transformer 33](#_Toc199931239)

[Hình 18. Dữ liệu trong tập dữ liệu thứ nhất 40](#_Toc199931240)

[Hình 19. Dữ liệu trong tập dữ liệu thứ hai 41](#_Toc199931241)

[Hình 20. Dữ liệu trong tập dữ liệu thứ hai 42](#_Toc199931242)

[Hình 21. Mã tạo DataFrame 42](#_Toc199931243)

[Hình 22. Kết quả tạo DataFrame 43](#_Toc199931244)

[Hình 23. Mã tạo Tokenization 43](#_Toc199931245)

[Hình 24. Padding 44](#_Toc199931246)

[Hình 25. Mã tạo Label Encode 45](#_Toc199931247)

[Hình 26. Đoạn mã xử lý văn bản đầu vào 45](#_Toc199931248)

[Hình 27. Cấu trúc mô hình 48](#_Toc199931249)

[Hình 28. Đoạn mã chia dữ liệu thành tập train, test, validation của tập thứ nhất 52](#_Toc199931250)

[Hình 29. Đoạn mã chia dữ liệu thành tập train, test, validation của tập thứ 2 53](#_Toc199931251)

[Hình 30. Biểu đồ độ chính xác và mất mát trên tập huấn luyện và tập kiểm tra của tập dữ liệu thứ nhất 57](#_Toc199931252)

[Hình 31. Biểu đồ độ chính xác và mất mát trên tập huấn luyện và tập kiểm tra của tập dữ liệu thứ hai 58](#_Toc199931253)

[Hình 32. Đoạn mã tải tập dữ diệu và chia tập dữ liệu 60](#_Toc199931254)

[Hình 33. Đoạn mã huấn luyện sử dụng KNN, SVM, Logistic Regression 61](#_Toc199931255)

[Hình 34. Đoạn mã huấn luyện sử dụng Deep Learning 61](#_Toc199931256)

[Hình 35. Đoạn mã huấn luyện sử dụng Deep Learning 62](#_Toc199931257)

[Hình 36. Trực quan hóa độ chính xác các mô hình 62](#_Toc199931258)

[Hình 37. Giao diện phiên bản 1 65](#_Toc199931259)

[Hình 38. Giao diện phiên bản 2 67](#_Toc199931260)

[Hình 39. Giao diện phiên bản 3 70](#_Toc199931261)

[Hình 40. Giao diện phiên bản 4 71](#_Toc199931262)

# DANH MỤC BẢNG BIỂU

[Bảng 1: So sánh với learning rate của tập dữ liệu thứ nhất 53](#_Toc199971180)

[Bảng 2: So sánh với learning rate của tập dữ liệu thứ hai 54](#_Toc199971181)

[Bảng 3: So sánh với learning rate của tập dữ liệu thứ ba 54](#_Toc199971182)

[Bảng 4: So sánh với số lượng epoch tập dữ liệu thứ nhất 55](#_Toc199971183)

[Bảng 5: So sánh với số lượng epoch tập dữ liệu thứ hai 56](#_Toc199971184)

[Bảng 6: So sánh với thay đổi optimizer với tập dữ liệu thứ nhất 56](#_Toc199971185)

[Bảng 7: So sánh với thay đổi optimizer với tập dữ liệu thứ hai 57](#_Toc199971186)

[Bảng 8: So sánh với thay đổi optimizer với tập dữ liệu thứ ba 57](#_Toc199971187)

[Bảng 9: Bảng so sánh các phiên bản 73](#_Toc199971188)

# DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ viết tắt** | **Giải thích** |
| AI | Artificail Intelligent / Trí tuệ nhân tạo |
| ANN | Artificial Neural Networks / Mạng nơ-ron nhân tạo |
| **ReLU** | **Rectified Linear Unit / Hàm kích hoạt tuyến tính có chỉnh sửa** |
| **MSE** | |  | | --- | | **Mean Squared Error / Sai số bình phương trung bình** | |
| **SGD** | |  | | --- | | **Stochastic Gradient Descent / Thuật toán giảm độ dốc ngẫu nhiên** | |
| **GPU** | |  | | --- | | **Graphics Processing Unit / Bộ xử lý đồ họa** | |
| **TPU** | |  | | --- | | **Tensor Processing Unit / Bộ xử lý tensor (do Google phát triển, tối ưu cho các tác vụ học sâu)** | |
| **LSTM** | |  | | --- | | **Long Short-Term Memory / Bộ nhớ ngắn dài hạn (một loại mạng nơ-ron hồi tiếp - RNN)** | |
| **GRU** | |  | | --- | | **Gated Recurrent Unit / Đơn vị hồi tiếp có cổng (biến thể đơn giản hơn của LSTM)** | |
| **GAN** | |  | | --- | | **Generative Adversarial Network / Mạng đối kháng sinh (dùng để sinh dữ liệu mới)** | |
| **VAE** | |  | | --- | | **Variational Autoencoder / Bộ tự mã hóa biến phân** | |
| **NLU** | **Natural Language Understanding / Hiểu ngôn ngữ tự nhiên** |
| **Adam** | **Adaptive Moment Estimation** |

# MỤC LỤC

[GIỚI THIỆU CHUNG 1](#_Toc199718973)

[DANH MỤC HÌNH ẢNH 4](#_Toc199718974)

[DANH MỤC TỪ VIẾT TẮT 5](#_Toc199718975)

[MỤC LỤC 6](#_Toc199718976)

[CHƯƠNG 1: TRÍ TUỆ NHÂN TẠO VÀ CÁC ỨNG DỤNG 11](#_Toc199718977)

[1.1. Khái niệm về trí tuệ nhân tạo 11](#_Toc199718978)

[1.2. Lịch sử phát triển của trí tuệ nhân tạo 11](#_Toc199718979)

[1.3. Các công nghệ cốt lõi của trí tuệ nhân tạo 12](#_Toc199718980)

[1.4. Các ứng dụng của trí tuệ nhân tạo 16](#_Toc199718981)

[1.5. Lợi ích và thách thức của trí tuệ nhân tạo 17](#_Toc199718982)

[1.6. Xu hướng phát triển trí tuệ nhân tạo trong tương lai 18](#_Toc199718983)

[1.7. Kết luận 18](#_Toc199718984)

[CHƯƠNG 2: CÁC KỸ THUẬT HỌC SÂU 19](#_Toc199718985)

[2.1. Giới thiệu chung 19](#_Toc199718986)

[2.2. Cơ bản về mạng nơ-ron 20](#_Toc199718987)

[2.2.1. Cấu trúc mạng 20](#_Toc199718988)

[2.2.2. Huấn luyện mạng 21](#_Toc199718989)

[2.2.3. Tối ưu hóa 21](#_Toc199718990)

[2.2.4. Tensor và phép toán 22](#_Toc199718991)

[2.3. Mạng nơ-ron tích chập – CNN 22](#_Toc199718992)

[2.3.1. Giới thiệu 22](#_Toc199718993)

[2.3.2. Lớp tích chập 23](#_Toc199718994)

[2.3.3. Lớp gộp - Pooling 25](#_Toc199718995)

[2.3.4. Bước nhảy - stride 26](#_Toc199718996)

[2.3.5. Hàm phi tuyến - ReLU 26](#_Toc199718997)

[2.4. Mạng nơ-ron hồi tiếp - RNN 27](#_Toc199718998)

[2.4.1. Tổng quan mạng nơ-ron hồi tiếp 27](#_Toc199718999)

[2.4.2. LSTM 29](#_Toc199719000)

[2.4.3. RNN nâng cao 30](#_Toc199719001)

[2.4.4. Ứng dụng của RNN 30](#_Toc199719002)

[2.5. Mô hình Sinh - Generative Models 30](#_Toc199719003)

[2.5.1. Mạng sinh đối kháng – GAN 30](#_Toc199719004)

[2.5.2. Variational Autoencoder -VAEs 32](#_Toc199719005)

[2.6. Kỹ thuật tối ưu hóa và điều chuẩn 36](#_Toc199719006)

[2.6.1. Tối ưu hóa 36](#_Toc199719007)

[2.6.2. Điều chuẩn 36](#_Toc199719008)

[2.7. Kết luận 37](#_Toc199719009)

[CHƯƠNG 3: ỨNG DỤNG HỌC SÂU CHO CHATBOT HỖ TRỢ Y TẾ 38](#_Toc199719010)

[3.1. Tổng quan về chatbot hỗ trợ y tế 38](#_Toc199719011)

[3.1.1. Định nghĩa và vai trò 38](#_Toc199719012)

[3.1.2. Vai trò của học sâu 38](#_Toc199719013)

[3.2. Mô tả dữ liệu và tiền xử lý 40](#_Toc199719014)

[3.2.1. Dữ liệu đầu vào 40](#_Toc199719015)

[3.2.2. Tiền xử lý dữ liệu 43](#_Toc199719016)

[3.2.3. Kết quả tiền xử lý 47](#_Toc199719017)

[3.3. Xây dựng và huấn luyện mô hình học sâu 49](#_Toc199719018)

[3.3.1. Cấu trúc mô hình 49](#_Toc199719019)

[3.3.2. Biên dịch mô hình 52](#_Toc199719020)

[3.3.3. Huấn luyện 53](#_Toc199719021)

[3.3.4. Huấn luyện khi thay đổi optimizer,learning rate và epoch 54](#_Toc199719022)

[3.4. Đánh giá mô hình 57](#_Toc199719056)

[CHƯƠNG 4: BÀI TẬP CHẠY CÁC PHIÊN BẢN DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG 60](#_Toc199719057)

[4.1. Phiên bản 1 60](#_Toc199719058)

[4.2. Phiên bản 2 65](#_Toc199719059)

[4.3. Phiên bản 3 67](#_Toc199719060)

[4.4. Phiên bản 4 69](#_Toc199719061)

[4.5. So sánh các phiên bản 70](#_Toc199719062)

[KẾT LUẬN 72](#_Toc199719093)

[TÀI LIỆU THAM KHẢO 73](#_Toc199719094)

CHƯƠNG 1: TRÍ TUỆ NHÂN TẠO VÀ CÁC ỨNG DỤNG

* 1. **Khái niệm về trí tuệ nhân tạo**

Trí tuệ Nhân tạo (Artificial Intelligence – AI) là lĩnh vực trong khoa học máy tính tập trung vào việc tạo ra các hệ thống có khả năng thực hiện các nhiệm vụ đòi hỏi trí tuệ con người, như học tập, suy luận, giải quyết vấn đề, hiểu ngôn ngữ và nhận diện hình ảnh. Thuật ngữ "AI" được John McCarthy đề xuất lần đầu vào năm 1956 tại hội thảo Dartmouth, đánh dấu khởi đầu cho một trong những cuộc cách mạng khoa học quan trọng nhất của thế kỷ 20 và 21.

AI không chỉ đơn thuần là một công nghệ, mà còn là một hệ sinh thái bao gồm nhiều lĩnh vực khác nhau như học máy, học sâu, xử lý ngôn ngữ, thị giác máy tính, robot học, và nhiều kỹ thuật khác. Chính sự giao thoa giữa các lĩnh vực này đã tạo nên những hệ thống thông minh ngày càng giống con người hơn về khả năng tư duy và hành động.

Trí tuệ nhân tạo là một lĩnh vực của khoa học máy tính tập trung vào việc phát triển các hệ thống và máy móc có khả năng thực hiện các nhiệm vụ đòi hỏi trí thông minh của con người. Các nhiệm vụ này bao gồm học tập, suy luận, giải quyết vấn đề, nhận thức và tương tác với môi trường. Theo John McCarthy, người tiên phong trong lĩnh vực AI, trí tuệ nhân tạo được định nghĩa là "khoa học và kỹ thuật tạo ra các máy móc thông minh". AI cho phép máy móc học hỏi từ dữ liệu, thích nghi với các tình huống mới và đưa ra quyết định dựa trên phân tích.

AI được chia thành hai loại chính: AI hẹp (Narrow AI) và AI tổng quát (General AI). AI hẹp được thiết kế để thực hiện các nhiệm vụ cụ thể, như nhận diện hình ảnh, dịch ngôn ngữ, hoặc đề xuất nội dung trên các nền tảng như Netflix. Trong khi đó, AI tổng quát hướng tới khả năng thực hiện bất kỳ nhiệm vụ trí tuệ nào mà con người có thể làm, nhưng hiện vẫn đang trong giai đoạn nghiên cứu. Ngoài ra, còn có khái niệm AI siêu thông minh (Superintelligent AI), ám chỉ các hệ thống vượt xa trí tuệ con người, nhưng đây vẫn là một viễn cảnh tương lai.

AI ngày nay đã trở thành một phần không thể thiếu trong cuộc sống, từ trợ lý ảo như Siri, Google Assistant đến các hệ thống tự động hóa trong sản xuất. Tuy nhiên, AI cũng đặt ra các câu hỏi về đạo đức, quyền riêng tư và tác động xã hội. Việc hiểu rõ khái niệm AI là nền tảng để khám phá các ứng dụng và tiềm năng của nó trong các lĩnh vực khác nhau

* 1. **Lịch sử phát triển của trí tuệ nhân tạo**

Lịch sử phát triển của trí tuệ nhân tạo bắt đầu từ những năm 1940, khi các nhà khoa học khởi đầu ý tưởng về máy móc thông minh. Năm 1950, Alan Turing đề xuất bài kiểm tra Turing, một phương pháp đánh giá khả năng của máy móc trong việc thể hiện hành vi thông minh tương đương con người. Năm 1956, hội nghị Dartmouth do John McCarthy tổ chức đã chính thức đặt tên cho lĩnh vực này là "trí tuệ nhân tạo", đánh dấu sự ra đời của AI như một ngành khoa học độc lập

Trong những năm 1960-1980, AI tập trung vào các hệ thống chuyên gia, sử dụng các quy tắc logic để hỗ trợ ra quyết định trong y tế, tài chính và kỹ thuật. Tuy nhiên, giai đoạn này gặp nhiều hạn chế do thiếu dữ liệu và sức mạnh tính toán. Thập niên 1990 chứng kiến sự tiến bộ với các thuật toán học máy, nổi bật là chiến thắng của máy tính Deep Blue của IBM trước kỳ thủ cờ vua Garry Kasparov vào năm 1997.

Sự bùng nổ của AI bắt đầu từ những năm 2010, nhờ vào sự phát triển của dữ liệu lớn (Big Data), sức mạnh tính toán của GPU, và các kỹ thuật học sâu (Deep Learning). Các mô hình AI như mạng nơ-ron sâu đã cải thiện đáng kể khả năng nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên và điều khiển robot. Ngày nay, AI được ứng dụng rộng rãi trong xe tự hành, trợ lý ảo, và chẩn đoán y tế. Tương lai của AI hứa hẹn sẽ mang lại những bước đột phá mới, nhưng cũng đi kèm với các thách thức về quản lý và đạo đức.

* 1. **Các công nghệ cốt lõi của trí tuệ nhân tạo**

Trí tuệ nhân tạo (AI) không phải là một thực thể riêng lẻ, mà là một tập hợp của nhiều công nghệ và kỹ thuật được phát triển để mô phỏng và mở rộng khả năng nhận thức, học hỏi và ra quyết định của con người. Những công nghệ cốt lõi này chính là nền tảng cho sự phát triển của các hệ thống AI ngày nay. Tùy theo ứng dụng cụ thể, các công nghệ có thể được triển khai độc lập hoặc phối hợp để hình thành những hệ thống thông minh phức tạp. Trí tuệ nhân tạo (AI) dựa trên nhiều công nghệ cốt lõi, cho phép máy móc thực hiện các nhiệm vụ thông minh.

Công nghệ đầu tiên là học máy (Machine Learning), giúp hệ thống học hỏi từ dữ liệu mà không cần lập trình chi tiết. Các thuật toán như hồi quy tuyến tính, cây quyết định và mạng nơ-ron được sử dụng để dự đoán và phân loại dữ liệu. Trong phần này, chúng ta sẽ khảo sát các công nghệ cốt lõi cấu thành nên trí tuệ nhân tạo, bao gồm học máy (machine learning), học sâu (deep learning), xử lý ngôn ngữ tự nhiên (natural language processing), thị giác máy tính (computer vision), và robot học (robotics).

Một trong những công nghệ trọng tâm đầu tiên là học máy (machine learning - ML), lĩnh vực cho phép các hệ thống máy tính học hỏi và cải thiện hiệu suất thông qua kinh nghiệm, cụ thể là qua dữ liệu, mà không cần được lập trình rõ ràng cho từng nhiệm vụ. ML không phải là một công nghệ mới, nhưng đã bùng nổ trong thập kỷ qua nhờ sự kết hợp giữa sức mạnh tính toán gia tăng, dữ liệu lớn, và những cải tiến về thuật toán. Học máy có thể được chia thành ba nhóm chính: học có giám sát (supervised learning), học không giám sát (unsupervised learning) và học tăng cường (reinforcement learning).

Trong học có giám sát, mô hình được huấn luyện với một tập dữ liệu đầu vào có gán nhãn, để dự đoán hoặc phân loại đầu ra trong những tình huống mới. Các thuật toán như hồi quy tuyến tính, hồi quy logistic, cây quyết định, và máy vector hỗ trợ (SVM) được sử dụng phổ biến trong lĩnh vực này. Học không giám sát, ngược lại, làm việc với dữ liệu chưa được gán nhãn, nhằm tìm ra các mẫu ẩn hoặc cấu trúc nội tại trong dữ liệu—ví dụ như phân cụm (clustering) với K-means hoặc phân tích thành phần chính (PCA). Học tăng cường đặc biệt thú vị vì nó mô phỏng cách con người học hỏi thông qua tương tác với môi trường, nhận phần thưởng hoặc hình phạt dựa trên hành vi—một cơ chế đã đưa đến thành công của các hệ thống chơi game như AlphaGo của DeepMind.

Một nhánh quan trọng và hiện đại hơn của học máy chính là học sâu (deep learning – DL), vốn được xem như một sự mở rộng phức tạp và mạnh mẽ của các mô hình mạng nơ-ron nhân tạo truyền thống. Học sâu sử dụng các kiến trúc mạng nơ-ron nhiều tầng để trích xuất và học biểu diễn đặc trưng từ dữ liệu. Điều này cho phép hệ thống xử lý những tập dữ liệu phi cấu trúc có độ phức tạp cao như hình ảnh, âm thanh, và văn bản, vốn là những dạng dữ liệu mà các phương pháp truyền thống gặp nhiều giới hạn.

Các kiến trúc phổ biến của học sâu bao gồm mạng nơ-ron tích chập (Convolutional Neural Networks – CNN), thường được sử dụng trong các nhiệm vụ liên quan đến xử lý ảnh và video, nhờ khả năng phát hiện và học các đặc trưng không gian trong hình ảnh. Trong khi đó, mạng nơ-ron hồi tiếp (Recurrent Neural Networks – RNN) và các biến thể như LSTM, GRU đã chứng minh hiệu quả cao trong xử lý chuỗi thời gian và dữ liệu có tính tuần tự như âm thanh hoặc văn bản. Một bước tiến lớn trong học sâu gần đây là sự ra đời của kiến trúc Transformer, khởi nguồn từ bài báo “Attention is All You Need” của Google vào năm 2017. Kiến trúc này đã thay đổi hoàn toàn cách tiếp cận trong xử lý ngôn ngữ tự nhiên và nhanh chóng được mở rộng sang nhiều lĩnh vực khác, tạo tiền đề cho sự ra đời của các mô hình ngôn ngữ lớn như BERT, GPT hay T5.

Tiếp theo, một trong những lĩnh vực ứng dụng quan trọng của AI là xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP), cho phép máy tính hiểu, diễn giải, tạo sinh và tương tác với ngôn ngữ của con người. NLP bao gồm nhiều tác vụ đa dạng như phân tích cú pháp, gán nhãn từ loại, phân tích thực thể có tên (NER), phân loại văn bản, phân tích cảm xúc, dịch máy và tóm tắt văn bản tự động. Nhờ vào NLP, các hệ thống AI ngày nay có thể đọc và hiểu các văn bản, trò chuyện với con người qua chatbot hoặc trợ lý ảo, thậm chí viết văn, sáng tác thơ và tổng hợp ý tưởng.

Các tiến bộ trong NLP phần lớn dựa trên sự phát triển của học sâu, đặc biệt là các mô hình embedding như Word2Vec, GloVe, và sau này là BERT và GPT. Những mô hình này đã giúp AI có khả năng biểu diễn từ ngữ dưới dạng vector số học phản ánh cả ngữ nghĩa và ngữ cảnh, từ đó cải thiện đáng kể hiệu quả của các tác vụ ngôn ngữ. NLP không chỉ giúp nâng cao chất lượng giao tiếp giữa người và máy, mà còn mở rộng ứng dụng trong nhiều lĩnh vực khác nhau như giáo dục (ví dụ: chấm bài tự động), y học (trích xuất thông tin từ hồ sơ bệnh án), và phân tích thị trường (đánh giá phản hồi khách hàng qua mạng xã hội).

Một trong những trụ cột quan trọng khác của trí tuệ nhân tạo là thị giác máy tính (Computer Vision – CV), lĩnh vực cho phép máy tính “nhìn thấy”, phân tích và hiểu được hình ảnh hoặc video. Từ việc nhận diện khuôn mặt trên mạng xã hội, đến giám sát an ninh bằng camera, hay điều khiển xe tự hành qua cảm biến thị giác – tất cả đều dựa vào những tiến bộ trong thị giác máy tính. Không giống như xử lý dữ liệu dạng số hay văn bản, thị giác máy tính phải xử lý khối lượng lớn dữ liệu phi cấu trúc với tính đa chiều và độ phức tạp cao, đòi hỏi các mô hình học sâu hiệu suất cao, đặc biệt là CNN.

Các tác vụ chính trong thị giác máy tính bao gồm: phân loại ảnh (image classification), phát hiện đối tượng (object detection), phân vùng ảnh (semantic segmentation), theo dõi đối tượng (object tracking), và tạo ảnh mới (image generation). Những tiến bộ trong lĩnh vực này đã mở rộng khả năng ứng dụng AI trong nhiều ngành công nghiệp như y tế (phân tích ảnh X-quang, MRI, CT để hỗ trợ chẩn đoán bệnh), nông nghiệp (giám sát cây trồng bằng drone), sản xuất (phát hiện lỗi sản phẩm) và giao thông (phân tích hành vi phương tiện và người đi bộ).

Các mô hình như YOLO (You Only Look Once), SSD (Single Shot Multibox Detector) hay Faster R-CNN đã giúp tăng tốc và nâng cao độ chính xác cho các tác vụ phát hiện vật thể trong thời gian thực. Sự kết hợp giữa thị giác máy tính và cảm biến vật lý, như lidar hay radar, cũng đóng vai trò quan trọng trong việc giúp robot hoặc phương tiện tự hành có được “nhận thức không gian” gần giống với con người.

Tiếp nối đó, robotics – lĩnh vực nghiên cứu và phát triển robot thông minh – là một phần không thể thiếu trong hệ sinh thái AI. Robot ngày nay không còn đơn thuần là những cỗ máy được lập trình để lặp lại một chuỗi hành động cố định, mà dần dần trở thành những hệ thống tự động có khả năng học hỏi, thích nghi và tương tác linh hoạt với môi trường. Sự kết hợp giữa AI với cơ khí, cảm biến và điều khiển giúp tạo ra các robot có khả năng nhận thức và phản ứng theo thời gian thực. Điều này đặc biệt hữu ích trong các tình huống môi trường phức tạp, không thể dự đoán hoàn toàn – như trong phẫu thuật, cứu hộ, hay robot cộng tác trong nhà máy sản xuất.

Robot ứng dụng AI có thể sử dụng thị giác máy tính để nhận diện vật thể, sử dụng NLP để giao tiếp với con người, hoặc dùng học tăng cường để học cách thực hiện các thao tác mới thông qua thử - sai và phản hồi từ môi trường. Trong công nghiệp, robot thông minh đang thay thế lao động tay chân trong các dây chuyền lắp ráp, đặc biệt tại các ngành như ô tô, điện tử, chế tạo chính xác. Trong lĩnh vực dịch vụ, robot có thể được sử dụng để hướng dẫn khách tại sân bay, phục vụ đồ ăn tại nhà hàng, hay hỗ trợ chăm sóc người già tại nhà.

Sự tích hợp giữa các công nghệ không chỉ dừng lại ở mức kỹ thuật mà còn mở ra hệ sinh thái AI toàn diện, nơi dữ liệu được thu thập, xử lý và phân tích theo chu trình khép kín. Một hệ thống AI thông minh hiện đại thường phải có: khả năng thu nhận dữ liệu (từ cảm biến, camera, mic), xử lý và hiểu ngữ nghĩa (thông qua ML, DL, NLP), ra quyết định (từ các mô hình dự đoán) và thực thi hành động (bằng robot hoặc giao diện tương tác). Mỗi công nghệ đảm nhiệm một vai trò quan trọng trong chuỗi này, và việc tối ưu hóa sự phối hợp giữa chúng là yếu tố then chốt dẫn đến thành công của toàn bộ hệ thống.

Các công nghệ cốt lõi của trí tuệ nhân tạo không chỉ là các công cụ riêng lẻ phục vụ các bài toán cụ thể, mà còn là những khối xây dựng nền tảng tạo nên toàn bộ năng lực trí tuệ của hệ thống máy móc hiện đại. Việc phân loại thành các công nghệ như học máy, học sâu, xử lý ngôn ngữ tự nhiên, thị giác máy tính và robot học chủ yếu là để thuận tiện trong nghiên cứu và phát triển, nhưng trên thực tế, các công nghệ này có sự đan xen và bổ trợ lẫn nhau rất mạnh mẽ.

Ở cấp độ kỹ thuật, mỗi công nghệ đóng vai trò riêng trong kiến trúc tổng thể của một hệ thống AI. Học máy và học sâu là hai lớp nền về khả năng “học” từ dữ liệu. Nếu coi AI như một bộ não nhân tạo thì học sâu chính là trung tâm xử lý thông tin thần kinh, với các lớp mạng nơ-ron tích chập, mạng hồi tiếp hoặc transformer đảm nhận vai trò tương tự vỏ não – xử lý đa tầng, phi tuyến và có khả năng biểu diễn trừu tượng cao. Trong khi đó, học máy truyền thống cung cấp những phương pháp kiểm soát, suy luận tuyến tính hoặc giải thích được, đặc biệt hữu dụng trong các hệ thống yêu cầu sự minh bạch và kiểm định rõ ràng.

Xử lý ngôn ngữ tự nhiên và thị giác máy tính có thể được xem là hai “giác quan” chủ đạo của hệ thống AI, tương ứng với khả năng đọc/nghe và nhìn/thị giác. Các công nghệ này không tự tồn tại, mà dựa trên các mô hình học sâu và kiến trúc đặc thù để xử lý dữ liệu phi cấu trúc, chẳng hạn như dùng CNN cho ảnh, hoặc dùng transformer cho văn bản. Việc tách riêng NLP và CV thành hai nhánh lớn là hoàn toàn có cơ sở, vì mỗi lĩnh vực có tập kỹ thuật, ngôn ngữ mô hình, bộ dữ liệu huấn luyện và chỉ số đánh giá riêng biệt.

Robot học, ở chiều ngược lại, không xử lý dữ liệu mà tập trung vào hành động – nó là cánh tay nối dài giúp AI có thể tương tác vật lý với thế giới. Nhưng để robot vận hành linh hoạt, nó vẫn cần các mô-đun nhận thức – tức là sự tích hợp từ học máy, CV, và NLP. Do đó, robot học là điểm hội tụ nơi các công nghệ cốt lõi được “hiện thực hóa” dưới dạng hành vi.

Một yếu tố kỹ thuật quan trọng khác là sự giao tiếp giữa các công nghệ trong hệ thống AI tổng thể. Các công nghệ này thường được triển khai dưới dạng mô-đun hoặc dịch vụ vi mô (microservices) có thể phối hợp thông qua các giao thức truyền thông, API nội bộ hoặc hệ thống điều phối phân tán. Việc chuẩn hóa các giao diện và dữ liệu trung gian (intermediate representations) cho phép NLP tạo ra dữ liệu đầu vào cho ML, hoặc CV cung cấp đầu vào cho hệ thống kiểm soát robot. Các nền tảng AI hiện đại như TensorFlow Extended, PyTorch Lightning hay NVIDIA Isaac đều hướng tới việc kết nối trơn tru các mô-đun công nghệ trong cùng một pipeline xử lý.

Ngoài ra, trong bối cảnh phát triển mô hình AI quy mô lớn (AI at scale), các công nghệ cốt lõi không chỉ cần mạnh về mặt thuật toán mà còn phải tương thích với hạ tầng phần cứng (GPU, TPU), cơ chế song song hóa (data/model parallelism), và khả năng tối ưu hóa (tối ưu gradient, giảm nhiễu, học liên tục). Đây là lý do vì sao hiểu sâu từng công nghệ – và mối liên kết giữa chúng – là điều kiện tiên quyết để xây dựng AI hiệu quả, an toàn và có khả năng mở rộng.

Tóm lại, khi bàn về trí tuệ nhân tạo, không thể chỉ dừng ở cấp độ thuật toán hoặc mô hình cụ thể. Cốt lõi của AI chính là sự kết hợp nhuần nhuyễn giữa các công nghệ nền tảng – mỗi công nghệ đảm nhận một chức năng cơ bản tương ứng với khả năng nhận thức và hành động. Khi những khối chức năng này được tích hợp một cách hợp lý và tối ưu, hệ thống AI sẽ đạt đến trạng thái thông minh thực sự – tức là có thể học hỏi, cảm nhận, suy luận và phản hồi linh hoạt trong thế giới thực.

* 1. **Các ứng dụng của trí tuệ nhân tạo**

Trí tuệ nhân tạo (AI) đã trở thành một công cụ mạnh mẽ, được ứng dụng rộng rãi trong nhiều lĩnh vực, mang lại giá trị to lớn cho xã hội. Các ứng dụng cụ thể của AI là trong lĩnh vực y tế, giao thông, giáo dục, giải trí, và thương mại điện tử.

**Y tế**

AI đang cách mạng hóa ngành y tế bằng cách hỗ trợ chẩn đoán, điều trị, và quản lý dữ liệu bệnh nhân. Các hệ thống AI như IBM Watson phân tích hình ảnh y tế (X-quang, MRI) để phát hiện sớm các bệnh như ung thư với độ chính xác cao. AI cũng hỗ trợ dự đoán các đợt bùng phát dịch bệnh dựa trên dữ liệu toàn cầu, như trong đại dịch COVID-19.

Trong phát triển thuốc, AI giúp xác định các hợp chất tiềm năng, giảm thời gian và chi phí nghiên cứu. Các ứng dụng chăm sóc cá nhân hóa sử dụng AI để phân tích dữ liệu di truyền, đề xuất phác đồ điều trị phù hợp với từng bệnh nhân. Ngoài ra, chatbot AI hỗ trợ tư vấn sức khỏe tâm lý, cung cấp dịch vụ 24/7. Robot phẫu thuật, như hệ thống Da Vinci, sử dụng AI để thực hiện các ca phẫu thuật phức tạp với độ chính xác cao. Tuy nhiên, việc ứng dụng AI trong y tế đòi hỏi đảm bảo tính minh bạch và bảo mật dữ liệu bệnh nhân. Trong tương lai, AI hứa hẹn sẽ tiếp tục cải thiện chất lượng chăm sóc sức khỏe toàn cầu.

**Giao thông**

AI đóng vai trò quan trọng trong việc hiện đại hóa ngành giao thông, đặc biệt là trong phát triển xe tự hành. Các công ty như Tesla và Waymo sử dụng thị giác máy tính và học sâu để giúp xe nhận diện chướng ngại vật, làn đường, và tín hiệu giao thông. AI cũng tối ưu hóa quản lý giao thông đô thị, giảm ùn tắc bằng cách phân tích dữ liệu từ camera và cảm biến.

Trong logistics, AI hỗ trợ tối ưu hóa lộ trình vận chuyển, giảm chi phí nhiên liệu và thời gian giao hàng. Các dịch vụ chia sẻ xe như Uber sử dụng AI để dự đoán nhu cầu và định giá động. Ngoài ra, AI cải thiện an toàn giao thông bằng cách phân tích hành vi lái xe và cảnh báo nguy cơ tai nạn. Trong hàng không, AI hỗ trợ điều khiển không lưu và dự báo thời tiết, đảm bảo an toàn bay. Những tiến bộ này giúp giao thông trở nên thông minh hơn, nhưng cũng đặt ra thách thức về quy định pháp lý và an ninh mạng.

**Giáo dục**

AI đang thay đổi cách học tập và giảng dạy bằng cách cá nhân hóa trải nghiệm giáo dục. Các nền tảng như Coursera hay Khan Academy sử dụng AI để đề xuất nội dung học tập phù hợp với trình độ và sở thích của học sinh. AI cũng hỗ trợ chấm điểm tự động các bài thi trắc nghiệm và bài viết, tiết kiệm thời gian cho giáo viên.

Chatbot giáo dục, như Duolingo, giúp học sinh luyện tập ngôn ngữ mọi lúc, mọi nơi. Trong giáo dục đặc biệt, AI cung cấp các công cụ hỗ trợ học sinh khuyết tật, chẳng hạn như chuyển đổi văn bản thành giọng nói. Các hệ thống phân tích dữ liệu học tập giúp giáo viên theo dõi tiến độ của học sinh và điều chỉnh phương pháp giảng dạy. Ngoài ra, AI hỗ trợ phát triển các môi trường học tập ảo, như thực tế ảo (VR), tăng cường trải nghiệm học tập. Tuy nhiên, việc sử dụng AI trong giáo dục cần đảm bảo công bằng và tránh phụ thuộc quá mức vào công nghệ.

**Giải trí**

AI đã thay đổi ngành giải trí bằng cách nâng cao trải nghiệm người dùng và sáng tạo nội dung. Các nền tảng như Netflix và Spotify sử dụng AI để phân tích sở thích người dùng, đề xuất phim, nhạc, hoặc podcast phù hợp. Trong sản xuất nội dung, AI hỗ trợ tạo nhạc, viết kịch bản, và thiết kế đồ họa, ví dụ như công cụ AI của Adobe. Các trò chơi điện tử tích hợp AI để tạo ra các nhân vật thông minh và môi trường động, nâng cao trải nghiệm chơi game.

AI cũng được sử dụng trong sản xuất phim, như tạo hiệu ứng đặc biệt hoặc chỉnh sửa video tự động. Trong lĩnh vực thể thao, AI phân tích dữ liệu trận đấu để đưa ra chiến lược thi đấu tối ưu. Tuy nhiên, việc sử dụng AI trong giải trí đặt ra vấn đề về bản quyền và tính xác thực của nội dung, đặc biệt với các công nghệ như deepfake. AI tiếp tục mở ra những cơ hội sáng tạo mới cho ngành giải trí.

**Thương mại điện tử**

AI là động lực thúc đẩy sự phát triển của thương mại điện tử, cải thiện trải nghiệm khách hàng và hiệu quả kinh doanh. Các nền tảng như Amazon sử dụng AI để đề xuất sản phẩm dựa trên lịch sử mua sắm và hành vi duyệt web của khách hàng. Chatbot AI cung cấp dịch vụ hỗ trợ khách hàng 24/7, trả lời câu hỏi và giải quyết vấn đề nhanh chóng. AI cũng hỗ trợ quản lý chuỗi cung ứng, dự đoán nhu cầu thị trường và tối ưu hóa kho hàng.

Trong marketing, AI phân tích dữ liệu để tạo ra các chiến dịch quảng cáo cá nhân hóa, tăng tỷ lệ chuyển đổi. Các công nghệ như nhận diện hình ảnh cho phép khách hàng tìm kiếm sản phẩm bằng cách tải ảnh lên. Ngoài ra, AI giúp phát hiện gian lận giao dịch, bảo vệ cả doanh nghiệp và khách hàng. Tuy nhiên, việc sử dụng AI trong thương mại điện tử cần đảm bảo quyền riêng tư của người dùng và tránh lạm dụng dữ liệu.

* 1. **Lợi ích và thách thức của trí tuệ nhân tạo**

Trí tuệ nhân tạo mang lại hàng loạt lợi ích nổi bật trên nhiều lĩnh vực. Về mặt kinh tế, AI giúp tự động hóa các quy trình sản xuất và dịch vụ, nâng cao năng suất lao động, giảm chi phí vận hành và mở ra mô hình kinh doanh mới. Trong y tế, AI hỗ trợ chẩn đoán, dự đoán và cá nhân hóa điều trị; trong giáo dục, AI giúp tạo nội dung dạy học thích ứng với từng cá nhân. Ngoài ra, AI góp phần nâng cao trải nghiệm người dùng, tối ưu hóa chuỗi cung ứng, phát hiện gian lận, và hỗ trợ ra quyết định chiến lược trong các tổ chức lớn.

Tuy nhiên, AI cũng đặt ra nhiều thách thức đáng lưu ý. Một trong những vấn đề chính là đạo đức và quyền riêng tư, đặc biệt khi AI xử lý dữ liệu cá nhân hoặc ra quyết định ảnh hưởng đến con người. Thiên kiến trong dữ liệu và thuật toán có thể dẫn đến các quyết định sai lệch, bất công. Ngoài ra, AI làm dấy lên lo ngại về thất nghiệp do tự động hóa, mất kiểm soát hệ thống, và lạm dụng công nghệ trong giám sát, chiến tranh hoặc thao túng thông tin. Sự thiếu hụt khung pháp lý và năng lực quản trị cũng là rào cản đối với việc triển khai AI một cách bền vững.

* 1. **Xu hướng phát triển trí tuệ nhân tạo trong tương lai**

AI đang bước vào giai đoạn phát triển mạnh mẽ với nhiều xu hướng quan trọng. Trước tiên là sự trỗi dậy của các mô hình ngôn ngữ lớn (Large Language Models – LLM) như GPT, Gemini, Claude… với khả năng xử lý ngôn ngữ và tạo sinh nội dung ở mức độ ngày càng giống con người. Cùng với đó, AI đa phương thức (multimodal AI) đang được chú trọng – đây là các hệ thống có thể đồng thời xử lý và hiểu văn bản, hình ảnh, âm thanh và video, mở rộng khả năng tương tác tự nhiên giữa người và máy.

Một xu hướng khác là AI biên (Edge AI) – các mô hình được triển khai trực tiếp trên thiết bị đầu cuối như điện thoại, xe hơi, drone… để tăng tốc độ phản hồi, giảm phụ thuộc vào đám mây và đảm bảo quyền riêng tư. Đồng thời, AI cũng đang được kết hợp chặt chẽ với các lĩnh vực khác như Internet vạn vật (IoT), blockchain, thực tế ảo (VR/AR), và điện toán lượng tử.

Cuối cùng, xu hướng phát triển AI theo hướng đáng tin cậy, minh bạch và có thể giải thích được (trustworthy & explainable AI) sẽ ngày càng trở nên quan trọng, nhằm tăng tính chấp nhận xã hội và đáp ứng các yêu cầu đạo đức – pháp lý toàn cầu.

* 1. **Kết luận**

Trí tuệ nhân tạo đã, đang và sẽ tiếp tục là lực đẩy cốt lõi cho sự đổi mới sáng tạo trong thế kỷ 21. Với nền tảng là các công nghệ học máy, học sâu, xử lý ngôn ngữ, thị giác máy tính và robot học, AI không chỉ mở rộng giới hạn năng lực của máy móc mà còn góp phần thay đổi cách con người sống, học, làm việc và tương tác. Tuy nhiên, sự phát triển nhanh chóng của AI cũng đặt ra nhiều thách thức, đòi hỏi sự điều tiết linh hoạt giữa đổi mới công nghệ và kiểm soát rủi ro. Việc nghiên cứu và ứng dụng AI một cách có trách nhiệm, minh bạch và nhân văn sẽ là chìa khóa để hiện thực hóa tiềm năng to lớn của công nghệ này trong tương lai.

# CHƯƠNG 2: CÁC KỸ THUẬT HỌC SÂU

## Giới thiệu chung

Phần này nhằm cung cấp một cái nhìn tổng quan về các kỹ thuật học sâu, vai trò quan trọng của chúng trong lĩnh vực học máy, và các ứng dụng thực tiễn trong đời sống. Học sâu không chỉ là một bước tiến vượt bậc trong trí tuệ nhân tạo mà còn là nền tảng cho nhiều ứng dụng mang tính cách mạng trong các lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên, và nhiều ngành công nghiệp khác.

Học sâu (Deep Learning) là một nhánh của học máy, tập trung vào việc sử dụng các mạng nơ-ron nhân tạo với nhiều lớp ẩn để mô hình hóa và giải quyết các bài toán phức tạp. Theo François Chollet trong cuốn *Deep Learning with Python*, học sâu được đặc trưng bởi khả năng tự động trích xuất đặc trưng từ dữ liệu thô mà không cần can thiệp thủ công. Các mạng nơ-ron sâu, với cấu trúc nhiều lớp, cho phép mô hình học được các biểu diễn dữ liệu ở các mức độ trừu tượng khác nhau, từ các đặc trưng cơ bản đến các khái niệm phức tạp.

Ví dụ, trong nhận diện hình ảnh, một mạng nơ-ron sâu có thể tự động học cách nhận biết các cạnh, hình dạng, và cuối cùng là các đối tượng cụ thể như khuôn mặt hay xe hơi, chỉ từ dữ liệu pixel thô. Điều này làm cho học sâu trở thành một công cụ mạnh mẽ trong việc xử lý các bài toán mà dữ liệu có cấu trúc phức tạp hoặc kích thước lớn.

Học sâu đã tạo ra những bước đột phá trong nhiều lĩnh vực nhờ khả năng xử lý khối lượng dữ liệu lớn và học hỏi từ các mẫu phức tạp. Một số ứng dụng nổi bật bao gồm:

* ***Thị giác máy tính***: Nhận diện đối tượng, phân đoạn hình ảnh, và phát hiện bất thường trong y học
* ***Xử lý ngôn ngữ tự nhiên***: Dịch máy, chatbot thông minh, và phân tích cảm xúc.
* ***Trí tuệ nhân tạo tổng quát***: Hỗ trợ phát triển các hệ thống tự động hóa, xe tự hành, và trợ lý ảo.

Sự phát triển của phần cứng (như GPU và CPU) cùng với các thư viện học sâu phổ biến như TensorFlow và PyTorch đã giúp học sâu trở nên dễ tiếp cận hơn, cho phép các nhà nghiên cứu và kỹ sư áp dụng nó vào nhiều bài toán thực tiễn.

Để làm rõ tiềm năng của học sâu, dưới đây là hai ứng dụng tiêu biểu:

* Nhận diện hình ảnh: Các mô hình CNN, như ResNet hay VGG, được sử dụng để phân loại hình ảnh trong các ứng dụng như nhận diện khuôn mặt hoặc phát hiện đối tượng trong xe tự hành. Ví dụ, một mô hình CNN có thể được huấn luyện để nhận diện các loài động vật trong ảnh với độ chính xác cao.
* Chatbot thông minh: Các mô hình RNN hoặc Transformer (như BERT) được sử dụng để xây dựng chatbot có khả năng hiểu và trả lời các câu hỏi của người dùng một cách tự nhiên. Ví dụ, một chatbot dịch vụ khách hàng có thể xử lý các yêu cầu đặt hàng hoặc giải đáp thắc mắc mà không cần can thiệp của con người.

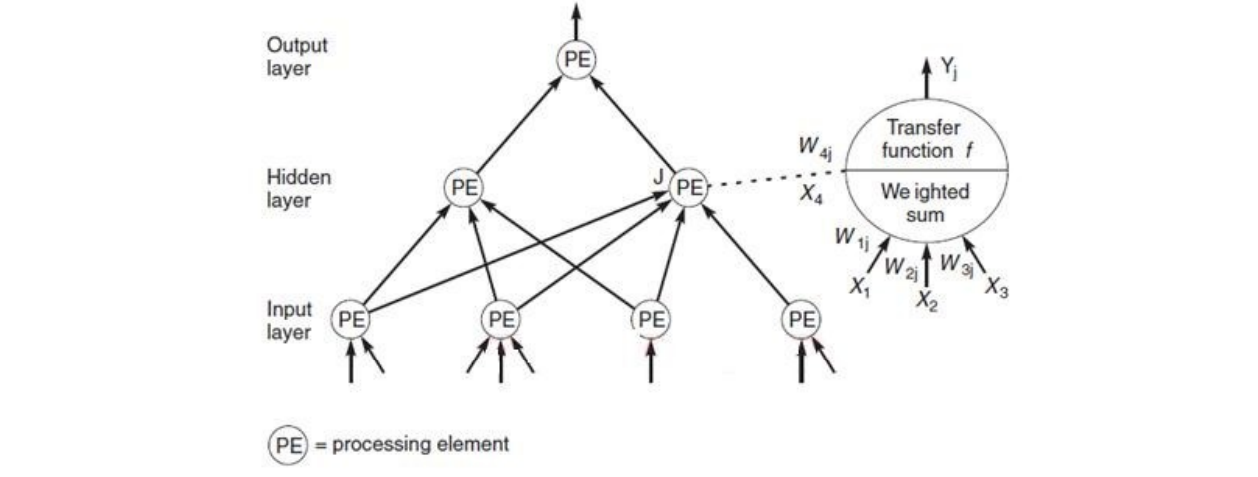
Những ví dụ này minh họa cách học sâu không chỉ giải quyết các bài toán phức tạp mà còn mang lại giá trị thực tiễn, từ cải thiện trải nghiệm người dùng đến tối ưu hóa quy trình kinh doanh.

Học sâu đã và đang định hình tương lai của trí tuệ nhân tạo, với khả năng xử lý dữ liệu phức tạp và tạo ra các giải pháp sáng tạo. Phần tiếp theo của chương sẽ đi sâu vào các kỹ thuật cụ thể, bắt đầu bằng cấu trúc và hoạt động của mạng nơ-ron, nhằm cung cấp nền tảng vững chắc cho việc áp dụng học sâu vào thực tiễn.

## Cơ bản về mạng nơ-ron

Mạng nơ-ron nhân tạo là một mô hình tính toán được lấy cảm hứng từ cách hoạt động của não bộ con người. Đây là nền tảng của học sâu, cho phép máy tính học hỏi từ dữ liệu một cách tự động và hiệu quả. Một mạng nơ-ron bao gồm nhiều lớp nơ-ron kết nối với nhau, trong đó mỗi nơ-ron thực hiện một phép toán đơn giản. Tuy nhiên, khi các nơ-ron này được kết hợp thành một mạng lớn, chúng có thể học được các mô hình và mối quan hệ phức tạp trong dữ liệu.

* + 1. Cấu trúc mạng



1. Cấu trúc mạng nơ-ron

Một mạng nơ-ron điển hình bao gồm ba loại lớp chính:

* ***Lớp đầu vào (input layer)***: Đây là lớp nhận dữ liệu thô từ bên ngoài. Ví dụ, trong bài toán phân loại ảnh, lớp đầu vào có thể nhận các giá trị pixel của ảnh. Trong xử lý ngôn ngữ tự nhiên, lớp này có thể nhận các từ hoặc ký tự của câu.
* ***Lớp ẩn (hidden layers)***: Đây là các lớp ở giữa, chịu trách nhiệm xử lý và biến đổi dữ liệu từ lớp đầu vào. Mỗi lớp ẩn có thể học được các đặc trưng hoặc biểu diễn phức tạp hơn từ dữ liệu. Một mạng nơ-ron có thể có một hoặc nhiều lớp ẩn, và số lượng lớp ẩn càng nhiều, mạng càng "sâu", từ đó có khả năng học được các biểu diễn phức tạp hơn.
* ***Lớp đầu ra (output layer)***: Đây là lớp cuối cùng, tạo ra kết quả dự đoán của mạng. Tùy thuộc vào bài toán, lớp đầu ra có thể có một hoặc nhiều nơ-ron. Ví dụ, trong bài toán phân loại nhị phân (chẳng hạn như phân biệt chó và mèo), lớp đầu ra có thể có một nơ-ron với giá trị từ 0 đến 1, biểu thị xác suất thuộc về một lớp. Trong bài toán hồi quy (dự đoán giá trị liên tục, như giá nhà), lớp đầu ra có thể có một nơ-ron trả về giá trị số.

Trong mạng nơ-ron, mỗi nơ-ron nhận đầu vào từ các nơ-ron ở lớp trước, nhân các đầu vào này với trọng số tương ứng, cộng với một độ lệch (bias), rồi đưa kết quả qua hàm kích hoạt để tạo đầu ra phi tuyến. Các hàm kích hoạt phổ biến bao gồm ReLU, trả về giá trị đầu vào nếu lớn hơn 0, ngược lại trả về 0, giúp mạng học các mối quan hệ phi tuyến một cách hiệu quả; và Sigmoid, trả về giá trị từ 0 đến 1, thường dùng trong phân loại nhị phân để biểu thị xác suất.

Ví dụ, trong bài toán phân loại ảnh chó và mèo, lớp đầu vào nhận 784 giá trị pixel từ ảnh 28x28. Các lớp ẩn sử dụng ReLU để học các đặc trưng từ đơn giản (cạnh, góc) đến phức tạp (tai, mắt), và lớp đầu ra, thường dùng Sigmoid, dự đoán xác suất ảnh là chó hoặc mèo. Trong chatbot, như mô hình Transformer, các nơ-ron và hàm kích hoạt như ReLU hỗ trợ xử lý dữ liệu văn bản, giúp phân tích ý định và tạo phản hồi tự nhiên, ví dụ trả lời câu hỏi về áo đỏ giá dưới 500.000 VNĐ dựa trên ngữ cảnh hội thoại.

* + 1. Huấn luyện mạng

Huấn luyện mạng nơ-ron là quá trình điều chỉnh các trọng số và độ lệch để mạng có thể dự đoán chính xác nhất có thể trên dữ liệu huấn luyện. Mục tiêu là giảm thiểu sai số giữa dự đoán của mạng và giá trị thực tế.

Hàm Mất Mát

Hàm mất mát (loss function) là công cụ quan trọng trong học sâu, dùng để đánh giá mức độ sai lệch giữa dự đoán của mạng nơ-ron và giá trị thực tế, từ đó hướng dẫn quá trình tối ưu hóa. Tùy vào bài toán, các hàm mất mát khác nhau được áp dụng. Cross-entropy thường được sử dụng cho bài toán phân loại, đo lường sự khác biệt giữa phân phối xác suất dự đoán và nhãn thực tế, với biến thể categorical cross-entropy dành cho phân loại đa lớp và binary cross-entropy cho phân loại nhị phân.

Trong khi đó, Mean Squared Error (MSE) phù hợp cho bài toán hồi quy, tính bình phương trung bình của sai số giữa giá trị dự đoán và thực tế. Ví dụ, trong phân loại nhị phân, nếu nhãn thực tế là 1 (chó) và mạng dự đoán xác suất 0.8, cross-entropy sẽ ghi nhận sai số nhỏ; nhưng nếu dự đoán 0.2, sai số sẽ lớn hơn, thúc đẩy điều chỉnh trọng số.

Trong ngữ cảnh chatbot, như khi xử lý ý định người dùng, cross-entropy giúp mô hình như BERT hoặc GPT phân loại chính xác ý định, cải thiện phản hồi dựa trên dữ liệu huấn luyện.

* + 1. Tối ưu hóa

Để giảm thiểu hàm mất mát trong quá trình huấn luyện các mô hình học sâu, các thuật toán tối ưu hóa như gradient giảm ngẫu nhiên (SGD) đóng vai trò quan trọng. SGD hoạt động bằng cách tính gradient của hàm mất mát dựa trên các trọng số, sau đó điều chỉnh trọng số theo hướng ngược lại của gradient để giảm giá trị mất mát, lặp lại quá trình này cho đến khi hàm mất mát hội tụ.

Thuật toán lan truyền ngược (Backpropagation) là cốt lõi để tính gradient hiệu quả, truyền sai số từ lớp đầu ra ngược về các lớp trước, từ đó cập nhật trọng số toàn mạng. Ngoài SGD, các biến thể tối ưu hóa như Adam, kết hợp động lượng và điều chỉnh tốc độ học cho từng tham số để hội tụ nhanh hơn, hoặc RMSprop, hiệu quả trong các bài toán có dữ liệu không ổn định bằng cách điều chỉnh tốc độ học dựa trên lịch sử gradient, cũng được sử dụng rộng rãi.

Ví dụ, khi huấn luyện mạng phân loại ảnh, nếu mô hình dự đoán sai một ảnh chó là mèo, SGD sẽ điều chỉnh trọng số để giảm sai số, cải thiện độ chính xác trong các lần lặp tiếp theo. Trong bối cảnh chatbot, các thuật toán này giúp tối ưu hóa các mô hình như Transformer hay GPT, đảm bảo phản hồi chính xác và tự nhiên hơn dựa trên dữ liệu huấn luyện.

* + 1. Tensor và phép toán

Trong học sâu, tensor là cấu trúc dữ liệu cơ bản, tổng quát hóa vector và ma trận để biểu diễn dữ liệu đa chiều. Tensor rank-0 là một số đơn lẻ (ví dụ: 5), rank-1 là mảng 1 chiều (ví dụ: [1, 2, 3]), rank-2 là mảng 2 chiều (ví dụ: [[1, 2], [3, 4]]), và rank-3 trở lên biểu diễn dữ liệu phức tạp như ảnh màu RGB với kích thước (chiều cao, chiều rộng, 3 kênh màu). Các phép toán tensor là nền tảng cho tính toán trong mạng nơ-ron.

Phép toán từng phần tử thực hiện các phép toán như cộng hoặc nhân trên từng phần tử tương ứng (ví dụ: [1, 2] + [3, 4] = [4, 6]). Phép tích tensor bao gồm nhân ma trận, tích vô hướng, hoặc các phép toán phức tạp hơn cho tensor đa chiều, như nhân ma trận trọng số với vector đầu vào trong một lớp nơ-ron. Reshaping cho phép thay đổi hình dạng tensor mà không thay đổi dữ liệu, ví dụ chuyển vector 1x9 thành ma trận 3x3. Các phép toán này được tối ưu hóa trên GPU, hỗ trợ xử lý dữ liệu lớn và mô hình phức tạp.

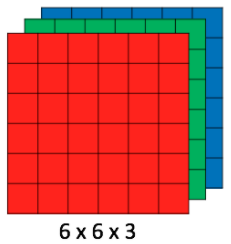
Trong chatbot, ví dụ, một lớp fully-connected sử dụng vector đầu vào, nhân với ma trận trọng số, cộng vector độ lệch, và áp dụng hàm kích hoạt để tạo phản hồi, tận dụng tensor để đảm bảo hiệu quả tính toán và độ chính xác của mô hình như Transformer hoặc GPT.

## Mạng nơ-ron tích chập – CNN

* + 1. Giới thiệu

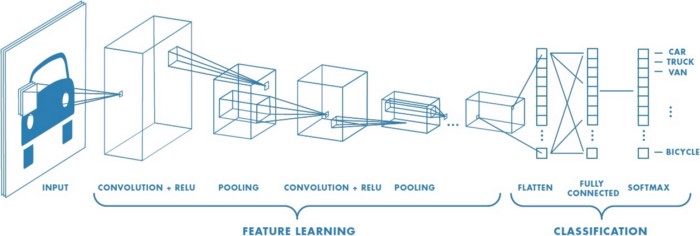
Trong mạng neural, mô hình mạng neural tích chập là 1 trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.

CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB 6x6x3 (3 ở đây là giá trị RGB).



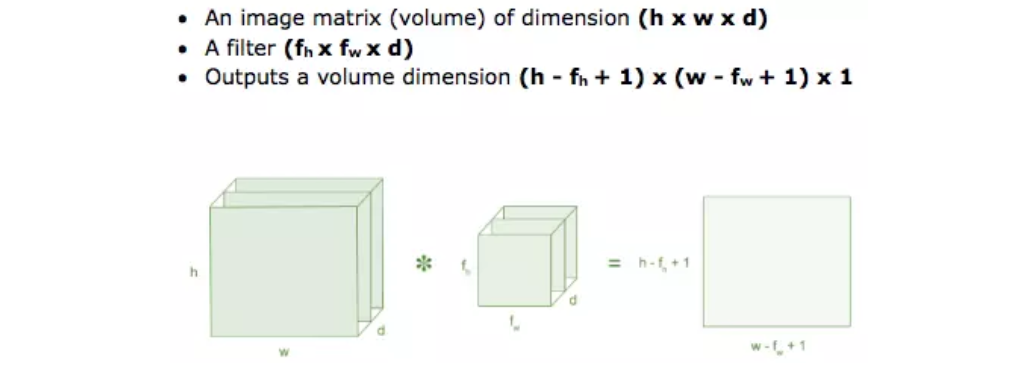
1. Mảng ma trận RGB 6x6x3

Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị



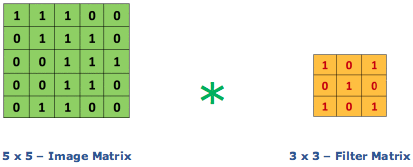
1. Luồng CNN xử lý hình ảnh đầu vào
   * 1. Lớp tích chập

Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vương nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.



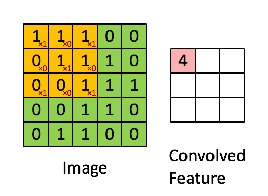
1. Lớp tích chập

Xem xét 1 ma trận 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như hình bên dưới.



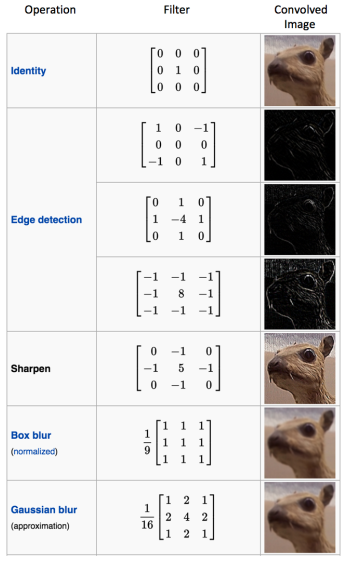
1. Ma trận 5x5 và bộ lọc 3x3

Sau đó, lớp tích chập của ma trận hình ảnh 5 x 5 nhân với ma trận bộ lọc 3 x 3 gọi là 'Feature Map' như hình bên dưới.



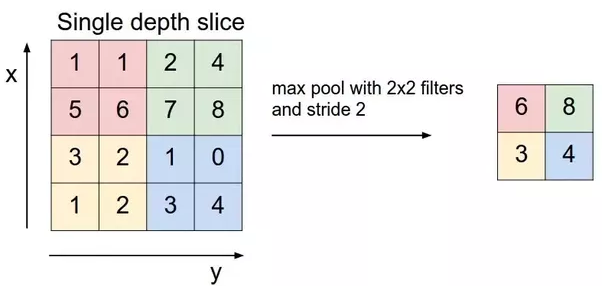
1. Lớp tích chập của ma trận 5x5 và bộ lọc 3x3

Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau.



1. Hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau
   * 1. Lớp gộp - Pooling

Lớp gộp được sử dụng để giảm kích thước không gian của bản đồ đặc trưng, từ đó giảm số lượng tham số và tăng hiệu quả tính toán. Loại phổ biến nhất là max-pooling, lấy phần tử lớn nhất từ ma trận đối tượng, hoặc lấy tổng trung bình. Tổng tất cả các phần tử trong map gọi là sum pooling

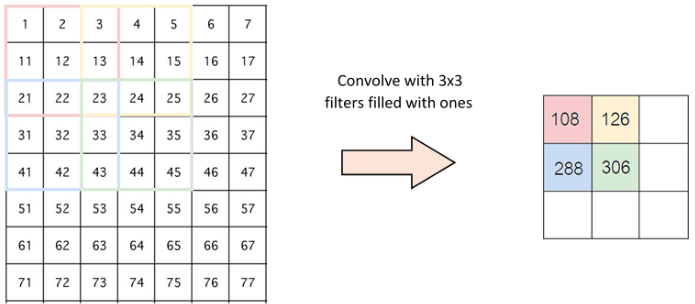


1. Lớp gộp

**Các loại gộp khác**

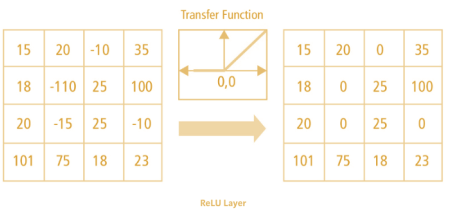
* + **Average-pooling**: Lấy giá trị trung bình thay vì giá trị lớn nhất.
  + **Global average pooling**: Giảm toàn bộ bản đồ đặc trưng thành một giá trị duy nhất, thường được dùng ở lớp cuối của CNN để tạo đầu ra phân loại.
    1. Bước nhảy - stride

Stride là số pixel thay đổi trên ma trận đầu vào. Khi stride là 1 thì ta di chuyển các kernel 1 pixel. Khi stride là 2 thì ta di chuyển các kernel đi 2 pixel và tiếp tục như vậy. Hình dưới là lớp tích chập hoạt động với stride là 2.



1. Stride
   * 1. Hàm phi tuyến - ReLU

ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là: ƒ (x) = max (0, x). Tại sao ReLU lại quan trọng: ReLU giới thiệu tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm.



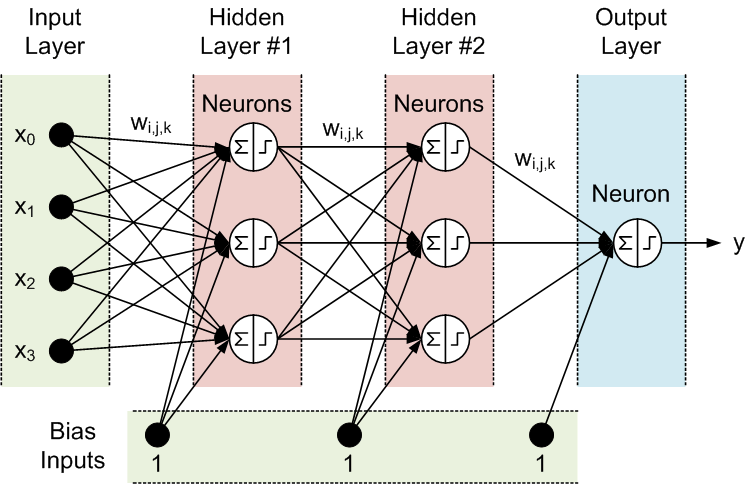
1. Hàm phi tuyến

Có 1 số hà phi tuyến khác như tanh, sigmoid cũng có thể được sử dụng thay cho ReLU. Hầu hết người ta thường dùng ReLU vì nó có hiệu suất tốt.

## Mạng nơ-ron hồi tiếp - RNN

* + 1. Tổng quan mạng nơ-ron hồi tiếp

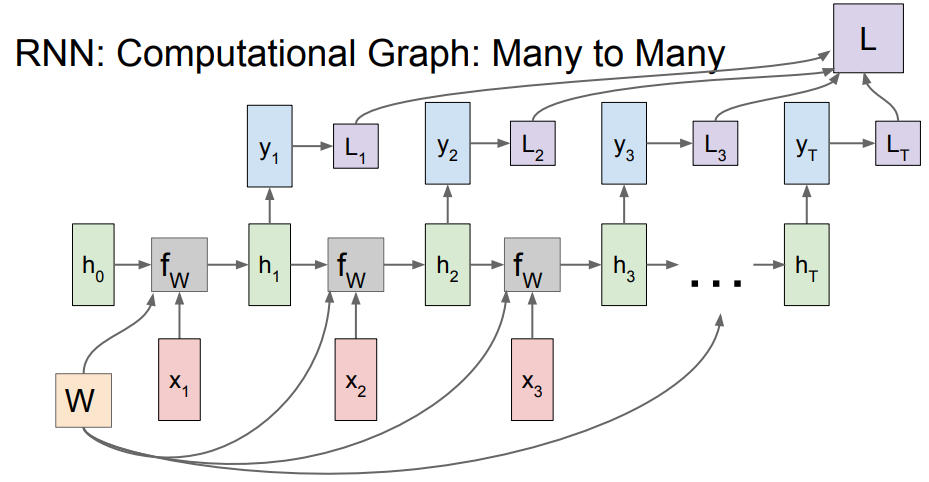
Để có thể hiểu rõ về RNN, trước tiên chúng ta cùng nhìn lại mô hình Neural Network dưới đây:



1. Mô hình Neural Network

Như đã biết thì Neural Network bao gồm 3 phần chính là Input layer, Hidden layer và Output layer, ta có thể thấy là đầu vào và đầu ra của mạng neuron này là độc lập với nhau. Như vậy mô hình này không phù hợp với những bài toán dạng chuỗi như mô tả, hoàn thành câu, ... vì những dự đoán tiếp theo như từ tiếp theo phụ thuộc vào vị trí của nó trong câu và những từ đằng trước nó.

Và như vậy RNN ra đời với ý tưởng chính là sử dụng một bộ nhớ để lưu lại thông tin từ từ những bước tính toán xử lý trước để dựa vào nó có thể đưa ra dự đoán chính xác nhất cho bước dự đoán hiện tại.



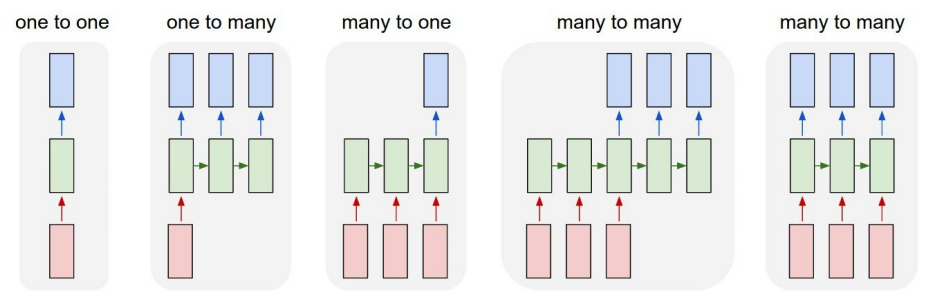
1. Mô hình RNN

Giải thích: Nếu như mạng Neural Network chỉ là input layer  đi qua hidden layer  và cho ra output layer  với đầy đủ kết nối giữa các layer thì trong RNN, các input  sẽ được kết hợp với hidden layer  bằng hàm  để tính toán ra hidden layer  hiện tại và output  sẽ được tính ra từ , là tập các trọng số và nó được ở tất cả các cụm, các ​là các hàm mất mát sẽ được giải thích sau. Như vậy kết quả từ các quá trình tính toán trước đã được "nhớ" bằng cách kết hợp thêm ​ tính ra  để tăng độ chính xác cho những dự đoán ở hiện tại. Cụ thể quá trình tính toán được viết dưới dạng toán như sau:

Hàm  chúng ta sẽ xử dụng hàm , công thức trên sẽ trở thành :

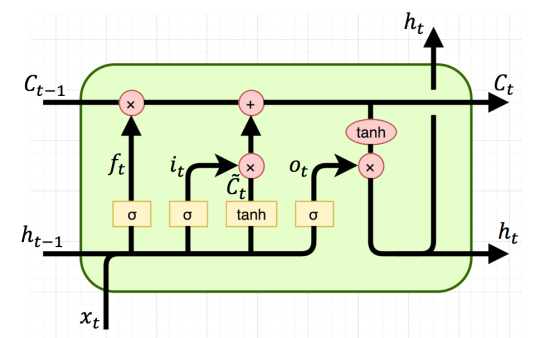
,, ​. Đối với mạng NN chỉ sử dụng một ma trận trọng số W duy nhất thì với RNN nó sử dụng 3 ma trận trọng số cho 2 quá trình tính toán:  kết hợp với "bộ nhớ trước"  và  kết hợp với  để tính ra "bộ nhớ của bước hiện tại" từ đó kết hợp với để tính ra

Ngoài mô hình Many to Many như ta thấy ở trên thì RNN còn rất nhiều dạng khác như sau:



1. Dạng khác của mô hình RNN
   * 1. LSTM

Ta có thể thấy là các state càng xa ở trước đó thì càng bị vanishing gradient và các hệ số không được update với các frame ở xa. Hay nói cách khác là RNN không học được từ các thông tin ở trước đó xa do vanishing gradient. Như vậy về lý thuyết là RNN có thể mang thông tin từ các layer trước đến các layer sau, nhưng thực tế là thông tin chỉ mang được qua một số lượng state nhất định, sau đó thì sẽ bị vanishing gradient, hay nói cách khác là model chỉ học được từ các state gần nó. Để khắc phục các vấn đề của RNN cơ bản, biến thể nâng cao là được phát triển.



1. Mô hình LSTM

Ở state thứ t của mô hình LSTM:

* Output:  ta gọi *c* là cell state, *h* là hidden state.
* Input: ​. Trong đó  là input ở state thứ t của modellà output của layer trước.

tương ứng với **f**orget gate, **i**nput gate và **o**utput gate

* Forget gate: =σ()
* Input gate:=σ()
* Output gate:  =σ()

**LSTM**: Sử dụng các **cổng** (gate) để kiểm soát luồng thông tin:

* Forget gate: Quyết định thông tin nào từ trạng thái trước cần quên.
* Input gate: Quyết định thông tin mới nào được thêm vào.
* Output gate: Quyết định đầu ra dựa trên trạng thái hiện tại.

Nhờ cơ chế này, LSTM có khả năng lưu trữ thông tin qua nhiều bước thời gian, phù hợp với các tác vụ yêu cầu nhớ lâu dài

* + 1. RNN nâng cao

Các mạng nơ-ron hồi tiếp nâng cao (RNN) được cải tiến để xử lý hiệu quả các bài toán chuỗi, với các kỹ thuật như Dropout hồi tiếp và RNN hai chiều. Dropout hồi tiếp áp dụng dropout lên trạng thái ẩn giữa các bước thời gian, giúp giảm quá khớp, đặc biệt trong các mô hình lớn, bằng cách ngẫu nhiên bỏ qua một số kết nối trong quá trình huấn luyện. RNN hai chiều xử lý chuỗi theo cả hai hướng (từ trái sang phải và ngược lại), cho phép mô hình nắm bắt ngữ cảnh đầy đủ hơn. Ví dụ, trong nhận diện giọng nói, từ hiện tại có thể phụ thuộc vào cả từ trước và sau, giúp cải thiện độ chính xác.

* + 1. Ứng dụng của RNN

Ứng dụng của RNN rất đa dạng. Trong dự báo chuỗi thời gian, RNN dự đoán giá cổ phiếu, thời tiết, hoặc lưu lượng giao thông dựa trên dữ liệu lịch sử. Trong dịch máy, RNN và các biến thể như LSTM hoặc GRU chuyển đổi văn bản giữa các ngôn ngữ, ví dụ từ tiếng Việt sang tiếng Anh. Về tạo văn bản, RNN hỗ trợ chatbot sinh ra phản hồi tự nhiên, như trả lời câu hỏi về áo đỏ giá dưới 500.000 VNĐ, hoặc tạo bài viết tự động.

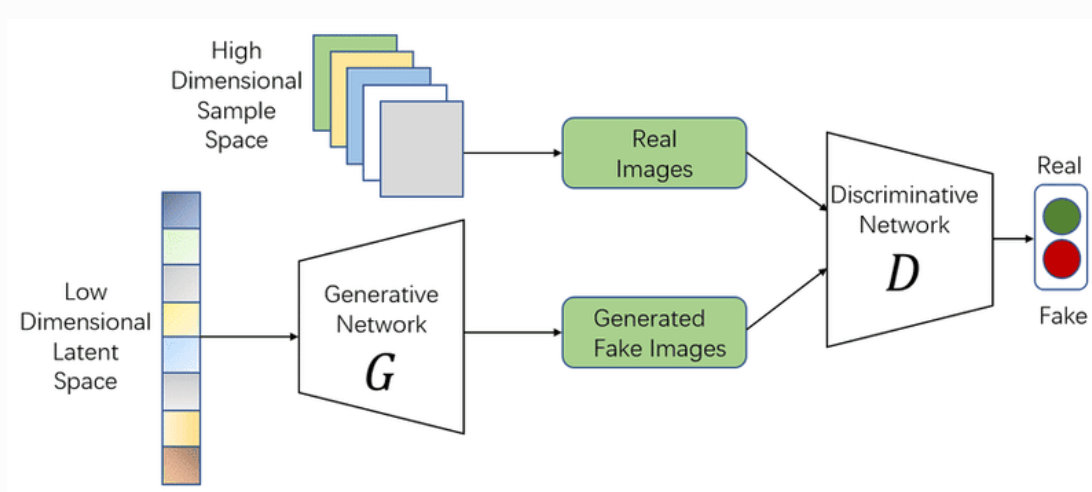
Trong nhận diện giọng nói, RNN xử lý tín hiệu âm thanh để chuyển thành văn bản, tăng cường khả năng hiểu ngôn ngữ tự nhiên. Các ứng dụng này tận dụng khả năng xử lý chuỗi của RNN, kết hợp với các kỹ thuật tối ưu như lan truyền ngược và hàm mất mát (như cross-entropy), để đạt hiệu suất cao trong các bài toán phức tạp.

## Mô hình Sinh - Generative Models

* + 1. Mạng sinh đối kháng – GAN

Được giới thiệu bởi Ian Goodfellow và các cộng sự vào năm 2014, Mạng Sinh Đối Kháng (Generative Adversarial Networks – GANs) là một trong những mô hình phổ biến nhất trong [Generative AI](https://fpt.ai/vi/bai-viet/generative-ai/). GANs bao gồm hai mạng nơ-ron đối kháng nhau: Mạng tạo (Generator) và Mạng phân biệt (Discriminator).

 Generator cố gắng tạo ra dữ liệu giả, trong khi Discriminator học cách phân biệt giữa dữ liệu thật và giả. Quá trình đối kháng này diễn ra liên tục cho đến khi cả hai mạng đạt đến trạng thái cân bằng: dữ liệu do Generator tạo ra đủ giống với dữ liệu thật để “đánh lừa” được Discriminator.



1. Mạng Sinh Đối Kháng

Chẳng hạn, để tạo hình ảnh khuôn mặt người, generator nhận một vector ngẫu nhiên làm đầu vào và tạo ra một hình ảnh khuôn mặt giả. Discriminator sau đó so sánh hình ảnh này với các hình ảnh khuôn mặt thật trong tập dữ liệu và xác định xem nó là thật hay giả. Dựa vào kết quả phân loại, generator được cải thiện để tạo ra hình ảnh khuôn mặt chân thực hơn, trong khi discriminator cũng học cách phân biệt chính xác hơn. Hai mô hình này liên tục cạnh tranh, giúp hệ thống tạo ra hình ảnh giống thật nhất.

Quá trình đào tạo mô hình Generative Adversarial Networks bao gồm các bước chính sau:

1. Khởi tạo (Initialization): Generator và Discriminator được thiết lập với các trọng số ngẫu nhiên.
2. Vòng lặp huấn luyện (Training Loop): Generator tạo dữ liệu giả, trong khi Discriminator xác định dữ liệu là thật hay giả dựa trên xác suất từ 0 đến 1.
3. Lan truyền ngược (Backpropagation): Discriminator sử dụng tín hiệu lỗi để cập nhật trọng số, sau đó lỗi này cũng được truyền ngược để điều chỉnh trọng số của Generator.
4. Tạo mẫu (Sampling): Sau khi huấn luyện, Generator có thể tạo dữ liệu mới dựa trên phân phối mà nó đã học.

Generative Adversarial Networks có thể học từ các phân phối dữ liệu phức tạp, đa phương thức để tạo ra các mẫu đa dạng và chất lượng cao. Tuy nhiên, việc đào tạo GAN có thể gặp khó khăn, chẳng hạn như Generator chỉ tạo ra các mẫu lặp lại. StyleGAN là biến thể của GANS được NVIDIA phát triển để giải quyết các vấn đề này.

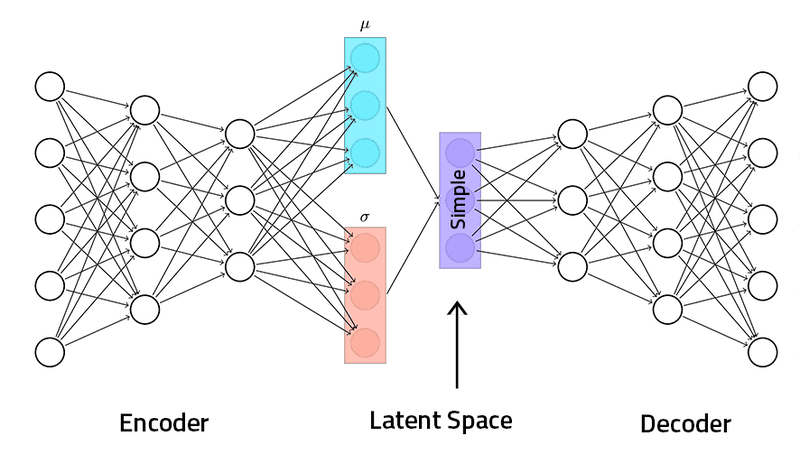
Mô hình học sâu này có khả năng tạo ra các hình ảnh chân dung con người với độ chân thực cao. Năm 2019, một kỹ sư của Uber đã sử dụng StyleGAN để tạo các khuôn mặt mới cho trang web “This Person Does Not Exist”. Wang đã bày tỏ sự kinh ngạc vì khả năng phân tích tất cả các đặc điểm có liên quan trên khuôn mặt của con người để tạo các ảnh chân dung giống như thật mỗi khi tải lại trang của Generative Models này.

* + 1. Variational Autoencoder -VAEs

So với Generative Adversarial Networks, Variational Autoencoder là một mô hình tạo sinh dễ kiểm soát đầu ra hơn. Là một dạng mở rộng của Autoencoder, VAEs không chỉ có khả năng tái tạo dữ liệu mà còn có thể biểu diễn dữ liệu trong không gian tiềm ẩn dưới dạng phân phối xác suất. Ý tưởng cốt lõi của Variational Autoencoders là mã hóa đầu vào thành một phân phối Gaussian, với các tham số trung bình (Mean) và phương sai (Variance).

Variational Autoencoders hoạt động dựa trên hai mạng chính: Encoder và Decoder. Encoder đảm nhận vai trò mã hóa dữ liệu đầu vào thành tham số của phân phối xác suất trong không gian tiềm ẩn. Decoder sử dụng các mẫu từ phân phối này để tái tạo lại dữ liệu đầu vào.

Mục tiêu chính của Variational Autoencoders là giảm lỗi tái tạo (Reconstruction Loss) và đảm bảo rằng các biểu diễn trong không gian tiềm ẩn liên tục và đầy đủ. Điều này mang lại lợi thế lớn khi ứng dụng vào các bài toán sinh dữ liệu và giảm chiều dữ liệu.



1. Variational Autoencoder

Quá trình huấn luyện VAE có thể được mô tả qua các bước sau:

Mã hóa (Encoding): Dữ liệu đầu vào được đưa vào bộ mã hóa, nơi dữ liệu được nén vào một không gian latent được giả định tuân theo phân phối Gaussian, bao gồm trung bình (μ) và độ lệch chuẩn (σ). Encoder được xây dựng bằng mạng nơ-ron sâu, học cách nén dữ liệu đầu vào thành các biểu diễn tiềm ẩn. VAEs biểu diễn dữ liệu thành một vùng không gian xác suất (tập hợp các phân phối cho mỗi mẫu đầu vào

Lấy mẫu (Sampling): Thay vì trực tiếp lấy mẫu từ phân phối xác suất, VAEs áp dụng một kỹ thuật đặc biệt gọi là Reparameterization Trick để giữ tính khả vi trong quá trình lan truyền ngược. Bằng cách biểu diễn *z = μ + σ \* ε*, trong đó *ε* là nhiễu được lấy mẫu từ Gaussian chuẩn, mô hình đảm bảo rằng gradient có thể được lan truyền ngược một cách chính xác.

Giải mã (Decoding): Decoder sử dụng điểm được lấy mẫu trong không gian tiềm ẩn (z) để tái tạo lại dữ liệu mới. Mục tiêu của giải mã là tái tạo dữ liệu đầu vào từ biểu diễn trong không gian latent.

Tính toán hàm loss: Hàm loss của Variational Autoencoders bao gồm:

* Reconstruction loss: Đo lường mức độ mà decoder tái tạo chính xác dữ liệu đầu vào để đánh giá độ giống giữa dữ liệu đầu vào và đầu ra.
* KL-divergence loss: Đo lường sự khác biệt giữa phân phối Gaussian đã học và phân phối thực tế trong không gian latent để đảm bảo phân phối trong không gian tiềm ẩn gần với Gaussian chuẩn.

Lan truyền ngược (Backpropagation): Dựa vào tín hiệu lỗi từ hàm loss, các trọng số của bộ mã hóa và bộ giải mã được cập nhật thông qua lan truyền ngược.

Sau khi được huấn luyện, Variational Autoencoders có thể mẫu lấy mẫu từ phân phối Gaussian để tạo dữ liệu mới, có cấu trúc chặt chẽ với , nén thông tin thành các biểu diễn có chiều thấp trong không gian latent hoặc Trích xuất đặc trưng quan trọng từ dữ liệu đầu vào. Tuy nhiên, do bản chất xác suất của mô hình, VAE có thể tạo ra các mẫu mờ, chất lượng thấp hoặc bị nhiễu.

Để cải thiện chất lượng, các phương pháp như đào tạo đối nghịch (adversarial training) và flow-based models đã được đề xuất nhằm tăng hiệu suất của VAE trong việc tạo dữ liệu rõ nét hơn.

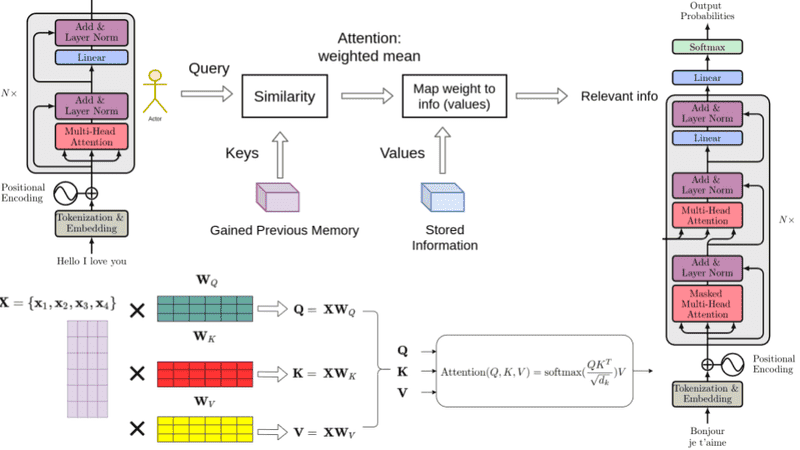
* + 1. Mô hình Transformer

Mô hình Transformer là một loại kiến trúc mạng nơ-ron xuất sắc trong việc xử lý dữ liệu tuần tự, thường được gắn liền với các [mô hình ngôn ngữ lớn](https://fpt.ai/vi/bai-viet/mo-hinh-ngon-ngu-lon/) (LLM). Kiến trúc Transformer được mô tả lần đầu trong bài báo nổi tiếng năm 2017 “Attention is All You Need” của Vaswani và cộng sự, hiện được coi là một bước ngoặt trong học sâu ([Deep Learning](https://fpt.ai/vi/bai-viet/deep-learning/)).

Transformer Model là một trong những mô hình dẫn dắt làn sóng Transformer AI. Các nhà nghiên cứu của Stanford đã gọi Transformer là “foundation models” (mô hình nền tảng) trong một bài báo vào tháng 8 năm 2021 vì họ cho rằng những mô hình này đang thúc đẩy một sự thay đổi mô hình trong AI. Họ nhận định rằng: “Quy mô và phạm vi tuyệt vời của các mô hình nền tảng trong vài năm qua đã mở rộng trí tưởng tượng của chúng ta về những gì có thể.”

Transformer Model có thể học cách hiểu ngữ cảnh và từ đó suy ra ý nghĩa bằng cách theo dõi các mối quan hệ trong dữ liệu tuần tự, như các từ trong câu nhờ cơ chế self-attention (tự chú ý). Cơ chế này giúp mô hình phát hiện cách mà dữ liệu, dù xa nhau trong chuỗi, có thể ảnh hưởng và phụ thuộc vào nhau.

Các kiến trúc Transformer là một sự phát triển của các mô hình sequence-to-sequence dựa trên mạng nơ-ron hồi quy (RNN) dùng cho dịch máy ([Machine Translation](https://fpt.ai/vi/bai-viet/machine-translation-la-gi/)). Hiện tại, thuật toán Transformer đã được ứng dụng rộng rãi học máy ([Machine Learning](https://fpt.ai/vi/bai-viet/machine-learning/)) và nhiều lĩnh vực khác của trí tuệ nhân tạo (AI), như thị giác máy tính ([Computer Vision](https://fpt.ai/vi/bai-viet/computer-vision-va-nhung-ung-dung-trong-nganh-tai-chinh-ngan-hang-bao-hiem/)), nhận dạng giọng nói và dự báo chuỗi thời gian.



1. Mô hình transformer

Các thành phần cốt lõi của kiến trúc Transformer:

**Token:**Trong khi các ký tự (chữ cái, số hoặc dấu câu) à đơn vị cơ bản mà con người sử dụng để biểu diễn ngôn ngữ, đơn vị nhỏ nhất của ngôn ngữ mà các mô hình AI sử dụng là token. Mỗi token được gán một số ID và những số ID này là cách các LLM điều hướng cơ sở dữ liệu từ vựng của chúng. Quá trình tokenization này làm giảm đáng kể sức mạnh tính toán cần thiết để xử lý văn bản.

**Mã hóa vị trí (Positional Encoding):**Trong ngôn ngữ của chúng ta, vị trí của từ trong câu rất quan trọng. Ví dụ, câu “Con mèo đuổi con chuột” và “Con chuột đuổi con mèo” có cùng các từ nhưng ý nghĩa hoàn toàn khác nhau. Khi Transformer xử lý một câu, nó chuyển mỗi từ thành một dãy vector và gán vị trí cho chúng. Các từ gần nhau trong câu sẽ có nhãn vị trí gần nhau để giúp mô hình hiểu rằng các từ này có liên quan mật thiết với nhau (từ “ngôi” và “nhà” trong cụm “ngôi nhà”). Mã hóa vị trí (positional encoding) giúp Transformer có thể hiểu cấu trúc câu và mối quan hệ giữa các từ dù nó xử lý tất cả các từ đồng thời.

**Multi-headed Attention:** Để nắm bắt nhiều cách đa diện mà các token có thể liên quan với nhau, các mô hình Transformer thực hiện multi-headed attention. Mô hình sẽ tính toán đồng thời nhiều mối quan hệ giữa các từ, tạo ra nhiều “đầu” attention để xem xét các góc độ khác nhau trong mối quan hệ giữa các từ và xác định chính xác nghĩa của từng từ trong bối cảnh câu. Trong các lớp cuối cùng của mỗi khối attention, đầu ra của các mạch song song được nối lại với nhau trước khi được gửi đến lớp feedforward tiếp theo, mỗi mạch học các trọng số khác nhau để nắm bắt một khía cạnh riêng biệt của ý nghĩa ngữ nghĩa.

Mô hình Transformer hoạt động chủ yếu dựa trên các khối mã hóa/giải mã (encoder/decoder), tương tự như các mạng nơ-ron truyền thống. Điểm đặc biệt là Transformer sử dụng cơ chế self-attention để hiểu và xác định nên chú ý đến phần nào của chuỗi dữ liệu tại bất kỳ thời điểm cụ thể nào. Cách thức hoạt động của mô hình gồm 4 bước như sau:

Bước 1: Mô hình “đọc” các chuỗi dữ liệu thô và chuyển đổi chúng thành các vector nhúng, sau đó sử dụng chúng để tính toán trọng số attention thông qua một loạt phép nhân ma trận. Các vector chính bao gồm:

* Vector truy vấn: Thông tin mà một token cụ thể đang tìm kiếm, được sử dụng để tính toán cách các token khác có thể ảnh hưởng đến ý nghĩa, sự kết hợp hoặc ý nghĩa ngầm của chính token này trong ngữ cảnh.
* Vector khóa: Thông tin mà mỗi token chứa. Sự căn chỉnh giữa truy vấn và khóa được sử dụng để tính toán trọng số attention phản ánh mức độ liên quan của chúng trong ngữ cảnh.
* Vector giá trị: Vector trả lại thông tin từ mỗi vector khóa, được điều chỉnh theo trọng số attention tương ứng. Đóng góp từ các khóa căn chỉnh mạnh với truy vấn được cân nhắc nặng hơn; đóng góp từ các khóa không liên quan đến truy vấn sẽ được cân nhắc gần với không.

Bước 2: Mô hình xác định các điểm tương đồng, tương quan và các phụ thuộc khác giữa mỗi vector bằng cách tính tích vô hướng giữa mỗi vector. Nếu các vector được căn chỉnh tốt, nhân chúng với nhau sẽ cho ra giá trị lớn. Nếu chúng không căn chỉnh, tích vô hướng của chúng sẽ nhỏ hoặc âm.

Bước 3: Các điểm căn chỉnh được chuyển đổi thành trọng số attention thông qua hàm kích hoạt softmax. Hàm này chuẩn hóa tất cả các giá trị về phạm vi từ 0 – 1 sao cho chúng tổng hợp lại bằng 1. Gán trọng số attention 0 giữa “Vector A” và “Vector B” có nghĩa là Vector B nên bị bỏ qua khi đưa ra dự đoán về Vector A. Gán cho Vector B trọng số attention 1 có nghĩa là nó nên nhận 100% sự chú ý của mô hình khi đưa ra quyết định về Vector A.

Bước 4: Các trọng số attention được sử dụng để nhấn mạnh hoặc giảm bớt ảnh hưởng của các phần tử đầu, giúp các mô hình Transformer tập trung vào hoặc bỏ qua thông tin cụ thể tại một thời điểm cụ thể.

## Kỹ thuật tối ưu hóa và điều chuẩn

* + 1. Tối ưu hóa

Tối ưu hóa trong học sâu là quá trình điều chỉnh mô hình để đạt hiệu suất tối ưu, tập trung vào các yếu tố như tốc độ học, kiến trúc mô hình và dung lượng. Điều chỉnh tốc độ học là yếu tố quan trọng: tốc độ học quá cao có thể khiến mô hình không hội tụ, dao động quanh giá trị tối ưu, trong khi tốc độ học quá thấp làm chậm quá trình huấn luyện. Các chiến lược như learning rate decay (giảm dần tốc độ học) hoặc các thuật toán adaptive learning rate như Adam và RMSprop tự động điều chỉnh tốc độ học, giúp mô hình hội tụ nhanh và hiệu quả hơn.

Sử dụng kiến trúc tốt hơn đòi hỏi chọn kiến trúc phù hợp với bài toán, ví dụ Transformer cho chatbot hoặc CNN cho xử lý ảnh, đồng thời tăng độ sâu hoặc thêm lớp để học các đặc trưng phức tạp hơn, như nhận diện ý định trong câu hỏi "áo đỏ giá dưới 500.000 VNĐ".

Tăng dung lượng mô hình bằng cách thêm nơ-ron hoặc lớp có thể cải thiện khả năng học, nhưng cần cẩn thận để tránh quá khớp, thường được giảm thiểu bằng kỹ thuật như dropout. Các phương pháp này, kết hợp với lan truyền ngược và hàm mất mát như cross-entropy, đảm bảo mô hình học sâu, chẳng hạn như GPT hoặc BERT, hoạt động hiệu quả trong các ứng dụng như chatbot, dịch máy, hoặc phân loại ảnh.

* + 1. Điều chuẩn

Điều chuẩn là tập hợp các kỹ thuật quan trọng trong học sâu nhằm giảm quá khớp và nâng cao khả năng tổng quát hóa của mô hình. Dropout hoạt động bằng cách ngẫu nhiên "tắt" một số nơ-ron trong quá trình huấn luyện với một xác suất nhất định, ngăn mô hình phụ thuộc quá nhiều vào một số nơ-ron cụ thể, từ đó tăng tính mạnh mẽ. Ví dụ, trong chatbot sử dụng Transformer, dropout giúp mô hình xử lý các câu hỏi như "áo đỏ giá dưới 500.000 VNĐ" mà không bị lệ thuộc vào một số đặc trưng ngữ cảnh cụ thể

Chuẩn hóa lô (Batch Normalization) chuẩn hóa đầu vào của mỗi lớp bằng cách trừ trung bình và chia cho độ lệch chuẩn của lô dữ liệu, giúp ổn định quá trình huấn luyện, giảm phụ thuộc vào giá trị ban đầu của tham số và cho phép sử dụng tốc độ học cao hơn, cải thiện hiệu suất của các mô hình như BERT hoặc GPT.

Tăng cường dữ liệu (Data Augmentation) biến đổi dữ liệu đầu vào, như xoay, lật, hoặc thay đổi độ sáng trong xử lý ảnh, hoặc thêm nhiễu từ ngữ trong văn bản, để tăng tính đa dạng của tập dữ liệu. Kỹ thuật này đặc biệt hữu ích khi tập dữ liệu nhỏ, giúp mô hình học được các biến thể khác nhau, ví dụ cải thiện khả năng hiểu các biến thể câu hỏi trong chatbot. Các kỹ thuật này, kết hợp với tối ưu hóa như Adam và hàm mất mát như cross-entropy, đảm bảo mô hình học sâu hoạt động hiệu quả và tổng quát hóa tốt trên các tác vụ như dịch máy, phân loại ảnh, hoặc tạo phản hồi tự nhiên.

## Kết luận

Học sâu đã và đang định hình lại cảnh quan của trí tuệ nhân tạo, trở thành động lực thúc đẩy những tiến bộ vượt bậc trong nhiều lĩnh vực, từ y tế, tài chính, đến nghệ thuật và giao thông. Các kỹ thuật cốt lõi như Mạng Nơ-ron Tích chập (CNN), Mạng Nơ-ron Hồi tiếp (RNN), Mạng Đối kháng Tạo (GAN), và Bộ mã hóa Tự động Biến phân (VAE) không chỉ đại diện cho những bước đột phá trong cách máy móc học hỏi từ dữ liệu mà còn mở ra những khả năng ứng dụng thực tiễn đầy tiềm năng. CNN, với khả năng trích xuất đặc trưng không gian từ hình ảnh thông qua các phép tích chập và lớp gộp, đã trở thành nền tảng cho các ứng dụng thị giác máy tính như phân loại hình ảnh như nhận diện bệnh lý từ ảnh y tế, phát hiện đối tượng như hệ thống xe tự lái, hay phân đoạn ảnh (phân tích khối u trong ảnh MRI).

Trong khi đó, RNN và các biến thể như LSTM, với cơ chế lưu trữ trạng thái, tỏ ra vượt trội trong xử lý dữ liệu tuần tự, từ dịch máy như Google Translate, dự báo chuỗi thời gian như dự đoán thời tiết, giá cổ phiếu, đến nhận diện giọng nói như trợ lý ảo như Alexa. Đồng thời, các mô hình sinh như GAN và VAE đã cách mạng hóa việc tạo nội dung, cho phép sinh ra hình ảnh, âm thanh, hoặc văn bản với độ chân thực đáng kinh ngạc. GAN, thông qua cuộc cạnh tranh giữa bộ sinh và bộ phân biệt, tạo ra những sản phẩm sáng tạo như ảnh nghệ thuật, chỉnh sửa khuôn mặt, hay dữ liệu giả lập cho huấn luyện, trong khi VAE, với cách tiếp cận dựa trên không gian tiềm ẩn, cung cấp sự ổn định và đa dạng trong các ứng dụng như khôi phục ảnh hoặc khám phá dữ liệu.

Những kỹ thuật này, khi kết hợp với các phương pháp tối ưu hóa (như điều chỉnh tốc độ học, chuẩn hóa lô) và điều chuẩn (như dropout, tăng cường dữ liệu), không chỉ cải thiện hiệu suất mà còn đảm bảo khả năng tổng quát hóa của mô hình trên các tập dữ liệu đa dạng. Nhìn về tương lai, học sâu đang tiến tới những chân trời mới với các chủ đề nâng cao như tích hợp học sâu với lập trình, nơi các mô hình AI có thể tự động viết mã hoặc tối ưu hóa quy trình phát triển phần mềm, và học suốt đời, cho phép hệ thống AI liên tục học hỏi và thích nghi với dữ liệu mới mà không quên kiến thức cũ . Những hướng phát triển này hứa hẹn sẽ khắc phục các hạn chế hiện tại, như nhu cầu về dữ liệu lớn hay khả năng thích ứng với môi trường thay đổi, đồng thời mở ra tiềm năng xây dựng các hệ thống AI thông minh hơn, linh hoạt hơn, và gần gũi hơn với cách con người học hỏi và sáng tạo. Chính sự kết hợp giữa các kỹ thuật hiện tại và những cải tiến trong tương lai sẽ tiếp tục đưa học sâu trở thành một công cụ không thể thiếu trong việc giải quyết các thách thức phức tạp của thế giới hiện đại.

# CHƯƠNG 3: ỨNG DỤNG HỌC SÂU CHO CHATBOT HỖ TRỢ Y TẾ

* 1. **Tổng quan về chatbot hỗ trợ y tế**

Chatbot hỗ trợ y tế là một ứng dụng trí tuệ nhân tạo (AI) sử dụng các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) để tương tác với người dùng, cung cấp thông tin y tế, tư vấn triệu chứng, hỗ trợ bác sĩ, hoặc giáo dục sức khỏe. Với sự phát triển của học sâu (deep learning), chatbot y tế có khả năng hiểu các câu hỏi phức tạp, phân tích ngữ cảnh, và đưa ra phản hồi chính xác, phù hợp với nhu cầu người dùng. Phần tổng quan này sẽ trình bày vai trò, ứng dụng, thách thức, và tầm quan trọng của học sâu trong việc xây dựng chatbot y tế, cùng với các ví dụ minh họa sử dụng TensorFlow.

* + 1. *Định nghĩa và vai trò*

Định nghĩa: Chatbot hỗ trợ y tế là một hệ thống AI được thiết kế để tương tác với người dùng thông qua văn bản hoặc giọng nói, cung cấp hỗ trợ y tế tức thời dựa trên dữ liệu hội thoại và kiến thức y khoa. Chatbot có thể hoạt động trên các nền tảng như ứng dụng di động, website, hoặc tích hợp vào hệ thống y tế điện tử. Chúng được huấn luyện trên dữ liệu hội thoại y tế (như cặp câu hỏi-phản hồi hoặc tài liệu y khoa) để trả lời các câu hỏi, phân tích triệu chứng, và đưa ra lời khuyên phù hợp. Ví dụ, khi người dùng hỏi: "Tôi bị đau đầu và sốt, phải làm sao?", chatbot sẽ phân tích câu hỏi, nhận diện các triệu chứng (đau đầu, sốt), và trả lời: "Bạn nên nghỉ ngơi, uống nhiều nước, và theo dõi nhiệt độ. Nếu triệu chứng kéo dài hoặc nặng hơn, hãy đến bác sĩ."

Chatbot giúp người dùng nhận diện triệu chứng và đưa ra gợi ý ban đầu, như nghỉ ngơi, uống thuốc giảm đau, hoặc khuyến nghị thăm khám bác sĩ. Điều này đặc biệt hữu ích ở các khu vực thiếu bác sĩ hoặc khi người dùng cần tư vấn nhanh.

Chatbot cung cấp thông tin về phòng ngừa bệnh tật, hướng dẫn sử dụng thuốc, hoặc khuyến khích lối sống lành mạnh, nâng cao nhận thức cộng đồng về chăm sóc sức khỏe. Chatbot cho phép người dùng ở vùng sâu vùng xa hoặc những người không có điều kiện gặp bác sĩ trực tiếp nhận được tư vấn sơ bộ, từ đó cải thiện khả năng tiếp cận dịch vụ y tế.

Chatbot y tế sử dụng các kỹ thuật học sâu để xử lý ngôn ngữ tự nhiên, cho phép chúng hiểu ý định của người dùng, trích xuất thông tin quan trọng, và tạo ra các phản hồi phù hợp. Ví dụ, khi người dùng hỏi: "Tôi bị đau đầu và sốt, phải làm sao?", chatbot cần phân tích câu hỏi, nhận diện triệu chứng (đau đầu, sốt), và trả lời dựa trên dữ liệu đã huấn luyện

* + 1. *Vai trò của học sâu*

Học sâu (deep learning) đóng vai trò cốt lõi trong việc nâng cao khả năng của chatbot hỗ trợ y tế, cho phép chúng hiểu ngôn ngữ tự nhiên, xử lý dữ liệu hội thoại phức tạp, và tạo ra các phản hồi chính xác, tự nhiên trong bối cảnh y tế. Các mô hình học sâu, như mạng nơ-ron hồi tiếp (RNN), LSTM, hoặc các biến thể Transformer, giúp chatbot nhận diện ý định (intent) của người dùng, trích xuất thực thể (entity) như triệu chứng hoặc tên bệnh, và tạo phản hồi phù hợp. Dựa trên đoạn code bạn cung cấp, mục này sẽ phân tích cách học sâu được áp dụng trong việc xây dựng chatbot y tế, với trọng tâm là mô hình LSTM được sử dụng trong code, các lợi ích cụ thể, và tiềm năng mở rộng trong tương lai.

Học sâu mang lại các khả năng quan trọng cho chatbot y tế, bao gồm:

*Hiểu ngôn ngữ tự nhiên*: Học sâu cho phép chatbot phân tích và hiểu các câu hỏi phức tạp hoặc không rõ ràng từ người dùng, bao gồm các câu sử dụng từ lóng, ngôn ngữ địa phương, hoặc cách diễn đạt không chuẩn. Ví dụ, khi người dùng hỏi: "Tui bị ho lâu, mệt nữa, là bị gì?", mô hình học sâu có thể nhận diện ý định (hỏi về triệu chứng) và trích xuất các thực thể (ho, mệt) để đưa ra phản hồi như: "Ho kéo dài và mệt mỏi có thể liên quan đến viêm phế quản hoặc nhiễm trùng. Bạn nên đi khám bác sĩ để được kiểm tra thêm."

*Xử lý chuỗi dữ liệu hội thoại*: Các mô hình như LSTM, được sử dụng trong code của bạn, rất hiệu quả trong việc xử lý chuỗi văn bản, cho phép chatbot ghi nhớ ngữ cảnh trong các câu hỏi dài. Điều này đặc biệt quan trọng trong y tế, nơi người dùng thường mô tả nhiều triệu chứng cùng lúc, ví dụ: "Tôi bị đau đầu, sốt nhẹ, và đau họng đã ba ngày."

*Phân loại ý định và tạo phản hồi*: Học sâu giúp chatbot phân loại ý định của người dùng (như hỏi về triệu chứng, yêu cầu thông tin thuốc, hoặc đặt lịch khám) và tạo phản hồi phù hợp. Code của bạn sử dụng một mô hình LSTM để phân loại ý định, với đầu ra là một phân phối xác suất trên các lớp ý định (softmax), giúp chatbot xác định ý định chính xác nhất.

*Khả năng học hỏi từ dữ liệu lớn*: Các mô hình học sâu có thể được huấn luyện trên khối lượng dữ liệu hội thoại lớn, như các cặp câu hỏi-phản hồi từ diễn đàn y tế hoặc hồ sơ bệnh án, để cải thiện độ chính xác và tính phù hợp của phản hồi.

Học sâu mang lại nhiều lợi ích cụ thể cho chatbot y tế, bao gồm:

***Hiểu ngữ cảnh phức tạp***: Tầng LSTM trong mô hình của bạn có khả năng ghi nhớ thông tin từ các từ trước đó trong câu, giúp chatbot hiểu các câu hỏi dài hoặc phức tạp. Ví dụ, nếu người dùng hỏi: "Tôi bị ho, sốt, và đau ngực, có phải bị viêm phổi không?", LSTM có thể liên kết các triệu chứng để đưa ra phản hồi chính xác hơn.

***Khả năng phân loại ý định chính xác***: Hàm kích hoạt softmax trong tầng Dense cuối cùng của mô hình tạo ra phân phối xác suất trên các ý định, giúp chatbot xác định đúng ý định của người dùng (như hỏi triệu chứng, yêu cầu tư vấn thuốc, hoặc hỏi về bệnh viện).

***Khả năng mở rộng***: Mô hình có thể được huấn luyện trên dữ liệu lớn hơn hoặc điều chỉnh để hỗ trợ các ngôn ngữ khác, như tiếng Việt, bằng cách sử dụng các bộ dữ liệu hội thoại y tế tiếng Việt

***Tính linh hoạt***: Mô hình LSTM có thể được thay thế hoặc kết hợp với các kiến trúc khác, như Transformer, để cải thiện hiệu suất trong các trường hợp cần xử lý ngữ cảnh dài hơn hoặc phức tạp hơn.

* 1. **Mô tả dữ liệu và tiền xử lý**

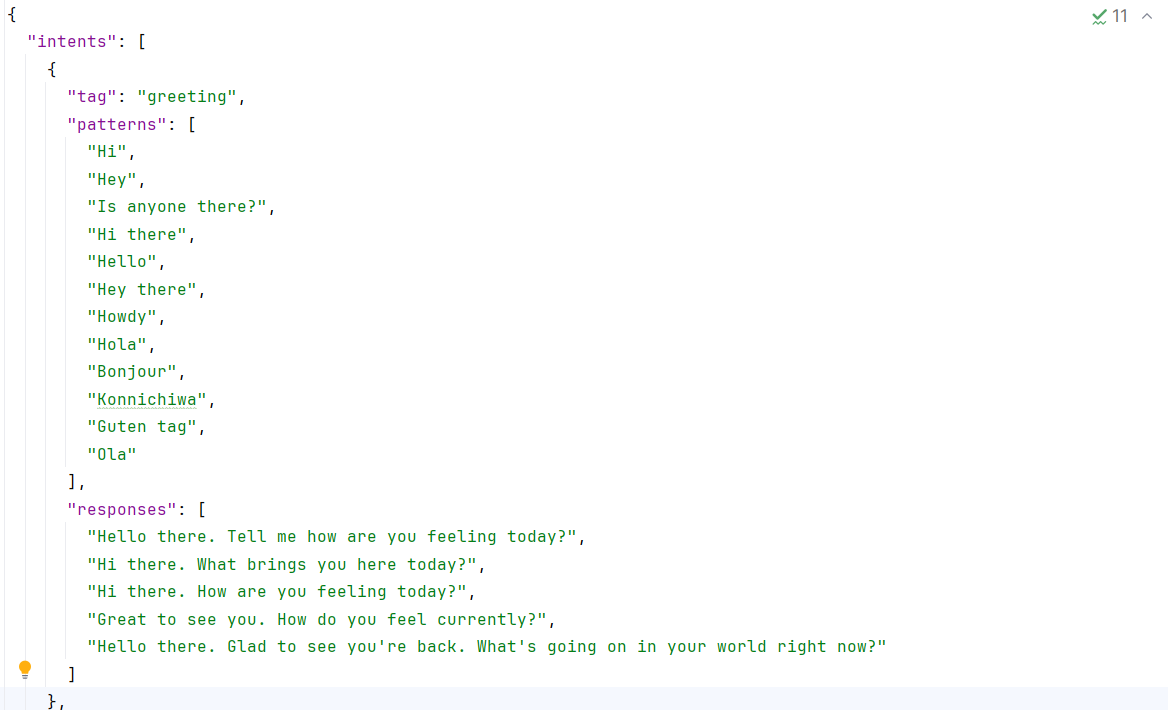
Mục đích của việc chuẩn bị và tiền xử lý dữ liệu đóng vai trò quan trọng trong việc xây dựng một mô hình học sâu hiệu quả cho chatbot. Quá trình này nhằm chuyển đổi dữ liệu thô, chẳng hạn như văn bản từ file , thành định dạng số mà mô hình có thể hiểu và xử lý. Bằng cách mã hóa dữ liệu văn bản thành các biểu diễn số, mô hình học sâu có thể nhận diện và học hỏi các đặc trưng ngữ nghĩa từ câu hỏi của người dùng. Đồng thời, việc đảm bảo các chuỗi đầu vào có độ dài đồng nhất giúp duy trì tính nhất quán trong quá trình huấn luyện, từ đó nâng cao hiệu suất của mô hình.

Bên cạnh đó, tiền xử lý dữ liệu còn bao gồm việc mã hóa nhãn ý định (intents) để mô hình có thể dự đoán chính xác ý định của người dùng dựa trên câu hỏi đầu vào. Điều này cho phép chatbot đưa ra các phản hồi phù hợp và đúng ngữ cảnh. Quá trình này không chỉ giúp tổ chức dữ liệu một cách hiệu quả mà còn giảm thiểu nhiễu, loại bỏ các yếu tố gây nhiễu loạn và tối ưu hóa hiệu quả huấn luyện. Nhờ vậy, mô hình có thể ánh xạ chính xác các câu hỏi đến các ý định tương ứng, tạo nền tảng cho một chatbot thông minh và phản hồi chính xác.

* + 1. *Dữ liệu đầu vào*

**Tập dữ liệu thứ nhất**

Dữ liệu đầu vào được lưu trữ trong file intents2.json, chứa thông tin về các ý định (intents) mà chatbot cần nhận diện. File này được lấy từ dữ liệu giao tiếp Mental Health Conversational Data. Kaggle và thiết kế để cung cấp thông tin về cách người dùng tương tác với chatbot và các phản hồi tương ứng.

****

1. Dữ liệu trong tập dữ liệu thứ nhất

File *intents2.json* là một danh sách các ý định (*intents*), mỗi ý định bao gồm ba thành phần chính:

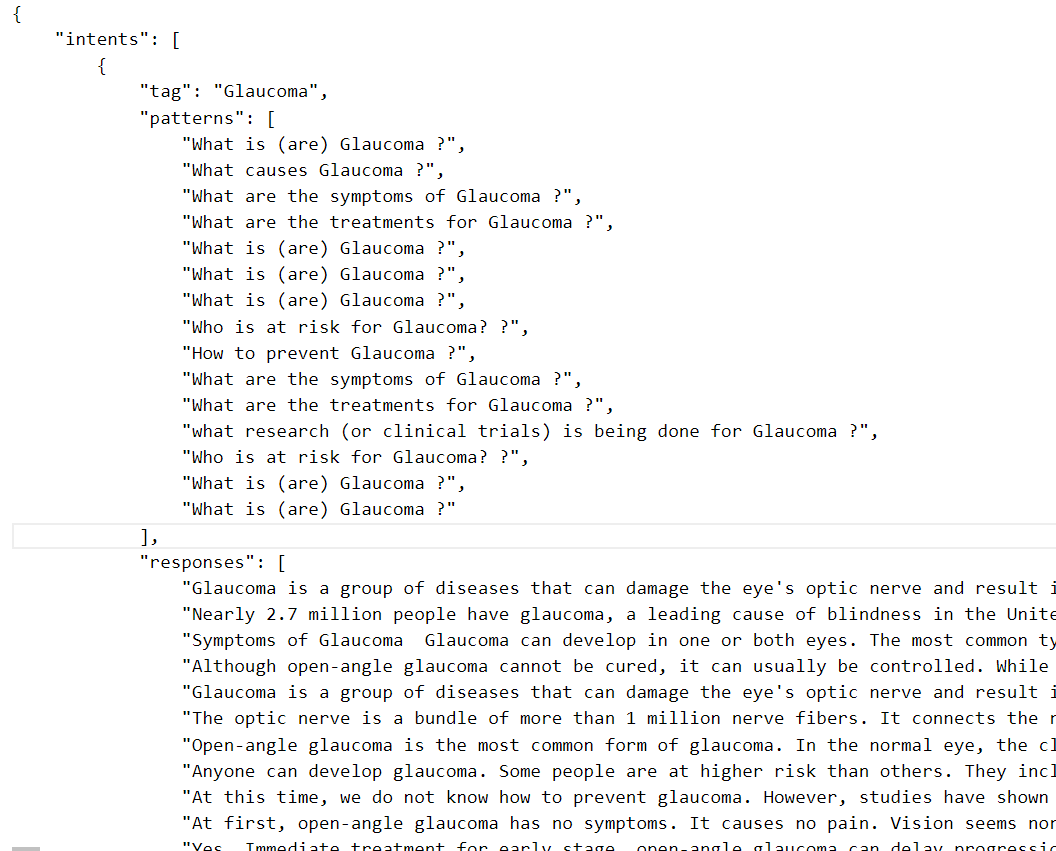
Thành phần đầu tiên là *tag*, tức nhãn ý định, đại diện cho mục đích của câu hỏi hoặc câu lệnh mà người dùng đưa ra. Ví dụ, các nhãn như "greeting" (chào hỏi), "creator" (tìm hiểu về người tạo), hay "goodbye" (tạm biệt) giúp mô hình phân loại ý định dựa trên nội dung đầu vào. Nhãn này đóng vai trò như một chỉ dấu để chatbot xác định đúng ý định của người dùng.

Thành phần thứ hai là *patterns* danh sách các câu hỏi hoặc câu lệnh mà người dùng có thể nhập, liên quan trực tiếp đến ý định tương ứng. Chẳng hạn, với ý định *"greeting"*, các patterns có thể bao gồm *"Hi"*, *"Hello"* hoặc các biến thể khác. Những mẫu câu này cung cấp dữ liệu đầu vào để mô hình học cách nhận diện ý định dựa trên ngôn ngữ tự nhiên của người dùng.

Cuối cùng là *responses* danh sách các câu trả lời mà chatbot có thể chọn ngẫu nhiên để phản hồi khi ý định được xác định. Ví dụ, với ý định *"greeting"*, chatbot có thể trả lời: *"Hello there. How are you feeling today?".* Cấu trúc này không chỉ giúp tổ chức dữ liệu một cách rõ ràng mà còn tạo điều kiện thuận lợi cho quá trình tiền xử lý và huấn luyện, đảm bảo chatbot phản hồi chính xác và phù hợp với ngữ cảnh.

**Tập dữ liệu thứ hai**

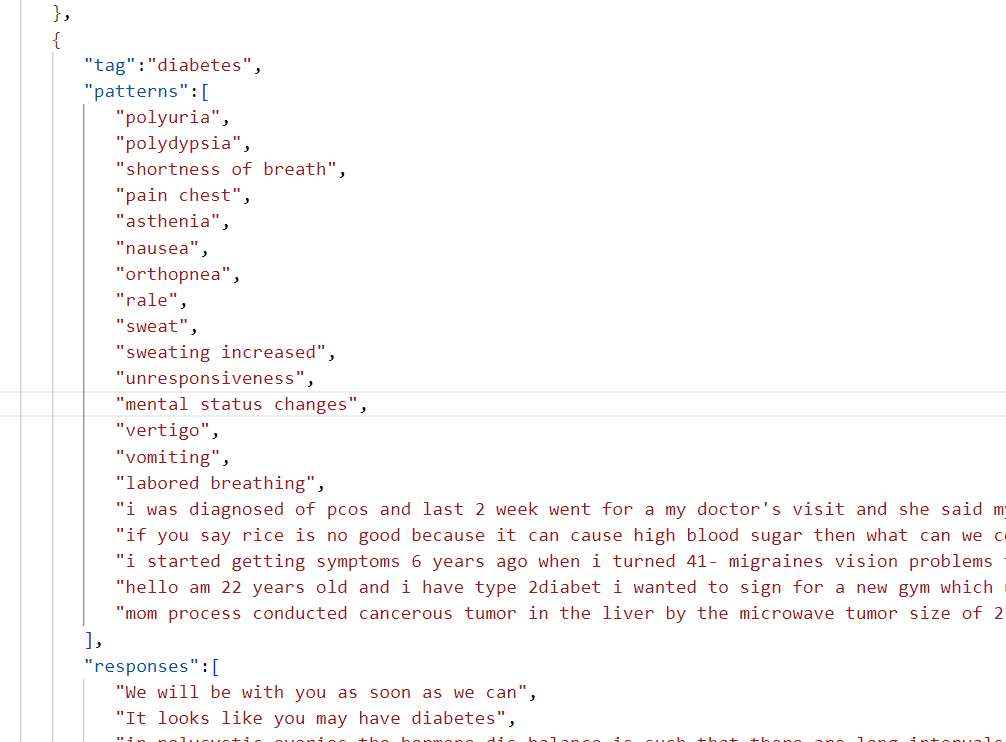
Dữ liệu đầu vào được lưu trữ trong file intents3.json, chứa thông tin về các ý định (intents) mà chatbot cần nhận diện. File này được lấy từ dữ liệu giao tiếp Medical Question Answering Dataset. Kaggle và thiết kế để cung cấp thông tin y tế như triệu chứng bệnh, bệnh này là bệnh gì và được chuyển đổi có cấu trúc giống với tập dữ liệu thứ nhất

****

1. Dữ liệu trong tập dữ liệu thứ hai

**Tập dữ liệu thứ 3**

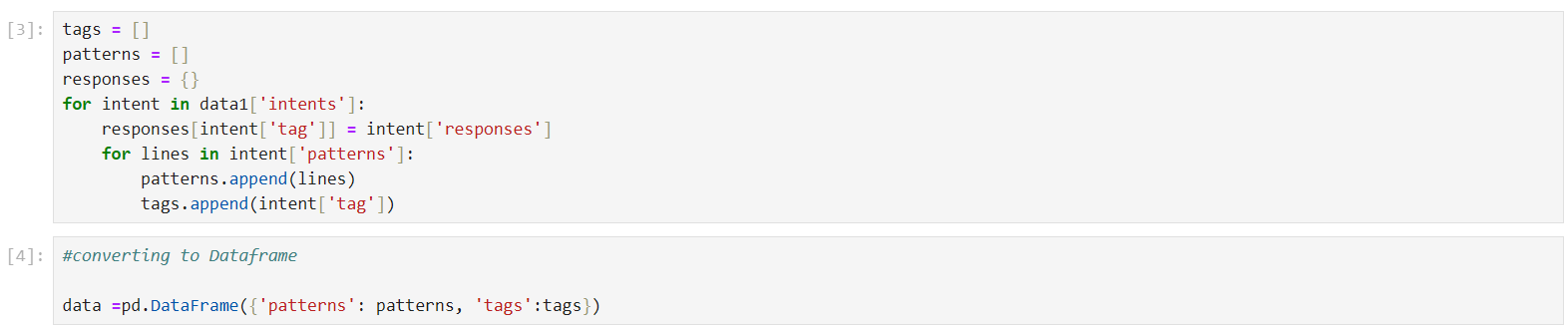
Dữ liệu đầu vào được lưu trữ trong file intents4.json, chứa thông tin về các ý định mà chatbot cần nhận diện **có tên là Health-Care-Chatbot bởi amberkakkar01.** Tập dữ liệu này bao gồm các intents về các bệnh như cảm lạnh, ho, đau đầu, với các mẫu câu hỏi và phản hồi tương ứng.

****

1. Dữ liệu trong tập dữ liệu thứ hai
   * 1. *Tiền xử lý dữ liệu*

Quá trình tiền xử lý dữ liệu bao gồm các bước chuyển đổi dữ liệu thô thành định dạng phù hợp cho mô hình học sâu, các bước tiền xử lý bao gồm:

**Tạo DataFrame**



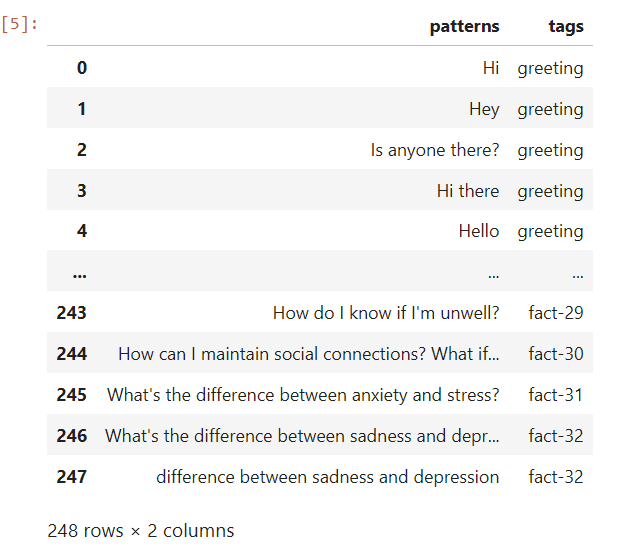
1. Mã tạo DataFrame

Giải thích:

Dữ liệu từ file intents2.json được đọc và xử lý để tạo ra hai danh sách chính phục vụ cho việc huấn luyện mô hình chatbot. Danh sách đầu tiên, *patterns*, bao gồm tất cả các câu hỏi hoặc câu lệnh được trích xuất từ trường "patterns" của mỗi ý định trong file. Danh sách thứ hai, *tags*, chứa các nhãn ý định tương ứng với từng câu trong danh sách patterns. Việc này đảm bảo rằng mỗi câu hỏi hoặc câu lệnh được gắn với một nhãn ý định cụ thể, như "greeting" hoặc "creator".

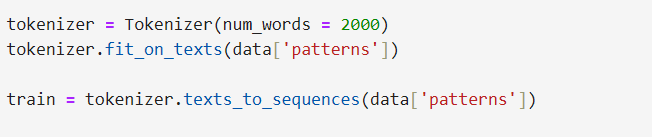
Sau khi được trích xuất, hai danh sách này được chuyển đổi thành một *pandas.DataFrame* với hai cột chính. Cột *patterns* lưu trữ các câu văn bản, chẳng hạn như "Hi" hoặc "who created you ", đại diện cho các đầu vào người dùng. Cột *tags* chứa các nhãn ý định tương ứng, chẳng hạn như "greeting" cho câu "Hi" hoặc "creator" cho câu "who created you". Cấu trúc DataFrame này giúp tổ chức dữ liệu một cách rõ ràng và thuận tiện, tạo điều kiện cho các bước tiền xử lý tiếp theo, như mã hóa văn bản và chuẩn hóa độ dài chuỗi, để chuẩn bị dữ liệu cho mô hình học sâu.

Kết quả là DataFrame này chứa 248 hàng:



1. Kết quả tạo DataFrame

**Tokenization**



1. Mã tạo Tokenization

Giải thích:

Quá trình tiền xử lý dữ liệu văn bản từ file *intents2.json* sử dụng *Tokenizer* trong thư viện *tensorflow.keras.preprocessing.text.Tokenizer* để mã hóa các câu văn bản thành các chuỗi số nguyên, phù hợp cho mô hình học sâu. Tham số *num\_words=2000*được thiết lập để giới hạn từ điển ở 2000 từ phổ biến nhất

Quá trình mã hóa bắt đầu với phương thức *fit\_on\_texts*, được áp dụng trên danh sách các câu trong *patterns* để xây dựng từ điển. Mỗi từ duy nhất trong các câu được gán một số nguyên, ví dụ: *"hi"* có thể được gán số 1, *"hello"* gán số 2. Sau đó, phương thức *texts\_to\_sequences* chuyển đổi mỗi câu thành một chuỗi số dựa trên từ điển này. Chẳng hạn, câu "Hi there" có thể được mã hóa thành [1, 3] (nếu *"hi"* là 1 và *"there"* là 3), trong khi câu *"who created you"* trở thành [4, 5, 6].

Mục đích của quá trình này là chuyển đổi dữ liệu văn bản thành dạng số mà mô hình học sâu có thể xử lý, đồng thời loại bỏ các từ hiếm (nếu vượt quá giới hạn *num\_words*) để giảm kích thước từ điển và tăng hiệu quả tính toán. Kết quả là mỗi câu trong *patterns* được biểu diễn dưới dạng một danh sách các số nguyên, và từ điển *(tokenizer.word\_index)* lưu trữ ánh xạ từ các từ sang số, ví dụ: {*'hi'*: 1, *'there'*: 3, ...}. Cấu trúc này tạo nền tảng cho các bước tiếp theo trong huấn luyện mô hình chatbot.

**Padding**



1. Padding

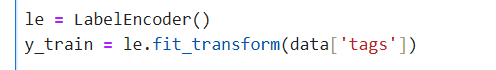
Giải thích:

Quá trình *pad\_sequences* trong thư viện *tensorflow .keras. preprocessing. sequence* được sử dụng để chuẩn hóa độ dài các chuỗi số nguyên, đảm bảo tất cả các chuỗi có cùng kích thước, phù hợp cho việc huấn luyện mô hình học sâu. Đầu tiên, quá trình xác định độ dài lớn nhất của chuỗi trong tập dữ liệu train, tức là số từ tối đa trong một câu. Các chuỗi ngắn hơn sẽ được đệm (*padding*) bằng số 0 (giá trị mặc định) để đạt được độ dài này, đảm bảo tính đồng nhất.

Ví dụ, nếu câu dài nhất trong tập dữ liệu có 5 từ, được mã hóa thành *[1, 2, 3, 4, 5]*, các chuỗi ngắn hơn sẽ được điều chỉnh như sau: câu *"Hi there"* *([1, 3])* trở thành *[1, 3, 0, 0, 0]*, và câu *"who created you"* *([4, 5, 6])* trở thành *[4, 5, 6, 0, 0]*. Quá trình này đảm bảo rằng mọi chuỗi đầu vào có cùng độ dài, cho phép mô hình xử lý dữ liệu một cách đồng nhất.

Mục đích của *pad\_sequences* là tạo ra một ma trận *x\_train* với kích thước (số mẫu, *input\_shape*), trong đó *input\_shape* là độ dài chuỗi sau khi được đệm. Ma trận này chứa các chuỗi số đã được chuẩn hóa, sẵn sàng để làm đầu vào cho mô hình học sâu. Kết quả là một tập dữ liệu đồng nhất, giúp tối ưu hóa quá trình huấn luyện và đảm bảo mô hình chatbot có thể xử lý các câu hỏi với độ dài khác nhau một cách hiệu quả.

**Label Encoding**



1. Mã tạo Label Encode

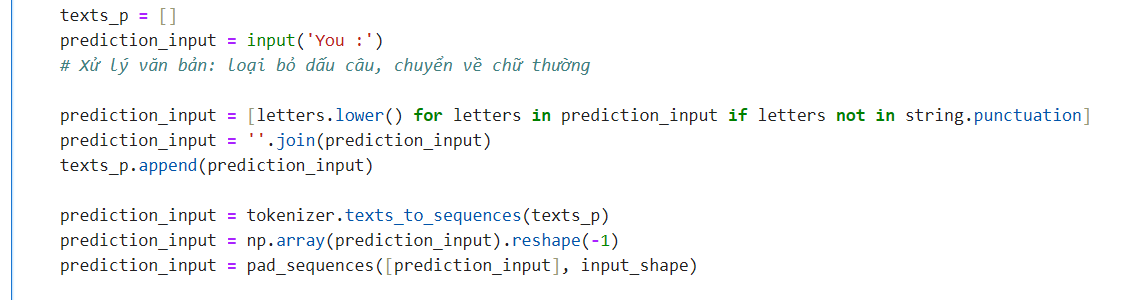
Giải thích:

Quá trình mã hóa nhãn ý định sử dụng ***LabelEncoder*** từ thư viện sklearn.preprocessing nhằm chuyển đổi các nhãn ***tags*** dạng văn bản thành các số nguyên, tạo điều kiện cho bài toán phân loại trong mô hình học sâu. Cụ thể, mỗi nhãn ý định duy nhất được gán một giá trị số nguyên, ví dụ: "greeting" được mã hóa thành 0, "creator" thành 1. Danh sách ***tags*** ban đầu sẽ được chuyển thành một mảng số nguyên, gọi là ***y\_train***.

Ví dụ, nếu danh sách ***tags*** có dạng ***["greeting", "greeting", "creator", "creator", ...]***, sau khi mã hóa, nó trở thành mảng ***[0, 0, 1, 1, ...]***. Mục đích của quá trình này là biến các nhãn văn bản thành dạng số, phù hợp với yêu cầu của hàm mất mát ***sparse\_categorical\_crossentropy***, vốn đòi hỏi nhãn phải ở dạng số nguyên để thực hiện phân loại chính xác.

Kết quả là mảng ***y\_train*** chứa các nhãn số tương ứng với từng mẫu câu trong ma trận ***x\_train***. Điều này đảm bảo rằng dữ liệu nhãn được chuẩn bị đầy đủ và đồng bộ với dữ liệu đầu vào, sẵn sàng cho việc huấn luyện mô hình chatbot để dự đoán ý định của người dùng một cách hiệu quả.

**Xử lý văn bản trong chatbot**

****

1. Đoạn mã xử lý văn bản đầu vào

Giải thích:

Quá trình chuẩn hóa đầu vào của người dùng là một bước quan trọng để đảm bảo dữ liệu mới được xử lý tương tự như dữ liệu huấn luyện, giúp mô hình dự đoán ý định chính xác. Đầu tiên, câu nhập từ người dùng được chuyển thành **chữ thường**, ví dụ: *"Hello"* được chuẩn hóa thành *"hello"*. Tiếp theo, các **dấu câu** được loại bỏ, chẳng hạn *"Hi!"* trở thành *"hi"*, và các ký tự được kết hợp thành một chuỗi hoàn chỉnh. Bước này giúp loại bỏ nhiễu và đảm bảo tính nhất quán trong dữ liệu văn bản.

Sau khi chuẩn hóa, câu văn bản được mã hóa thành chuỗi số nguyên bằng phương thức ***texts\_to\_sequences*** của ***Tokenizer***, sử dụng từ điển đã xây dựng từ dữ liệu huấn luyện. Ví dụ, câu *"hi there"* có thể được chuyển thành ***[1,3]***. Tiếp theo, ***pad\_sequences*** được áp dụng để đệm chuỗi số này, đảm bảo độ dài bằng ***input\_shape*** (độ dài chuỗi tối đa trong dữ liệu huấn luyện), ví dụ: *[1, 3]* trở thành *[1, 3, 0, 0, 0]* nếu ***input\_shape*** là 5.

Mục đích của các bước này là đảm bảo đầu vào của người dùng được xử lý giống hệt dữ liệu huấn luyện, từ định dạng văn bản đến cấu trúc số. Điều này cho phép mô hình học sâu nhận diện và dự đoán ý định của câu hỏi một cách chính xác, duy trì hiệu suất ổn định khi xử lý các câu hỏi thực tế từ người dùng.

* + 1. *Kết quả tiền xử lý*

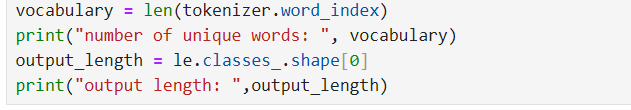
Sau khi thực hiện các bước tiền xử lý, dữ liệu được chuyển đổi thành dạng sẵn sàng cho huấn luyện mô hình. Các kết quả cụ thể bao gồm:

1. Kích thước đầu vào



Kích thước đầu vào (*input\_shape*) là một tham số quan trọng trong quá trình tiền xử lý dữ liệu cho mô hình học sâu của chatbot. Nó được xác định dựa trên độ dài của chuỗi sau khi áp dụng *pad\_sequences*, tức là số từ tối đa trong một câu thuộc tập dữ liệu huấn luyện. Ví dụ, nếu câu dài nhất trong danh sách patterns có 18 từ, thì *input\_shape* sẽ được thiết lập là 18. Điều này đảm bảo rằng tất cả các chuỗi đầu vào, bao gồm cả dữ liệu huấn luyện và câu hỏi từ người dùng, đều có cùng độ dài sau khi được đệm bằng số 0, tạo ra một ma trận đồng nhất để mô hình xử lý hiệu quả.

1. Số lượng từ duy nhất (vocabulary) và số lượng nhãn đầu ra (output\_length)



**Vocabulary** được xác định dựa trên ***tokenizer.word\_index***, một từ điển ánh xạ mỗi từ duy nhất trong dữ liệu ***patterns*** sang một số nguyên. Số lượng từ duy nhất, tức là kích thước của từ điển này, được gọi là ***vocabulary***. Ví dụ, nếu dữ liệu chứa 100 từ khác nhau sau khi sử dụng ***Tokenizer***, thì ***vocabulary = 100***. Con số này phản ánh phạm vi từ vựng mà mô hình cần xử lý, thường nhỏ hơn giới hạn ***num\_words***

**Output\_length** là số lượng lớp ý định, tức là số nhãn duy nhất trong danh sách ***tags***. Nó đại diện cho số lượng ý định mà mô hình cần dự đoán. Ví dụ, nếu có 5 nhãn ý định khác nhau như "greeting", "creator", "goodbye", v.v., thì ***output\_length = 5***. Thông số này xác định số lớp đầu ra trong bài toán phân loại, phù hợp với hàm mất mát như ***sparse\_categorical\_crossentropy***.

Cả hai thông số này đều quan trọng để định hình kiến trúc mô hình, đảm bảo dữ liệu đầu vào và đầu ra được xử lý chính xác và hiệu quả. Kết quả chạy:



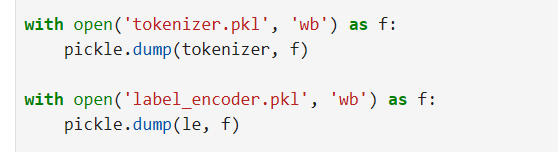
1. Dữ liệu cuối cùng

**Trong quá trình tiền xử lý dữ liệu cho mô hình chatbot, các thành phần đầu vào và nhãn được tổ chức một cách có hệ thống để phù hợp với việc huấn luyện mô hình học sâu. Đầu vào (*x\_train*) là một ma trận có kích thước (số mẫu, *input\_shape*), trong đó số mẫu là số câu trong danh sách *patterns*, và *input\_shape* là độ dài chuỗi tối đa sau khi được đệm bằng pad\_sequences. Mỗi hàng của ma trận này chứa một chuỗi số nguyên đại diện cho một câu đã được mã hóa và đệm, ví dụ: *[1, 3, 0, 0, 0]* cho câu "*Hi there*" nếu *input\_shape* = 5.**

**Nhãn (*y\_train*) là một mảng có kích thước (*số mẫu*,), chứa các nhãn số nguyên tương ứng với từng câu trong *x\_train*. Các nhãn này được tạo ra bằng cách sử dụng *LabelEncoder*, ánh xạ các nhãn ý định dạng văn bản (như "*greeting*", "*creator*") thành các số nguyên (ví dụ: 0, 1, ...). Ví dụ, nếu có 28 mẫu câu, *y\_train* có thể là *[0, 0, 1, 1, ...]*, trong đó mỗi số đại diện cho một ý định.**

**Từ điển và bộ mã hóa nhãn đóng vai trò quan trọng trong việc lưu trữ ánh xạ. Từ điển (*tokenizer.word\_index*) lưu trữ ánh xạ từ các từ sang số nguyên, ví dụ: {*'hi'*: 1, *'there'*: 3, ...}, giúp mã hóa và giải mã văn bản. Bộ mã hóa nhãn (*le.classes*\_) lưu trữ ánh xạ từ nhãn ý định sang số, ví dụ: {*'greeting'*: 0, *'creator'*: 1, ...}, đảm bảo mô hình có thể liên kết các dự đoán số với ý định cụ thể. Các cấu trúc này giúp duy trì tính nhất quán giữa dữ liệu huấn luyện và dữ liệu thực tế từ người dùng, hỗ trợ quá trình dự đoán chính xác.**

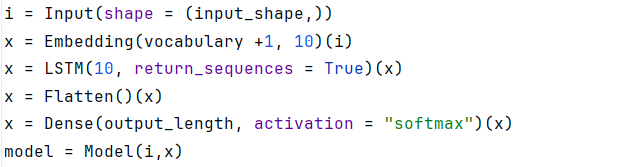
1. Lưu trữ tiền xử lý



Quá trình lưu trữ dữ liệu tiền xử lý được thực hiện bằng cách sử dụng thư viện pickle để lưu *Tokenizer* và *LabelEncoder*, đảm bảo tính nhất quán khi sử dụng lại trong các giai đoạn dự đoán hoặc huấn luyện tiếp theo. Cụ thể, đoạn mã sử dụng *with open('tokenizer.pkl', 'wb') as f: và pickle.dump(tokenizer, f)* để lưu đối tượng *tokenizer* vào file *"tokenizer.pkl"*. Tương tự, *with open('label\_encoder.pkl', 'wb') as f: và pickle.dump(le, f)* được dùng để lưu đối tượng *LabelEncoder (le)* vào file *"label\_encoder.pkl"*. Điều này giúp bảo toàn từ điển ánh xạ từ → số và nhãn → số đã được xây dựng từ dữ liệu huấn luyện.

Giải thích: Việc lưu *Tokenizer* và *LabelEncoder* bằng pickle cho phép sử dụng lại các đối tượng này khi dự đoán, đảm bảo rằng quá trình mã hóa văn bản và nhãn của câu hỏi người dùng phù hợp với dữ liệu huấn luyện. Điều này quan trọng để duy trì sự nhất quán và đảm bảo độ chính xác trong các lần huấn luyện hoặc dự đoán tiếp theo, đặc biệt khi dữ liệu mới được đưa vào hệ thống chatbot.

* 1. **Xây dựng và huấn luyện mô hình học sâu**
     1. *Cấu trúc mô hình*

****



1. Cấu trúc mô hình

Mô hình được xây dựng với một kiến trúc tuần tự, bao gồm các lớp được thiết kế để xử lý dữ liệu văn bản đã được tiền xử lý. Dữ liệu đầu vào (*x\_train*) là một ma trận kích thước (28, input\_shape), trong đó 28 là số mẫu và input\_shape là độ dài chuỗi tối đa sau khi padding (ví dụ: 18 nếu câu dài nhất có 18 từ). Dữ liệu nhãn (*y\_train*) là mảng kích thước (28,), chứa các nhãn số nguyên từ 0 đến output\_length - 1 (ví dụ: 0 cho "greeting", 1 cho "creator", v.v.).

**Input Layer: Nhận chuỗi số có độ dài input\_shape**

Lớp đầu vào được định nghĩa bằng *Input(shape=(input\_shape,))*, với *input\_shape* được xác định từ x*\_train.shape[1]*. Đây là độ dài chuỗi tối đa sau khi sử dụng *pad\_sequences*, đảm bảo tất cả chuỗi đầu vào có cùng kích thước.

*input\_shape* được lấy trực tiếp từ dữ liệu để phản ánh chính xác độ dài tối đa của các câu trong tập huấn luyện (ví dụ: 18). Điều này đảm bảo mô hình có thể xử lý mọi chuỗi đầu vào một cách đồng nhất mà không mất thông tin.

Nếu chọn một giá trị cố định lớn hơn (ví dụ: 50), các chuỗi ngắn sẽ được đệm quá nhiều số 0, gây lãng phí tài nguyên tính toán. Ngược lại, nếu chọn nhỏ hơn (ví dụ: 10), các chuỗi dài hơn sẽ bị cắt bớt, dẫn đến mất thông tin quan trọng. Lấy giá trị thực tế từ dữ liệu là cách tối ưu nhất.

**Embedding Layer: Chuyển đổi token thành vector 10 chiều**

Lớp *Embedding (vocabulary + 1, 10)* ánh xạ các token số ( từ 0 đến vocabulary ) thành vector 10 chiều. *vocabulary* là số từ duy nhất trong từ điển (*len(tokenizer.word\_index)*), ví dụ: 50 nếu có 50 từ duy nhất, và cộng thêm 1 để tính giá trị *padding (0)*.

Với tập dữ liệu nhỏ kích thước vector 10 chiều là đủ để biểu diễn ngữ nghĩa của từ mà không làm tăng quá nhiều tham số mô hình. Tổng số tham số của lớp này là *(vocabulary + 1) \* 10* (ví dụ: *51 \* 10 = 510* nếu *vocabulary = 50*), giữ cho mô hình nhẹ và tránh nguy cơ quá khớp. Trong các bài toán lớn hơn (như xử lý ngôn ngữ tự nhiên với hàng triệu từ), kích thước embedding thường dao động từ 50 đến 300, nhưng với dữ liệu nhỏ, 10 chiều là lựa chọn hợp lý để giảm độ phức tạp tính toán.

Nếu chọn kích thước lớn hơn (ví dụ: 50 hoặc 100), số tham số sẽ tăng đáng kể (ví dụ: *51 \* 50 = 2550* hoặc *51 \* 100 = 5100*), làm tăng nguy cơ quá khớp và đòi hỏi nhiều dữ liệu hơn để huấn luyện hiệu quả. Nếu chọn nhỏ hơn (ví dụ: 5), khả năng biểu diễn ngữ nghĩa của từ sẽ bị hạn chế, làm giảm hiệu suất của mô hình trong việc hiểu ngữ cảnh.

**LSTM Layer: Xử lý dữ liệu tuần tự với 10 đơn vị**

Lớp *LSTM(10, return\_sequences=True)* sử dụng 10 đơn vị (units) để xử lý dữ liệu tuần tự. Tham số *return\_sequences=True* đảm bảo rằng đầu ra là một chuỗi *(kích thước: số mẫu, input\_shape, 10)*, thay vì chỉ lấy trạng thái cuối cùng.

Số đơn vị 10 được chọn để phù hợp với tập dữ liệu nhỏ và độ phức tạp thấp. Với các câu trung bình 3-5 từ và tối đa 18 từ, 10 đơn vị là đủ để học các mẫu ngữ nghĩa và ngữ cảnh dài hạn mà không làm mô hình quá phức tạp. Số tham số của lớp LSTM được tính gần đúng là *4 \* (input\_dim + units + 1) \* units*, trong đó *input\_dim = 10 (từ Embedding)* và *units = 10*, tức là khoảng *4 \* (10 + 10 + 1) \* 10 = 840* tham số, một con số hợp lý cho dữ liệu nhỏ.

Chọn return\_sequences=True vì đầu ra của LSTM sẽ được đưa vào lớp Flatten, việc giữ toàn bộ chuỗi đầu ra cho phép khai thác thông tin từ tất cả các bước thời gian (ví dụ: 18 bước nếu input\_shape = 18). Điều này quan trọng trong bài toán phân loại ý định, nơi toàn bộ ngữ cảnh của câu (từ đầu đến cuối) cần được xem xét để dự đoán chính xác.

Nếu tăng số đơn vị (ví dụ: 50), số tham số sẽ tăng lên khoảng *4 \* (10 + 50 + 1) \* 50 = 12,200*, làm mô hình phức tạp hơn, dễ dẫn đến quá khớp trên dữ liệu nhỏ. Nếu giảm số đơn vị (ví dụ: 5), số tham số giảm xuống *4 \* (10 + 5 + 1) \* 5 = 320*, nhưng mô hình có thể không đủ khả năng học các mẫu ngữ cảnh phức tạp, dẫn đến dưới khớp. Tương tự, nếu không dùng *return\_sequences=True*, đầu ra chỉ là trạng thái cuối (kích thước: số mẫu, 10), làm mất thông tin từ các bước thời gian trước, giảm khả năng hiểu ngữ cảnh toàn diện.

**Flatten Layer: Chuyển đổi đầu ra của LSTM thành vector 1 chiều**

Lớp *Flatten()* chuyển đổi đầu ra của *LSTM* (kích thước: số mẫu, input\_shape, 10) thành vector 1 chiều (kích thước: *số mẫu, input\_shape \* 10*). Ví dụ: nếu *input\_shape = 18*, đầu ra của *Flatten* sẽ là (số mẫu, 180).

*Flatten* là cách đơn giản và hiệu quả để kết nối đầu ra của *LSTM* với lớp *Dense*, vốn yêu cầu đầu vào 1 chiều. Phương pháp này giữ lại toàn bộ thông tin từ chuỗi, đảm bảo không mất dữ liệu trước khi dự đoán

**Dense Layer: Lớp đầu ra với số nơ-ron bằng output\_length, hàm kích hoạt softmax**

Lớp *Dense(output\_length, activation="softmax")* tạo đầu ra với số nơ-ron bằng số nhãn ý định (*output\_length*, ví dụ: 5 nếu có 5 ý định như *"greeting"*, *"creator"*, *"goodbye"*, v.v.). Hàm kích hoạt *softmax* chuyển đổi đầu ra thành xác suất cho từng lớp, với tổng xác suất bằng 1.

Số nơ-ron phải bằng số lớp ý định để mô hình có thể dự đoán đúng nhãn. output\_length được tính từ *le.classes\_.shape[0]*, đảm bảo khớp với số nhãn thực tế trong dữ liệu (ví dụ: 5). Số tham số của lớp này là *(input\_shape \* 10 + 1) \* output\_length*, ví dụ: nếu *input\_shape = 18* và *output\_length = 5*, thì *(18 \* 10 + 1) \* 5 = 905* tham số.

*Softmax* là hàm kích hoạt tiêu chuẩn cho bài toán phân loại đa lớp, vì nó cung cấp xác suất cho từng lớp, cho phép mô hình chọn nhãn có xác suất cao nhất (ví dụ: *[0.1, 0.6, 0.2, 0.05, 0.05]* → chọn nhãn 1).

Số nơ-ron không thể khác *output\_length*, nếu không mô hình sẽ không thể dự đoán đúng số lớp (ví dụ: nếu đặt 10 nơ-ron cho 5 lớp, sẽ không có ý nghĩa). Hàm kích hoạt sigmoid (dành cho phân loại nhị phân) hoặc linear (dành cho hồi quy) không phù hợp, vì bài toán yêu cầu phân loại đa lớp với xác suất chuẩn hóa.

* + 1. *Biên dịch mô hình*

**

**Hàm mất mát**

Hàm mất mát sparse\_categorical\_crossentropy được sử dụng để đo lường sai số giữa nhãn thực tế (y\_train, ví dụ: [0, 0, 1, 1, ...]) và nhãn dự đoán (xác suất từ softmax). Hàm này phù hợp cho bài toán phân loại đa lớp khi nhãn được mã hóa dưới dạng số nguyên, như trong trường hợp sử dụng LabelEncoder. sparse\_categorical\_crossentropy không yêu cầu nhãn ở dạng one-hot encoding, tiết kiệm bộ nhớ so với categorical\_crossentropy. Ví dụ: với 5 lớp, one-hot encoding sẽ tạo mảng kích thước (28, 5), trong khi sparse\_categorical\_crossentropy chỉ cần mảng (28,) chứa các số từ 0 đến 4.

Nếu nhãn đã được mã hóa one-hot (ví dụ: [1, 0, 0, 0, 0] cho "greeting"), categorical\_crossentropy sẽ phù hợp hơn, nhưng ở đây nhãn là số nguyên, nên sparse\_categorical\_crossentropy là lựa chọn tối ưu. Các hàm như mse (mean squared error, dành cho hồi quy) không phù hợp với bài toán phân loại, vì chúng không tối ưu hóa xác suất dự đoán.

**Bộ tối ưu: adam**

Bộ tối ưu adam (Adaptive Moment Estimation) được sử dụng để điều chỉnh trọng số mô hình trong quá trình huấn luyện. Adam kết hợp ưu điểm của momentum (theo dõi gradient trung bình) và RMSProp (theo dõi bình phương gradient), tự động điều chỉnh tốc độ học dựa trên các giá trị gradient trước đó. Điều này giúp adam hội tụ nhanh và ổn định, đặc biệt với dữ liệu nhỏ và mô hình đơn giản như trong bài toán này. Tham số mặc định của adam (learning rate = 0.001) thường hoạt động tốt mà không cần tinh chỉnh.

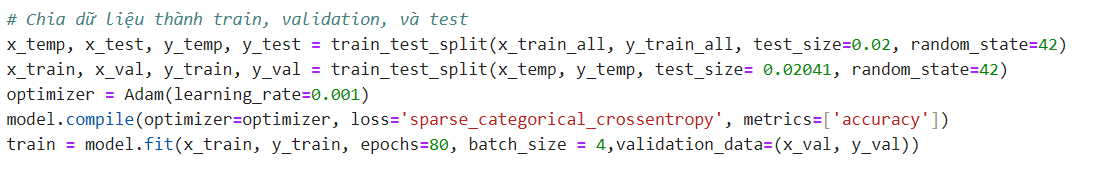
Bộ tối ưu như SGD (Stochastic Gradient Descent) yêu cầu điều chỉnh thủ công tốc độ học và có thể hội tụ chậm hơn, đặc biệt trên dữ liệu nhỏ. Các bộ tối ưu khác như RMSProp hoặc Adagrad cũng có thể được sử dụng, nhưng adam là lựa chọn phổ biến vì sự cân bằng giữa tốc độ và ổn định, không cần cấu hình phức tạp.

**Độ chính xác (accuracy)**

Độ chính xác (*metrics=['accuracy']*) được chọn để đánh giá hiệu suất mô hình, đo tỷ lệ dự đoán đúng trên tổng số mẫu, accuracy là thước đo trực quan và phù hợp cho bài toán phân loại đa lớp, đặc biệt khi các lớp có phân bố tương đối cân bằng. Với dữ liệu nhỏ, accuracy cung cấp cái nhìn tổng quát về hiệu suất mà không cần phân tích chi tiết hơn.

* + 1. *Huấn luyện*

**Tập dữ liệu thứ nhất**

**

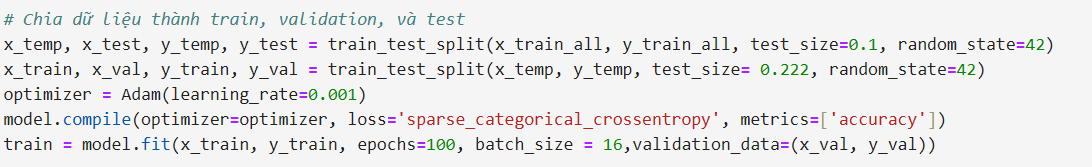
1. Đoạn mã chia dữ liệu thành tập train, test, validation của tập thứ nhất

Quá trình huấn luyện mô hình học sâu cho chatbot được thực hiện thông qua hàm *model.fit* trong thư viện TensorFlow/Keras, với mục tiêu tối ưu hóa trọng số mô hình để nhận diện ý định người dùng từ dữ liệu văn bản. Cụ thể, mô hình được huấn luyện trên tập dữ liệu *x\_train* (các chuỗi số đã mã hóa và padding) và nhãn y\_train (các ý định đã mã hóa), với số vòng lặp là 80 epochs. Tham số *validation\_data=(x\_val, y\_val)* được sử dụng để đánh giá hiệu suất mô hình trên tập validation sau mỗi epoch, giúp theo dõi quá trình học và phát hiện hiện tượng quá khớp. Số lượng epochs lớn cho phép mô hình có đủ thời gian để học các mẫu ngữ nghĩa phức tạp, nhưng đòi hỏi sự theo dõi chặt chẽ thông qua tập validation để đảm bảo mô hình không học quá mức, dẫn đến giảm hiệu suất trên dữ liệu mới.

Việc phân chia dữ liệu thành các tập train, validation, và test được thực hiện qua hai bước sử dụng hàm train\_test\_split từ thư viện sklearn.model\_selection. Ở bước đầu tiên, dữ liệu tổng thể (x\_train\_all, y\_train\_all) được chia với test\_size=0.02, nghĩa là 2% dữ liệu được dành cho tập test, còn 98% còn lại được dùng để tiếp tục chia thành tập train và validation. Ở bước thứ hai, phần 98% dữ liệu còn lại được chia tiếp với test\_size=0.02041, tương đương 2% tổng dữ liệu cho tập validation, và 96% tổng dữ liệu cho tập train. Kết quả, tỷ lệ cuối cùng đạt được là 96% train, 2% validation, và 2% test. Tham số random\_state=42 được sử dụng trong cả hai bước để đảm bảo tính tái hiện, nghĩa là dữ liệu được chia giống hệt mỗi lần chạy, hỗ trợ việc thử nghiệm và so sánh kết quả.

Tỷ lệ phân chia 96/2/2 được lựa chọn vì đây là một cách tiếp cận hiệu quả cho dữ liệu nhỏ, đảm bảo cân bằng giữa việc học và đánh giá. Với 96% dữ liệu dành cho tập train, mô hình có đủ dữ liệu để học các đặc trưng và mẫu ngữ nghĩa, giảm nguy cơ dưới khớp (underfitting), đặc biệt khi số lượng mẫu hạn chế. Tập validation chiếm 2% tổng dữ liệu, đủ lớn để cung cấp đánh giá đáng tin cậy về hiệu suất mô hình trong quá trình huấn luyện, giúp điều chỉnh siêu tham số như số epochs hoặc phát hiện sớm quá khớp. Tập test, chiếm 2%, được giữ riêng biệt để đánh giá độc lập sau huấn luyện, đảm bảo rằng mô hình có khả năng tổng quát hóa trên dữ liệu chưa thấy, phản ánh chính xác hiệu quả thực tế khi triển khai chatbot.

**Tập dữ liệu thứ hai**

****

1. Đoạn mã chia dữ liệu thành tập train, test, validation của tập thứ 2

Đoạn code thực hiện việc chia tập dữ liệu hội thoại y tế thành các tập train, validation, và test một cách hợp lý để huấn luyện mô hình học sâu. Cụ thể, dữ liệu ban đầu (x\_train\_all, y\_train\_all) được chia thành tập test (x\_test, y\_test) với tỷ lệ 10% (test\_size=0.1) bằng hàm train\_test\_split từ thư viện sklearn, sử dụng random\_state=42 để đảm bảo tính tái lập. Phần dữ liệu còn lại (x\_temp, y\_temp) tiếp tục được chia thành tập train (x\_train, y\_train) và tập validation (x\_val, y\_val) với tỷ lệ validation là 22.2% (test\_size=0.222) so với phần dữ liệu còn lại, tương đương với khoảng 20% tổng dữ liệu ban đầu. Như vậy, tỷ lệ cuối cùng giữa tập train, validation, và test là khoảng 70:20:10, một tỷ lệ phổ biến trong huấn luyện mô hình học máy, giúp đảm bảo mô hình được huấn luyện trên lượng dữ liệu đủ lớn (70%), đồng thời có tập validation (20%) để đánh giá hiệu suất trong quá trình huấn luyện và tập test (10%) để kiểm tra khả năng tổng quát hóa. Việc sử dụng random\_state=42 trong cả hai lần chia dữ liệu giúp đảm bảo tính nhất quán, rất quan trọng khi thử nghiệm và tối ưu mô hình chatbot y tế

**Tập dữ liệu thứ ba**

* + 1. *Huấn luyện khi thay đổi optimizer,learning rate và epoch*

**Thay đổi learing rate**

|  |  |  |
| --- | --- | --- |
| Learning rate | Train Accuracy | Best Val Accuracy |
| 0.0001 | 0.069364 | 0.04 |
| 0.0010 | 0.751445 | 0.10 |
| 0.0100 | 0.994220 | 0.22 |
| 0.1000 | 0.994220 | 0.10 |

Bảng 1: So sánh với learning rate của tập dữ liệu thứ nhất

|  |  |  |
| --- | --- | --- |
| Learning rate | Train Accuracy | Best Val Accuracy |
| 0.0001 | 0.169364 | 0.08 |
| 0.0010 | 0.976879 | 0.55 |
| 0.0100 | 0.794220 | 0.45 |
| 0.1000 | 0.74220 | 0.33 |

Bảng 2: So sánh với learning rate của tập dữ liệu thứ hai

|  |  |  |
| --- | --- | --- |
| Learning rate | Train Accuracy | Best Val Accuracy |
| 0.0001 | 0.249380 | 0.02 |
| 0.0010 | 0.532625 | 0.30 |
| 0.0100 | 0.310000 | 0.45 |
| 0.1000 | 0.842200 | 0.33 |

Bảng 3: So sánh với learning rate của tập dữ liệu thứ ba

Khi sử dụng learning rate là 0.0001, mô hình gần như không học được gì từ dữ liệu. Kết quả cho thấy độ chính xác trên tập huấn luyện chỉ đạt khoảng 6.9%, còn độ chính xác trên tập validation chỉ là 4%. Điều này cho thấy tốc độ học quá chậm khiến mô hình không hội tụ đủ trong số epoch đã cho. Mô hình ở trường hợp này rơi vào tình trạng underfitting — tức là không thể nắm bắt được mối quan hệ giữa dữ liệu đầu vào và nhãn đầu ra.

Khi tăng learning rate lên 0.001, mô hình bắt đầu học tốt hơn. Độ chính xác trên tập huấn luyện đạt khoảng 75%, cho thấy mô hình đã có khả năng ghi nhớ các mẫu dữ liệu. Tuy nhiên, độ chính xác trên tập validation chỉ đạt 10%, phản ánh mô hình vẫn chưa khái quát hóa được tốt và có dấu hiệu overfitting nhẹ — học tốt trên tập huấn luyện nhưng không áp dụng tốt trên dữ liệu mới. Điều này cho thấy learning rate này đã khá ổn, nhưng vẫn cần điều chỉnh hoặc kết hợp thêm kỹ thuật regularization.

Khi learning rate tăng lên 0.1, độ chính xác trên tập huấn luyện vẫn duy trì ở mức rất cao (99.4%), nhưng độ chính xác trên tập validation lại giảm còn 10%, tương đương với trường hợp learning rate = 0.001. Điều này cho thấy mô hình đã học quá nhanh, và có thể đã bỏ qua các điểm tối ưu quan trọng do bước nhảy quá lớn trong quá trình cập nhật trọng số. Learning rate quá cao thường khiến quá trình học trở nên dao động, không ổn định, hoặc nhanh chóng overfit, như được thể hiện rõ trong kết quả này.

Từ các kết quả trên, ta thấy rằng lựa chọn learning rate phù hợp là cực kỳ quan trọng trong quá trình huấn luyện mô hình. Learning rate quá nhỏ khiến mô hình học chậm và không hiệu quả; quá lớn thì khiến mô hình không hội tụ ổn định hoặc overfit nhanh chóng. Trong trường hợp này, learning rate = 0.01 là lựa chọn tốt nhất

**Thay đổi epoch**

Từ lựa chọn trên với learning rate = 0.01, thay đổi epoch để so sánh:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epoch | Train Accuracy | Best Val Accuracy | Final Loss | Final Val Loss |
| 10 | 0.971098 | 0.18 | 0.116067 | 7.062705 |
| 20 | 0.976879 | 0.20 | 0.037863 | 8.469006 |
| 50 | 0.994220 | 0.08 | 0.021991 | 9.386338 |
| 100 | 0.994220 | 0.20 | 0.045385 | 16.180819 |
| 150 | 0.624277 | 0.14 | 6.247488 | 32.868279 |

Bảng 4: So sánh với số lượng epoch tập dữ liệu thứ nhất

*Epoch*: Đây là số lần mô hình được "xem" toàn bộ dữ liệu huấn luyện. Mỗi lần như vậy gọi là một epoch. Ví dụ, epoch = 10 nghĩa là mô hình đã học qua toàn bộ dữ liệu 10 lần. Nếu epoch quá ít, mô hình có thể chưa học đủ. Nếu quá nhiều, mô hình có thể học "quá kỹ", dẫn đến không đoán tốt với dữ liệu mới

*Train Accuracy* (độ chính xác trên tập huấn luyện): Là tỷ lệ mô hình đoán đúng trên dữ liệu nó được học. Nếu chỉ số này cao thì có nghĩa mô hình đang học rất tốt trên dữ liệu đã biết.

*Best Val Accuracy* (độ chính xác cao nhất trên tập kiểm tra/validation): Đây là độ chính xác tốt nhất mà mô hình đạt được trên dữ liệu chưa từng thấy trong quá trình học. Nó cho biết mô hình có thể đoán tốt với dữ liệu mới hay không.

*Final Loss* (độ sai số cuối cùng trên tập huấn luyện): Loss là chỉ số cho biết mô hình đoán sai nhiều hay ít. Loss thấp nghĩa là mô hình đang đoán khá đúng. Final Loss là giá trị loss ở lần học cuối cùng.

*Final Val Loss* (độ sai số cuối cùng trên tập kiểm tra): Cũng giống như Final Loss, nhưng tính trên tập kiểm tra – tức là dữ liệu mô hình chưa từng được học. Nếu Final Val Loss cao mà Final Loss thấp, điều này cho thấy mô hình học quá kỹ trên tập huấn luyện và không đoán tốt trên dữ liệu mới –*overfitting*.

Kết quả huấn luyện mô hình theo các giá trị epoch cho thấy sự khác biệt rõ rệt về khả năng học và khái quát hóa của mô hình. Khi số epoch còn thấp (10–20), mô hình đạt độ chính xác trên tập huấn luyện rất cao (khoảng 97–98%) và đồng thời giữ được độ chính xác trên tập validation ở mức tương đối ổn định (18–20%). Điều này cho thấy mô hình đang học tốt mà chưa bị quá khớp (overfitting).

Tuy nhiên, khi số epoch tăng lên 50 và 100, mặc dù train accuracy gần như đạt tuyệt đối (99.4%), nhưng best val accuracy lại giảm xuống (thậm chí chỉ còn 8% ở epoch = 50), và final val loss tăng mạnh. Đây là dấu hiệu rõ ràng của overfitting: mô hình ghi nhớ dữ liệu huấn luyện thay vì học được các đặc trưng tổng quát.

Đáng chú ý, tại epoch = 150, train accuracy bất ngờ giảm mạnh xuống chỉ còn 62%, kèm theo final val loss tăng vọt lên 32.86. Điều này cho thấy mô hình có thể đã quá học lâu dẫn đến hiện tượng mất khả năng tối ưu do learning rate cao khiến quá trình học trở nên không ổn định.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epoch | Train Accuracy | Best Val Accuracy | Final Loss | Final Val Loss |
| 10 | 0.459442 | 0.235099 | 4.062639 | 15.088922 |
| 20 | 0.459442 | 0.189294 | 3.536958 | 20.293917 |
| 50 | 0.809419 | 0.271523 | 1.707418 | 29.433266 |
| 100 | 0.780123 | 0.252208 | 2.637911 | 37.708961 |
| 150 | 0.792408 | 0.210265 | 5.808945 | 132.840134 |

Bảng 5: So sánh với số lượng epoch tập dữ liệu thứ hai

**Thay đổi optimizer**

Từ lựa chọn trên với learning rate = 0.01 và epoch = 100, thay đổi optimizer để so sánh:

|  |  |  |
| --- | --- | --- |
| Optimizer | Train Accuracy | Best Val Accuracy |
| adam | 0.988439 | 0.18 |
| sgd | 0.080925 | 0.04 |
| rmsprop | 0.982659 | 0.16 |

Bảng 6: So sánh với thay đổi optimizer với tập dữ liệu thứ nhất

Adam là một optimizer kết hợp giữa phương pháp momentum và RMSprop, tự động điều chỉnh learning rate cho từng tham số dựa trên trung bình mũ của gradient và bình phương gradient trước đó. Nhờ đó, Adam thường hội tụ nhanh và ổn định hơn trên nhiều bài toán khác nhau.

SGDlà phương pháp tối ưu cơ bản nhất, cập nhật trọng số dựa trên gradient tại mỗi bước. Tuy nhiên, SGD với learning rate cố định 0.01 trong trường hợp này thể hiện khả năng hội tụ kém, có thể do learning rate không phù hợp hoặc do không sử dụng momentum, dẫn đến độ chính xác huấn luyện rất thấp.

RMSprop là một biến thể của SGD có khả năng tự điều chỉnh learning rate cho mỗi tham số, dựa trên trung bình mũ bình phương của gradient. RMSprop thường hoạt động tốt trên các mạng hồi tiếp như LSTM, giúp tăng tốc hội tụ.

Adam đạt độ chính xác huấn luyện cao nhất (98.84%) và giá trị tốt nhất trên tập validation (18%), chứng tỏ nó hiệu quả trong việc tối ưu mô hình. RMSprop cũng thể hiện kết quả rất tốt với độ chính xác huấn luyện 98.27% và validation 16%, chỉ thấp hơn Adam một chút. Ngược lại, SGD có hiệu suất kém, độ chính xác huấn luyện chỉ đạt 8.09% và validation chỉ 4%, cho thấy mô hình không học được nhiều với optimizer này trong điều kiện hiện tại.

|  |  |  |
| --- | --- | --- |
| Optimizer | Train Accuracy | Best Val Accuracy |
| adam | 0.981777 | 0.399823 |
| sgd | 0.005315 | 0.005319 |
| rmsprop | 0.126803 | 0.070035 |

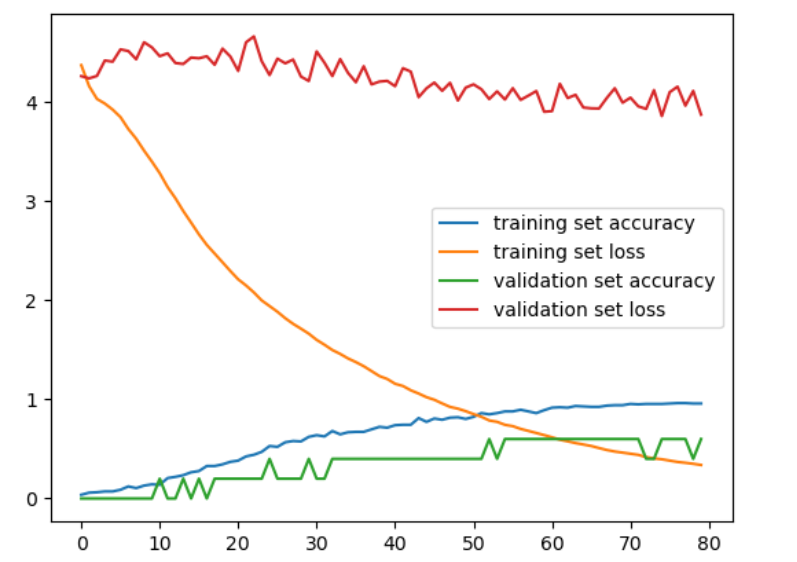
Bảng 7: So sánh với thay đổi optimizer với tập dữ liệu thứ hai

|  |  |  |
| --- | --- | --- |
| Optimizer | Train Accuracy | Best Val Accuracy |
| adam | 0.322706 | 0.305732 |
| sgd | 0.238399 | 0.105319 |
| rmsprop | 0.260716 | 0.170035 |

Bảng 8: So sánh với thay đổi optimizer với tập dữ liệu thứ ba

* 1. **Đánh giá mô hình**

**Tập dữ liệu thứ nhất**

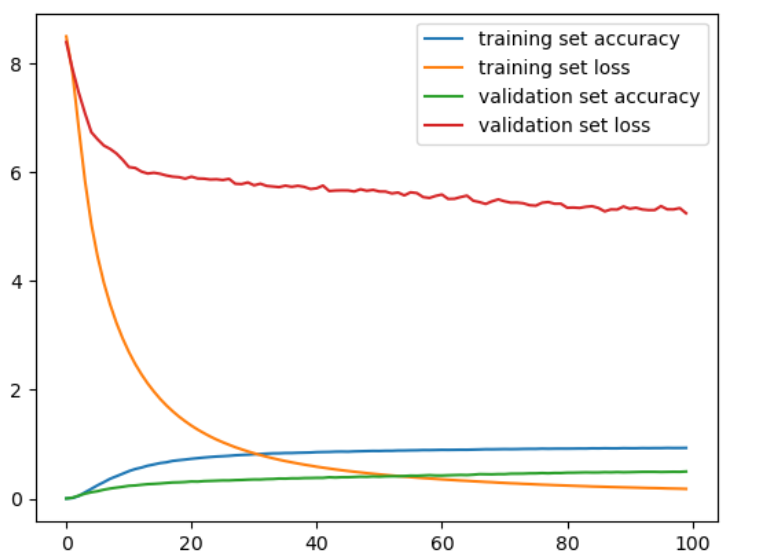


1. Biểu đồ độ chính xác và mất mát trên tập huấn luyện và tập kiểm tra của tập dữ liệu thứ nhất

Dựa trên biểu đồ, mô hình thể hiện hiệu suất đáng chú ý trong quá trình huấn luyện. Độ chính xác của tập huấn luyện (training set accuracy) tăng dần và đạt mức ổn định cao, trong khi độ chính xác của tập kiểm tra (validation set accuracy) cũng duy trì ở mức tương đối ổn định, dao động quanh một giá trị thấp hơn nhưng vẫn cho thấy khả năng tổng quát hóa tốt của mô hình. Sự khác biệt giữa hai chỉ số này tuy không quá lớn, nhưng vẫn là một điểm cần được xem xét để đánh giá thêm về khả năng áp dụng thực tế.

Về mặt mất mát , mất mát của tập huấn luyện giảm mạnh trong giai đoạn đầu và sau đó ổn định ở mức thấp, phản ánh quá trình học hiệu quả trên dữ liệu huấn luyện. Tuy nhiên, mất mát của tập kiểm tra lại có xu hướng tăng dần theo số lượng epoch, điều này là một tín hiệu rõ ràng của hiện tượng overfitting. Mô hình dường như đang học quá mức chi tiết trên dữ liệu huấn luyện, dẫn đến việc không còn khả năng dự đoán tốt trên dữ liệu chưa thấy trước đó.

**Tập dữ liệu thứ hai**



1. Biểu đồ độ chính xác và mất mát trên tập huấn luyện và tập kiểm tra của tập dữ liệu thứ hai

Dựa trên biểu đồ và số liệu, đường cong training set accuracy (màu xanh lam) tăng nhanh trong khoảng 20 epoch đầu, từ mức thấp lên gần 1.0, và duy trì ổn định ở mức cao. Cụ thể, các giá trị training accuracy trong 5 epoch cuối được ghi nhận là [0.9289, 0.9318, 0.9305, 0.9308, 0.9321], cho thấy độ chính xác trung bình trên tập huấn luyện đạt khoảng 93%. Điều này chỉ ra rằng mô hình LSTM, được xây dựng trong code, đã học tốt các đặc trưng từ dữ liệu hội thoại y tế trong file intents2.json. Song song đó, đường cong training set loss (màu cam) giảm mạnh từ giá trị ban đầu xuống gần 0 trong 20 epoch đầu, và các giá trị training loss trong 5 epoch cuối là [0.1916, 0.1877, 0.1853, 0.1829, 0.1797], cho thấy loss trung bình khoảng 0.18. Sự giảm đều đặn của loss và độ chính xác cao trên tập huấn luyện cho thấy mô hình hội tụ tốt, có khả năng phân loại ý định (intent) từ các câu hỏi y tế (như "What is Glaucoma?" hoặc "What are the symptoms of Glaucoma?") một cách chính xác trên dữ liệu huấn luyện.

Đường cong validation set accuracy (màu xanh lá) tăng chậm từ khoảng 0.1 lên gần 0.5 trong 20 epoch đầu, nhưng sau đó dao động quanh mức 0.5 đến 0.6 cho đến epoch 100. Các giá trị validation accuracy trong 5 epoch cuối là [0.4902, 0.4921, 0.4939, 0.4930, 0.4988], với giá trị trung bình khoảng 49.3%. Điều này cho thấy mô hình hoạt động kém trên tập kiểm định, không đạt được hiệu suất mong muốn. Tương tự, đường cong validation set loss (màu đỏ) khởi đầu ở mức cao (gần 8) và giảm xuống còn khoảng 4-5, nhưng dao động mạnh và không giảm thêm. Các giá trị validation loss trong 5 epoch cuối là [5.3726, 5.3132, 5.3120, 5.3358, 5.2412], với giá trị trung bình khoảng 5.31. Sự chênh lệch lớn giữa training accuracy/loss (93% và 0.18) và validation accuracy/loss (49.3% và 5.31) là dấu hiệu rõ ràng của hiện tượng quá khớp (overfitting). Mô hình đã học quá tốt trên tập huấn luyện nhưng không tổng quát hóa được trên tập kiểm định, điều này có thể làm giảm khả năng của chatbot trong việc xử lý các câu hỏi y tế mới hoặc đa dạng hơn.

# CHƯƠNG 4: BÀI TẬP CHẠY CÁC PHIÊN BẢN DỰ ĐOÁN BỆNH TIỂU ĐƯỜNG

* 1. **Phiên bản 1**

Phiên bản có:

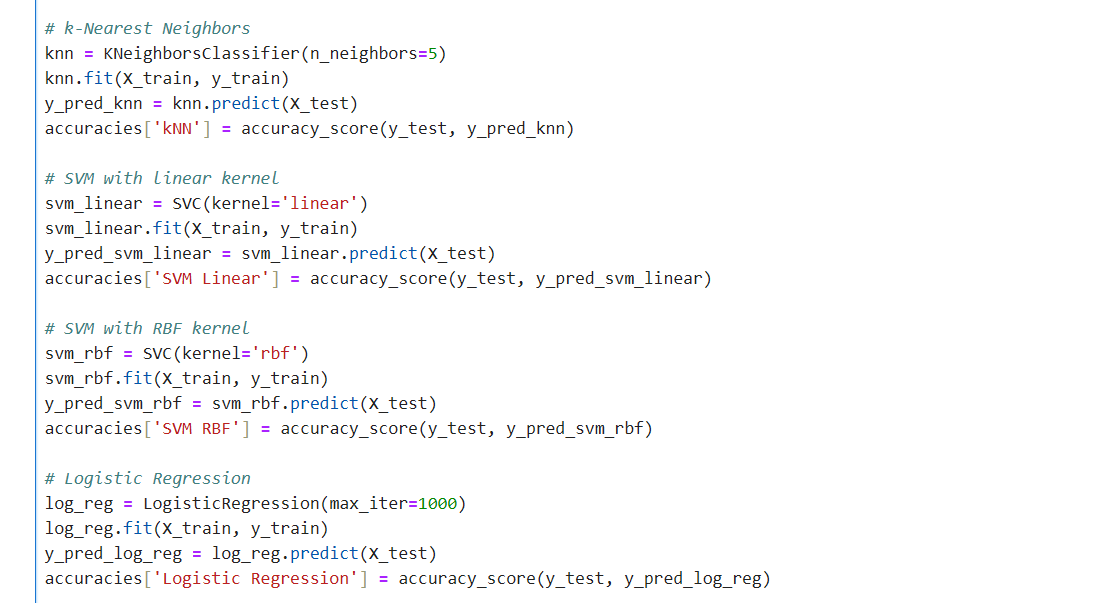
***FlaskAPI***: FlaskAPI là một phần mở rộng của framework Flask, được thiết kế để xây dựng các API RESTful trong Python một cách đơn giản và hiệu quả. Nó cung cấp các công cụ để tạo các endpoint xử lý các yêu cầu HTTP như GET, POST, PUT, DELETE, và trả về dữ liệu ở định dạng JSON hoặc XML, phù hợp với các ứng dụng web hoặc di động. Với tính nhẹ và dễ tùy chỉnh, FlaskAPI cho phép lập trình viên nhanh chóng triển khai các dịch vụ backend, tích hợp dễ dàng với cơ sở dữ liệu và các thư viện khác, lý tưởng cho các dự án quy mô nhỏ đến trung bình.

***Giao diện HTML***: Giao diện HTML là nền tảng để xây dựng cấu trúc hiển thị của một trang web, định dạng nội dung như văn bản, hình ảnh, biểu mẫu, bảng, và liên kết. Các thẻ HTML, chẳng hạn như <div>, <p>, hoặc <input>, được sử dụng để tổ chức và trình bày thông tin một cách trực quan cho người dùng

Đoạn mã huấn luyện:



1. Đoạn mã tải tập dữ diệu và chia tập dữ liệu

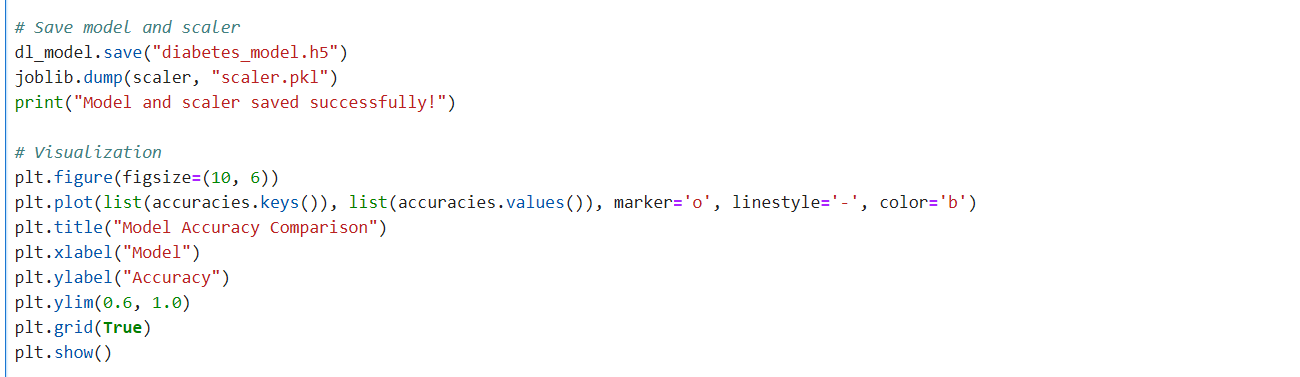


1. Đoạn mã huấn luyện sử dụng KNN, SVM, Logistic Regression

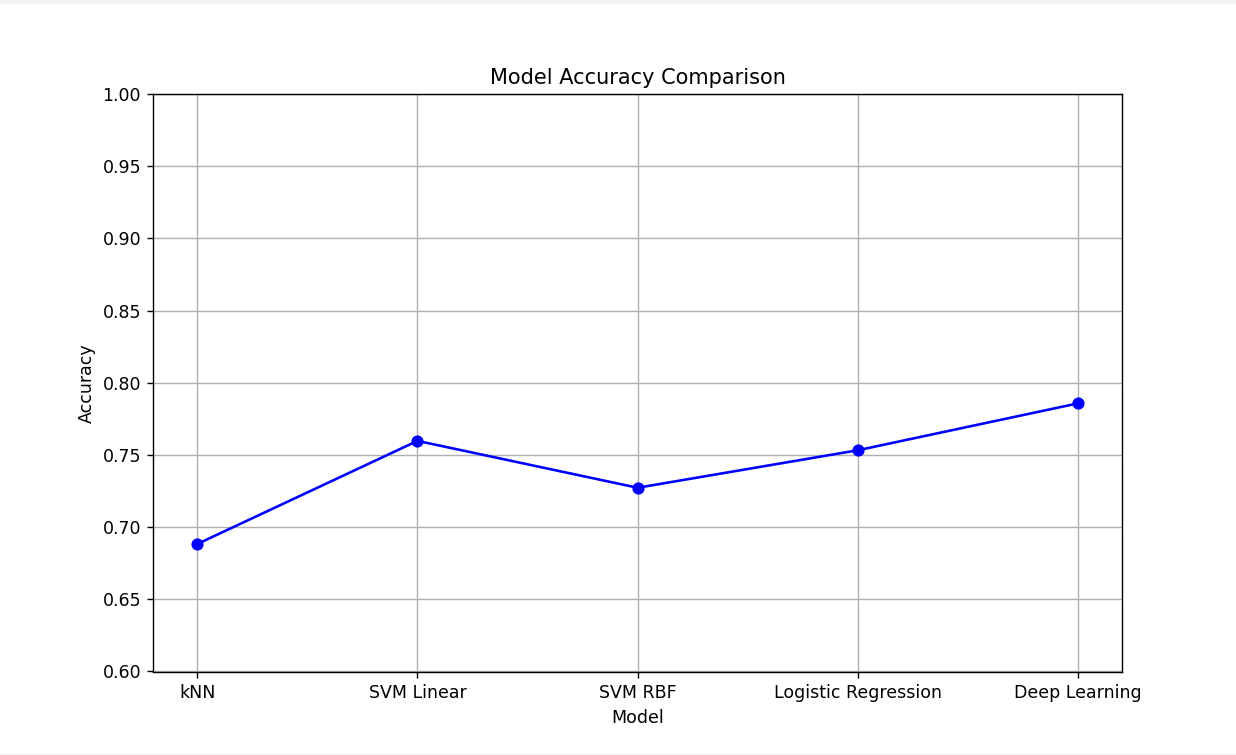
Đoạn code trên triển khai bốn mô hình học máy để phân loại dữ liệu: K-Nearest Neighbors (KNN), SVM với kernel tuyến tính (Linear), SVM với kernel RBF, và Logistic Regression. Đầu tiên, KNN được khởi tạo với 5 hàng xóm gần nhất, sau đó huấn luyện trên tập dữ liệu huấn luyện (X\_train, y\_train) và dự đoán trên tập kiểm tra (X\_test), với độ chính xác được tính bằng hàm accuracy\_score. Tương tự, hai mô hình SVM (kernel tuyến tính và RBF) cùng Logistic Regression (với 1000 vòng lặp tối đa) cũng được huấn luyện và đánh giá hiệu suất trên cùng tập dữ liệu, nhằm so sánh độ chính xác giữa các phương pháp.



1. Đoạn mã huấn luyện sử dụng Deep Learning



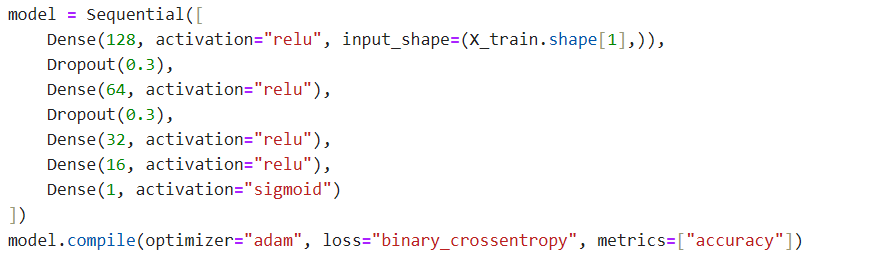
1. Đoạn mã huấn luyện sử dụng Deep Learning



1. Trực quan hóa độ chính xác các mô hình

Deep Learning có độ chính xác cao nhất trong số các mô hình. Các mô hình truyền thống như Logistic Regression, SVM, kNN cũng hoạt động khá ổn. Sự khác biệt giữa các mô hình không quá lớn, nhưng Deep Learning có tiềm năng mở rộng tốt hơn. Deep Learning (học sâu) là một nhánh của Machine Learning, sử dụng mạng nơ-ron nhân tạo nhiều tầng để học các đặc trưng phức tạp từ dữ liệu. Trong bài toán này, chúng tôi xây dựng một mô hình mạng nơ-ron với kiến trúc gồm 6 lớp (trong đó có 4 lớp Dense và 2 lớp Dropout) nhằm phân loại bệnh nhân có/không có nguy cơ mắc bệnh tiểu đường.

**Kiến trúc mô hình**

****

***Dense layers***: Các lớp dense học các đặc trưng từ dữ liệu đầu vào. Mỗi neuron trong lớp kết nối với tất cả neuron ở lớp trước, tạo ra các biểu diễn phức tạp. Chúng đóng vai trò quan trọng trong việc trích xuất thông tin từ dữ liệu.

***Dropout layers***: Dropout giúp giảm overfitting bằng cách tắt ngẫu nhiên một số neuron trong quá trình huấn luyện. Điều này buộc mô hình học các biểu diễn mạnh mẽ hơn, không phụ thuộc vào một số neuron cụ thể

***Activation function ReLU***: Hàm kích hoạt ReLU được sử dụng trong các lớp ẩn để học các quan hệ phi tuyến tính. Nó chuyển đổi các giá trị âm thành 0 và giữ nguyên giá trị dương. ReLU giúp mô hình học nhanh hơn và tránh vấn đề tiêu biến gradient.

***Sigmoid***: Hàm sigmoid được dùng ở lớp đầu ra cho bài toán phân loại nhị phân. Nó ánh xạ đầu ra thành giá trị từ 0 đến 1, biểu thị xác suất. Hàm này phù hợp cho việc dự đoán nhãn 0 hoặc 1.

**Huấn luyện mô hình**

Mô hình được huấn luyện trong 100 epochs, với kích thước batch = 10, sử dụng bộ dữ liệu được chuẩn hóa.





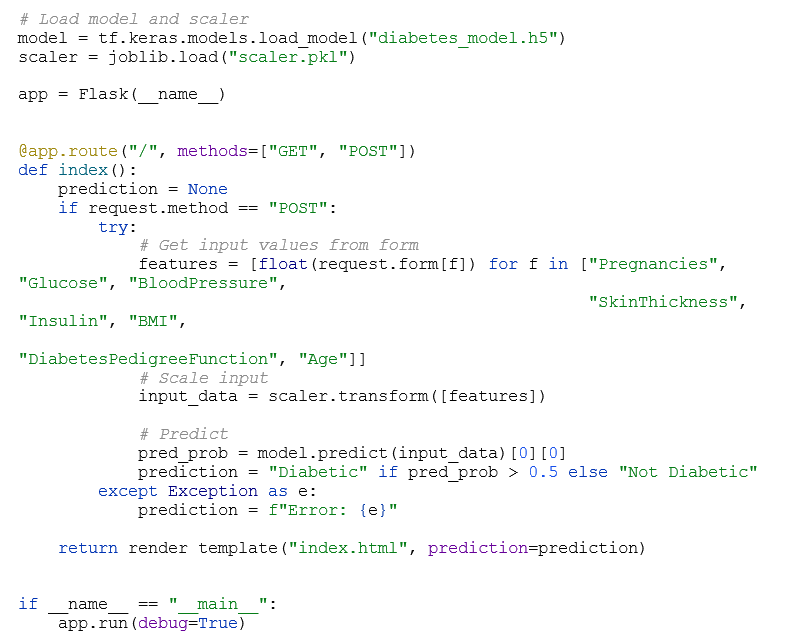
***Adam optimizer***: Adam là một bộ tối ưu hóa mạnh mẽ và phổ biến, kết hợp gradient descent và momentum. Nó điều chỉnh tốc độ học tập thích ứng cho từng tham số, giúp hội tụ nhanh. Adam được sử dụng rộng rãi trong các mô hình học sâu.

***Binary crossentropy***: Binary crossentropy là hàm mất mát lý tưởng cho bài toán phân loại nhị phân. Nó đo lường sự khác biệt giữa dự đoán và nhãn thực tế, với giá trị từ 0 đến 1. Hàm này giúp mô hình tối ưu hóa hiệu quả cho phân loại 0 hoặc 1.

**Kết quả mô hình**

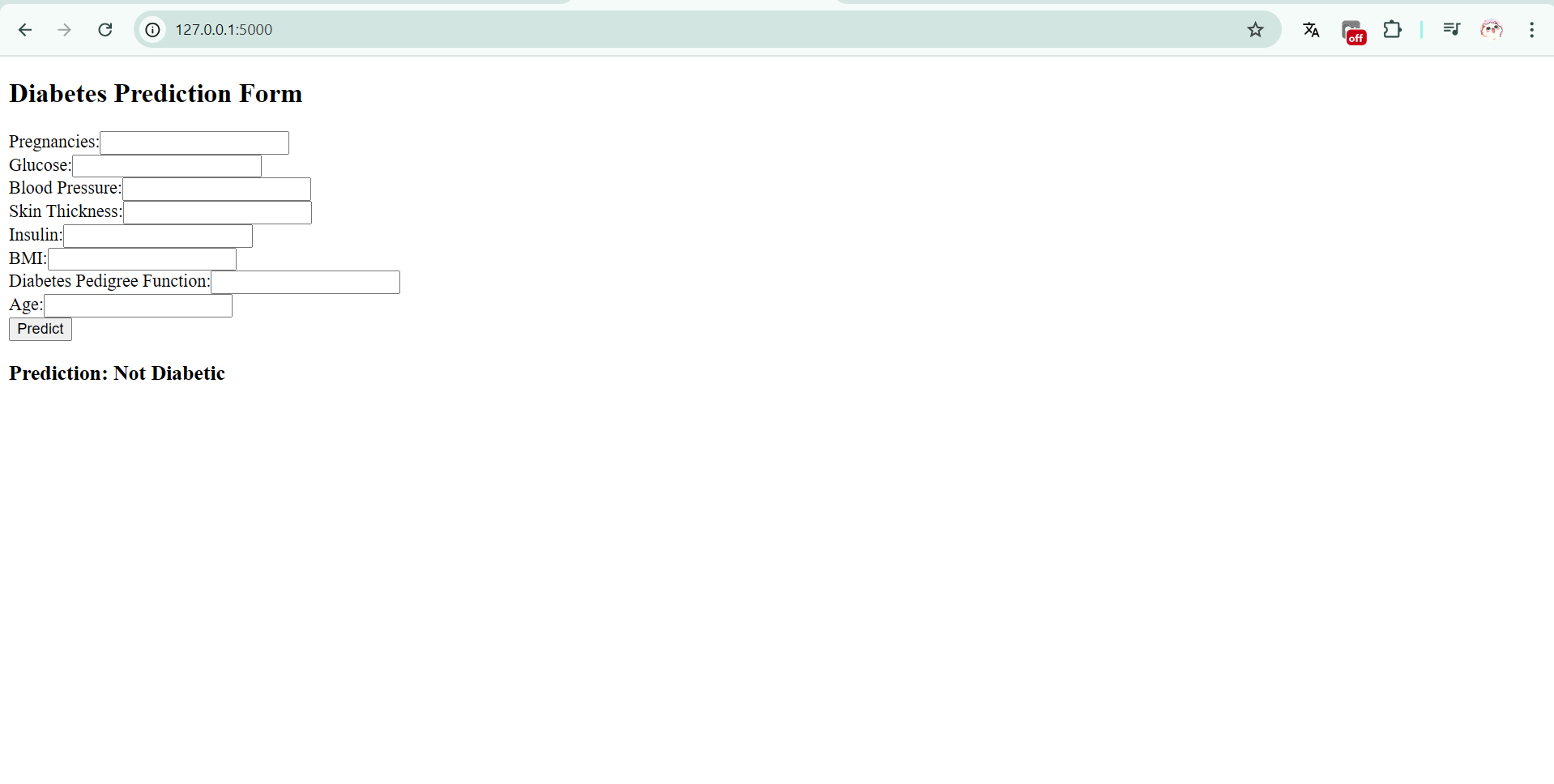
Mô hình đạt độ chính xác cao trên tập kiểm thử, cao nhất trong số các mô hình thử nghiệm. Điều này cho thấy Deep Learning có khả năng học sâu hơn và phát hiện được các mối quan hệ phức tạp giữa các đặc trưng y tế. Deep Learning là hướng tiếp cận hiện đại, mạnh mẽ và phù hợp với bài toán dữ liệu y tế có nhiều đặc trưng và phi tuyến tính. Tuy nhiên, mô hình cần dữ liệu chất lượng, phải được tinh chỉnh cẩn thận để đạt hiệu quả cao nhất.

**Load model**



**Tạo file index.html**





1. Giao diện phiên bản 1
   1. **Phiên bản 2**

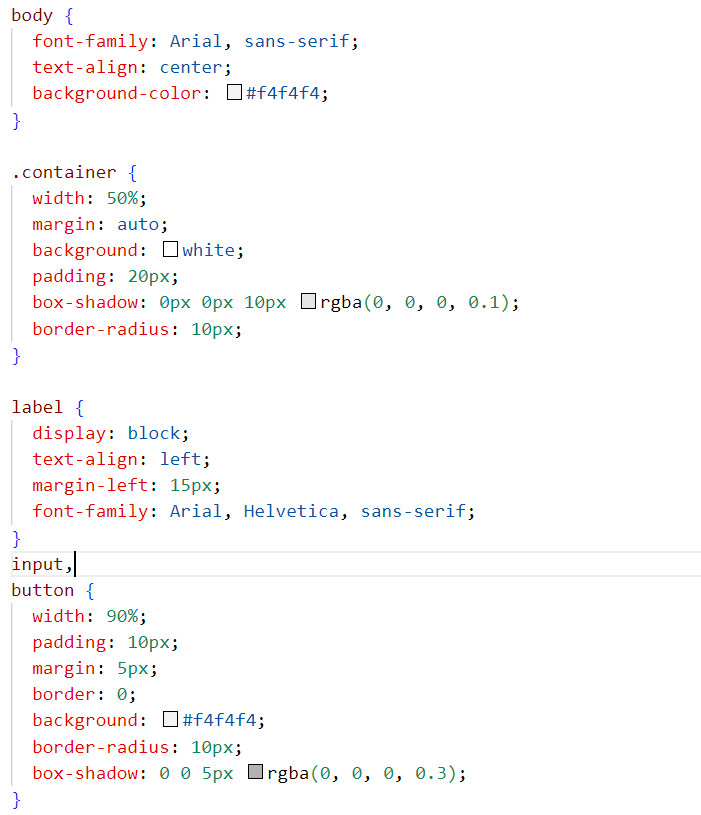
Điểm cải tiến: Node JS cho fontend, UI bằng CSS, mẫu tương tác với JS

**Node.js cho frontend**: Node.js thường được biết đến như một môi trường runtime phía backend, nhưng trong phát triển frontend, nó được dùng để hỗ trợ các công cụ như Webpack, Vite, hay quản lý gói qua npm. Nó cho phép lập trình viên chạy JavaScript để xây dựng và tối ưu hóa giao diện người dùng trước khi triển khai. Điều này giúp quá trình phát triển frontend trở nên hiệu quả và tích hợp tốt với các framework như React hay Vue.

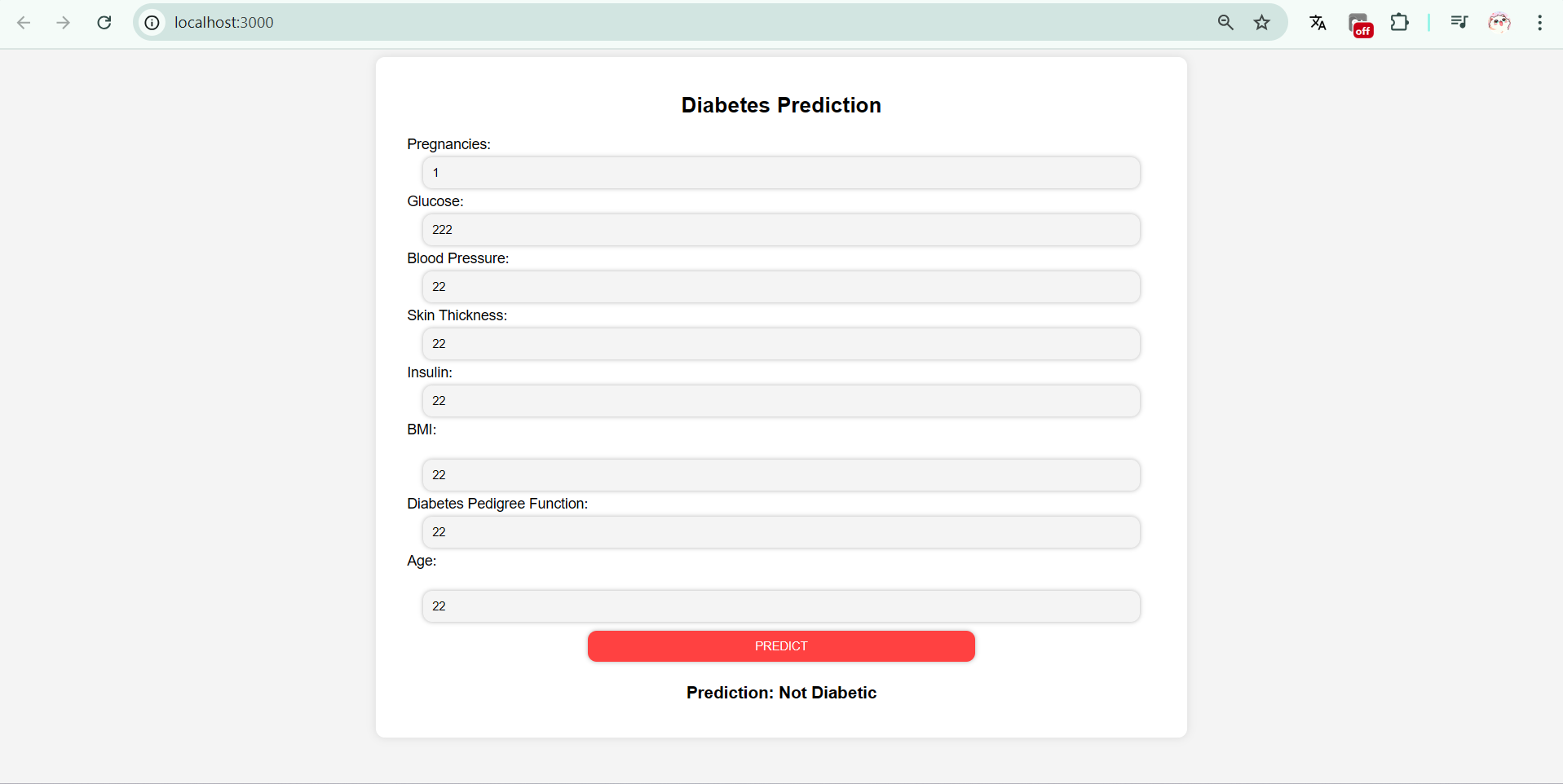
**Modern UI with CSS**: Giao diện người dùng hiện đại (Modern UI) sử dụng CSS (Cascading Style Sheets) để tạo ra các thiết kế đẹp mắt và đáp ứng. CSS là ngôn ngữ định kiểu, giúp tùy chỉnh màu sắc, bố cục, phông chữ, và hiệu ứng như hover hay animation, ví dụ: button:hover {background-color: #4CAF50;} làm nút đổi màu khi di chuột. CSS kết hợp với các kỹ thuật như Flexbox hoặc Grid giúp giao diện linh hoạt trên nhiều thiết bị.

**Interactive form với JS**: Mẫu (form) tương tác được tạo bằng JavaScript để tăng tính năng động cho giao diện web. JS xử lý các sự kiện như khi người dùng nhập dữ liệu, gửi form, hoặc hiển thị thông báo, ví dụ: kiểm tra tính hợp lệ của email trước khi gửi. Điều này cải thiện trải nghiệm người dùng bằng cách cung cấp phản hồi tức thì mà không cần tải lại trang.

**Tạo file styles.css**

****

**Kết quả**



1. Giao diện phiên bản 2
   1. **Phiên bản 3**

Điểm cải tiến: Cải thiện độ chính xác của mô hình, dùng nhiều lớp hơn, có BatchNormalization, LeakyReLU, Dropout, Optimizer: AdamW (mạnh hơn Adam), EarlyStopping để tránh overfiting, API của mô hình sẽ được triển khai trong Docker

Docker là một nền tảng ảo hóa cấp container, cho phép đóng gói ứng dụng cùng toàn bộ môi trường phụ thuộc (thư viện, cấu hình, hệ điều hành) vào một container độc lập. Container này đảm bảo API chạy ổn định và nhất quán trên mọi hệ thống, từ máy phát triển đến server sản xuất, mà không gặp vấn đề về sự khác biệt môi trường



**Chi tiết cải tiến**

BatchNormalization()

Chuẩn hóa đầu vào của từng lớp về phân phối chuẩn (mean = 0, std = 1). Giúp mô hình hội tụ nhanh hơn, giảm hiện tượng "internal covariate shift". Hạn chế overfitting, đặc biệt là khi có nhiều tầng ẩn.

tf.keras.layers.LeakyReLU(alpha=0.1)

Một dạng ReLU cải tiến giúp tránh hiện tượng "dead neuron" khi đầu vào nhỏ hơn 0. Luôn cho ra giá trị khác 0 → mô hình học tốt hơn với dữ liệu phức tạp.

Dropout(0.4) # hoặc 0.3

Tắt ngẫu nhiên các node trong quá trình huấn luyện. Ngăn mô hình học quá khớp (overfitting). Ở đây dùng tỉ lệ dropout lớn hơn mô hình cũ (0.4 → mạnh hơn), phù hợp cho mô hình nhiều tầng.

Mô hình tổng quát:

Input → Dense → BatchNorm → LeakyReLU → Dropout → ... → Dense

Dùng AdamW thay vì Adam

from tensorflow.keras.optimizers import AdamW

AdamW = Adam + Weight Decay (regularization trực tiếp lên trọng số).

Lợi ích:

+ Tránh overfitting tốt hơn.

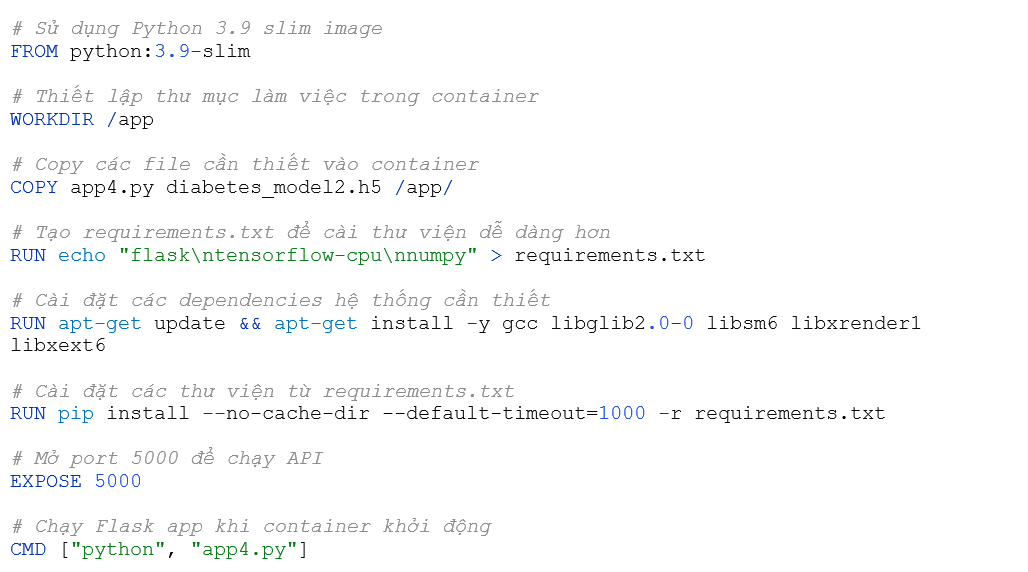
+ Ổn định hơn với dữ liệu không đồng đều.

+ Tối ưu hóa giống như L2 Regularization một cách rõ ràng hơn Adam thường.

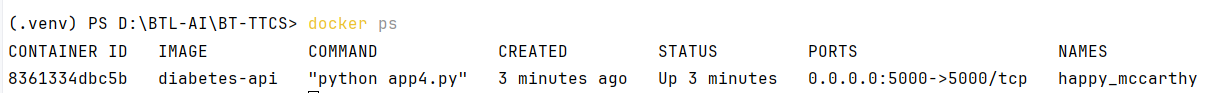
EarlyStopping(monitor="val\_loss", patience=10, restore\_best\_weights=True)

Theo dõi val\_loss: nếu không giảm sau 10 lần (patience=10), sẽ dừng huấn luyện. restore\_best\_weights=True: đảm bảo trọng số tốt nhất sẽ được lưu lại, tránh việc lấy model tại epoch cuối. Cách hiệu quả để ngăn overfitting, tiết kiệm thời gian huấn luyện. Chuẩn hóa toàn bộ dữ liệu đầu vào giúp: Các feature có cùng phân phối → mô hình hội tụ nhanh hơn

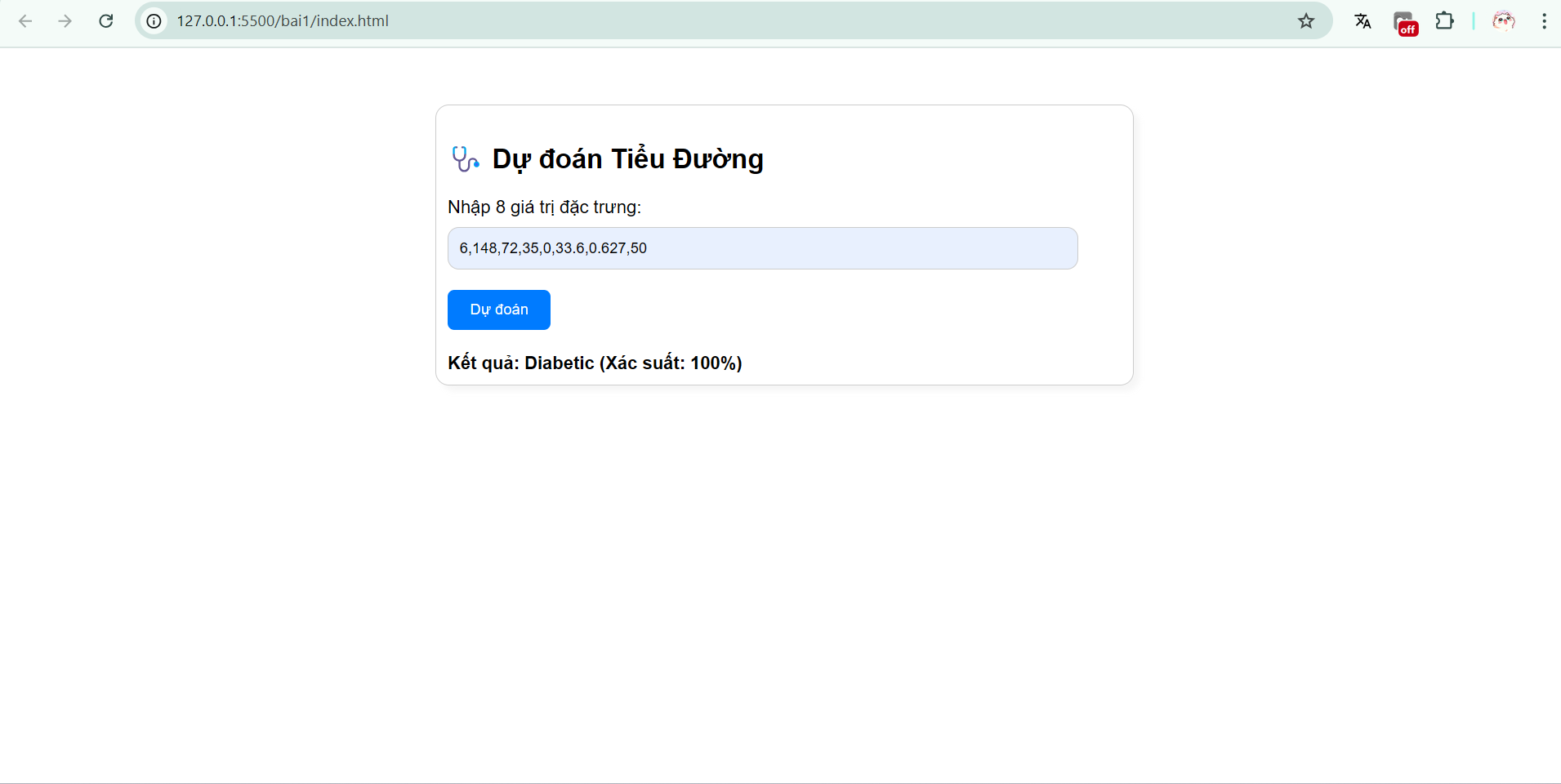
**Tạo Dockerfile để deploy trên Docker**



Kiểm tra xem Docker có đang chạy hay không



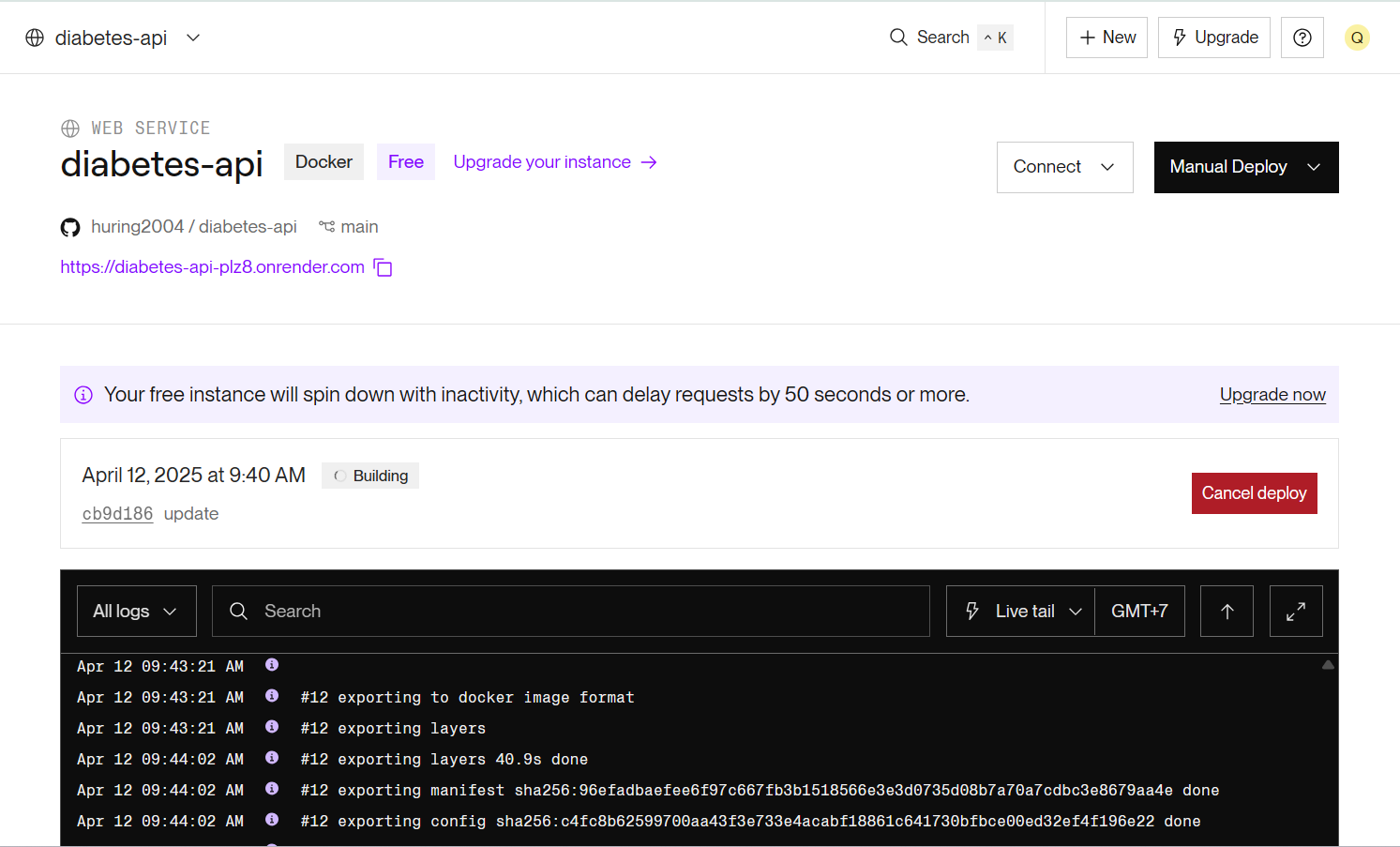
**Kết quả**



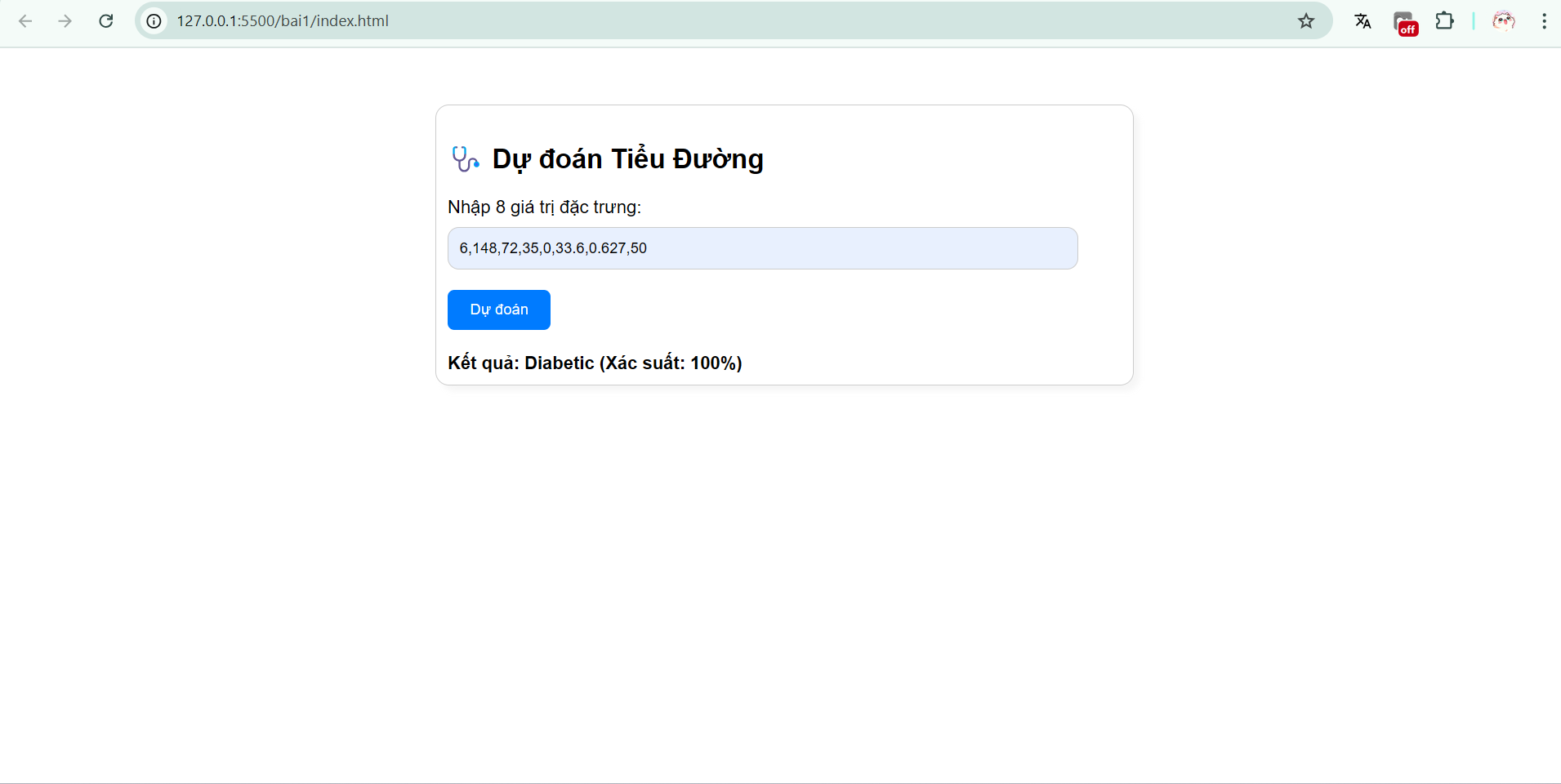
1. Giao diện phiên bản 3
   1. **Phiên bản 4**

Điểm cải tiến: Deploy trên Render

Render là gì? Render là một nền tảng Platform as a Service (PaaS) dựa trên đám mây, cho phép triển khai và quản lý ứng dụng web, API, hoặc cơ sở dữ liệu một cách dễ dàng mà không cần quản lý hạ tầng phức tạp. Render hỗ trợ nhiều ngôn ngữ lập trình (Node.js, Python, Ruby, v.v.) và cả Docker, cho phép chạy bất kỳ ứng dụng nào trong container. Các tính năng chính bao gồm: triển khai tự động từ Git, chứng chỉ TLS miễn phí, bảo vệ DDoS, mạng riêng (private networking), và tích hợp cơ sở dữ liệu như PostgreSQL hoặc Redis. Render cung cấp tầng miễn phí (free tier) với tài nguyên giới hạn (ví dụ: 500MB RAM, 0.1 CPU), phù hợp cho các dự án nhỏ hoặc thử nghiệm, và các gói trả phí để mở rộng quy mô



**Kết quả**



1. Giao diện phiên bản 4
   1. **So sánh các phiên bản**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tiêu chí** | **Version 1** | **Version 2** | **Version 3** | **Version 4** |
| Giao diện | HTML đơn giản | Node.js + CSS đẹp | HTML + JS đơn giản | HTML, JS đơn giản |
| Kỹ thuật | Python, Flask | Python, Flask, Node.js | Python, HTML/JS, Docker | Python, ML nâng cao |
| Ứng dụng | Web | Web | Web | Web |
| Triển khai | Local(frontend) | Local (frontend/backend) | Docker container | Render |
| Accuracy mô hình | Cơ bản | Cơ bản | Tối ưu hóa bằng kỹ thuật mới | Sử dụng lại của version 3 |

Bảng 9: Bảng so sánh các phiên bản

# KẾT LUẬN

Báo cáo đã cung cấp cái nhìn toàn diện về trí tuệ nhân tạo và học sâu, từ khái niệm, lịch sử, đến các công nghệ cốt lõi như mạng nơ-ron tích chập, hồi tiếp, và mô hình sinh. Các kỹ thuật tối ưu hóa và điều chuẩn được trình bày để nâng cao hiệu quả mô hình. Ứng dụng cụ thể của học sâu trong chatbot y tế với các tập dữ liệu chăm sóc sức khỏe tinh thần, hay y tế được phân tích chi tiết, từ tiền xử lý dữ liệu, xây dựng và huấn luyện mô hình, đến đánh giá hiệu suất khi điều chỉnh các tham số như optimizer, learning rate và epoch. Báo cáo nhấn mạnh tiềm năng của AI trong cải thiện sức khỏe tinh thần, đồng thời khẳng định tầm quan trọng của việc tối ưu hóa và triển khai mô hình để đạt hiệu quả cao trong thực tiễn.

# TÀI LIỆU THAM KHẢO

[1] Chollet, F. (2021). Deep Learning with Python, Second Edition. Manning Publications.

[2] Lee, W.-M. (2019). Python Machine Learning. Wiley.

[3] Mental Health Conversational Data. Kaggle. Truy cập từ *https://www.kaggle.com/datasets/elvis23/mental-health-conversational-data*

[4] Medquad Medical Question Answer . Kaggle. Truy cập từ *https://www.kaggle.com/datasets/pythonafroz/medquad-medical-question-answer-for-ai-research*