

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
**KHOA CÔNG NGHỆ THÔNG TIN 1**

---



**BÁO CÁO THỰC TẬP CƠ SỞ**

<b>Giảng viên hướng dẫn</b>	<b>: PGS.TS Trần Đình Quế</b>
<b>Họ và tên sinh viên</b>	<b>: Đình Quang Hưng</b>
<b>Mã sinh viên</b>	<b>: B22DCCN407</b>
<b>Lớp</b>	<b>: D22CQCN11-B</b>
<b>Nhóm thực tập</b>	<b>: 30</b>

*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

**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 “noron”, 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.

**Tăng tính tương tác:** 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 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.

**Cải thiện tính khả thi trong triển khai:** Phần mã lập trình liên quan đến GPT-2 trong mục 3.4.2 trước đây thiếu hướng dẫn chi tiết, gây khó khăn cho người muốn triển khai thực tế. Để khắc phục, phần này đã được mở rộng với các hướng dẫn cụ thể, bao gồm danh sách các thư viện cần thiết (như TensorFlow, PyTorch, hoặc Hugging Face Transformers), các yêu cầu phần cứng tối thiểu (ví dụ: GPU

NVIDIA với bộ nhớ tối thiểu 16GB), và quy trình xử lý dữ liệu đầu vào, từ tiền xử lý văn bản đến định dạng dữ liệu cho huấn luyện. Ngoài ra, các đoạn mã mẫu đã được bổ sung chú thích rõ ràng, giải thích từng bước và cung cấp ví dụ về cách tích hợp mã vào một hệ thống thực tế. Những cải tiến này đảm bảo rằng người đọc, kể cả những người mới bắt đầu, có thể dễ dàng áp dụng mã lập trình vào các dự án thực tế.

**Bổ sung thảo luận về yêu cầu tài nguyên tính toán:** Phần “Huấn luyện Chatbot sử dụng GPT-2” trước đây không đề cập đến các yêu cầu về phần cứng, thời gian huấn luyện, hay chi phí tính toán, khiến tài liệu thiếu thực tiễn. Để khắc phục, một mục con mới có tiêu đề “yêu cầu tài nguyên tính toán” đã được thêm vào, cung cấp thông tin chi tiết về cấu hình phần cứng cần thiết, chẳng hạn như sử dụng GPU (ví dụ: NVIDIA A100) hoặc TPU để huấn luyện GPT-2. Phần này cũng thảo luận về thời gian huấn luyện ước tính dựa trên kích thước tập dữ liệu và cấu hình phần cứng, cũng như các yếu tố ảnh hưởng đến chi phí tính toán, bao gồm việc sử dụng dịch vụ đám mây như AWS hoặc Google Cloud. Các đề xuất về cách tối ưu hóa tài nguyên, chẳng hạn như giảm kích thước mô hình hoặc sử dụng kỹ thuật huấn luyện hiệu quả, cũng được trình bày để hỗ trợ người triển khai thực tế.

**Thực hiện đề xuất chỉnh sửa về yêu cầu tài nguyên:** Theo đề xuất, một mục con mới, “yêu cầu tài nguyên và tối ưu hóa huấn luyện”, đã được bổ sung vào Chương 3. Mục này nêu rõ cấu hình phần cứng tối thiểu, chẳng hạn như GPU với 12GB VRAM hoặc TPU v3 cho huấn luyện GPT-2, và liệt kê các thư viện cần thiết như PyTorch 2.0 hoặc TensorFlow 2.8. Các ví dụ cụ thể về cách cấu hình môi trường huấn luyện trên các nền tảng như Google Colab hoặc máy chủ cục bộ cũng được đưa vào, giúp người đọc dễ dàng triển khai và tối ưu hóa quy trình huấn luyện trong các điều kiện tài nguyên khác nhau.

**Bổ sung phần đánh giá hiệu suất mô hình:** Để giải quyết thiếu sót về việc đánh giá hiệu suất mô hình, tài liệu đã bổ sung một mục mới trong Chương 3 có tiêu đề “đánh giá hiệu suất mô hình”. Mục này thảo luận chi tiết về các chỉ số đánh giá phổ biến như độ chính xác, F1-score, BLEU, và perplexity, giải thích ý nghĩa của từng chỉ số trong bối cảnh phát triển chatbot. Ví dụ, chỉ số BLEU được mô tả là phù hợp để đánh giá chất lượng văn bản sinh ra, trong khi perplexity được sử dụng để đo lường khả năng dự đoán của mô hình ngôn ngữ. Phần này cũng cung cấp các ví dụ thực tế về cách áp dụng các chỉ số này để đánh giá GPT-2 trong các tác vụ hội thoại, kèm theo hướng dẫn cách tính toán chúng bằng các công cụ như scikit-learn hoặc các thư viện NLP chuyên dụng.

**Thực hiện đề xuất chỉnh sửa về đánh giá hiệu suất:** Theo đề xuất, các mục con mới về “đánh giá hiệu suất mô hình” đã được thêm vào Chương. Trong Chương 2, mục 2.7 tập trung vào các khái niệm lý thuyết về đánh giá mô hình học

sâu, bao gồm mô tả các chỉ số như F1-score, BLEU, ROUGE, và perplexity, cùng với các ví dụ minh họa cách áp dụng chúng trong các ứng dụng thực tế như phân loại văn bản hoặc sinh văn bản. Trong Chương 3, mục đi sâu vào việc đánh giá chất lượng hội thoại của chatbot, cung cấp các kịch bản thực tế để đo lường hiệu suất GPT-2, chẳng hạn như sử dụng BLEU để so sánh câu trả lời của chatbot với câu trả lời chuẩn, hoặc sử dụng khảo sát người dùng để đánh giá mức độ tự nhiên của hội thoại. Các công cụ và mã mẫu để tính toán các chỉ số này cũng được cung cấp, đảm bảo người đọc có thể áp dụng chúng vào các dự án thực tế một cách dễ dàng.

## DANH MỤC TỪ VIẾT TẮT

<b>Từ viết tắt</b>	<b>Giải thích</b>
AI	Artificial 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
NLG	Natural Language Generation / Sinh ngôn ngữ tự nhiên
GPT-2	Generative Pre-trained Transformer 2 / Mô hình Transformer được huấn luyện trước (phiên bản 2)
NLU	Natural Language Understanding / Hiểu ngôn ngữ tự nhiên

## MỤC LỤC

GIỚI THIỆU CHUNG.....	1
DANH MỤC HÌNH VẼ.....	2
DANH MỤC TỪ VIẾT TẮT.....	3
MỤC LỤC.....	4
CHƯƠNG 1: TRÍ TUỆ NHÂN TẠO VÀ CÁC ỨNG DỤNG.....	6
1.1. Khái niệm về trí tuệ nhân tạo.....	6
1.2. Lịch sử phát triển của trí tuệ nhân tạo.....	7
1.3. Các công nghệ cốt lõi của trí tuệ nhân tạo.....	7
1.4. Các ứng dụng của trí tuệ nhân tạo.....	12
1.5. Lợi ích và thách thức của trí tuệ nhân tạo.....	14
1.6. Xu hướng phát triển trí tuệ nhân tạo trong tương lai.....	15
1.7. Kết luận.....	15
CHƯƠNG 2: CÁC KỸ THUẬT HỌC SÂU.....	17
2.1. Giới thiệu chung.....	17
2.2. Cơ bản về mạng nơ-ron.....	18
2.2.1. Cấu trúc mạng.....	18
2.2.2. Huấn luyện mạng.....	20
2.2.3. Tối ưu hóa.....	20
2.2.4. Tensor và phép toán.....	21
2.3. Mạng nơ-ron tích chập – CNN.....	21
2.3.1. Giới thiệu.....	21
2.3.2. Lớp tích chập.....	22
2.3.3. Lớp gộp - Pooling.....	24
2.3.4. Bước nhảy - stride.....	25
2.3.5. Hàm phi tuyến - ReLU.....	25
2.4. Mạng nơ-ron hồi tiếp - RNN.....	26
2.4.1. Tổng quan mạng nơ-ron hồi tiếp.....	26
2.4.2. LSTM.....	28



2.4.3.	RNN nâng cao.....	30
2.4.4.	Ứng dụng của RNN.....	30
2.5.	Mô hình Sinh - Generative Models.....	30
2.5.1.	Mạng sinh đối kháng – GAN.....	30
2.5.2.	Variational Autoencoder -VAEs.....	32
2.6.	Kỹ thuật tối ưu hóa và điều chuẩn.....	36
2.6.1.	Tối ưu hóa.....	36
2.6.2.	Điều chuẩn.....	37
2.7.	Kết luận.....	38
CHƯƠNG 3: ỨNG DỤNG HỌC SÂU CHO CHATBOT .....		40
3.1.	Tổng quan về Chatbot.....	40
3.2.	Các thành phần chính của Chatbot dựa trên học sâu.....	40
3.2.1.	Hiểu ngôn ngữ tự nhiên – NLU .....	40
3.2.2.	Quản lý hội thoại - Dialogue Management.....	40
3.2.3.	Tạo ngôn ngữ tự nhiên – NLG.....	41
3.3.	Các mô hình học sâu phổ biến trong Chatbot.....	41
3.3.1.	Transformer và các biến thể.....	41
3.3.2.	Recurrent Neural Networks (RNN) và LSTM.....	42
3.3.3.	Reinforcement Learning - RL.....	42
3.4.	Huấn luyện Chatbot sử dụng GPT-2 .....	42
3.4.1.	Mô tả.....	43
3.4.2.	Mã lập trình.....	43
KẾT LUẬN.....		48
TÀI LIỆU THAM KHẢO.....		49

## CHƯƠNG 1: TRÍ TUỆ NHÂN TẠO VÀ CÁC ỨNG DỤNG

### 1.1. 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 (AI) 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.

## **1.2.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.

## **1.3.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.

#### **1.4. 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.

### **1.5.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.

## **1.6.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.

## **1.7.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

### 2.1. 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.

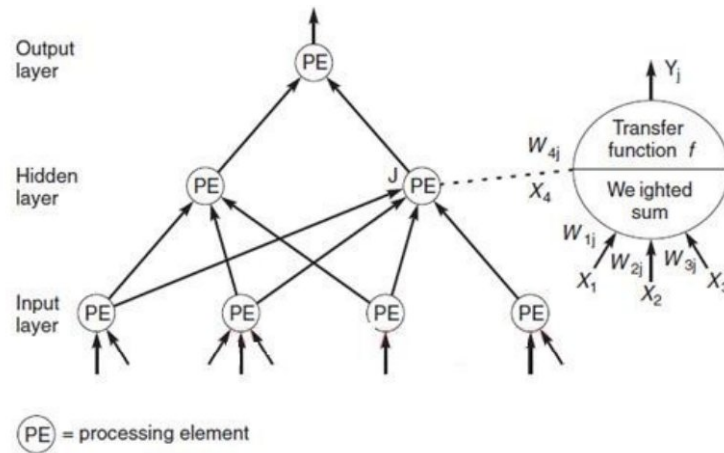
Những ví dụ này minh họa cách học sâu không chỉ giải quyết các bài toán phức tạp mà còn mang lại giá trị thực tiễn, từ cải thiện trải nghiệm người dùng đến tối ưu hóa quy trình kinh doanh.

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.

## **2.2. 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.

### *2.2.1. Cấu trúc mạng*



Hình 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.

### 2.2.2. 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.

### 2.2.3. 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.

#### 2.2.4. 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.

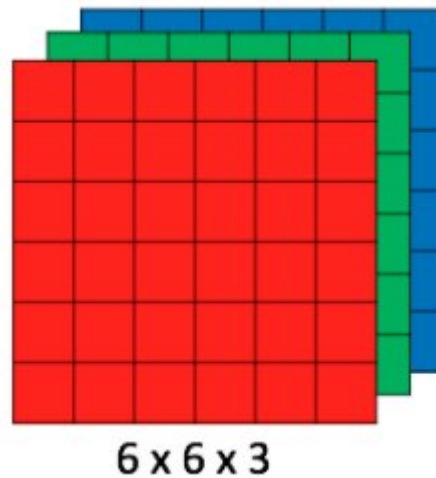
Trong chatbot, ví dụ, một lớp fully-connected sử dụng vector đầu vào, nhân với ma trận trọng số, cộng vector độ lệch, và áp dụng hàm kích hoạt để tạo phản hồi, tận dụng tensor để đảm bảo hiệu quả tính toán và độ chính xác của mô hình như Transformer hoặc GPT.

### 2.3. Mạng nơ-ron tích chập – CNN

#### 2.3.1. Giới thiệu

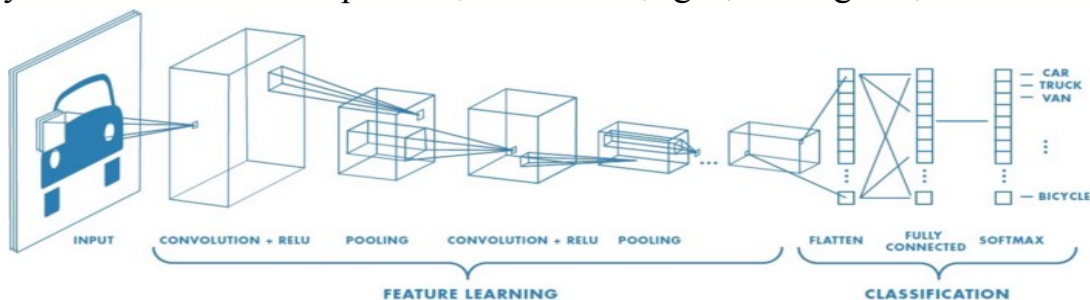
Trong mạng neural, mô hình mạng neural tích chập là 1 trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.

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  $H \times W \times D$  (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB  $6 \times 6 \times 3$  (3 ở đây là giá trị RGB).



Hình x: mảng ma trận RGB  $6 \times 6 \times 3$

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 (Kernels), 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

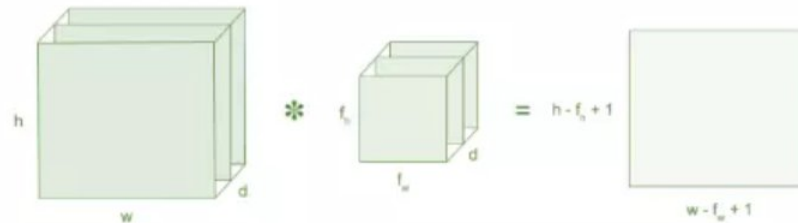


### 2.3.2. Lớp tích chập

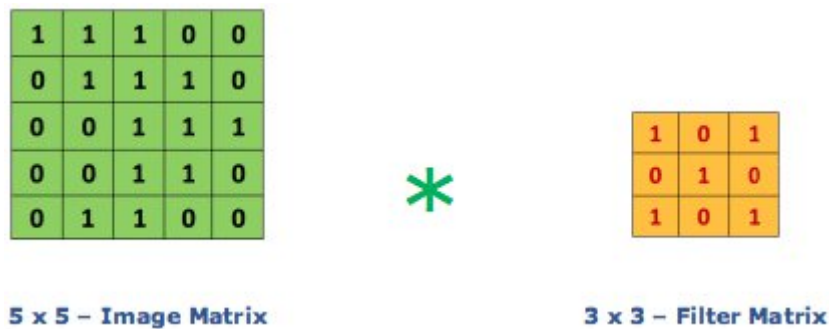
Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh

bằng cách sử dụng các ô vuông 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.

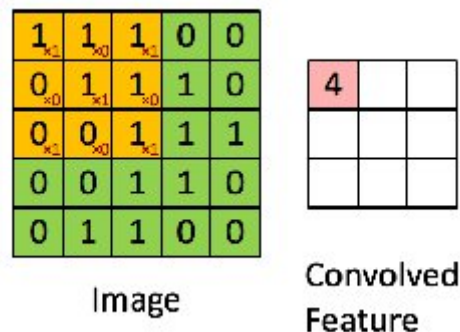
- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f<sub>h</sub> x f<sub>w</sub> x d)**
- Outputs a volume dimension **(h - f<sub>h</sub> + 1) x (w - f<sub>w</sub> + 1) x 1**



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.



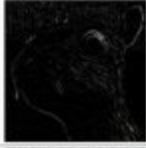


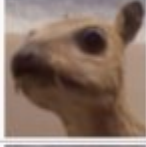



Sau đó, lớp tích chập của ma trận hình ảnh 5 x 5 nhân với ma trận bộ lọc 3 x 3 gọi là 'Feature Map' như hình bên dưới.



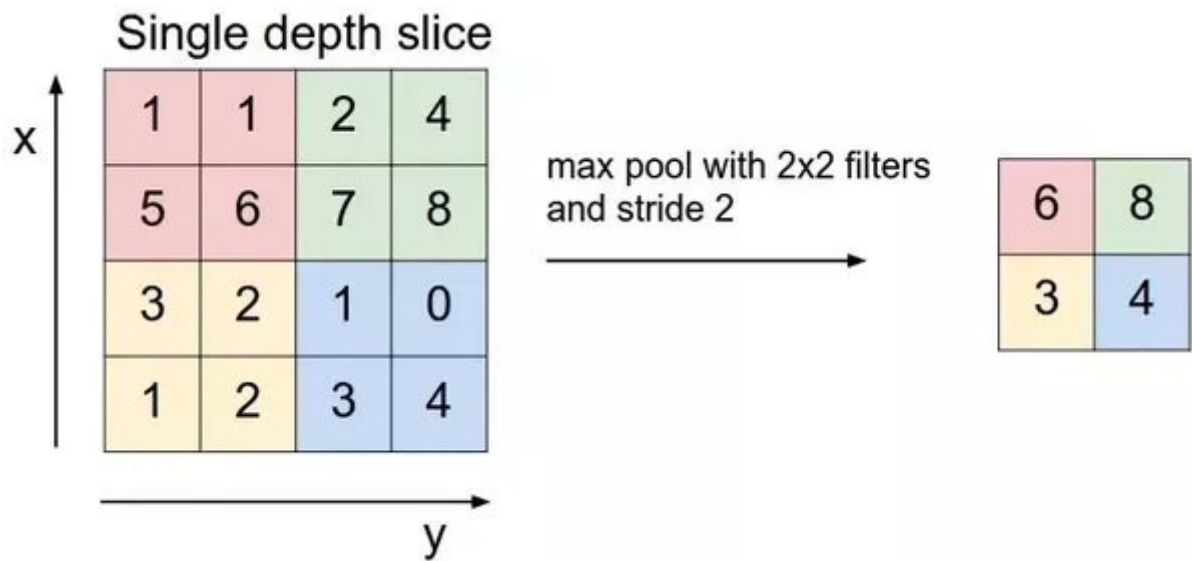
Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc.

Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

### 2.3.3. 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

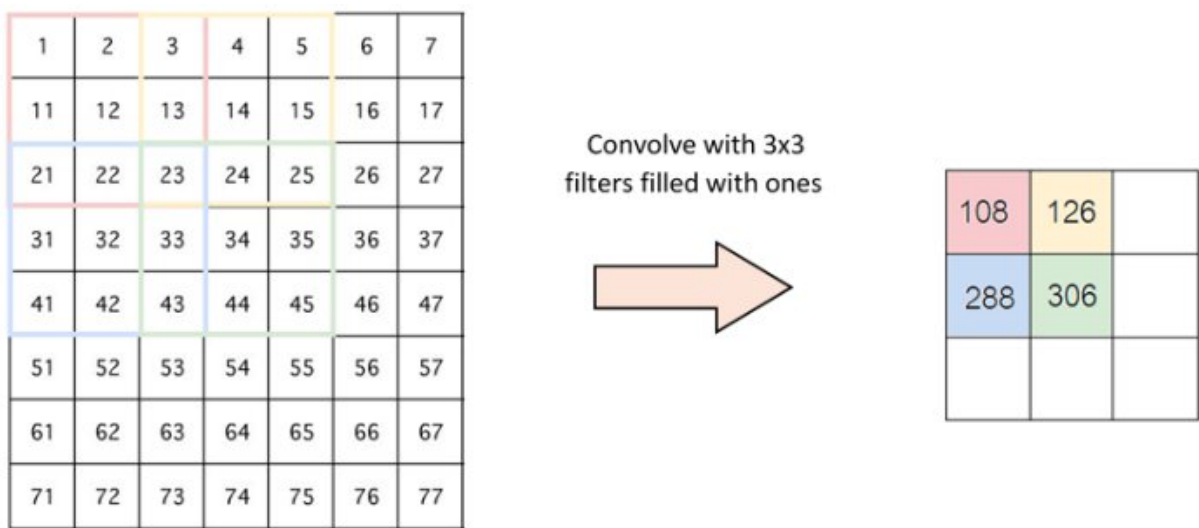


### Các loại gộp khác

- 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.

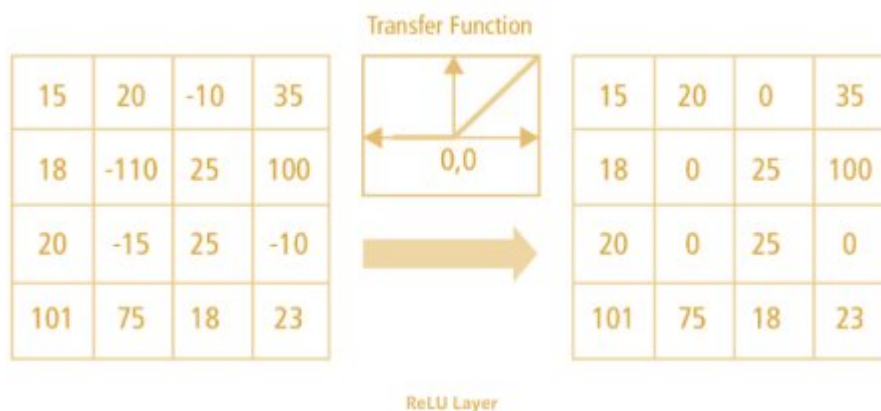
#### 2.3.4. Bước nhảy - stride

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.



#### 2.3.5. Hàm phi tuyến - ReLU

ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là:  $f(x) = \max(0, x)$ . 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.



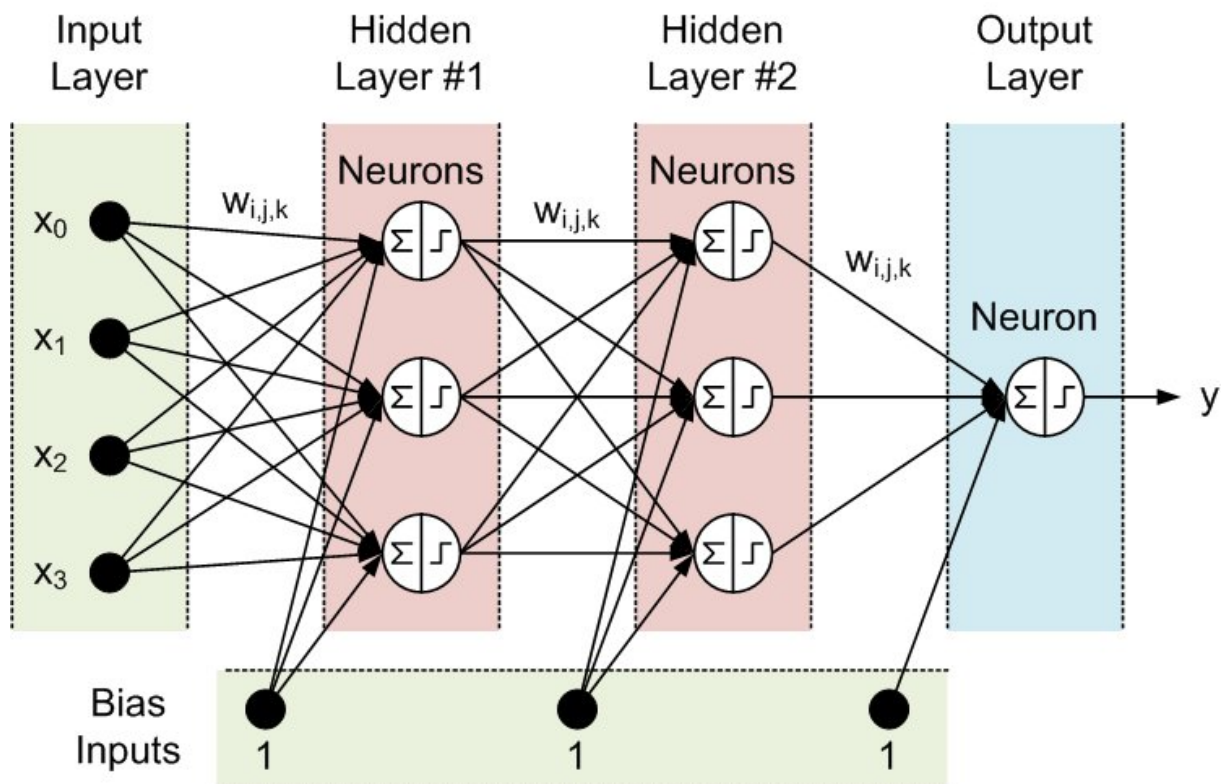
Có 1 số hàm phi tuyến khác như tanh, sigmoid cũng có thể được sử dụng thay cho ReLU. Hầu hết người ta thường dùng ReLU vì nó có hiệu suất tốt.

## 2.4. Mạng nơ-ron hồi tiếp - RNN

### 2.4.1. Tổng quan mạng nơ-ron hồi tiếp

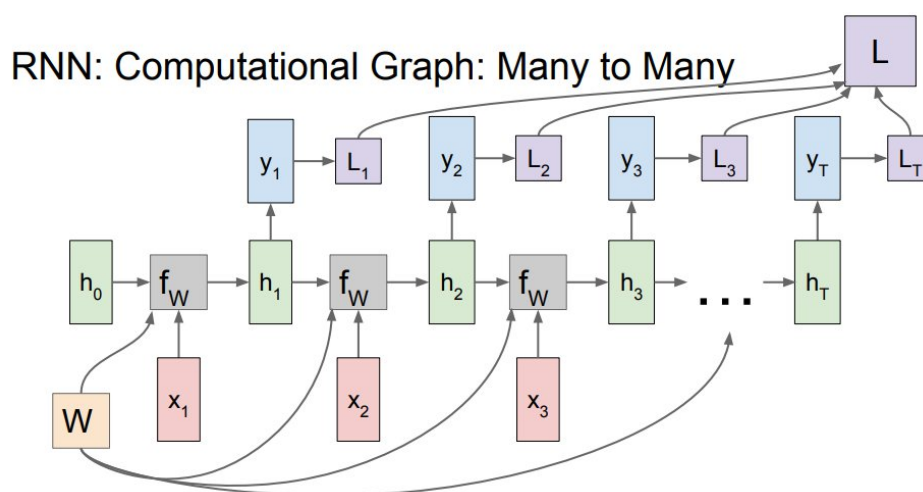
Để có thể hiểu rõ về RNN, trước tiên chúng ta cùng nhìn lại mô hình Neural Network dưới đây:





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 đứng trước nó.

Và như vậy RNN ra đời với ý tưởng chính là sử dụng một bộ nhớ để lưu lại thông tin 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.



Giải thích: Nếu như mạng Neural Network chỉ là input layer  $x$  đi qua hidden layer  $h$  và cho ra output layer  $y$  với đầy đủ kết nối giữa các layer thì trong RNN, các input  $x_t$  sẽ được kết hợp với hidden layer  $h_{t-1}$  bằng hàm  $f_W$  để tính toán ra hidden layer  $h_t$  hiện tại và output  $y_t$  sẽ được tính ra từ  $h_t$ ,  $W$  là tập các trọng số và nó được ở tất cả các cụm, các  $L_1, L_2, L_3, \dots, L_t$  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  $h_{t-1}$  tính ra  $h_t$  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:  $h_t = f_W(h_{t-1}, x_t)$

Hàm  $f_W$  chúng ta sẽ sử dụng hàm  $\tanh$ , công thức trên sẽ trở thành :

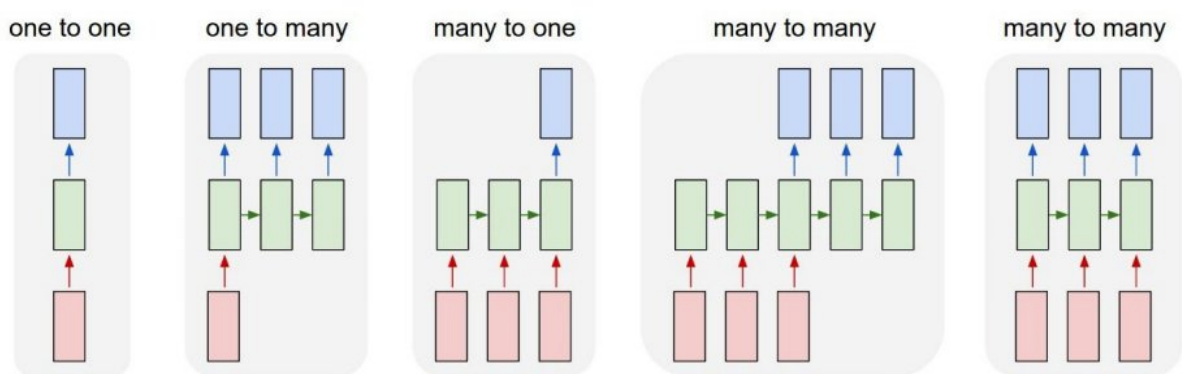
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Đến đây có 3 thứ mới xuất hiện:  $W_{xh}$

,  $W_{hh}$ ,  $W_{hy}$ . 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:  $W_{hh}$  kết hợp với "bộ nhớ trước"  $h_{t-1}$  và  $W_{xh}$  kết hợp với  $x_t$  để tính ra "bộ nhớ của bước hiện tại"  $h_t$  từ đó kết hợp với  $W_{hy}$  để tính ra  $y_t$

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

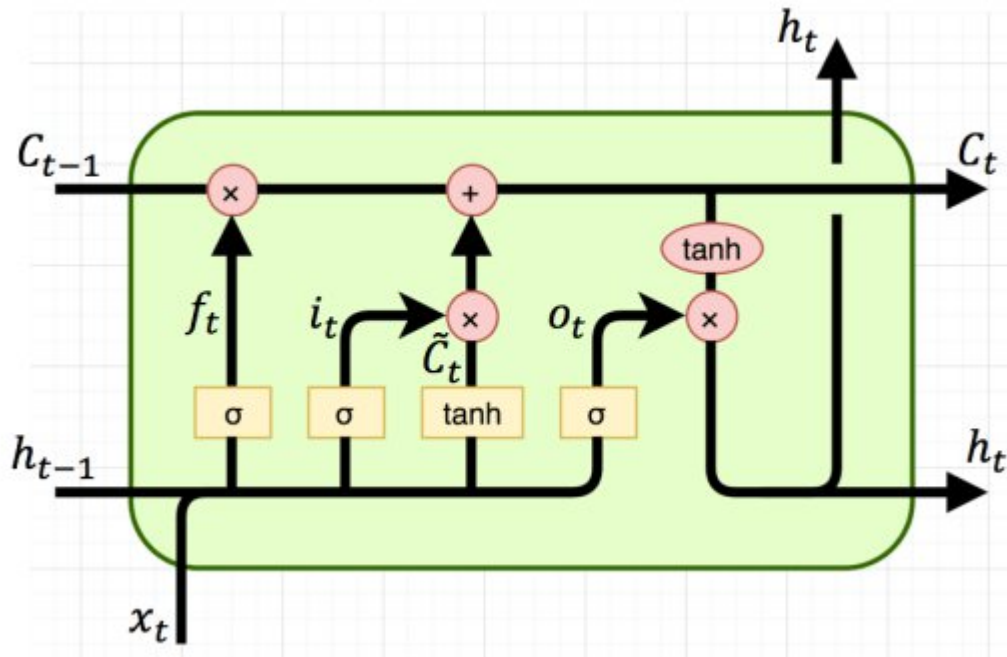


#### 2.4.2. LSTM

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.



Hình x: Mô hình LSTM

Ở state thứ  $t$  của mô hình LSTM:

- Output:  $c_t, h_t$  ta gọi  $c$  là cell state,  $h$  là hidden state.
- Input:  $c_{t-1}, h_{t-1}, x_t$ . Trong đó  $x_t$  là input ở state thứ  $t$  của model  $c_{t-1}, h_{t-1}$  là output của layer trước.

$f_t, i_t, o_t$  tương ứng với forget gate, input gate và output gate

- Forget gate:  $f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b_f)$
- Input gate:  $i_t = \sigma(U_i * x_t + W_i * h_{t-1} + b_i)$
- Output gate:  $o_t = \sigma(U_o * x_t + W_o * h_{t-1} + b_o)$

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

#### 2.4.3. 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.

#### 2.4.4. 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.

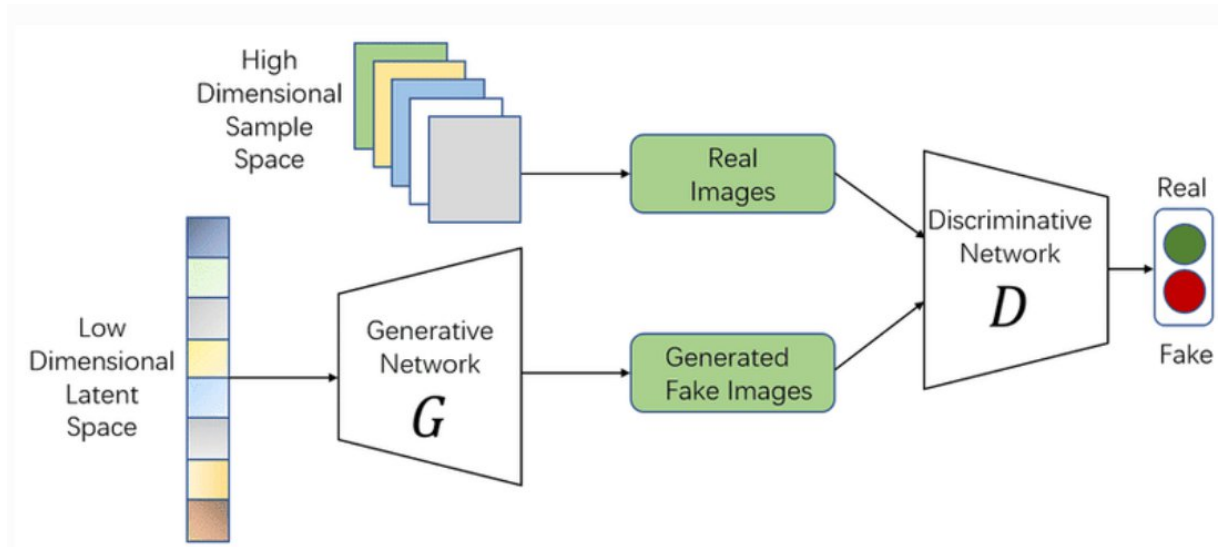
## 2.5. Mô hình Sinh - Generative Models

### 2.5.1. 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. 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.



Hình x: Mô hình 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.

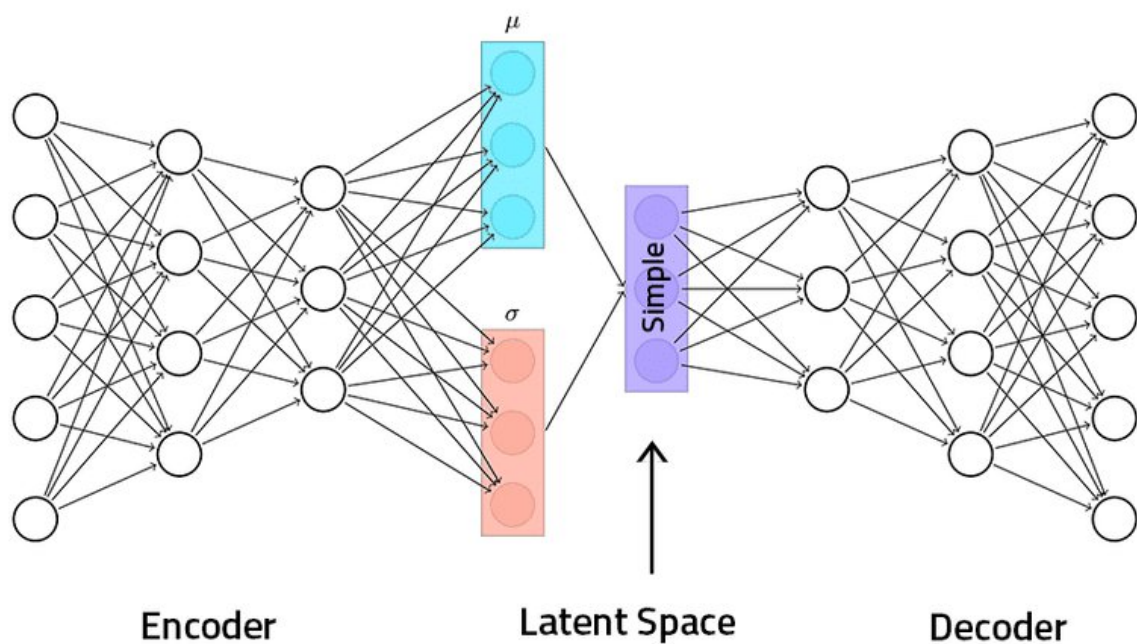
Mô hình học sâu này có khả năng tạo ra các hình ảnh chân dung con người với độ chân thực cao. Năm 2019, một kỹ sư của Uber đã sử dụng StyleGAN để tạo các khuôn mặt mới cho trang web “This Person Does Not Exist”. Wang đã bày tỏ sự kinh ngạc vì khả năng phân tích tất cả các đặc điểm có liên quan trên khuôn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

### 2.5.2. *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.



Hình x: 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 ( $\mu$ ) và độ lệch chuẩn ( $\sigma$ ). 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  $z = \mu + \sigma * \epsilon$ , trong đó  $\epsilon$  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.

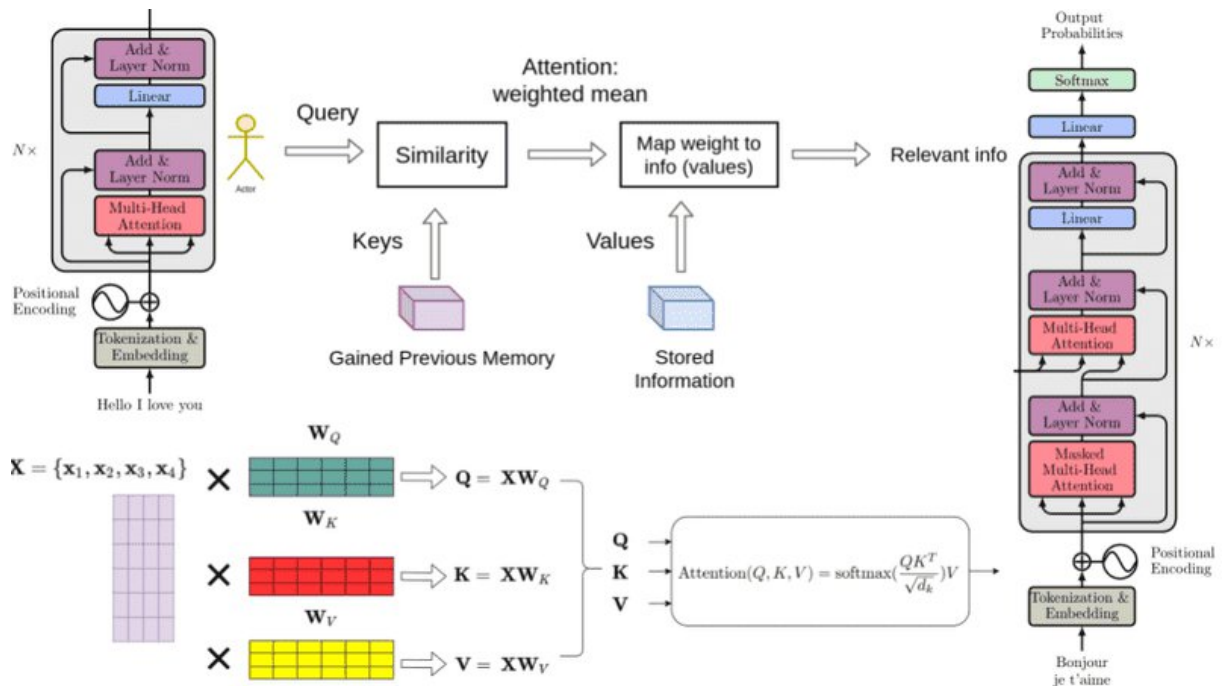
### 2.5.3. Mô hình Transformer

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 (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).

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). Hiện tại, thuật toán Transformer đã được ứng dụng rộng rãi học máy (Machine Learning) và nhiều lĩnh vực khác của trí tuệ nhân tạo (AI), như thị giác máy tính (Computer Vision), nhận dạng giọng nói và dự báo chuỗi thời gian.



Hình x: Mô hình transformer

Các thành phần cốt lõi của kiến trúc Transformer:

**Token:** Trong khi các ký tự (chữ cái, số hoặc dấu câu) là 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.

## **2.6. Kỹ thuật tối ưu hóa và điều chuẩn**

### *2.6.1. 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.

### 2.6.2. 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.

## 2.7. 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

### 3.1. Tổng quan về Chatbot

Chatbot là các hệ thống phần mềm mô phỏng giao tiếp với con người thông qua văn bản hoặc giọng nói. Chúng được ứng dụng rộng rãi trong các lĩnh vực như dịch vụ khách hàng, giáo dục, y tế, và thương mại điện tử. Học sâu với khả năng xử lý dữ liệu phức tạp thông qua các mạng nơ-ron nhân tạo, đã nâng cao đáng kể hiệu suất của chatbot, đặc biệt trong việc hiểu ngôn ngữ tự nhiên, tạo phản hồi, và học hỏi từ tương tác.

Học sâu đóng vai trò quan trọng trong việc phát triển các chatbot hiện đại nhờ khả năng xử lý ngôn ngữ tự nhiên (NLP) hiệu quả. Nhờ các mô hình học sâu, chatbot có thể hiểu ý định người dùng, phân tích ngữ cảnh và tạo ra các phản hồi tự nhiên, phù hợp. Các mô hình như Transformer, BERT, GPT và LSTM được sử dụng phổ biến, cho phép chatbot học hỏi từ khối lượng dữ liệu lớn, từ đó cải thiện hiệu suất theo thời gian. Những mô hình này giúp chatbot không chỉ xử lý các cuộc đối thoại phức tạp mà còn cung cấp trải nghiệm giao tiếp mượt mà, gần gũi với con người.

### 3.2. Các thành phần chính của Chatbot dựa trên học sâu

#### 3.2.1. Hiểu ngôn ngữ tự nhiên – NLU

Hiểu ngôn ngữ tự nhiên (NLU) là yếu tố cốt lõi giúp chatbot phân tích và hiểu câu hỏi hoặc yêu cầu của người dùng một cách chính xác. Các mô hình học sâu như BERT hoặc RoBERTa được sử dụng để trích xuất ý định (intent) và thực thể (entity) từ văn bản, nhờ khả năng phân tích ngữ cảnh hai chiều, nắm bắt mối quan hệ phức tạp giữa các từ.

Ví dụ, trong một chatbot thương mại điện tử, khi người dùng hỏi "Tôi muốn mua một chiếc áo đỏ giá dưới 500.000 VNĐ", NLU sẽ xác định ý định là "mua hàng" và các thực thể bao gồm "áo", "màu đỏ", "giá dưới 500.000 VNĐ". Trong khi BERT và RoBERTa vượt trội trong việc hiểu ngữ cảnh sâu, các mô hình như CNN hoặc LSTM thường được sử dụng cho các tác vụ phân loại ý định đơn giản hơn, cung cấp hiệu quả trong các tình huống ít phức tạp. Nhờ các mô hình này, chatbot có thể phản hồi chính xác và phù hợp với nhu cầu người dùng.

#### 3.2.2. Quản lý hội thoại - Dialogue Management

Quản lý hội thoại là thành phần cốt lõi trong chatbot, chịu trách nhiệm quyết định phản hồi phù hợp dựa trên trạng thái hội thoại và ý định người dùng. Các mô hình học sâu như Reinforcement Learning hoặc Transformer-based Dialogue

Models được sử dụng để tối ưu hóa luồng hội thoại, đảm bảo phản hồi nhất quán và phù hợp ngữ cảnh.

Ví dụ, khi người dùng hỏi tiếp "Có áo nào rẻ hơn không?" trong một chatbot thương mại điện tử, hệ thống sẽ dựa vào lịch sử hội thoại (về yêu cầu áo đỏ giá dưới 500.000 VNĐ) để đề xuất sản phẩm phù hợp, chẳng hạn như áo giá thấp hơn. Các mô hình Transformer tận dụng lịch sử hội thoại để tạo phản hồi tự nhiên, trong khi Deep Q-Networks giúp tối ưu hóa chiến lược hội thoại, đảm bảo chatbot đưa ra các quyết định hiệu quả và cải thiện trải nghiệm người dùng theo thời gian.

### 3.2.3. Tạo ngôn ngữ tự nhiên – NLG

Sinh ngôn ngữ tự nhiên là thành phần quan trọng trong chatbot, chịu trách nhiệm tạo ra các câu trả lời tự nhiên, dễ hiểu và phù hợp với ngữ cảnh. Các mô hình học sâu như GPT-3, T5 hoặc các biến thể Transformer được sử dụng để sinh văn bản, đảm bảo phản hồi mang tính giao tiếp và thân thiện.

Ví dụ, khi người dùng hỏi về áo đỏ giá dưới 500.000 VNĐ, chatbot có thể trả lời: "Chúng tôi có áo đỏ giá 450.000 VNĐ, bạn muốn xem chi tiết không?" thay vì một câu trả lời khô khan, cứng nhắc. Trong đó, GPT-3 nổi bật trong việc tạo văn bản sáng tạo và tự nhiên, trong khi T5 được tối ưu cho các nhiệm vụ cụ thể, cho phép chatbot sinh ra phản hồi chính xác, phù hợp với yêu cầu của người dùng và ngữ cảnh hội thoại.

## 3.3. Các mô hình học sâu phổ biến trong Chatbot

### 3.3.1. Transformer và các biến thể

Transformer là nền tảng cốt lõi của các mô hình học sâu hiện đại như BERT, GPT, và T5, tận dụng cơ chế Attention để xử lý dữ liệu theo ngữ cảnh, cho phép hiểu và tạo văn bản hiệu quả. Cơ chế Attention giúp mô hình tập trung vào các phần quan trọng của dữ liệu đầu vào, chẳng hạn như các từ liên quan trong một câu, thay vì xử lý tuần tự như RNN. Trong chatbot, Transformer cho phép xử lý các câu hỏi phức tạp và tạo phản hồi mạch lạc, tự nhiên.

Ví dụ, một chatbot giáo dục sử dụng GPT có thể trả lời câu hỏi "Giải thích thuyết tương đối" bằng cách sinh ra câu trả lời chi tiết, chẳng hạn: "Thuyết tương đối của Einstein bao gồm thuyết tương đối hẹp và rộng. Tương đối hẹp giải thích rằng tốc độ ánh sáng là hằng số và thời gian có thể giãn nở tùy theo vận tốc, trong khi tương đối rộng liên quan đến lực hấp dẫn và độ cong không-thời gian..." Phản hồi này được tạo nhờ khả năng của Transformer trong việc nắm bắt ngữ cảnh dài và kết hợp thông tin từ dữ liệu huấn luyện.

Tương tự, trong ngữ cảnh thương mại điện tử, Transformer giúp chatbot trả lời câu hỏi như "Tôi muốn mua áo đỏ giá dưới 500.000 VNĐ" bằng cách phân tích ý định, thực thể và lịch sử hội thoại, tạo phản hồi như "Chúng tôi có áo đỏ giá 450.000 VNĐ, bạn muốn xem chi tiết không?" Các kỹ thuật như dropout, chuẩn hóa lô, và tối ưu hóa bằng Adam đảm bảo Transformer hoạt động hiệu quả, giảm quá khớp và tăng khả năng tổng quát hóa.

### 3.3.2. Recurrent Neural Networks (RNN) và LSTM

Mạng nơ-ron hồi tiếp (RNN) và biến thể Long Short-Term Memory (LSTM) phù hợp với dữ liệu tuần tự như hội thoại, nhờ khả năng xử lý thông tin theo thời gian. Tuy nhiên, chúng kém hiệu quả hơn Transformer trong việc nắm bắt ngữ cảnh dài do vấn đề tiêu biến gradient và giới hạn về bộ nhớ. RNN và LSTM thường được sử dụng trong các chatbot đơn giản, chẳng hạn như chatbot trả lời câu hỏi FAQ. Ví dụ, một chatbot ngân hàng có thể sử dụng LSTM để trả lời các câu hỏi định dạng như "Lãi suất vay hiện tại là bao nhiêu?" với phản hồi chính xác như "Lãi suất vay hiện tại là 7% mỗi năm." Các mô hình này xử lý tốt các câu hỏi ngắn, có cấu trúc, nhưng không hiệu quả bằng Transformer khi cần xử lý các câu hỏi phức tạp hoặc hội thoại dài.

### 3.3.3. Reinforcement Learning - RL

RL tối ưu hóa chiến lược hội thoại bằng cách học từ phản hồi của người dùng, đặc biệt trong các chatbot cần thích nghi với hành vi khách hàng. Trong RL, chatbot được coi là một tác nhân (agent) tương tác với môi trường (người dùng), nhận thưởng (reward) dựa trên chất lượng phản hồi. Ứng dụng nổi bật là trong chatbot thương mại điện tử, nơi RL giúp học cách đề xuất sản phẩm phù hợp.

Ví dụ, nếu khách hàng hỏi "Tôi muốn mua áo đỏ giá dưới 500.000 VNĐ" và từ chối gợi ý ban đầu như "Áo đỏ giá 450.000 VNĐ", chatbot sử dụng RL (như Deep Q-Networks) để điều chỉnh, đề xuất sản phẩm khác như "Áo đỏ giá 400.000 VNĐ với giao hàng miễn phí." RL tối ưu hóa chiến lược bằng cách tối đa hóa thưởng tích lũy, dựa trên phản ứng tích cực hoặc tiêu cực của người dùng. Kết hợp với các kỹ thuật như dropout và hàm mất mát cross-entropy, RL giúp chatbot cải thiện khả năng phản hồi tự nhiên và phù hợp ngữ cảnh theo thời gian, dù không mạnh bằng Transformer trong xử lý ngữ cảnh dài.

## 3.4. Huấn luyện Chatbot sử dụng GPT-2

GPT-2 là một mô hình ngôn ngữ tự nhiên tiên tiến do OpenAI phát triển, được giới thiệu vào năm 2019. Dựa trên kiến trúc Transformer, GPT-2 được huấn luyện trên một tập dữ liệu văn bản khổng lồ từ internet, cho phép nó tạo ra văn bản giống con người với khả năng hiểu ngữ cảnh và sinh ra các câu trả lời mạch lạc. Mô hình

này có nhiều kích thước khác nhau (từ 124 triệu đến 1,5 tỷ tham số), phù hợp cho các ứng dụng như chatbot, tạo nội dung, và dịch thuật. Điểm nổi bật của GPT-2 là khả năng học không giám sát và fine-tuning dễ dàng trên các tập dữ liệu cụ thể, như sử dụng thư viện Transformers của Hugging Face để huấn luyện chatbot

### 3.4.1. Mô tả

Mô hình được sử dụng trong dự án này là GPT-2 phiên bản nhỏ, một mô hình ngôn ngữ mạnh mẽ được thiết kế để tạo văn bản dựa trên đầu vào. GPT-2 được chọn nhờ khả năng xử lý ngôn ngữ tự nhiên hiệu quả, phù hợp cho các tác vụ sinh văn bản như trả lời câu hỏi hoặc tạo hội thoại.

Tập dữ liệu huấn luyện bao gồm các đoạn hội thoại đơn giản, được lưu trữ dưới dạng file JSON. Dữ liệu này chứa các cặp câu hỏi và trả lời từ hai người tham gia, được ký hiệu là participant1 và participant2. Mỗi đoạn hội thoại được định dạng với các token đặc biệt như `<startofstring>`, `<bot>`: và `<endofstring>` để hỗ trợ quá trình huấn luyện và suy luận. Ví dụ, một câu hỏi như "What do you do?" sẽ được ghép với câu trả lời tương ứng để tạo thành chuỗi văn bản hoàn chỉnh.

Mục tiêu của dự án là huấn luyện mô hình GPT-2 để trả lời các câu hỏi cơ bản, chẳng hạn như các câu hỏi thường gặp (FAQ) trong dịch vụ khách hàng. Bằng cách sử dụng tập dữ liệu hội thoại, mô hình sẽ học cách tạo ra các câu trả lời phù hợp và tự nhiên, đáp ứng nhu cầu hỗ trợ khách hàng một cách hiệu quả.

### 3.4.2. Mã lập trình

```
!pip install transformers
from torch.utils.data import Dataset
import json
class ChatData(Dataset):
    def __init__(self, path: str, tokenizer):
        self.data = json.load(open(path, "r"))
        self.X = []
        for i in self.data:
            for j in i['dialog']:
                self.X.append(j['text'])
        for idx, i in enumerate(self.X):
            try:
                self.X[idx] = "<startofstring> "+i+"<bot>: "+self.X[idx+1]+ " <endofstring>"
            except:
                break
        self.X = self.X[:1000]
        print(self.X[0])

        self.X_encoded = tokenizer(self.X, max_length = 40, truncation = True, padding = "max_length", return_tensors="pt")
        self.input_ids = self.X_encoded['input_ids']
        self.attention_mask = self.X_encoded['attention_mask']

    def __len__(self):
        return len(self.X)
    def __getitem__(self, idx):
        return (self.input_ids[idx], self.attention_mask[idx])
```

- Giải thích:

Lớp `ChatData` được định nghĩa để xử lý dữ liệu hội thoại dưới dạng JSON, kế thừa từ lớp `Dataset` trong PyTorch, nhằm chuẩn bị dữ liệu cho việc huấn luyện mô hình ngôn ngữ như GPT-2. Trong phương thức khởi tạo `__init__`, lớp nhận vào hai tham số: `path` (đường dẫn đến file JSON chứa dữ liệu hội thoại) và `tokenizer` (đối tượng `tokenizer` để mã hóa văn bản). Phương thức này mở file JSON bằng `json.load(open(path, 'r'))` và lưu dữ liệu vào thuộc tính `self.X`. Sau đó, nó khởi tạo một danh sách `self.X` rỗng để lưu trữ các chuỗi hội thoại đã được xử lý, và lưu `tokenizer` vào `self.tokenizer` để sử dụng sau này.

Tiếp theo, phương thức `__init__` xử lý dữ liệu hội thoại từ JSON. Nó lặp qua từng đoạn hội thoại trong `self.data['dialog']` để xây dựng chuỗi văn bản. Với mỗi đoạn hội thoại, code sử dụng vòng lặp để duyệt qua từng lượt nói, lấy nội dung từ `self.X[idx]['text']`. Chuỗi văn bản được định dạng bằng cách thêm các token đặc biệt: `<startofstring>` ở đầu, `<bot>`: giữa các lượt nói, và `<endofstring>` ở cuối. Nếu có lỗi trong quá trình xử lý (ví dụ, chỉ số `idx+1` vượt quá độ dài danh sách), vòng lặp sẽ dừng lại bằng câu lệnh `break`. Sau khi xử lý, danh sách `self.X` chứa 1000 chuỗi văn bản đầu tiên (nếu dữ liệu lớn hơn) để giới hạn kích thước tập dữ liệu. Cuối cùng, mỗi chuỗi văn bản được mã hóa bằng `tokenizer` với các tham số như `max_length=40`, `truncation=True`, và `padding='max_length'` để tạo ra `input_ids` và `attention_mask`, được lưu vào `self.X_encoded`.

Phương thức `__len__` được định nghĩa để trả về độ dài của tập dữ liệu, tức là số lượng chuỗi văn bản trong `self.X`. Điều này giúp PyTorch biết kích thước của tập dữ liệu khi sử dụng trong `DataLoader`. Phương thức này đơn giản chỉ cần trả về giá trị `len(self.X)`.

Phương thức `__getitem__` được triển khai để truy xuất một mẫu dữ liệu tại chỉ số `idx`. Phương thức này lấy `input_ids` và `attention_mask` tương ứng với chỉ số `idx` từ `self.X_encoded`, sau đó trả về một bộ ba giá trị: chuỗi văn bản gốc (`self.X[idx]`), tensor `input_ids`, và tensor `attention_mask`. Các tensor này được sử dụng trực tiếp trong quá trình huấn luyện mô hình, đảm bảo dữ liệu đã được mã hóa và định dạng đúng.



```

from transformers import GPT2LMHeadModel, GPT2Tokenizer
from torch.optim import Adam
from torch.utils.data import DataLoader
import tqdm
import torch

def train(chatData, model, optim):
    epochs = 10

    for i in tqdm.tqdm(range(epochs)):
        for X, a in chatData:
            X = X.to(device)
            a = a.to(device)
            optim.zero_grad()
            model.to(device)
            loss = model(X, attention_mask=a, labels=X).loss
            loss.backward()
            optim.step()
            torch.save(model.state_dict(), "model_state.pt")

def infer(input):
    input = "<startofstring> " + input + "<bot>:"
    input = tokenizer(input, return_tensors="pt")
    X = input['input_ids'].to(device)
    a = input['attention_mask'].to(device)
    output = model.generate(X, attention_mask=a)
    output = tokenizer.decode(output[0])
    return output

tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
tokenizer.add_special_tokens({"pad_token": "<pad>", "bos_token": "<startofstring>",
                             "eos_token": "<endofstring>"})

tokenizer.add_tokens("<bot>:")
device = "cuda" if torch.cuda.is_available() else "mps" if torch.backends.mps.is_available() else "cpu"

model = GPT2LMHeadModel.from_pretrained("gpt2")
model.resize_token_embeddings(len(tokenizer))

chatData = ChatData("/content/chat_data.json", tokenizer)
chatData = DataLoader(chatData, batch_size=64)
model.train()

optim = Adam(model.parameters())

```

## - Giải thích:

Code bắt đầu với việc nhập các thư viện cần thiết để huấn luyện mô hình GPT-2. Các thư viện bao gồm *GPT2LMHeadModel* và *GPT2Tokenizer* từ *transformers* để làm việc với mô hình và mã hóa văn bản, *Adam* từ *torch.optim* để tối ưu hóa mô hình, và *DataLoader* từ *torch.utils.data* để xử lý dữ liệu theo *batch*. Ngoài ra, thư viện *tqdm* được nhập để hiển thị thanh tiến trình trong quá trình huấn luyện, và *torch* được sử dụng để làm việc với tensor và thiết bị (CPU/GPU).

Hàm *train* được định nghĩa để thực hiện quá trình huấn luyện mô hình. Hàm này nhận vào ba tham số: *chatData* (dữ liệu hội thoại), *model* (mô hình GPT-2), và *optim* (bộ tối ưu). Quá trình huấn luyện diễn ra trong *10 epoch*, với mỗi *epoch*, code lặp qua từng batch dữ liệu trong *chatData*. Đối với mỗi batch, dữ liệu *X* (*input\_ids*) và *a* (*attention\_mask*) được chuyển sang thiết bị *device* (có thể là CPU, CUDA, hoặc MPS). Bộ tối ưu được làm sạch *gradient* bằng *optim.zero\_grad()*, sau đó mô hình tính toán loss bằng cách truyền *X*, *attention\_mask=a*, và *labels=X* vào mô hình. Loss được tính toán ngược bằng *loss.backward()*, và tham số mô hình được cập nhật qua *optim.step()*. Cuối mỗi *epoch*, trạng thái mô hình được lưu vào file *model\_state.pt* bằng *torch.save*.

Hàm *infer* được thiết kế để suy luận và tạo văn bản từ đầu vào người dùng. Hàm nhận một chuỗi văn bản đầu vào, thêm các token đặc biệt *<startofstring>* và *<bot>:* để định dạng đúng theo cách mô hình đã được huấn luyện. Chuỗi này sau đó được mã hóa bằng *tokenizer* với *return\_tensors="pt"* để tạo *tensor*. Các tensor đầu vào (*input\_ids* và *attention\_mask*) được chuyển sang *device* để đảm bảo đồng bộ với thiết bị của mô hình. Mô hình tạo văn bản bằng phương thức *generate*, và kết quả được giải mã bằng *tokenizer.decode*, bỏ qua các token đặc biệt (*skip\_special\_tokens=True*). Văn bản được tạo ra cuối cùng được trả về.

Phần khởi tạo mô hình và dữ liệu bắt đầu bằng việc thiết lập *tokenizer*. *GPT2Tokenizer* được tải từ mô hình pretrained "*gpt2*", sau đó được bổ sung các token đặc biệt như `<pad>`, `<startofstring>`, `<endofstring>`, và `<bot>`: để hỗ trợ định dạng dữ liệu hội thoại. Thiết bị device được xác định dựa trên phần cứng khả dụng, ưu tiên CUDA (GPU), sau đó là MPS (cho Mac), và cuối cùng là CPU. Mô hình *GPT2LMHeadModel* cũng được tải từ "*gpt2*", sau đó được điều chỉnh kích thước embedding để khớp với *tokenizer* bằng *resize\_token\_embeddings*. Mô hình được chuyển sang *device* để đảm bảo tính toán diễn ra trên thiết bị đã chọn.

Cuối cùng, dữ liệu hội thoại được chuẩn bị bằng cách sử dụng lớp *ChatData*, với đường dẫn file JSON là `/content/chat_data.json` và *tokenizer* đã khởi tạo. Dữ liệu này sau đó được đưa vào *DataLoader* với *batch\_size=64* để xử lý theo *batch*. Bộ tối ưu *Adam* được khởi tạo với tham số của mô hình, sau đó hàm *train* được gọi để bắt đầu quá trình huấn luyện. Code đảm bảo rằng tất cả *tensor* và mô hình đều được đặt trên cùng một thiết bị để tránh lỗi xung đột thiết bị.

```
[12] train(chatData, model, optim)
0%|          | 0/10 [00:00<?, ?it/s]`loss_type=None` was set in the config but it is unrecognised.Using the default loss: `ForCausalMLLoss`.
100%|██████████| 10/10 [02:28<00:00, 14.86s/it]
```

```
while True:
    user_input=input()
    print(infer(user_input))

... helo
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
<startofstring> helo <bot>: i am not sure what that is . i am sure you are a great person . <endofstring>
nice to meet you
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
<startofstring> nice to meet you <bot>: How's it going? <endofstring><pad><pad><pad><pad><pad><pad><pad><pad><pad>
1 + 1 = ?
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
<startofstring> 1 + 1 = ? <bot>: Yes, it is, it is a great day <endofstring><pad><pad><pad><pad><pad><pad><pad><pad><pad>
no, you know about math
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
<startofstring> no, you know about math <bot>: what is going on <endofstring><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
hmm, i hate you
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
<startofstring> hmm, i hate you <bot>: I'm fine <endofstring><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
how are you
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
<startofstring> how are you <bot>: What are you doing? <endofstring><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
hmm, fuck you
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
<startofstring> hmm, fuck you <bot>: hello there! <endofstring><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>

```

Code bắt đầu với một vòng lặp vô hạn *while True*, được thiết kế để liên tục nhận đầu vào từ người dùng và tạo ra phản hồi từ mô hình. Trong vòng lặp này, hàm *input()* được sử dụng để lấy chuỗi văn bản từ người dùng, sau đó được truyền vào hàm *infer()* để xử lý. Kết quả từ *infer()* được in ra màn hình, cho phép mô hình tương tác với người dùng theo thời gian thực. Vòng lặp này sẽ tiếp tục chạy cho đến khi chương trình bị dừng thủ công, tạo ra một giao diện đơn giản để kiểm tra khả năng sinh văn bản của mô hình.

Hàm *infer()* nhận một chuỗi văn bản đầu vào và thực hiện quá trình suy luận để tạo ra phản hồi. Đầu tiên, chuỗi đầu vào được định dạng bằng cách thêm các

token đặc biệt `<startofstring>` và `<bot>`., sau đó được mã hóa bằng `tokenizer` để tạo `tensor` (`input_ids` và `attention_mask`). Các `tensor` này được chuyển sang thiết bị device (như CPU hoặc GPU) để đảm bảo tương thích với mô hình. Mô hình sau đó sử dụng phương thức `generate()` để tạo văn bản dựa trên đầu vào, với các tham số như `max_length` để giới hạn độ dài đầu ra. Kết quả được giải mã bằng `tokenizer.decode()` và trả về, bỏ qua các token đặc biệt để tạo ra văn bản dễ đọc.

Kết quả từ quá trình suy luận cho thấy mô hình đã được huấn luyện với dữ liệu hội thoại và có thể tạo ra các phản hồi cơ bản. Khi người dùng nhập "hello", mô hình trả về *"I am not sure what that is. I am sure you are a great person."* kèm theo các token `<startofstring>`, `<bot>`., và `<endofstring>`, cùng với các token padding (`<pad>`). Khi người dùng hỏi *"nice to meet you"* hoặc *"what are you doing"*, mô hình sinh ra các câu như *"Yes, it is. It is a great day"* hoặc *"I have no idea what that is"*, cho thấy mô hình đang cố gắng tạo câu trả lời dựa trên dữ liệu huấn luyện, nhưng chất lượng phản hồi còn hạn chế do dữ liệu huấn luyện có thể không đủ phong phú. Các câu hỏi như *"do you know math"* cũng được trả lời với các câu chung chung, kèm theo padding, phản ánh việc mô hình chưa được tối ưu hóa hoàn toàn..

## KẾT LUẬN

Dựa trên các phân tích và triển khai mã nguồn trong chương này, có thể rút ra một số kết luận quan trọng. Việc sử dụng mô hình GPT-2 phiên bản nhỏ để xây dựng chatbot đã chứng minh tiềm năng của học sâu trong việc tạo ra các hệ thống giao tiếp tự nhiên và linh hoạt. Ví dụ mã lập trình cung cấp một nền tảng cơ bản, minh họa cách huấn luyện và suy luận với dữ liệu hội thoại, tuy nhiên hiệu suất của chatbot phụ thuộc lớn vào chất lượng và quy mô của tập dữ liệu. Các lưu ý về yêu cầu phần cứng, như ưu tiên sử dụng GPU để tăng tốc độ huấn luyện, cũng nhấn mạnh tầm quan trọng của tài nguyên tính toán trong việc phát triển mô hình. Hơn nữa, việc tinh chỉnh mô hình thông qua điều chỉnh siêu tham số hoặc kỹ thuật như LoRA mở ra cơ hội cải thiện hiệu quả, đặc biệt khi xử lý các tập dữ liệu thực tế lớn hơn và đa dạng hơn.

Nhìn chung, học sâu, với các mô hình như Transformer và GPT, đã đặt nền móng cho sự phát triển vượt bậc của chatbot, cho phép chúng hiểu và phản hồi ngôn ngữ một cách thông minh hơn. Tuy nhiên, thách thức vẫn còn tồn tại, đặc biệt trong việc thích nghi với ngôn ngữ địa phương và tối ưu hóa hiệu suất trên các thiết bị khác nhau. Kết luận này không chỉ khẳng định giá trị của các ví dụ thực hành trong chương, mà còn khuyến khích tiếp tục nghiên cứu và phát triển để tạo ra các chatbot tiên tiến hơn, đáp ứng tốt hơn nhu cầu của người dùng trong tương lai.

## 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.