

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC SAO ĐỎ



ĐỒ ÁN TỐT NGHIỆP

TÊN ĐỀ TÀI: **ỨNG DỤNG DEEPMLEARNING XÂY DỰNG WEBSITE CHUYỂN VĂN BẢN THÀNH HÌNH ẢNH**

Ngành: Công Nghệ Thông Tin

Sinh viên thực hiện: TRƯỜNG VĂN TUẤN
NGÔ THỊ HẢI YÊN

Lớp: DK10 – CNTT

Giảng viên hướng dẫn: HOÀNG THỊ AN

Hải Dương - 2023

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin chung:

Họ và tên nhóm sinh viên thực hiện:

TT	Mã sinh viên	Họ và tên sinh viên	Lớp, khóa
1	1900671	Trương Văn Tuấn	DK10-CNTT
2	1900506	Ngô Thị Hải Yên	DK10-CNTT

Ngành: Công nghệ thông tin

Trình độ đào tạo: Đại học

Thời gian thực hiện đồ án tốt/khoa luận: từ ngày 01/08/2023 đến 01/12/2023

Giảng viên hướng dẫn: Hoàng Thị An

Học hàm, học vị: Thạc sĩ

2. Tên đề tài:

Ứng dụng deeplearning xây dựng website chuyển văn bản thành hình ảnh

3. Điều kiện cho trước:

- Tài liệu về xử lý ngôn ngữ tự nhiên.
- Tài liệu về thư viện Gradio, Ngrok.
- Tài liệu về thư viện Pytorch, Stable Diffusion.

4. Nhiệm vụ chính của đồ án/khoa luận:

- Nghiên cứu kỹ thuật xử lý ngôn ngữ tự nhiên (NLP).
- Nghiên cứu thư viện Gradio, Ngrok thiết kế website.
- Nghiên cứu thư viện Pytorch, Stable Diffusion chuyển văn bản thành hình ảnh.
- Xây dựng website chuyển văn bản thành hình ảnh theo mô tả.

5. Sản phẩm:

- Báo cáo thuyết minh đồ án: 02 bản bìa cứng, 05 bản bìa mềm.
- Website chuyển văn bản thành hình ảnh theo mô tả: Người dùng nhập văn bản mô tả, website trả về ảnh theo mô tả.
- Đĩa CD (phần mềm, nội dung đồ án, slide thuyết trình).

Hải Dương, ngày tháng năm 2023

TL. HIỆU TRƯỞNG
PHÓ TRƯỞNG KHOA
(Ký, ghi rõ họ và tên và đóng dấu)

GIẢNG VIÊN HƯỚNG DẪN
(Ký, ghi rõ họ và tên)

Phạm Văn Kiên

Hoàng Thị An

LỜI CAM ĐOAN

Tôi xin cam đoan các kết quả đưa ra trong đồ án/khóa luận tốt nghiệp này là các kết quả thu được trong quá trình nghiên cứu, thực nghiệm của tôi dưới sự hướng dẫn của Giảng viên ThS. Hoàng Thị An, không sao chép bát kỳ kết quả nghiên cứu nào của các tác giả khác.

Nội dung nghiên cứu có tham khảo và sử dụng một số thông tin, tài liệu từ các nguồn tài liệu đã được liệt kê trong danh mục các tài liệu tham khảo.

Nếu sai tôi xin chịu mọi hình thức kỷ luật theo quy định.

Hải Dương, ngày tháng..... năm.....

Sinh viên thực hiện
(Ký, ghi rõ họ và tên)

Trương Văn Tuấn
Ngô Thị Hải Yến

Đại học Sao Đỏ

MỤC LỤC

MỞ ĐẦU	1
Chương 1. CƠ SỞ LÝ THUYẾT	3
1.1. Deep Learning	3
1.1.1. Khái niệm.....	3
1.1.2. Cách thức hoạt động của Deep Learning.....	3
1.1.3. Ứng dụng của Deep Learning.....	4
1.2. Tổng quan Website AI.....	7
1.2.1. Khái niệm.....	7
1.2.2. Ứng dụng của Website Ai.....	7
1.3. Tổng quan về mạng nơ-ron [4]	8
1.3.1. Khái niệm.....	8
1.3.2. Các thành phần cơ bản.....	9
1.3.3. Kiến trúc mạng	10
1.3.4. Huấn luyện mạng	12
1.3.5. Mạng nơ-ron tích chập.....	13
Chương 2. STABLE DIFFUSION	20
2.1. Định nghĩa [1]	20
2.2. Kiến trúc của Stable Diffusion	20
2.3. Nhiễu.....	21
2.3.1. Các loại nhiễu ảnh	22
2.3.2. Các phương pháp khử nhiễu	22
2.4. Cách hoạt động	22
2.5. Các thành phần của Stable Diffusion.....	25
2.6. Mạng UNET [2].....	28
2.6.1. Giới thiệu về mạng UNET	28
2.6.2. Ý tưởng đằng sau UNET	28
2.6.3. Kiến trúc UNET.....	29
2.6.4. Tính toán loss, IoU và DSC	30
Chương 3. XÂY DỰNG CHƯƠNG TRÌNH	32
3.1. Mô tả bài toán	32
3.2. Dữ liệu đào tạo.....	32
3.3. Công cụ sử dụng	34

3.3.1. Sử dụng Visual Studio Code với Github	34
3.3.2. Sử dụng Google Colab.....	36
3.4. Thiết kế ứng dụng	38
3.4.1. Thiết kế giao diện	38
3.4.2. Thiết kế các chức năng chính	41
3.5. Thiết lập chương trình trên Google Colab	46
3.5.1. Thiết lập hệ thống	46
3.5.2. Thiết lập môi trường	46
3.5.3. Tạo các thành phần cần thiết cho chương trình	46
3.5.4. Thực thi chương trình	47
3.6. Kết quả chương trình	47
KẾT LUẬN	50
TÀI LIỆU THAM KHẢO	51

Đại học Sao Đỏ

DANH MỤC HÌNH ẢNH

Hình 1.1.	Cách thức hoạt động của Deep Learning.....	3
Hình 1.2.	Ứng dụng trong phân tích cảm xúc.....	4
Hình 1.3.	Ứng dụng trong xe tự lái.....	5
Hình 1.4.	Ứng dụng trong trợ lý ảo	5
Hình 1.5.	Ứng dụng trong chăm sóc sức khỏe	6
Hình 1.6.	Ứng dụng trong mạng xã hội	6
Hình 1.7.	Cấu trúc của tế bào nơ-ron sinh học	8
Hình 1.8.	Mô hình của một nơ-ron nhân tạo được gán nhãn k.....	9
Hình 1.9.	Ví dụ mạng nơ-ron có 1 lớp ẩn.....	10
Hình 1.10.	Mạng nơ-ron truyền thẳng nhiều lớp ẩn h	11
Hình 1.11.	Mạng hồi quy i.....	11
Hình 1.12.	Phân loại mạng nơ-ron.....	11
Hình 1.13.	Kiến trúc truyền thống của một mạng CNN.....	13
Hình 1.14.	Ví dụ mô hình CNN.....	13
Hình 1.15.	Minh họa Convolution.....	14
Hình 1.16.	Minh họa tầng kết nối đầy đủ	14
Hình 1.17.	Minh họa mạng CNN.....	14
Hình 1.18.	Minh họa các bộ lọc có kích thước khác nhau	15
Hình 1.19.	Minh họa độ trượt S	15
Hình 1.20.	Minh họa tính tương thích của tham số	16
Hình 1.21.	Minh họa tính trường thu cảm tầng k	18
Hình 2.1.	Kiến trúc của Stable Diffusion[1].....	21
Hình 2.2.	Nhiều trong ảnh	21
Hình 2.3.	Thêm nhiều vào ảnh.....	23
Hình 2.4.	Minh họa tạo tập dữ liệu	23
Hình 2.5.	Minh họa dự đoán nhiều với lượng nhiều 3	24
Hình 2.6.	Minh họa khử nhiễu ảnh	24
Hình 2.7.	Minh họa bộ tạo ảnh	25
Hình 2.8.	Minh họa bộ giải mã ảnh	26
Hình 2.9.	Dữ liệu đào tạo clip.....	27
Hình 2.10.	Cấu trúc của mô hình clip	27
Hình 2.11.	Cấu trúc UNET	29
Hình 2.12.	Ví dụ về phát hiện biển báo dừng	31
Hình 2.13.	Công thức tình IoU	31
Hình 3.1.	Hình ảnh minh họa bài toán	32
Hình 3.2.	Kết quả tạo văn bản thành hình ảnh trên LAION-Aesthetic	33

Hình 3.3. Tạo hình ảnh có điều kiện bằng văn bản	33
Hình 3.4. Điều chỉnh bộ dự đoán nhiễu.....	33
Hình 3.5. Tập dữ liệu đầu vào	34
Hình 3.6. Khởi tạo mã nguồn cục bộ mới trên github	35
Hình 3.7. Đồng bộ và đẩy mã nguồn	35
Hình 3.8. Mã nguồn đã di chuyển lên github	36
Hình 3.9. Tạo sổ tay mới trên colab (cách 1)	36
Hình 3.10. Thay đổi kiểu phần cứng trên Google Colab	37
Hình 3.11. Thay đổi kiểu phần cứng	37
Hình 3.12. Thực thi mã trên colab	38
Hình 3.13. Giao diện.....	38
Hình 3.14. Giao diện textbox.....	39
Hình 3.15. Giao diện nút bấm.....	39
Hình 3.16. Giao diện thanh progressbar	40
Hình 3.17. Giao diện chương trình với văn bản chi tiết	47
Hình 3.18. Chú chó đeo nơ	48
Hình 3.19. Giọt sương cầu vòng.....	48
Hình 3.20. Nhà ga xe lửa Châu Âu.....	49

DANH MỤC BẢNG

Bảng1.1. So sánh các phương pháp Zero-padding	16
Bảng 1.2. Độ phức tạp của mô hình	17
Bảng 1.3. Những biến thể khác của ReLU	18

Đại học Sao Đỏ

MỞ ĐẦU

1. Tính cấp thiết của đề tài

Việc phát triển tạo ra website chuyên đổi văn bản thành ảnh sử dụng trí tuệ nhân tạo là xu thế phát triển của nhiều lĩnh vực. Trong thế giới ngày nay, nhu cầu về chất lượng hình ảnh ngày càng tăng, không chỉ trong lĩnh vực giải trí mà còn trong y tế, nghệ thuật số, và nhiều lĩnh vực khác. Để đáp ứng và vượt qua những thách thức này, phát triển website chuyên đổi văn bản thành hình ảnh đã trở thành một công nghệ cực kỳ cần thiết và quan trọng.

Ngoài ra, việc sử dụng mô hình Stable Diffusion còn mở ra những khả năng mới trong lĩnh vực nghệ thuật số và thiết kế đồ họa. Những người làm việc trong ngành này có thể sử dụng công nghệ để tạo ra những tác phẩm số độc đáo, sáng tạo hơn và ấn tượng hơn.

Một ứng dụng cụ thể khác của Stable Diffusion là trong việc nâng cao trải nghiệm người dùng trên các trang web và ứng dụng. Việc sử dụng hình ảnh chất lượng cao có thể tăng cường sự tương tác và giao tiếp với người dùng, tạo ra một trải nghiệm trực quan và đẹp mắt. Xuất phát từ những điều này, nhóm đề tài chúng em lựa chọn đề tài "Ứng dụng deeplearning xây dựng website chuyên đổi văn bản thành hình ảnh" với mong muốn là tạo ra website có thể chuyển đổi mô tả thông tin sang hình ảnh sang tạo và đa dạng.

2. Mục tiêu nghiên cứu

Nghiên cứu kỹ thuật phát triển website và lĩnh vực của trí tuệ nhân tạo, quá trình xây dựng phát triển website chuyên đổi văn bản thành hình ảnh.

3. Đối tượng nghiên cứu

- Nghiên cứu các kỹ thuật lập trình phát triển website với thư viện Gradio trên Python.
- Nghiên cứu phương pháp lưu trữ và xử lý mô hình học máy trong Python.
- Nghiên cứu các kỹ thuật sử dụng các kỹ thuật Deep Learning.
- Nghiên cứu sử dụng mô hình máy học Stable Diffusion.

4. Phương pháp nghiên cứu

Phương pháp nghiên cứu tài liệu: Nghiên cứu các tạp chí công nghệ, đồ án và tài liệu chuyên ngành.

Phương pháp thực nghiệm: Thiết kế, xây dựng, chạy thử nghiệm ứng dụng và đánh giá kết quả.

5. Phạm vi nghiên cứu

Ứng dụng Stable Diffusion tạo website chuyên văn bản thành hình ảnh.

6. Ý nghĩa khoa học và thực tiễn của đồ án

Ý nghĩa khoa học: Đồ án tạo ra phương pháp xây dựng website chuyên đổi văn bản thành hình ảnh sử dụng các kỹ thuật của trí tuệ nhân tạo.

Ý nghĩa thực tiễn: Tạo ra website sử dụng trên tất cả các thiết bị, giúp hỗ trợ con người có nội dung sáng tạo hơn về hình, hỗ trợ tìm kiếm hình ảnh đa dạng theo mô tả, làm phong phú thêm nguyên tài nguyên hỗ trợ học máy cũng như xử lý ảnh.

7. Kết cấu của đồ án

Đồ án trình bày gồm 3 chương:

Chương 1. Cơ sở lý thuyết: Trình bày lý thuyết về Deeplearning, tổng quan về website sử dụng AI và mạng nơ-ron quá đó là cơ sở thực hiện đồ án thực hiện phát triển đồ án.

Chương 2. Stable Diffusion: Trình bày chi tiết về Stable Diffusion: Định nghĩa, kiến trúc, cách hoạt động, các thành phần của Stable Diffusion. Ngoài ra còn trình bày về mạng UNET, nhiều ảnh hưởng đến chất lượng Stable Diffusion.

Chương 3. Xây dựng chương trình: Trình bày bài toán chuyển đổi văn bản thành hình ảnh, cách thức thiết kế ứng dụng và triển khai ứng dụng trên máy chủ Google Colab.

Chương 1. CƠ SỞ LÝ THUYẾT

1.1. Deep Learning

1.1.1. Khái niệm

Deep Learning, hay học sâu, là một lĩnh vực con của Machine Learning, trong đó máy tính được huấn luyện để tự học và cải thiện thông qua các thuật toán. Deep Learning được xây dựng trên các khái niệm phức tạp hơn rất nhiều và chủ yếu hoạt động với các mạng nơ-ron nhân tạo để mô phỏng khả năng tư duy và suy nghĩ của con người.

Mặc dù các khái niệm liên quan đến mạng nơ-ron nhân tạo và Deep Learning đã xuất hiện từ những năm 1960, nhưng chúng đã bị giới hạn bởi khả năng tính toán và lượng dữ liệu có sẵn vào thời điểm đó. Gần đây, tiến bộ trong việc xử lý dữ liệu lớn (Big Data) đã cho phép tận dụng tối đa khả năng của mạng nơ-ron nhân tạo.

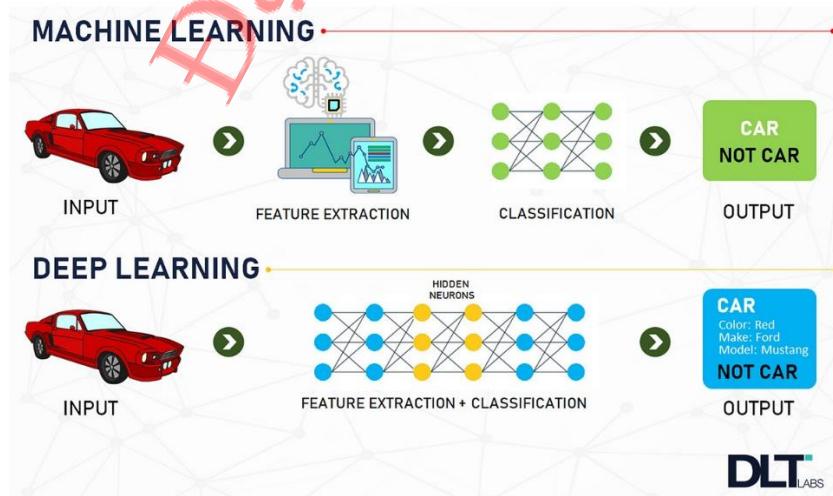
Mạng nơ-ron nhân tạo là động lực chính cho sự phát triển của Deep Learning. Mạng nơ-ron sâu (DNN) bao gồm nhiều lớp nơ-ron khác nhau, có khả năng thực hiện các tính toán phức tạp. Deep Learning đang phát triển rất nhanh chóng và được xem là một bước đột phá quan trọng trong Machine Learning.

1.1.2. Cách thức hoạt động của Deep Learning

Deep Learning là một phương pháp trong Machine Learning, trong đó mạng nơ-ron nhân tạo được sử dụng để mô phỏng khả năng tư duy của bộ não con người.

Mạng nơ-ron bao gồm nhiều lớp khác nhau, và số lượng lớp càng nhiều thì mạng càng “sâu”. Mỗi lớp chứa các nút mạng được liên kết với các lớp khác. Mỗi kết nối giữa các nút có một trọng số tương ứng, và trọng số này ảnh hưởng đến mạng nơ-ron theo mức độ của nó.

Mỗi nút trong mạng nơ-ron có một hàm kích hoạt, có nhiệm vụ chuyển đổi đầu ra từ nút đó. Dữ liệu được đưa vào mạng nơ-ron sẽ đi qua từng lớp và cuối cùng trả về kết quả tại lớp đầu ra, được gọi là output layer.



Hình 1.1. Cách thức hoạt động của Deep Learning

Trong quá trình huấn luyện mô hình mạng no-ron, các trọng số sẽ được thay đổi và nhiệm vụ của mô hình là tìm ra bộ giá trị của trọng số sao cho phán đoán là tốt nhất.

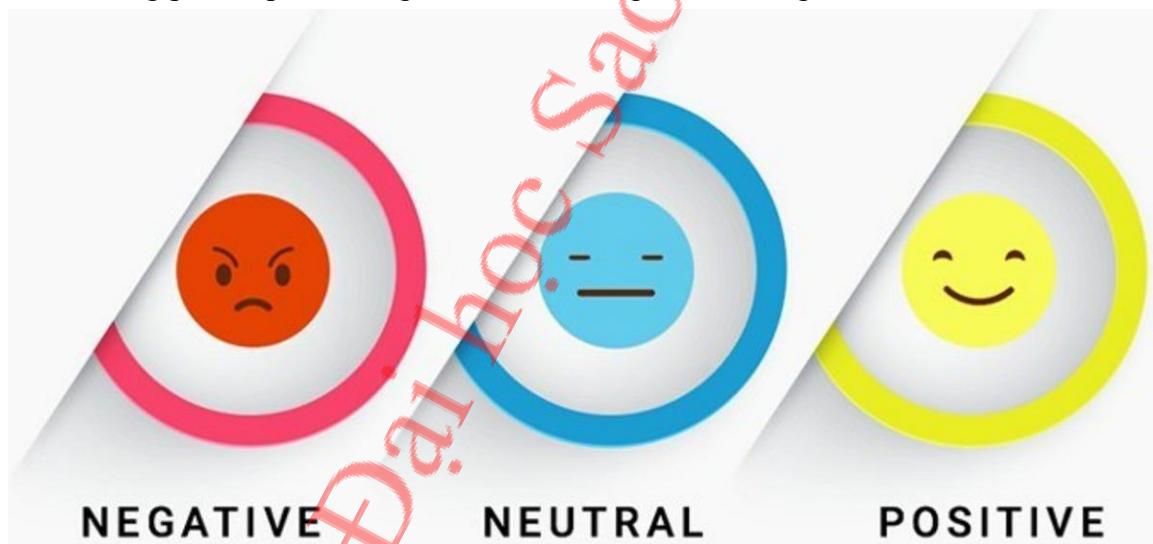
Các hệ thống Deep Learning yêu cầu phần cứng phải rất mạnh để có thể xử lý được lượng dữ liệu lớn và thực hiện các phép tính phức tạp. Nhiều mô hình Deep Learning có thể mất nhiều tuần, thậm chí nhiều tháng để triển khai trên những phần cứng tiên tiến nhất hiện nay.

1.1.3. Ứng dụng của Deep Learning

Deep Learning, trong kiến trúc mạng no-ron, được áp dụng rộng rãi trong các tác vụ yêu cầu tính toán mạnh, xử lý dữ liệu lớn và độ phức tạp cao. Dưới đây là một số ứng dụng phổ biến của Deep Learning trong cuộc sống hàng ngày:

- Phân tích cảm xúc

Trong lĩnh vực phân tích cảm xúc, Deep Learning được áp dụng để xử lý ngôn ngữ tự nhiên, phân tích văn bản và thống kê để hiểu và dự đoán cảm xúc của con người. Các công ty có thể sử dụng Deep Learning để hiểu và dự đoán cảm xúc của khách hàng dựa trên các đánh giá, bình luận, tweet, v.v. và từ đó đưa ra các chiến lược kinh doanh và marketing phù hợp cho từng nhóm đối tượng khách hàng.



Hình 1.2. Ứng dụng trong phân tích cảm xúc

- Xe tự lái

Công nghệ xe tự lái đang trở thành một xu hướng thú vị, và nó được xây dựng dựa trên các mạng no-ron tiên tiến. Đơn giản, các mô hình Deep Learning được sử dụng để nhận dạng các đối tượng trong môi trường xung quanh xe, tính toán khoảng cách đến các phương tiện khác, xác định vị trí làn đường, tín hiệu giao thông, v.v. Dựa trên thông tin này, các quyết định tối ưu và nhanh chóng được đưa ra. Tesla là một trong những hãng xe tiên phong trong lĩnh vực này.



Hình 1.3. Ứng dụng trong xe tự lái

- Trợ lý ảo

Trợ lý ảo (Virtual Digital Assistant hay Virtual Assistant) là một chương trình có khả năng hiểu ngôn ngữ tự nhiên và có thể trả lời các câu hỏi hoặc hoàn thành nhiệm vụ dựa trên khẩu lệnh (voice command). Các trợ lý ảo như Siri, Google Assistant hay Alexa cũng sử dụng Deep Learning để hiểu và xử lý các câu lệnh và yêu cầu từ người dùng. Điều này bao gồm việc nhận dạng giọng nói, chuyển đổi lời nói thành văn bản, hiểu ý định của người dùng và cung cấp phản hồi phù hợp.



Hình 1.4. Ứng dụng trong trợ lý ảo

- Chăm sóc sức khỏe

Deep Learning cũng có đóng góp không nhỏ vào lĩnh vực y tế, trong đó phổ biến gồm có các mô hình dự đoán tình trạng bệnh, chẩn đoán ung thư, phân tích kết quả MRI, X-ray.

Deep Learning hỗ trợ người dùng, bao gồm bác sĩ và bệnh nhân rất nhiều trong việc phân tích dữ liệu về sức khỏe, dự đoán tình trạng bệnh, chẩn đoán ung thư, phân tích kết quả chụp chiếu MRI, tia X-ray,...

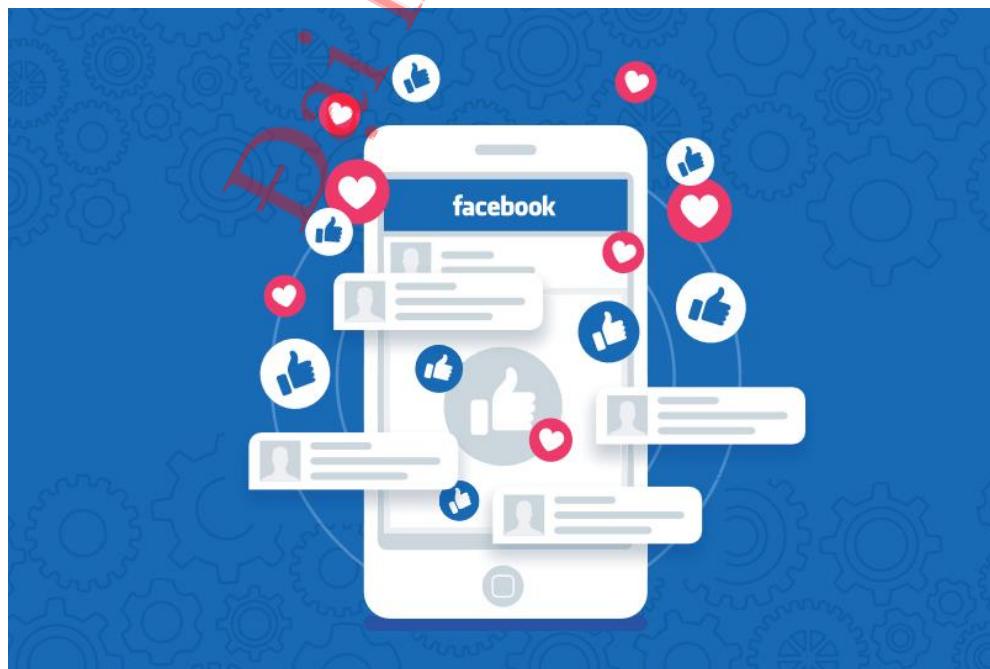


Hình 1.5. Ứng dụng trong chăm sóc sức khỏe

- Mạng xã hội

Các nền tảng mạng xã hội, bao gồm Twitter và Instagram, đã áp dụng Deep Learning để cải thiện các dịch vụ của họ. Bằng cách sử dụng mạng nơ-ron nhân tạo, những nền tảng này phân tích một lượng lớn dữ liệu để hiểu về sở thích và lựa chọn của người dùng. Ví dụ, Twitter sử dụng Deep Learning để cung cấp gợi ý nội dung và tìm kiếm phù hợp. Instagram cũng sử dụng Deep Learning để phát hiện và ngăn chặn hành vi bạo lực trên nền tảng, từ việc chặn các bình luận vi phạm đến các hình ảnh không phù hợp.

Facebook cũng là một mạng xã hội áp dụng Deep Learning vào các sản phẩm của mình. Các thuật toán mạng nơ-ron sâu được sử dụng để đề xuất trang, bạn bè, dịch vụ và nhận dạng khuôn mặt trên nền tảng.



Hình 1.6. Ứng dụng trong mạng xã hội

1.2. Tổng quan Website AI

1.2.1. Khái niệm

Khái niệm AI (viết tắt của Artificial Intelligence) trên website chỉ đề cập đến các chương trình máy tính phức tạp có khả năng tự học và đưa ra các quyết định mà không cần lập trình trước. Công nghệ này mô phỏng quá trình suy nghĩ và hoạt động của con người cho máy móc, đặc biệt là hệ thống máy tính.

Trong khi đó, website (còn gọi là trang web hoặc trang mạng) là một trang được xuất bản trên ít nhất một máy chủ web và được xác định bằng một tên miền chung. Nội dung liên quan đến website được xuất bản trên ít nhất một máy chủ web.

Trí tuệ nhân tạo (AI) trên website là việc tích hợp và triển khai các công nghệ AI trong quá trình xây dựng và phát triển các trang web. Điều này nhằm mục đích cải thiện trải nghiệm người dùng, tối ưu hóa quy trình kinh doanh, và mang lại giá trị gia tăng thông qua khả năng tự động hóa và thông minh hóa.

1.2.2. Ứng dụng của Website Ai

Trí tuệ nhân tạo (AI) đã từng bước chứng minh vai trò không thể phủ nhận của mình trong sự phát triển của các ứng dụng web hiện đại. Việc tích hợp AI vào các trang web không chỉ là một cách để nâng cao trải nghiệm người dùng, mà còn là chìa khóa mở ra những khả năng mới trong việc tối ưu hóa quy trình kinh doanh và tạo ra giá trị gia tăng.

- Chatbots và trò chuyện tự động

Một trong những định hình mạnh mẽ của sự hiện đại hóa này là việc sử dụng chatbots và hệ thống trò chuyện tự động. Các chatbot không chỉ đơn thuần là công cụ hỗ trợ khách hàng tức thì, mà còn trở thành đối tác tương tác thông minh, sử dụng xử lý ngôn ngữ tự nhiên để hiểu và phản hồi theo cách tự nhiên nhất có thể. Điều này giúp giảm thời gian đáp ứng, tăng tính linh hoạt và tạo ra trải nghiệm người dùng tương tác và hiệu quả.

- Tìm kiếm thông minh

Một lĩnh vực quan trọng khác mà trí tuệ nhân tạo ảnh hưởng mạnh mẽ đến là tìm kiếm thông minh. Công nghệ máy học đã được áp dụng để cá nhân hóa kết quả tìm kiếm, dựa trên lịch sử tìm kiếm và hành vi người dùng. Việc này không chỉ mang lại sự thuận tiện khi tra cứu thông tin mà còn làm tăng hiệu suất làm việc trên web.

- Cải thiện quảng cáo và tiếp thị

Gợi ý sản phẩm và dịch vụ là một khía cạnh quan trọng khác của ứng dụng AI trên web. Hệ thống gợi ý sử dụng học máy để hiểu rõ hơn về sở thích và hành vi mua sắm cá nhân của người dùng. Điều này giúp doanh nghiệp tối ưu hóa chiến lược bán hàng, nâng cao khả năng tương tác và tăng cường sự hài lòng của khách hàng.

- Phân tích cảm xúc, ý kiến

Một khía cạnh quan trọng nữa là phân tích ý kiến và tình cảm trên mạng. Các

thuật toán phân tích ý kiến giúp doanh nghiệp đánh giá tình cảm và phản hồi của người dùng trên các nền tảng xã hội. Điều này không chỉ là một cách để đo lường hiệu suất của sản phẩm hoặc dịch vụ, mà còn là công cụ đắc lực để dự báo xu hướng và điều chỉnh chiến lược tiếp theo.

- **Tự động hóa công việc**

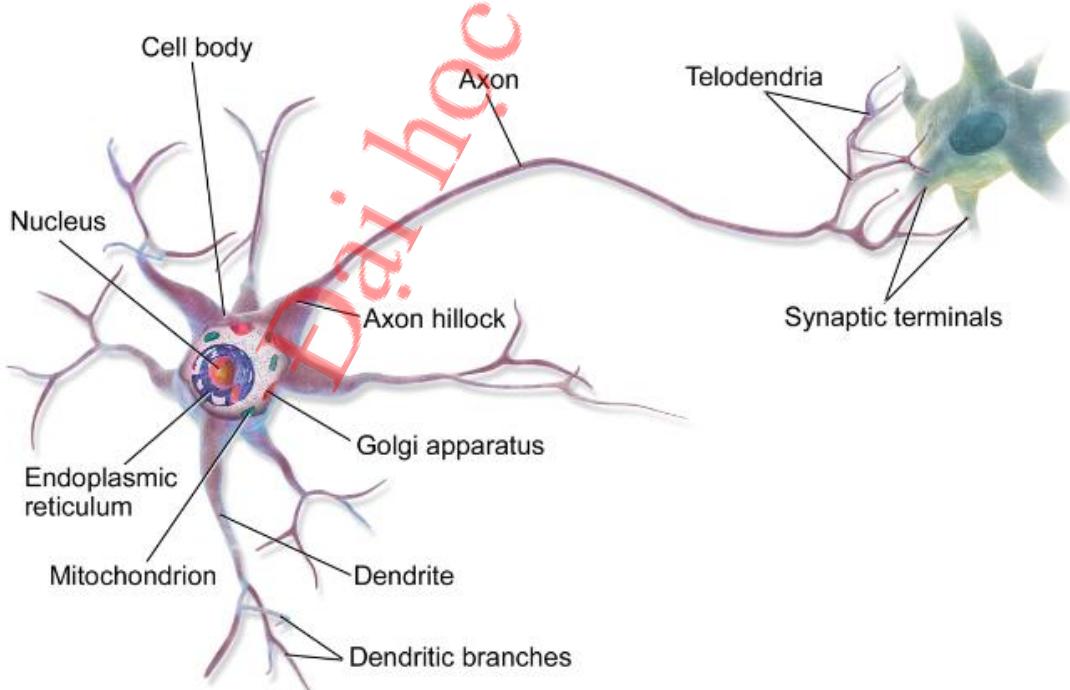
Tự động hóa nhiệm vụ lặp đi lặp lại trong quy trình kinh doanh là một ứng dụng khác của AI trên web. Việc này giúp doanh nghiệp tiết kiệm thời gian và tài nguyên bằng cách tự động hóa các nhiệm vụ như xử lý đơn hàng, kiểm tra tồn kho và quản lý dữ liệu.

Cuối cùng, sự tích hợp của trí tuệ nhân tạo vào các ứng dụng web không chỉ là một xu hướng, mà là một hành trình không ngừng của sự sáng tạo và cải tiến. Với sự phát triển liên tục của công nghệ, chúng ta có thể hứa hẹn thêm nhiều tiến bộ và ứng dụng mới mẻ trong tương lai.

1.3. Tổng quan về mạng nơ-ron [4]

1.3.1. Khái niệm

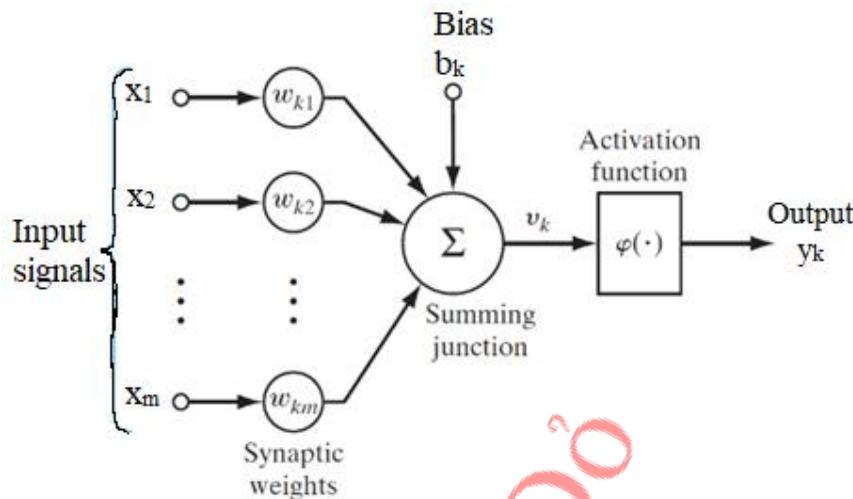
Mạng nơ-ron (Neural network) hay một số tác giả khác dịch là mạng neural hoặc mạng noron. Khi nói một nơ-ron (hoặc noron) đơn lẻ, từ tiếng anh là neuron. Những ý tưởng xây dựng các mô hình mạng nơ-ron nhân tạo bắt nguồn từ việc khám phá ra các cơ chế hoạt động đơn giản của mạng nơ-ron sinh học (biological neural network).



Hình 1.7. Cấu trúc của tế bào nơ-ron sinh học

Các nhà nghiên cứu đã tìm cách chuyển đổi những hiểu biết về cách thức hoạt động của các tế bào thần kinh sinh học thành các mô hình mạng nơ-ron nhân tạo (Artificial Neural Network) có thể hoạt động được trên máy tính.

Mô hình của một nơ-ron đơn lẻ, được xem như đơn vị xử lý thông tin cơ bản của một mạng nơ-ron. Các nơ-ron này được sử dụng để xây dựng thành các mạng nơ-ron có kiến trúc phức tạp sẽ được trình bày ở các phần sau.



Hình 1.8. Mô hình của một nơ-ron nhân tạo được gán nhãn k

Các đặc trưng cơ bản của một mạng nơ-ron:

- Một tập các đơn vị xử lý (các nơ-ron nhân tạo).
- Trạng thái kích hoạt hay đầu ra của đơn vị xử lý.
- Liên kết giữa các đơn vị. Xét tổng quát, mỗi liên kết được định nghĩa bởi một trọng số w_{jk} cho ta biết hiệu ứng của đơn vị j có trên đơn vị k .
- Mỗi luật lan truyền quyết định cách tính tín hiệu của từng đơn vị từ đầu vào của nó.
- Một hàm kích hoạt hay hàm chuyển (activation function, transfer function) xác định mức độ kích hoạt khác dựa trên mức độ kích hoạt hiện tại.
- Một đơn vị điều chỉnh (độ lệch) (bias, offset) của mỗi đơn vị.
- Phương pháp thu thập thông tin (luật học – learning rule).
- Môi trường hệ thống có thể hoạt động.

1.3.2. Các thành phần cơ bản

Mỗi nơ-ron (nút) là một đơn vị xử lý thông tin của mạng nơ-ron, là yếu tố cơ bản để cấu tạo thành mạng nơ-ron. Một mạng nơ-ron gồm 3 thành phần cơ bản sau:

- Đơn vị xử lý

Còn được gọi là một nơ-ron hay một nút (node), thực hiện một công việc rất đơn giản: Nhận tín hiệu vào từ các đơn vị phía trước hay một nguồn bên ngoài và sử dụng chúng để tính tín hiệu ra sẽ được lan truyền sang các đơn vị khác.

- Hàm kết hợp

Mỗi một đơn vị trong một mạng kết hợp các giá trị đưa vào nó thông qua các liên kết với các đơn vị khác, sinh ra một giá trị gọi là net input. Hàm thực hiện nhiệm vụ này gọi là hàm kết hợp (combination function), được định nghĩa bởi một luật lan truyền cụ thể.

- **Hàm kích hoạt**

Phần lớn các đơn vị trong mạng nơ-ron chuyển net input bằng cách sử dụng một hàm vô hướng gọi là hàm kích hoạt, kết quả của hàm này là một giá trị gọi là mức độ kích hoạt của đơn vị (unit's activation). Loại trừ khả năng đơn vị đó thuộc lớp ra, giá trị kích hoạt được đưa vào một hay nhiều đơn vị khác. Các hàm kích hoạt thường bị ép vào một khoảng giá trị xác định, do đó thường được gọi là các hàm bẹp (squashing).

1.3.3. Kiến trúc mạng

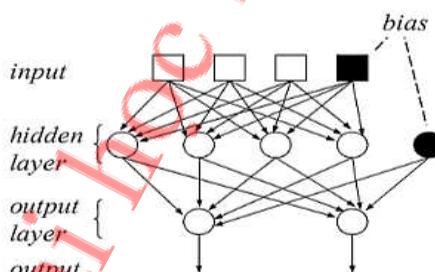
Kiến trúc của một mạng nơ-ron được xác định bởi:

- Số lượng các tín hiệu đầu vào và đầu ra.
- Số lượng các lớp
- Số lượng các nơ-ron trong mỗi lớp
- Số lượng các trọng số (các liên kết) đối với mỗi nơ-ron.
- Cách thức các nơ-ron (trong một lớp, giữa các lớp) liên kết với nhau.
- Những nơ-ron nào nhận các tín hiệu điều chỉnh lỗi.

Một mạng nơ-ron cần phải có:

- Một lớp đầu vào (input layer)
- Một lớp đầu ra (output layer)
- Không, một, hoặc nhiều lớp ẩn (hidden layer(s))

Ví dụ:



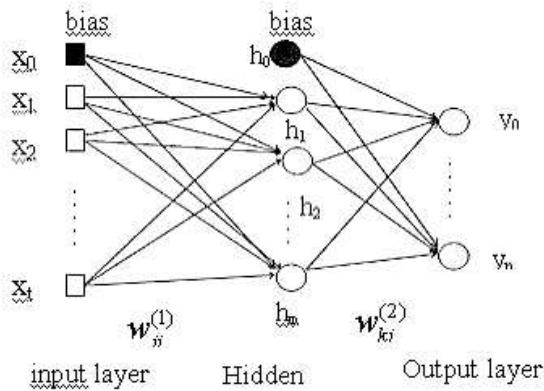
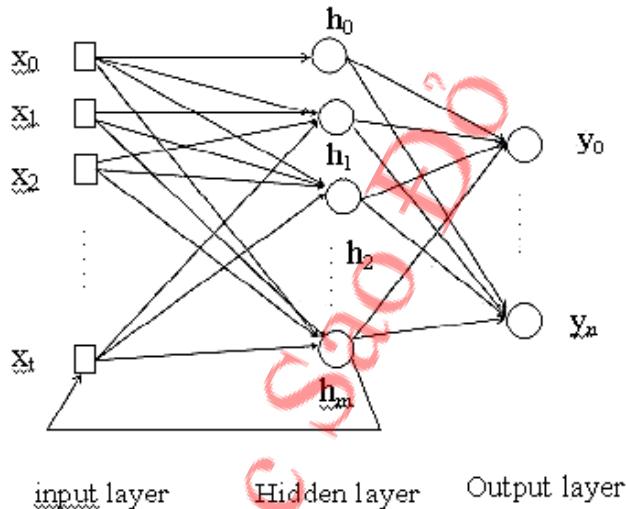
Hình 1.9. Ví dụ mạng nơ-ron có 1 lớp ẩn

Ví dụ về mạng nơ-ron có một lớp ẩn như hình 1.9 mạng nơ-ron có:

- Đầu vào: 3 tín hiệu
- Đầu ra: 2 giá trị
- Tổng cộng có 6 nơ-ron: 4 nơ-ron ở lớp ẩn, 2 nơ-ron ở đầu ra.

a) *Phân loại mạng theo kiểu kết nối các nơ-ron*

Dựa theo kiểu kết nối, các mạng nơ-ron truyền thẳng (feedforward Neural Network) và mạng hồi quy (recurrent Neural Network). Trong mạng truyền thẳng, các kết nối đi theo một hướng nhất định, không tạo thành chu trình. Ngược lại, các mạng hồi quy cho phép các kết nối nơ-ron tạo thành chu trình, với đỉnh là các nơ-ron và cung là các kết nối giữa chúng. Các nơ-ron nhận tín hiệu vào gọi là nơ-ron vào, các nơ-ron đưa thông tin ra gọi là nơ-ron ra, các nơ-ron còn lại gọi là nơ-ron ẩn.

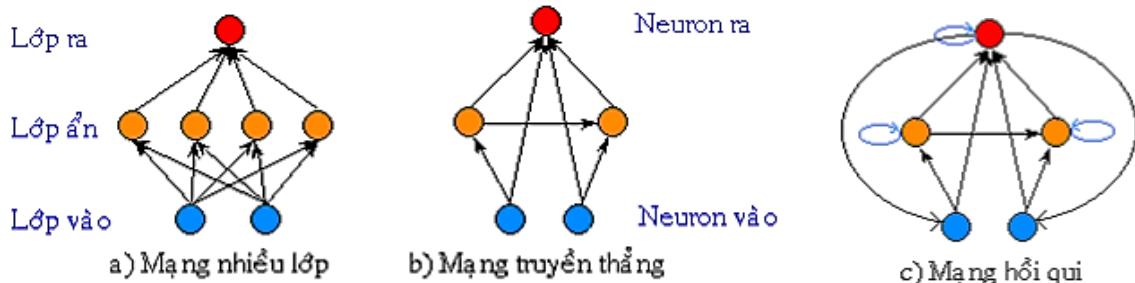
Hình 1.10. Mạng nơ-ron truyền thẳng nhiều lớp ẩn h 

Hình 1.11. Mạng hồi quy

b) Phân loại theo số lớp nơ-ron

Các nơ-ron trong mạng có thể được tổ chức thành các lớp theo nguyên tắc các nơ-ron ở lớp này, chỉ được nối với các nơ-ron ở lớp khác, không cho phép kết nối giữa các nơ-ron trên cùng lớp, hoặc từ nơ-ron lớp dưới lên nơ-ron lớp trên, cũng không cho phép kết nối nhảy qua 1 lớp. Lớp nhận tín hiệu vào gọi là lớp vào, lớp đưa thông tin ra gọi là lớp ra, các lớp ở giữa gọi là lớp ẩn.

Thông thường lớp vào không tham gia quá trình tính toán của mạng nơ-ron nên khi tính số lớp người ta không kể lớp vào.



Hình 1.12. Phân loại mạng nơ-ron

1.3.4. Huấn luyện mạng

Chức năng của một mạng nơron được quyết định bởi các nhân tố như: hình trạng mạng (số lớp, số đơn vị trên mỗi tầng, và cách mà các lớp được liên kết với nhau) và các trọng số của các liên kết bên trong mạng.

Hình trạng của mạng thường là cố định, và các trọng số được quyết định bởi một thuật toán huấn luyện (training algorithm). Tiến trình điều chỉnh các trọng số để mạng “nhận biết” được quan hệ giữa đầu vào và đích mong muốn được gọi là học (learning) hay huấn luyện (training).

Rất nhiều thuật toán học đã được phát minh để tìm ra tập trọng số tối ưu làm giải pháp cho các bài toán. Các thuật toán đó có thể chia làm ba nhóm chính:

- **Học có giám sát (Supervised learning)**

Là giải thuật điều chỉnh các trọng số kết nối dựa vào sự khác biệt giữa ngõ ra thực tế của mạng (actual network output) và ngõ ra mong muốn (target hoặc desired network output) ứng với một tập tín hiệu vào. Học giám sát đòi hỏi phải có một teacher hay supervisor cung cấp ngõ ra mong muốn, vì thế người ta gọi là học có thầy hay học có giám sát.

- **Học không giám sát (Unsupervised Learning).**

Trong phương pháp học không giám sát, không đòi hỏi tập ngõ ra mong muốn, vì thế người ta gọi là học không thầy hay học không giám sát. Trong quá trình huấn luyện, chỉ có tập dữ liệu vào được đưa vào mạng, mạng tự điều chỉnh theo nguyên tắc gộp các mẫu dữ liệu vào có đặc điểm tương tự thành từng nhóm.

Mạng nơ-ron không được giám sát được huấn luyện bằng cách để mạng nơ-ron liên tục tự điều chỉnh theo các đầu vào mới. Chúng được sử dụng để rút ra các suy luận từ các tập dữ liệu bao gồm dữ liệu đầu vào mà không có các phản hồi được gắn nhãn.

- **Học tăng cường (reinforcement learning).**

Học tăng cường là một giải thuật đặc biệt của học có giám sát. Thay vì phải do thầy cung cấp ngõ ra mong muốn thì giải thuật này sẽ nhờ một “chuyên gia” (critic) ước lượng ngõ ra tốt nhất ứng với một ngõ vào cho trước. Tiêu biểu cho học tăng cường là giải thuật di truyền (Genetic Algorithm - GA)

Học tăng cường là một loại kỹ thuật học máy trong đó tác nhân máy tính học cách thực hiện một tác vụ thông qua các tương tác thử và lỗi lặp đi lặp lại với môi trường động. Cách tiếp cận học tập này cho phép người đại diện đưa ra một loạt quyết định nhằm tối đa hóa số liệu thưởng cho nhiệm vụ mà không cần sự can thiệp của con người và không được lập trình rõ ràng để đạt được nhiệm vụ.

Các chương trình AI được huấn luyện với học tập cũng cố đánh bại người chơi con người trong các trò chơi trên bàn cờ như cờ vây và cờ vua, cũng như trò chơi điện tử. Mặc dù học tăng cường không phải là một khái niệm mới, nhưng tiến bộ gần đây

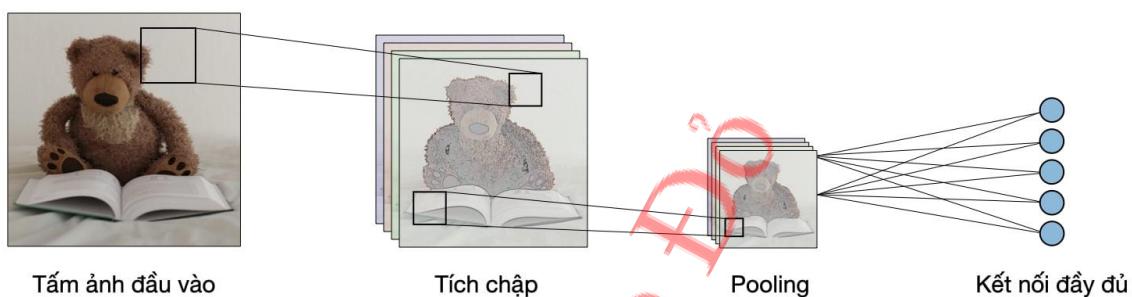
trong học sâu và sức mạnh tính toán đã giúp nó có thể đạt được một số kết quả đáng chú ý trong lĩnh vực trí tuệ nhân tạo.

1.3.5. Mạng nơ-ron tích chập

1.3.5.1. Cách thức hoạt động của CNN

Một mạng nơ-ron tích chập có thể có hàng chục hoặc hàng trăm lớp mà mỗi lớp học để phát hiện các đặc điểm khác nhau của hình ảnh.

CNN bao gồm một lớp đầu vào, một lớp đầu ra và nhiều lớp ẩn ở giữa. Các lớp này thực hiện các hoạt động làm thay đổi dữ liệu với mục đích tìm hiểu các đặc trưng cụ thể cho dữ liệu. Một số lớp phổ biến nhất là: Convolution, ReLU, Pooling, ReLU.

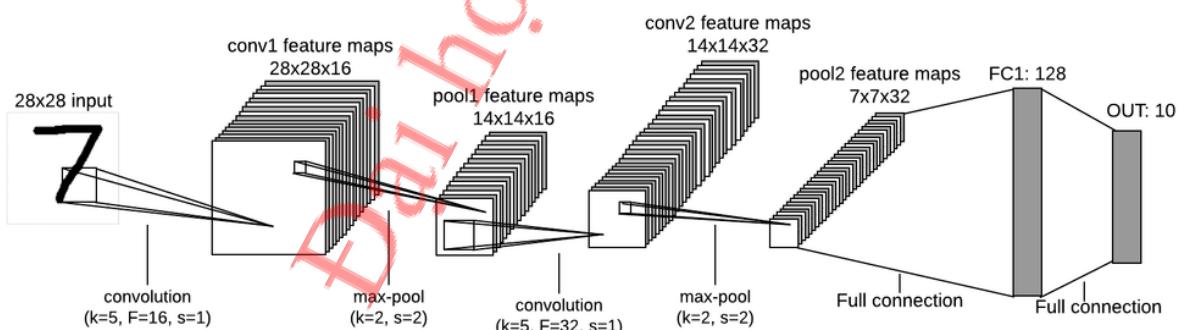


Hình 1.13. Kiến trúc truyền thống của một mạng CNN

1.3.5.2. Mô hình tổng quát của CNN

Mô hình tổng quát của một mạng CNN:

Input image → Convolutional layer (Conv) + Pooling layer (Pool) → Fully connected layer (FC) → Output.

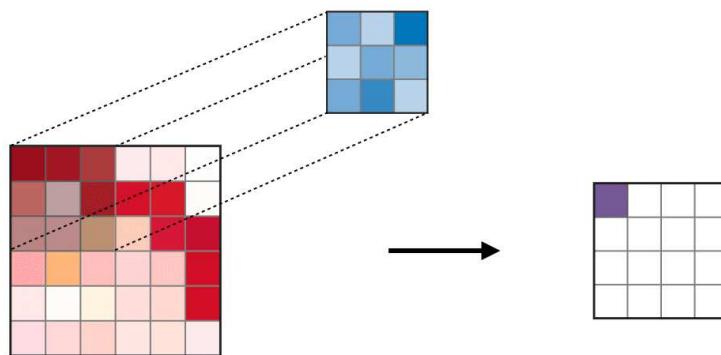


Hình 1.14. Ví dụ mô hình CNN

Convolution: Đặt các hình ảnh đầu vào thông qua một tập hợp các bộ lọc tích chập, mỗi bộ lọc trong số đó sẽ kích hoạt một số đặc trưng nhất định từ hình ảnh.

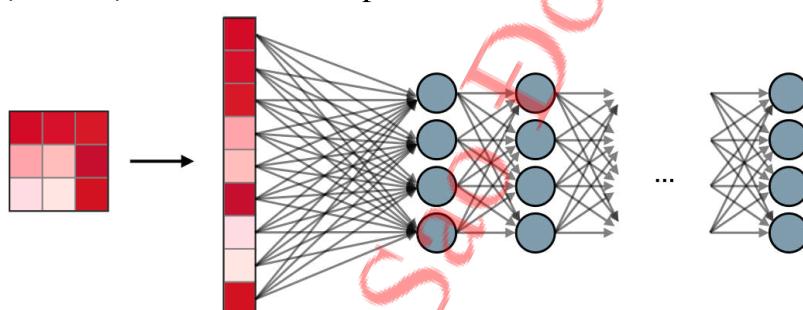
ReLU: Cho phép huấn luyện nhanh hơn và hiệu quả hơn bằng cách ánh xạ các giá trị âm về 0 và duy trì các giá trị dương. Điều này đôi khi được gọi là **kích hoạt**, bởi vì chỉ các đặc trưng đã được kích hoạt mới được chuyển sang lớp tiếp theo.

Pooling: Đơn giản hóa đầu ra bằng cách thực hiện lấy mẫu xuống phi tuyến, giảm số lượng các tham số mà mạng cần tìm hiểu.



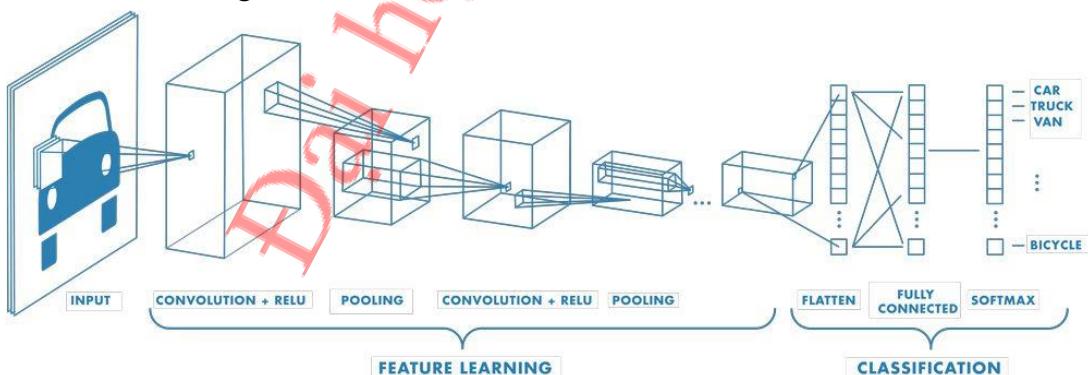
Hình 1.15. Minh họa Convolution

Đầu vào đó được kết nối đến tất cả nơ-ron. Trong mô hình mạng CNNs, các tầng kết nối dày đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.



Hình 1.16. Minh họa tầng kết nối dày đủ

Các thao tác này được lặp lại trên hàng chục hoặc hàng trăm lớp, với mỗi lớp học để xác định các đặc trưng khác nhau.



Hình 1.17. Minh họa mạng CNN

Hình 1.17 là ví dụ về một mạng có nhiều lớp chập. Các bộ lọc được áp dụng cho mỗi hình ảnh huấn luyện ở các độ phân giải khác nhau và đầu ra của mỗi hình ảnh biến đổi được sử dụng làm đầu vào cho lớp tiếp theo.

Giống như một mạng nơ-ron truyền thống, CNN có trọng số và độ lệch. Mô hình học các giá trị này trong quá trình huấn luyện và nó liên tục cập nhật chúng với mỗi ví dụ huấn luyện mới. Tuy nhiên, trong trường hợp của CNN, trọng số và độ lệch là giống nhau đối với tất cả các nơ-ron ẩn trong một lớp nhất định.

Điều này có nghĩa là tất cả các nơ-ron ẩn đang phát hiện cùng một đặc điểm, chẳng hạn như một cạnh hoặc một đốm màu, ở các vùng khác nhau của hình ảnh. Điều này làm cho mạng có khả năng dịch các đối tượng trong một hình ảnh. Ví dụ: một mạng được huấn luyện để nhận dạng ô tô sẽ có thể làm như vậy ở bất kỳ nơi nào có ô tô trong hình ảnh.

Sau khi tìm hiểu các đặc trưng trong nhiều lớp, kiến trúc của CNN chuyển sang phân loại.

Lớp tiếp theo đến lớp cuối cùng là lớp được kết nối đầy đủ tạo ra một vector có K chiều trong đó K là số lớp mà mạng có thể dự đoán. Vector này chứa các xác suất cho mỗi lớp của bất kỳ hình ảnh nào đang được phân loại.

Lớp cuối cùng của kiến trúc CNN sử dụng một lớp phân loại như softmax để cung cấp đầu ra phân loại.

1.3.5.3. Điều chỉnh các siêu tham số

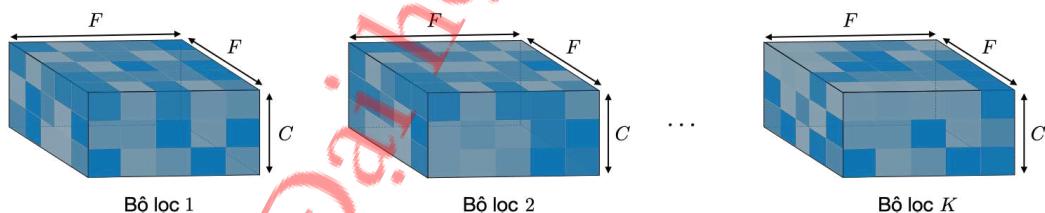
- Các siêu tham số của bộ lọc

Tầng tích chập chứa các bộ lọc mà rất quan trọng cho ta khi biết ý nghĩa đằng sau các siêu tham số của chúng.

- Các chiều của một bộ lọc

Một bộ lọc kích thước $F \times F$ áp dụng lên đầu vào chứa C kênh (channels) thì có kích thước tổng thể là $F \times F \times C$ thực hiện phép tích chập trên đầu vào kích thước $I \times I \times C$ và cho ra một feature map (hay còn gọi là activation map) có kích thước $O \times O \times 1$.

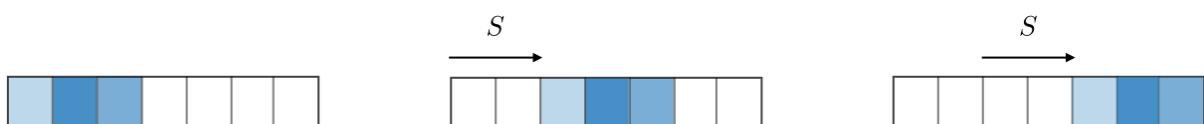
Lưu ý: Việc áp dụng K bộ lọc có kích thước $F \times F$ cho ra một feature map có kích thước O: $O \times O \times K$.



Hình 1.18. Minh họa các bộ lọc có kích thước khác nhau

- Stride

Đối với phép tích chập hoặc phép pooling, độ trượt S ký hiệu số pixel mà cửa sổ sẽ di chuyển sau mỗi lần thực hiện phép tính.

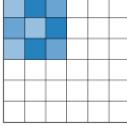
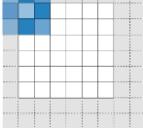


Hình 1.19. Minh họa độ trượt S

- Zero-padding

Zero-padding là tên gọi của quá trình thêm P số không vào các biên của đầu vào. Giá trị này có thể được lựa chọn thủ công hoặc một cách tự động bằng một trong ba những phương pháp mô tả bên dưới:

Bảng 1.1. So sánh các phương pháp Zero-padding

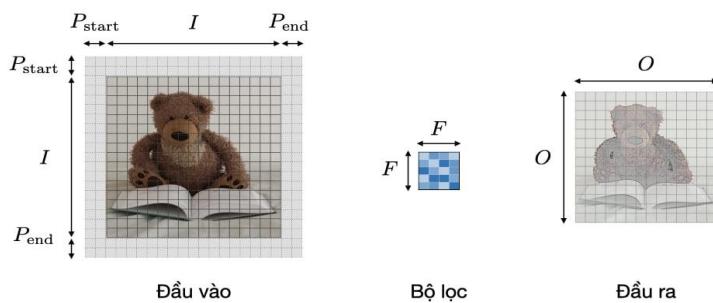
Phương pháp	Valid	Same	Full
Giá trị	$P = 0$	$P_{start} = \left\lfloor \frac{S \left\lceil \frac{I}{S} \right\rceil - I + F - S}{2} \right\rfloor$ $P_{end} = \left\lfloor \frac{S \left\lceil \frac{I}{S} \right\rceil - I + F - S}{2} \right\rfloor$	$P_{start} \in [0, F - 1]$ $P_{end} = F - 1$
Minh họa			
Mục đích	<ul style="list-style-type: none"> Không sử dụng padding. Bỏ phép tích chập cuối nếu số chiều không khớp. 	<ul style="list-style-type: none"> Sử dụng padding để làm cho feature map có kích thước $\left\lceil \frac{I}{S} \right\rceil$ Kích thước đầu ra thuận lợi về mặt toán học. Còn được gọi là 'half padding'. 	<ul style="list-style-type: none"> Padding tối đa sao cho các phép tích chập có thể được sử dụng tại các rìa của đầu vào Bộ lọc 'thầy' được đầu vào từ đầu đến cuối

- Điều chỉnh siêu tham số

- Tính tương thích của tham số trong tầng tích chập

Bằng cách ký hiệu I là độ dài kích thước đầu vào, F là độ dài của bộ lọc, P là số lượng zero padding, S là độ trượt, ta có thể tính được độ dài O của feature map theo một chiều bằng công thức:

$$O = \frac{I - F + P_{start} + P_{end}}{S} + 1$$



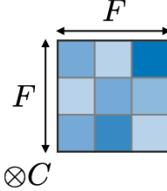
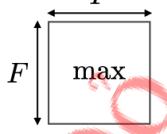
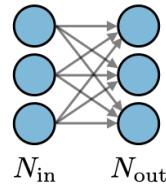
Hình 1.20. Minh họa tính tương thích của tham số

Lưu ý: Trong một số trường hợp, $P_{start}=P_{end} \triangleq P$, ta có thể thay thế $P_{start}+P_{end}$ bằng $2P$ trong công thức trên.

- Hiểu về độ phức tạp của mô hình

Để đánh giá độ phức tạp của một mô hình, cách hữu hiệu là xác định số tham số mà mô hình đó sẽ có. Trong một tầng của mạng nơ-ron tích chập, nó sẽ được tính toán như bảng 1.2:

Bảng 1.2. Độ phức tạp của mô hình

	CONV	POOL	FC
Minh họa	 $\times K$	 \max	
Kích thước đầu vào	$I \times I \times C$	$I \times I \times C$	N_{in}
Kích thước đầu ra	$O \times O \times K$	$O \times O \times C$	N_{out}
Số lượng tham số	$(F \times F \times C + 1) \cdot K$	0	$(N_{in} + 1) \times N_{out}$
Lưu ý	<ul style="list-style-type: none"> Một tham số bias với mỗi bộ lọc. Trong đa số trường hợp, $S < F$. Một lựa chọn phổ biến cho K là $2C$. 	<ul style="list-style-type: none"> Phép pooling được áp dụng lên từng kênh (channel-wise) Trong đa số trường hợp, $S = F$ 	<ul style="list-style-type: none"> Đầu vào được làm phẳng. Mỗi nơ-ron có một tham số bias. Số nơ-ron trong một tầng FC phụ thuộc vào ràng buộc kết cấu.

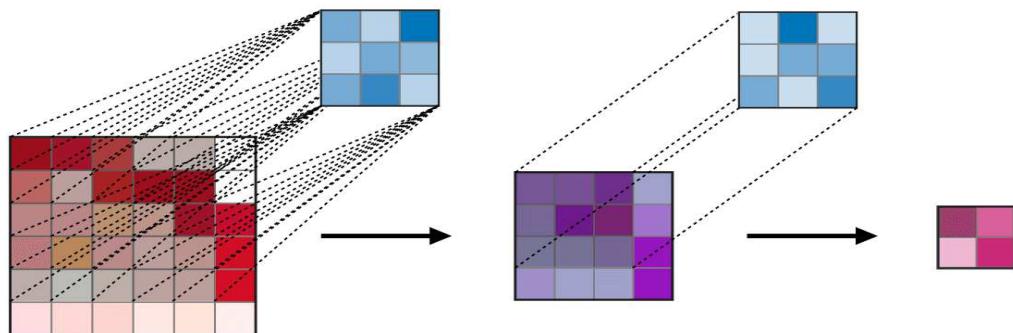
- Trường thu cảm

Trường thu cảm (receptive field) tại tầng k là vùng được ký hiệu $R_k \times R_k$ của đầu vào mà những pixel của activation map thứ k có thể "nhìn thấy".

Gọi F_j là kích thước bộ lọc của tầng j và S_i là giá trị độ trượt của tầng i và để thuận tiện, ta mặc định $S_0=1$, trường thu cảm của tầng k được tính toán bằng công thức:

$$R_k = 1 + \sum_{j=1}^k (F_j - 1) \prod_{i=0}^{j-1} S_i \quad (2.1)$$

Trong ví dụ hình 1.27, ta có $F_1=F_2=3$ và $S_1=S_2=1$, nên cho ra được $R_2=1+2\cdot1+2\cdot1=5$.



Hình 1.21. Minh họa tính trường thu cảm tầng k

1.3.5.4. Các hàm kích hoạt thường gấp

- **Rectified Linear Unit**

Tầng rectified linear unit (ReLU) là một hàm kích hoạt gg được sử dụng trên tất cả các thành phần. Mục đích của nó là tăng tính phi tuyến tính cho mạng. Những biến thể khác của ReLU được tổng hợp ở bảng dưới:

Bảng 1.3. Những biến thể khác của ReLU

ReLU	Leaky ReLU	ELU
$g(z)=\max(0,z)$	$g(z)=\max(\epsilon z, z)$ với $\epsilon \ll 1$	$g(z)=\max(\alpha(ez-1), z)$ với $\alpha \ll 1$
• Độ phức tạp phi tuyến tính có thể thông dịch được về mặt sinh học	• Gán vấn đề ReLU chết cho những giá trị âm	• Khả vi tại mọi nơi

- **Softmax**

Bước softmax có thể được coi là một hàm logistic tổng quát lấy đầu vào là một vector chứa các giá trị $x \in \mathbb{R}^n$ và cho ra là một vector gồm các xác suất $p \in \mathbb{R}^n$ thông qua một hàm softmax ở cuối kiến trúc.

Huấn luyện mạng phức tạp từ đầu hay sử dụng các mạng được huấn luyện trước để nhanh chóng học các nhiệm vụ mới.

Tạo mạng sâu mới cho các nhiệm vụ phân loại và hồi quy hình ảnh bằng cách xác định kiến trúc mạng và huấn luyện mạng từ đầu. Việc tinh chỉnh mạng phân loại hình ảnh được huấn luyện trước bằng cách học chuyển tiếp thường nhanh hơn và dễ dàng hơn nhiều so với việc huấn luyện từ đầu. Sử dụng mạng sâu được huấn luyện trước

cho phép nhanh chóng tìm hiểu các tác vụ mới mà không cần xác định và huấn luyện mạng mới, có hàng triệu hình ảnh hoặc có GPU mạnh mẽ.

Sau khi xác định kiến trúc mạng, ta phải xác định các tham số huấn luyện bằng cách sử dụng hàm trainingOptions. Sau đó, có thể huấn luyện mạng bằng cách sử dụng trainNetwork. Sử dụng mạng được huấn luyện để dự đoán nhãn lớp hoặc phản hồi số.

Ta có thể huấn luyện mạng nơ-ron phức hợp trên một CPU, một GPU, nhiều CPU hoặc GPU hoặc song song trên một cụm hoặc trong đám mây. Chỉ định môi trường thực thi bằng cách sử dụng chức năng trainingOptions.

Đại học Sao Đỏ

Chương 2. STABLE DIFFUSION

2.1. Định nghĩa [1]

Stable Diffusion là một mô hình học tập sâu, tạo hình ảnh được phát hành vào năm 2022. Nó chủ yếu được sử dụng để tạo ra các hình ảnh chi tiết theo mô tả văn bản, mặc dù nó cũng có thể được áp dụng cho các tác vụ khác như inpainting, outpainting và tạo hình ảnh từ hình ảnh Bản dịch được dịch bởi text prompt.

Stable Diffusion là một Latent Diffusion Models một loạt các mạng lưới thần kinh tổng thể sâu được phát triển bởi nhóm Compvis tại LMU Munich. Mô hình đã được phát hành bởi sự hợp tác của Stability AI, Compvis LMU và Runway với sự hỗ trợ từ Eleutherai và Laion. Vào tháng 10 năm 2022, sự ổn định AI đã huy động được 101 triệu đô la Mỹ trong một vòng do Lightspeed Venture Partners và Coatue Management.

Mã số và trọng số mô hình của Stable Diffusion đã được phát hành công khai, và nó có thể chạy trên hầu hết các phần cứng tiêu dùng được trang bị GPU khiêm tốn với ít nhất 8 GB VRAM. Điều này đánh dấu sự khởi đầu từ các mô hình văn bản-hình ảnh độc quyền trước đây như Dall-E và Midjourney, chỉ có thể truy cập thông qua các dịch vụ đám mây.

2.2. Kiến trúc của Stable Diffusion

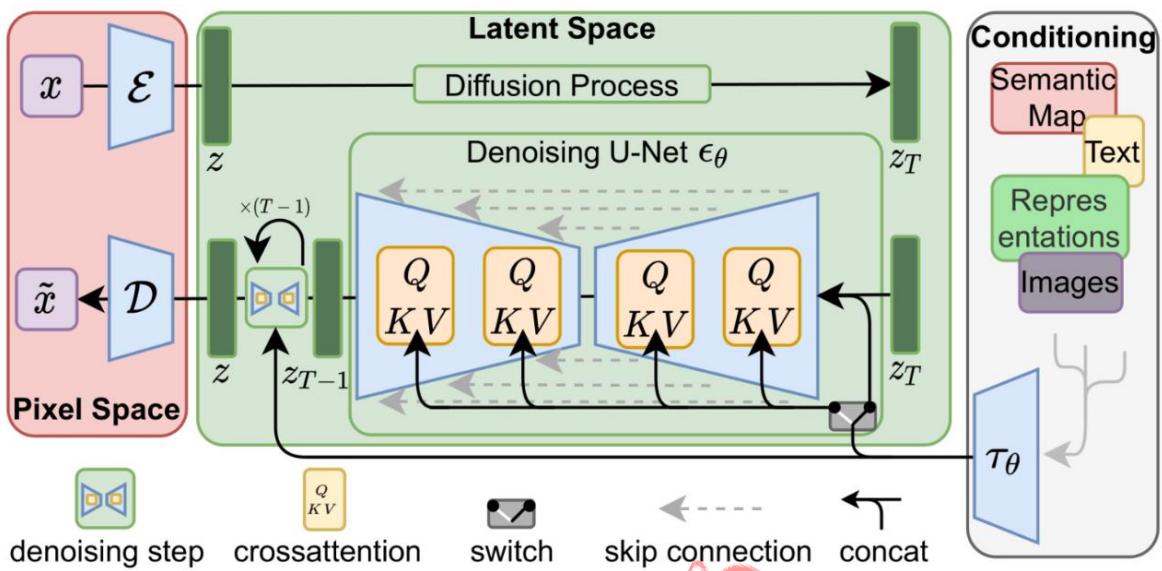
Stable Diffusion sử dụng một biến thể của mô hình khuếch tán (DM), được gọi là Latent Diffusion Models (LDM). Được giới thiệu vào năm 2015, các mô hình khuếch tán được đào tạo với mục tiêu loại bỏ các ứng dụng liên tiếp của nhiều Gaussian trên các hình ảnh đào tạo có thể được coi là một chuỗi tự động hóa giảm nhiễu. Stable Diffusion bao gồm 3 phần: bộ tự động biến thể (VAE), U-NET và bộ mã hóa văn bản tùy chọn.

Bộ mã hóa VAE nén hình ảnh từ không gian pixel đến một không gian tiềm ẩn nhỏ hơn, chụp một ý nghĩa ngữ nghĩa cơ bản hơn của hình ảnh. Nhiều Gaussian được áp dụng lặp đi lặp lại cho biểu diễn tiềm ẩn được nén trong quá trình khuếch tán chuyển tiếp.

Khối U-NET, bao gồm xương sống resnet, làm mất đi đầu ra từ khuếch tán phía trước để có được biểu diễn tiềm ẩn.

Cuối cùng, bộ giải mã VAE tạo ra hình ảnh cuối cùng bằng cách chuyển đổi biểu diễn trở lại không gian pixel. Bước khử nhiễu có thể có các điều kiện trên một chuỗi văn bản, hình ảnh và các phương thức khác. Dữ liệu điều kiện được mã hóa được tiếp xúc với các mạng U phân hủy thông qua cơ chế tham gia chéo.

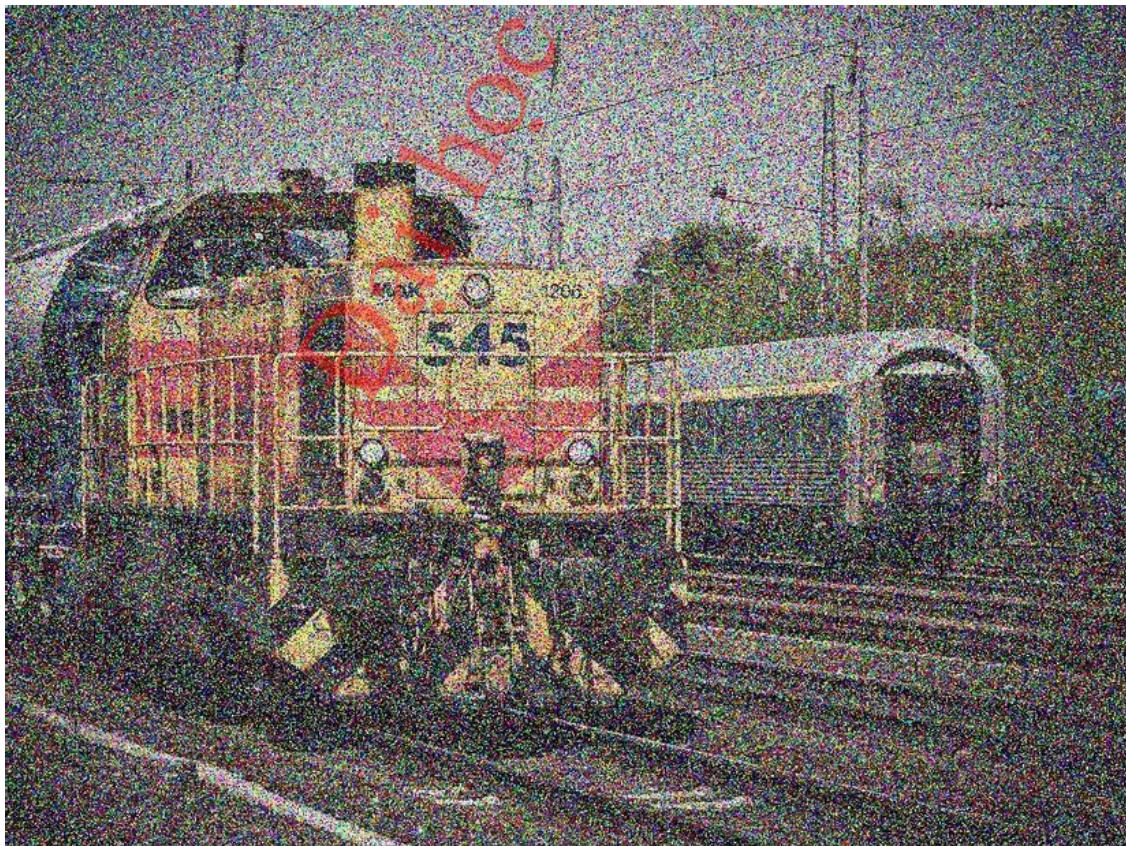
Bộ sử dụng điều kiện trên văn bản, bộ mã hóa văn bản clip VIT-L/14 cố định được sử dụng để chuyển đổi lời nhắc văn bản thành một không gian nhúng. Các nhà nghiên cứu chỉ ra tăng hiệu quả tính toán để đào tạo và tạo ra như một lợi thế của LDM.



Hình 2.1. Kiến trúc của Stable Diffusion[1]

2.3. Nhiễu

Nhiễu có thể hiểu là hiện tượng méo tín hiệu cơ bản gây cản trở quá trình quan sát hình ảnh và trích xuất thông tin. Với sự gia tăng mạnh mẽ của việc tạo ra hình ảnh kỹ thuật số thường được chụp trong điều kiện không khí/ánh sáng kém, các phương pháp khôi phục hình ảnh đã trở thành công cụ không thể thiếu trong kỹ nguyên phân tích có sự hỗ trợ của máy tính.



Hình 2.2. Nhiễu trong ảnh

2.3.1. Các loại nhiễu ảnh

Nhiễu Gauss: Nhiễu này có được do bản chất rò rỉ rác của bức xạ (hệ thống ghi ảnh bằng cách đếm các photon (lượng tử ánh sáng). Mỗi pixel trong ảnh nhiễu là tổng giá trị pixel đúng và pixel ngẫu nhiên.

Nhiễu muối – tiêu (Salt & Pepper noise): Nhiễu này sinh ra do xảy ra sai số trong quá trình truyền dữ liệu. Những pixel đơn được đặt luân phiên mang giá trị bằng 0 hay giá trị cực đại tạo ra hình chấm dạng muối tiêu trên ảnh.

Nhiễu Shot hay nhiễu Poisson: Nhiễu này sinh ra do trong quá trình thu nhận, số lượng lớn hạt photon đã tập trung vào một điểm và chúng đã tạo ra nhiễu tại điểm đó. Nhiễu được đặc trưng bởi hàm mật độ phân bố xác suất Poisson, nên được gọi là nhiễu Poisson

Nhiễu Speckle hay nhiễu đốm: Là loại nhiễu phát sinh do ảnh hưởng của điều kiện môi trường lên cảm biến hình ảnh trong quá trình thu nhận hình ảnh. Nhiễu lốm đốm hầu như được phát hiện trong trường hợp ảnh y tế, ảnh Radar hoạt động và ảnh Radar khẩu độ tổng hợp (SAR).

2.3.2. Các phương pháp khử nhiễu

Vì bộ lọc là một phương tiện chính để xử lý hình ảnh, một số lượng lớn các bộ lọc không gian đã được áp dụng để làm giảm hình ảnh, có thể được phân loại thêm thành hai loại như sau:

Lọc tuyến tính: Các bộ lọc tuyến tính thường được sử dụng để loại bỏ nhiễu trong miền không gian, nhưng chúng không giữ được kết cấu hình ảnh

- Bộ lọc trung bình đã được áp dụng để giảm nhiễu Gaussian, tuy nhiên, nó có thể làm mịn ảnh quá mức với nhiễu cao.

- Bộ lọc Wiener đã được sử dụng nhằm khắc phục nhược điểm của lọc trung bình, nhưng nó cũng có thể dễ dàng làm mờ các cạnh.

Lọc phi tuyến tính:

- Bộ lọc trung vị và bộ lọc trung vị có trọng số có thể được loại bỏ nhiễu mà không cần bất kỳ phương pháp nhận dạng nào.

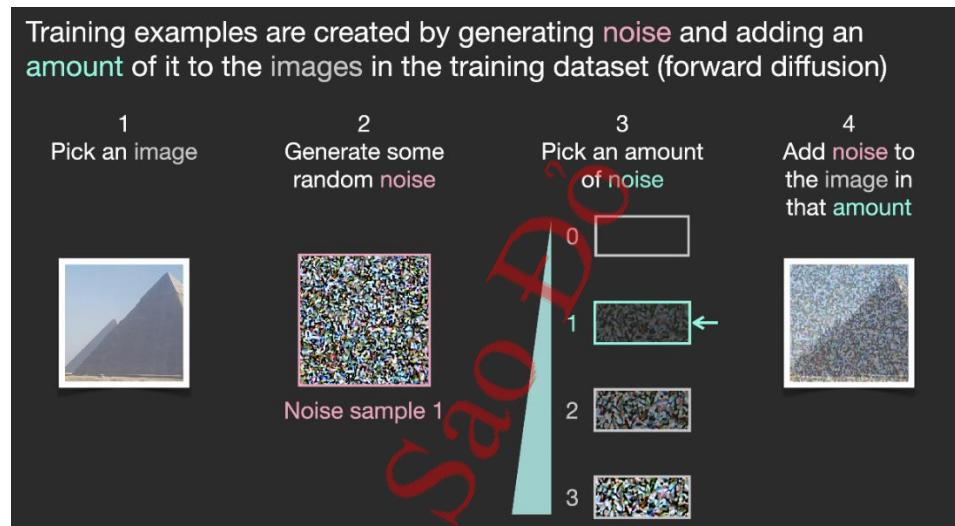
- Bộ lọc Bilateral được sử dụng rộng rãi để làm nhiễu hình ảnh do bản chất là bộ lọc làm mịn không tuyến tính, bảo toàn cạnh và giảm nhiễu. Giá trị cường độ của mỗi pixel được thay thế bằng giá trị cường độ trung bình có trọng số từ các pixel lân cận. Tuy vậy bộ lọc này có nhược điểm rằng việc thực hiện brute-force của nó mất khá nhiều thời gian $O(Nr^2)O(Nr^2)$, cực kỳ cao khi bán kính kernel lớn.

2.4. Cách hoạt động

Ý tưởng trung tâm của việc tạo ra hình ảnh với các diffusion models phụ thuộc vào thực tế là có các mô hình tầm nhìn máy tính mạnh mẽ. Dựa ra một bộ dữ liệu đủ lớn, các mô hình này có thể học các hoạt động phức tạp. Các Diffusion models tạo hình ảnh bằng cách đóng khung vấn đề như sau:

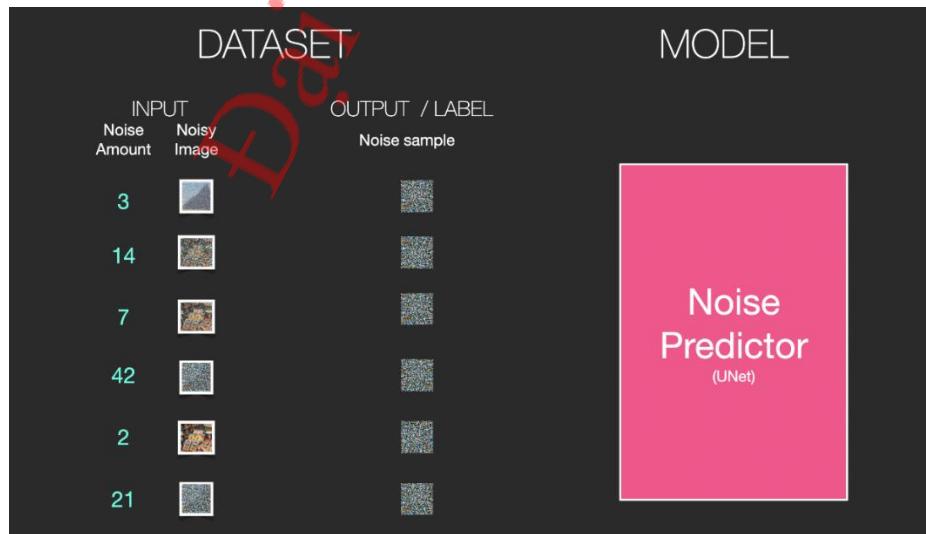
- Bước đầu có một hình ảnh, tạo ra nhiều và thêm nó vào hình ảnh.
- Sử dụng cùng một công thức này để tạo ra nhiều ví dụ đào tạo nhằm đào tạo thành phần trung tâm của mô hình.

Mặc dù ví dụ này hiển thị một số giá trị lượng nhiễu từ hình ảnh (amount 0, không có nhiễu) đến tổng nhiễu (amount 4, tổng nhiễu), vì vậy có thể dễ dàng kiểm soát lượng nhiễu cần thêm vào hình ảnh và do đó có thể trải rộng nó trên hàng chục gồm nhiều bước, tạo hàng chục ví dụ huấn luyện cho mỗi hình ảnh cho tất cả các hình ảnh trong tập dữ liệu huấn luyện.



Hình 2.3. Thêm nhiễu vào ảnh

Có thể dễ dàng kiểm soát mức độ nhiễu để thêm vào hình ảnh và vì vậy có thể lan truyền nó trên hàng chục của các bước, tạo hàng chục ví dụ đào tạo cho mỗi hình ảnh cho tất cả các hình ảnh trong một bộ dữ liệu đào tạo.

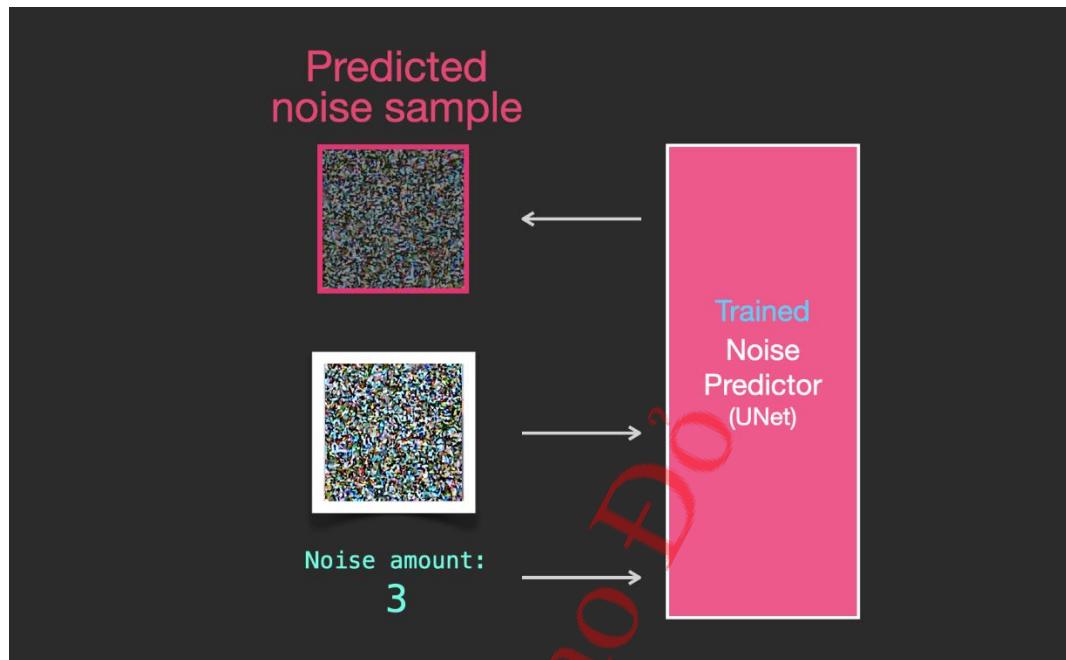


Hình 2.4. Minh họa tạo tập dữ liệu

Sử dụng tập dữ liệu này, để huấn luyện bộ dự đoán nhiễu và tạo ra một bộ dự đoán nhiễu cho mô hình sau này.

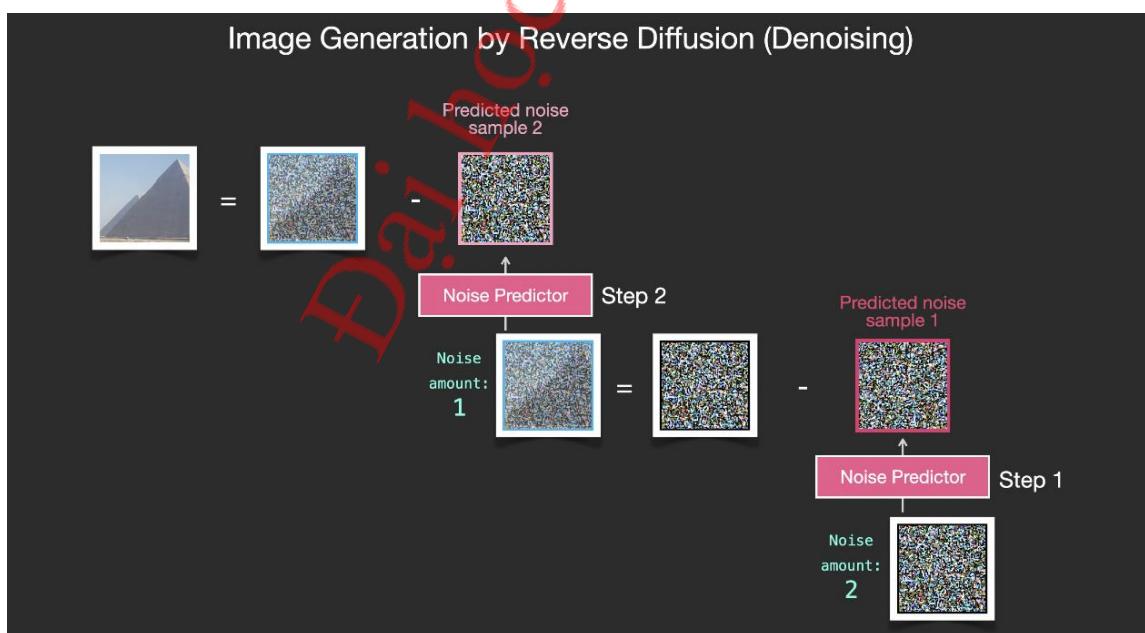
- Tạo hình ảnh bằng cách loại bỏ nhiễu

Bộ dự đoán lượng nhiễu đã được đào tạo có thể tạo ra lượng nhiễu và số bước khử nhiễu, đồng thời có thể dự đoán một phần nhiễu.



Hình 2.5. Minh họa dự đoán nhiễu với lượng nhiễu 3

Trình dự đoán nhiễu được đào tạo có thể thấy được nhiễu và số lượng bước khử nhiễu và có thể dự đoán một phần nhỏ nhiễu.



Hình 2.6. Minh họa khử nhiễu ảnh

Nhiễu được lấy mẫu được dự đoán để nếu trừ nhiễu khỏi hình ảnh, sẽ nhận được một hình ảnh gần hơn với hình ảnh mà mô hình được đào tạo (không phải là hình ảnh chính xác, mà là bản phân phối - sắp xếp pixel nơi bầu trời thường Màu xanh và trên

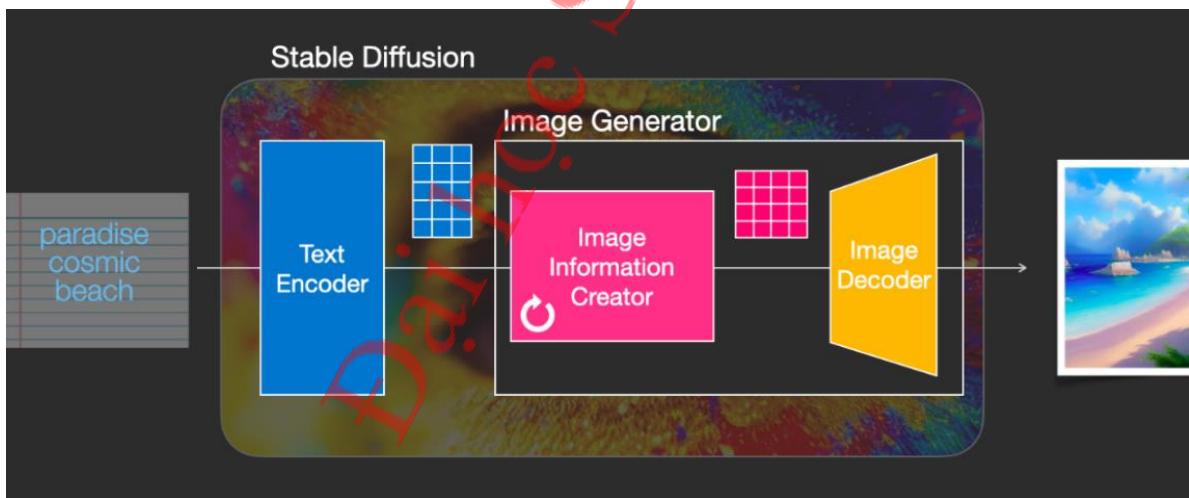
mặt đất, mọi người có hai mắt, mèo nhìn theo một cách nhất định - đôi tai nhọn và rõ ràng không bị ẩn tượng).

Nếu tập dữ liệu huấn luyện là các hình ảnh có tính thẩm mỹ cao (ví dụ: LAION Aesthetics, mà mô hình đã được đào tạo), thì hình ảnh thu được sẽ có xu hướng đẹp về mặt thẩm mỹ. Nếu đào tạo mô hình trên hình ảnh của logo, kết quả mô hình thu được sẽ tạo ra một mô hình tạo logo.

2.5. Các thành phần của Stable Diffusion

2.5.1. Bộ tạo thông tin hình ảnh

Thành phần này là bí mật của Stable Diffusion. Đó là nơi đạt được rất nhiều hiệu suất so với các mô hình trước đó đạt được. Thành phần này chạy cho nhiều bước để tạo thông tin hình ảnh. Đây là tham số các bước trong các giao diện và thư viện khuếch tán ổn định thường mặc định là 50 hoặc 100. Trình tạo thông tin hình ảnh hoạt động hoàn toàn trong không gian thông tin hình ảnh (hoặc không gian tiềm ẩn). Thuộc tính này làm cho nó nhanh hơn các mô hình khuếch tán trước đây hoạt động trong không gian pixel. Về mặt kỹ thuật, thành phần này được tạo thành từ một mạng lưới thần kinh unet và thuật toán lập lịch. Từ "Diffusion" mô tả những gì xảy ra trong thành phần này. Đó là cách xử lý từng bước thông tin dẫn đến một hình ảnh chất lượng cao được tạo ra cuối cùng (bởi thành phần tiếp theo, bộ giải mã hình ảnh).



Hình 2.7. Minh họa bộ tạo ảnh

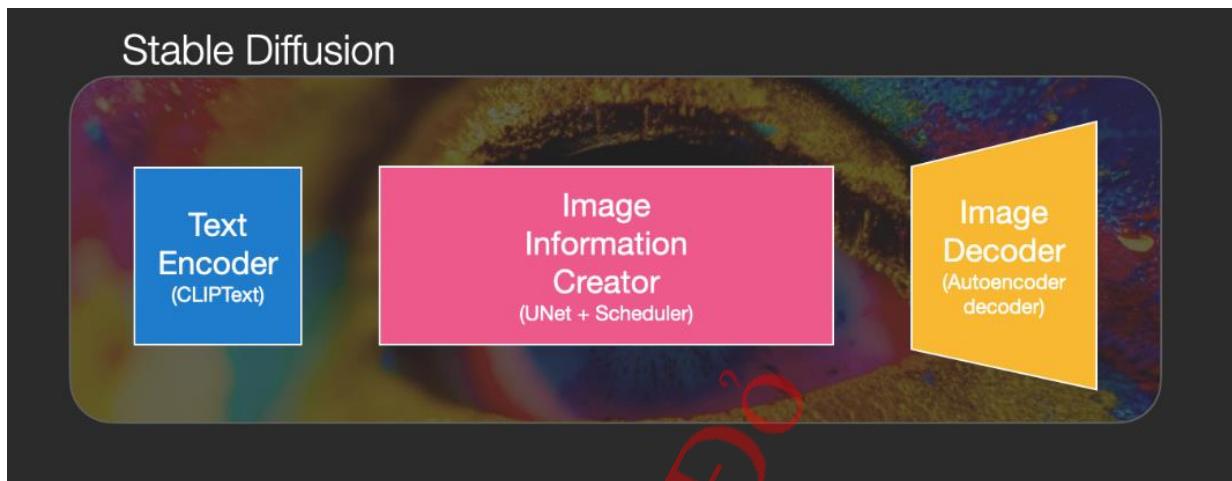
2.5.2. Bộ giải mã hình ảnh

Bộ giải mã hình ảnh vẽ một hình ảnh từ thông tin nó nhận được từ bộ tạo thông tin. Nó chỉ chạy một lần vào cuối quá trình để tạo ra hình ảnh pixel cuối cùng.

Bộ giải mã hình ảnh ba thành phần chính (mỗi thành phần có mạng lưới thần kinh riêng) tạo nên Stable Diffusion:

- ClipText cho mã hóa văn bản. Nhập ký tự. Đầu ra: 77 vectơ nhúng mã thông báo, mỗi thứ 768 kích thước.

- Unet + lập lịch trình để xử lý/khuếch tán thông tin dần dần trong không gian thông tin (tiềm ẩn). Đầu vào: Nhúng văn bản và một mảng đa chiều bắt đầu (danh sách các số có cấu trúc, còn được gọi là tensor) được tạo thành từ nhiều. Đầu ra: Bộ giải mã tự động mảng



Hình 2.8. Minh họa bộ giải mã ảnh

U-NET có phần bộ mã hóa và một phần bộ giải mã bao gồm các khối Resnet. Bộ mã hóa nén một biểu diễn hình ảnh vào biểu diễn hình ảnh độ phân giải thấp hơn và bộ giải mã giải mã biểu diễn hình ảnh độ phân giải thấp hơn trở lại biểu diễn hình ảnh độ phân giải cao hơn ban đầu được cho là ít nhiễu hơn. Cụ thể hơn, đầu ra U-NET dự đoán phần dư nhiễu có thể được sử dụng để tính toán biểu diễn hình ảnh từ chối dự đoán. Để ngăn chặn U-NET mất thông tin quan trọng trong khi lấy mẫu xuống, các kết nối cắt ngắn thường được thêm vào giữa các bộ dữ liệu lấy mẫu của bộ mã hóa vào các bộ dữ liệu lấy mẫu của bộ giải mã.

Ngoài ra, U-NET khuếch tán ổn định có thể điều chỉnh đầu ra của nó trên các phần tử văn bản thông qua các lớp tập thể chéo. Các lớp tập thể chéo được thêm vào cả bộ mã hóa và bộ giải mã của U-NET thường giữa các khối Resnet.

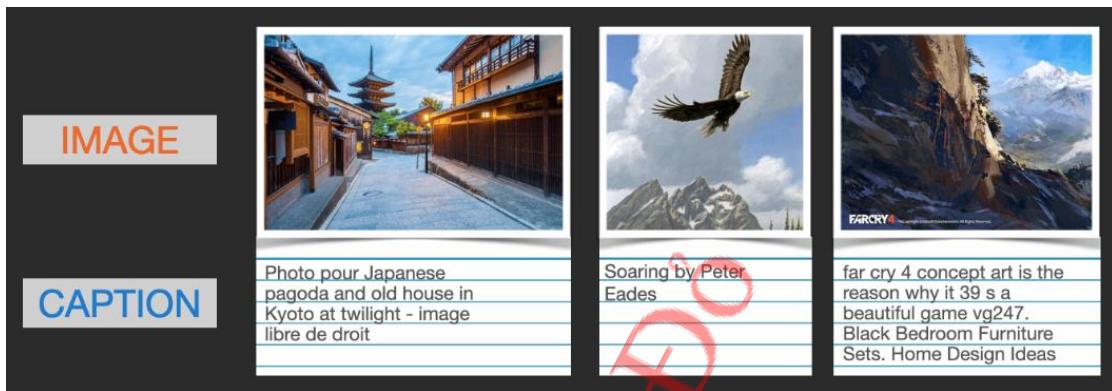
- Auto encoder Decoder: vẽ hình ảnh cuối cùng bằng cách sử dụng mảng thông tin được xử lý. Đầu vào: Mảng thông tin được xử lý (Kích thước: (4,64,64)) Đầu ra: Hình ảnh kết quả (Kích thước: (3, 512, 512) là (red/green/blue, width, height))

Mô hình VAE có hai phần, bộ mã hóa và bộ giải mã. Bộ mã hóa được sử dụng để chuyển đổi hình ảnh thành biểu diễn tiềm ẩn chiết thấp, sẽ đóng vai trò là đầu vào cho mô hình U-NET. Bộ giải mã, ngược lại, biến đổi biểu diễn tiềm ẩn trở lại thành một hình ảnh. Trong quá trình đào tạo khuếch tán tiềm ẩn, bộ mã hóa được sử dụng để có được các biểu diễn tiềm ẩn (tọa độ) của hình ảnh cho quá trình khuếch tán phía trước, áp dụng ngày càng nhiều tiếng ồn ở mỗi bước. Trong quá trình suy luận, các tọa độ bị khử được tạo ra bởi quá trình khuếch tán ngược được chuyển đổi trở lại thành hình ảnh bằng bộ giải mã VAE. Như chúng ta sẽ thấy trong quá trình suy luận, chúng ta chỉ cần bộ giải mã VAE.

2.5.3. Bộ mã hóa văn bản A Transformer Language Model

A Transformer Language Model: được sử dụng làm thành phần hiểu ngôn ngữ lấy dấu nhắc văn bản và tạo ra các nhúng mã thông báo. Diffusion Models ổn định được phát hành sử dụng cliptext (một mô hình dựa trên GPT).

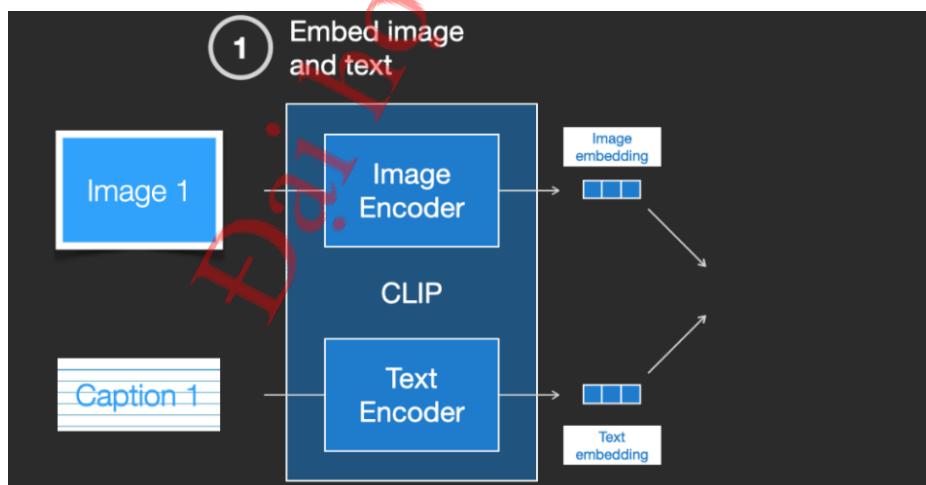
Clip được đào tạo trên một bộ dữ liệu hình ảnh và chú thích của chúng, chỉ với 400 triệu hình ảnh và chú thích của chúng:



Hình 2.9. Dữ liệu đào tạo clip

Clip là sự kết hợp của bộ mã hóa hình ảnh và bộ mã hóa văn bản. Quá trình đào tạo của nó có thể được đơn giản hóa để nghĩ đến việc chụp ảnh và chú thích của nó.

Clip là sự kết hợp giữa bộ mã hóa hình ảnh và bộ mã hóa văn bản. Quá trình đào tạo của nó có thể được đơn giản hóa bằng cách nghĩ đến việc chụp một bức ảnh và chú thích của nó. Chúng tôi mã hóa cả hai bằng bộ mã hóa hình ảnh và văn bản tương ứng.



Hình 2.10. Cấu trúc của mô hình clip

Bộ mã hóa văn bản chịu trách nhiệm chuyển đổi prompt đầu vào, ví dụ: "Một Astronaut cưỡi một con ngựa" vào một không gian nhúng có thể được hiểu bởi U-net. Nó thường là một bộ mã hóa dựa trên máy biến áp đơn giản, ánh xạ một chuỗi các mã thông báo đầu vào thành một chuỗi các tác phẩm truyền văn bản tiềm ẩn. Lấy cảm hứng từ Imagen, khuếch tán ổn định không đào tạo bộ mã hóa văn bản trong quá trình đào tạo và chỉ cần sử dụng trình mã hóa văn bản đã được đào tạo của clip, cliptextmodel.

2.6. Mạng UNET [2]

2.6.1. Giới thiệu về mạng UNET

Kiến trúc U-Net, được xuất bản lần đầu tiên vào năm 2015, đã là một cuộc cách mạng trong lĩnh vực học sâu. Kiến trúc đã giành chiến thắng trong Hội nghị chuyên đề quốc tế về thách thức theo dõi tế bào Y sinh (ISBI) năm 2015 ở nhiều hạng mục với tỷ lệ chênh lệch lớn. Một số công trình của họ bao gồm phân đoạn cấu trúc tế bào thần kinh trong các ngăn xếp hiển vi điện tử và hình ảnh hiển vi ánh sáng truyền qua.

Với kiến trúc U-Net này, việc phân đoạn hình ảnh có kích thước 512X512 có thể được tính toán bằng GPU hiện đại trong một khoảng thời gian nhỏ. Đã có nhiều biến thể và sửa đổi của kiến trúc này do thành công phi thường của nó. Một số trong số chúng bao gồm LadderNet, U-Net với sự chú ý, U-Net tích hợp lặp lại và dư (R2-UNet) và U-Net với các khối còn lại hoặc các khối có kết nối dày đặc.

Mặc dù U-Net là một thành tựu đáng kể trong lĩnh vực học sâu, nhưng điều cần thiết không kém là phải hiểu các phương pháp trước đây đã được sử dụng để giải quyết loại nhiệm vụ tương tự như vậy. Một trong những ví dụ chính cuối cùng là phương pháp cửa sổ trượt, đã chiến thắng thách thức phân đoạn EM tại ISBI vào năm 2012 với tỷ suất lợi nhuận lớn. Cách tiếp cận cửa sổ trượt có thể tạo ra một loạt các bản vá mẫu ngoài tập dữ liệu đào tạo ban đầu.

Kết quả này là do nó đã sử dụng phương pháp thiết lập mạng kiến trúc cửa sổ trượt bằng cách tạo nhãn lớp của mỗi pixel thành các đơn vị riêng biệt bằng cách cung cấp một vùng cục bộ (bản vá) xung quanh pixel đó. Một thành tựu khác của kiến trúc này là nó có thể bản địa hóa khá dễ dàng trên bất kỳ tập dữ liệu đào tạo nào cho các nhiệm vụ tương ứng.

Tuy nhiên, cách tiếp cận cửa sổ trượt có hai nhược điểm chính mà kiến trúc U-Net đã khắc phục. Vì mỗi pixel được xem xét riêng biệt, nên các bản vá kết quả chúng tôi chồng chéo lên nhau rất nhiều. Do đó, rất nhiều dự phòng tổng thể đã được tạo ra. Một hạn chế khác là quy trình đào tạo tổng thể khá chậm và tiêu tốn nhiều thời gian và nguồn lực. Tính khả thi của hoạt động của mạng bị nghi ngờ do những lý do sau.

U-Net là một kiến trúc thanh lịch giải quyết hầu hết các vấn đề xảy ra. Nó sử dụng khái niệm mạng tích tụ đầy đủ cho cách tiếp cận này. Mục đích của U-Net là nắm bắt cả các đặc điểm của ngữ cảnh cũng như bản địa hóa. Quá trình này được hoàn thành thành công bởi kiểu kiến trúc được xây dựng. Ý tưởng chính của việc triển khai là sử dụng các lớp hợp đồng kế tiếp, được theo sau bởi các toán tử upsampling để đạt được kết quả đầu ra có độ phân giải cao hơn trên các hình ảnh đầu vào.

2.6.2. Ý tưởng đằng sau UNET

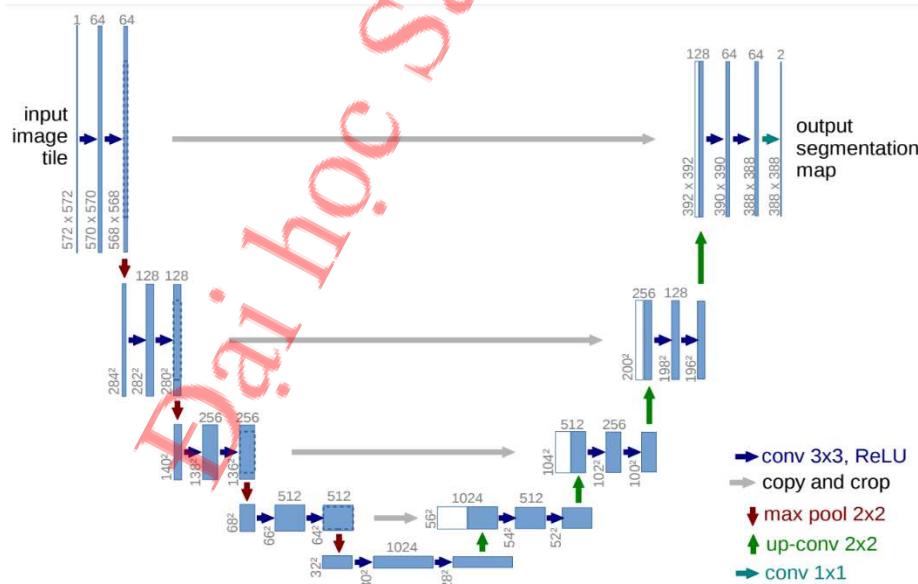
Ý tưởng chính đằng sau CNN là học feature mapping của hình ảnh và dùng nó để tạo ra feature mapping mức cao hơn. Điều này hoạt động tốt trong các bài toán phân

loại vì hình ảnh được chuyển đổi thành một vector sử dụng thêm để phân loại. Nhưng trong phân vùng hình ảnh, chúng ta không chỉ cần chuyển đổi feature map thành một vector mà còn tái tạo lại một hình ảnh từ vector này. Đây là một tác vụ khó. Toàn bộ ý tưởng của UNet được xoay quanh vấn đề này.

Khi chuyển đổi một hình ảnh thành một vector, chúng ta đã học feature mapping của hình ảnh, vậy tại sao không sử dụng cùng một mapping để chuyển đổi nó thành hình ảnh. Đây là công thức đằng sau UNet. Sử dụng các feature map tương tự được sử dụng khi tạo vector để mở rộng vector thành hình ảnh được phân vùng. Điều này sẽ bảo vệ tính toàn vẹn cấu trúc của hình ảnh, làm giảm sự biến dạng rất nhiều. Hãy tìm hiểu về kiến trúc một cách ngắn gọn hơn.

2.6.3. Kiến trúc UNET

Hình dạng của kiến trúc được hình thành có dạng chữ 'U' và do đó có tên như sau. Chỉ cần nhìn vào cấu trúc và vô số yếu tố liên quan đến quá trình xây dựng kiến trúc này, chúng ta có thể hiểu rằng mạng được xây dựng là một mạng phức hợp đầy đủ. Họ đã không sử dụng bất kỳ lớp nào khác như dày đặc hoặc làm phẳng hoặc các lớp tương tự khác. Biểu diễn trực quan cho thấy một đường dẫn hợp đồng ban đầu theo sau là một đường dẫn mở rộng.



Hình 2.11. Cấu trúc UNET

Kiến trúc cho thấy rằng một hình ảnh đầu vào được chuyển qua mô hình và sau đó nó được theo sau bởi một vài lớp phức hợp với chức năng kích hoạt ReLU. Chúng ta có thể nhận thấy rằng kích thước hình ảnh đang giảm từ 572X572 xuống 570X570 và cuối cùng là 568X568. Lý do cho việc giảm này là do họ đã sử dụng các chập không được đếm (được định nghĩa là các chập là "hợp lệ"), dẫn đến giảm kích thước tổng thể. Ngoài các khối Convolution, chúng ta cũng nhận thấy rằng chúng ta có một khối mã hóa ở phía bên trái, tiếp theo là khối bộ giải mã ở phía bên phải.

Kiến trúc này gồm 3 phần: contraction (encoder), bottleneck và expansion (decoder). Phần contraction được tạo bởi nhiều khối. Mỗi khối gồm hai lớp CNN có kernel size là 3×3 , theo sau bởi một lớp 2×2 max pooling. Số lượng kernel hay feature map tăng gấp đôi sau mỗi khối, do đó kiến trúc này có thể học những đặc trưng phức tạp một cách hiệu quả. Lớp ở vị trí thấp nhất làm trung gian giữa các lớp ở phần contraction và phần expansion. Nó dùng 2 lớp 3×3 CNN, theo sau bởi 1 với 2×2 up convolution.

Nhung trái tim của kiến trúc này nằm ở phần expansion. Tương tự như contraction, nó cũng bao gồm nhiều khối. Mỗi khối gồm hai lớp 3×3 CNN theo sau bởi một lớp upsampling. Sau mỗi khối số lượng feature map được giảm đi một nửa để đảm bảo tính đối xứng. Tuy nhiên đầu vào của mỗi khối expansion sẽ được nối thêm feature map của khối contraction tương ứng. Hành động này sẽ đảm bảo các đặc trưng được học trong quá trình contracting ảnh sẽ được dùng để tái tạo nó. Số lượng khối expansion bằng với số lượng khối contraction. Sau cùng, kết quả sẽ được đưa qua một lớp 3×3 CNN khác với số feature map bằng với số vùng mong muốn.

Khối mã hóa có sự giảm kích thước hình ảnh liên tục với sự trợ giúp của các lớp tổng hợp tối đa của bước 2. Chúng tôi cũng có các lớp tích tụ lặp lại với số lượng bộ lọc ngày càng tăng trong kiến trúc bộ mã hóa. Khi chúng tôi tiếp cận khía cạnh bộ giải mã, chúng tôi nhận thấy số lượng bộ lọc trong các lớp phức hợp bắt đầu giảm cùng với việc lấy mẫu tăng dần ở các lớp sau lên đến đầu. Chúng tôi cũng nhận thấy rằng việc sử dụng các kết nối bỏ qua kết nối các đầu ra trước đó với các lớp trong khối bộ giải mã.

Kết nối bỏ qua này là một khái niệm quan trọng để bảo toàn sự mượt mà từ các lớp trước để chúng phản ánh mạnh mẽ hơn về các giá trị tổng thể. Chúng cũng được khoa học chứng minh là tạo ra kết quả tốt hơn và dẫn đến sự hội tụ mô hình nhanh hơn. Trong khối tích chập cuối cùng, chúng ta có một vài lớp tích chập theo sau là lớp tích chập cuối cùng. Lớp này có bộ lọc 2 với chức năng thích hợp để hiển thị kết quả đầu ra. Lớp cuối cùng này có thể được thay đổi tùy theo mục đích mong muốn của dự án mà bạn đang cố gắng thực hiện.

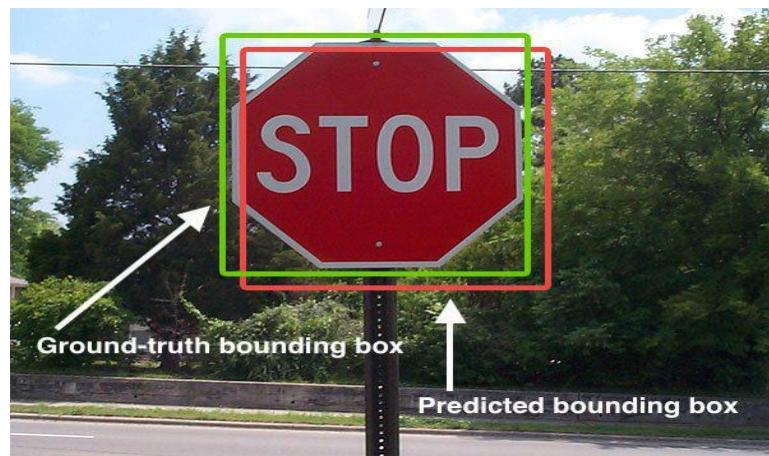
2.6.4. Tính toán loss, IoU và DSC

a) Intersection over Union (IoU)

Intersection over Union là một số liệu đánh giá được sử dụng để đo độ chính xác của bộ phát hiện đối tượng trên một tập dữ liệu cụ thể.

Intersection over Union chỉ đơn giản là một thước đo đánh giá. Bất kỳ thuật toán nào cung cấp các hộp giới hạn dự đoán dưới dạng đầu ra đều có thể được đánh giá bằng IoU.

Dưới đây, một ví dụ trực quan về hộp giới hạn sự thật cơ bản so với hộp giới hạn dự đoán:



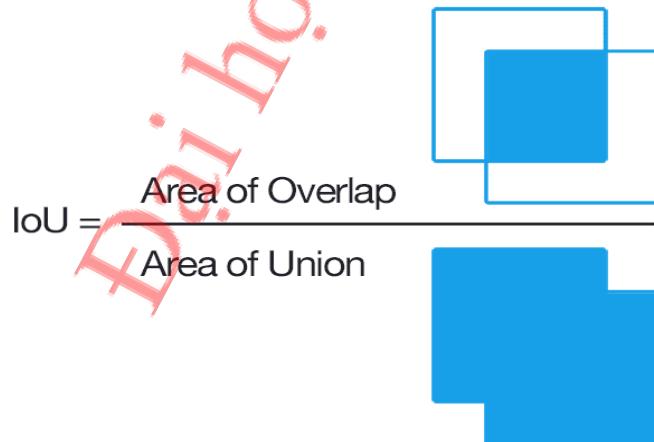
Hình 2.12. Ví dụ về phát hiện biển báo dừng

Hộp giới hạn dự đoán được vẽ bằng màu đỏ trong khi hộp giới hạn chân lý nền (nghĩa là được gán nhãn bằng tay) được vẽ bằng màu xanh lục.

Do đó, Giao điểm tính toán qua Union có thể được xác định thông qua:

IoU là diện tích trùng lặp giữa phân đoạn được dự đoán và sự thật cơ bản chia cho diện tích hợp nhất giữa phân đoạn được dự đoán và sự thật cơ bản , như được hiển thị trên hình bên trái. Chỉ số này nằm trong khoảng từ 0–1 (0–100%) với 0 biểu thị không có chồng chéo và 1 biểu thị phân đoạn chồng chéo hoàn hảo.

Đối với phân đoạn nhị phân (hai lớp) hoặc nhiều lớp , IoU trung bình của hình ảnh được tính bằng cách lấy IoU của mỗi lớp và lấy trung bình của chúng . (Nó được triển khai hơi khác trong mã).



Hình 2.13. Công thức tính IoU

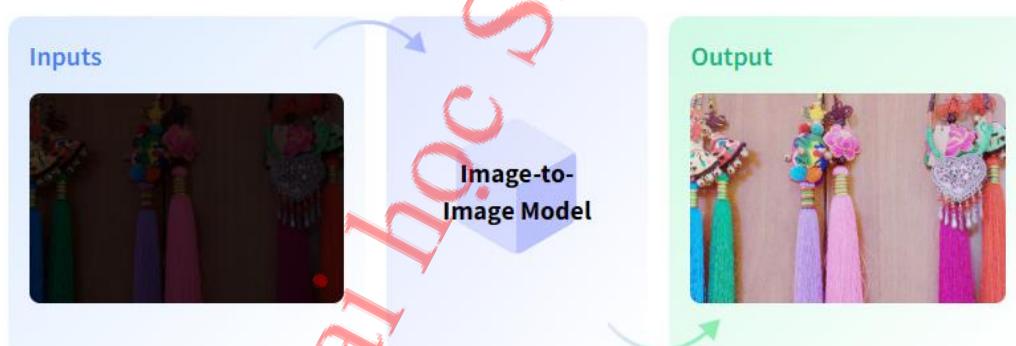
Chương 3. XÂY DỰNG CHƯƠNG TRÌNH

3.1. Mô tả bài toán

Bài toán text-to-image nói riêng đang là lĩnh vực đang được các công nghệ cũng như giới khoa học đặc biệt quan tâm. Các mô hình text-to-image model có khả năng sáng tạo ra những hình ảnh rất thú vị từ đoạn văn bản mô tả cho trước. Những mô hình là công cụ hỗ trợ con người rất nhiều trong việc xây dựng những content hình ảnh và nhiều ứng dụng khác.

Những năm trước đã thấy rất nhiều tiến bộ trong các mô hình có thể tạo ra những hình ảnh ngày càng tốt hơn (và thực tế hơn) với chủ thích bằng văn bản, nhưng với mức độ chưa từng có của phạm vi và tính linh hoạt trong nội dung và phong cách của các hình ảnh mà nó có thể tạo ra, Dall-e là đỉnh cao của những gì người ta có thể mong đợi từ mô hình tạo hình ảnh văn bản, một năm sau đó, Dall-e và các mô hình khác với tốc độ xử lý chậm hơn và khả năng xử lý có nhiều hạn chế.

Chương trình này có thể tiếp nhận mô tả bằng ngôn ngữ viết và tạo ra một hình ảnh hoàn toàn độc đáo bằng thuật toán AI. Ví dụ: đầu vào “một con cáo trên cây” sẽ trả về bức ảnh con cáo đang ngồi trên cây” hoặc “phi hành gia với chiếc bánh mì trên tay” sẽ có kết quả sẽ đúng như những gì người dùng đang hình dung trong đầu..



Hình 3.1. Hình ảnh minh họa bài toán

Khi phân tích sự tương đồng giữa các mô hình mới nhất: một cách tiếp cận hoàn toàn mới lạ đối với việc tạo hình ảnh văn bản thông qua việc sử dụng các mô hình Stable Diffusion với những thứ vượt trội hơn hẳn. Nó đang trả về kết quả chính xác trong khi giữ thời gian phản hồi khá thấp. Mô hình tạo ra hình ảnh vô điều kiện, inpainting và siêu phân giải, trong khi giảm đáng kể các yêu cầu tính toán so với DMS dựa trên pixel.

3.2. Dữ liệu đào tạo

Stable Diffusion được đào tạo trên các cặp hình ảnh và chủ thích được lấy từ Laion-5b, một bộ dữ liệu có sẵn công khai có nguồn gốc từ dữ liệu thu thập thông thường được loại bỏ khỏi web, trong đó các cặp văn bản hình ảnh 5 tỷ được phân loại dựa trên ngôn ngữ, được lọc vào các bộ dữ liệu riêng biệt theo độ phân giải, Khả năng dự đoán có chứa hình mờ và dự đoán điểm "thảm mỹ" (ví dụ: chất lượng thị giác chủ quan).

Bộ dữ liệu được tạo bởi Laion, một tổ chức phi lợi nhuận của Đức nhận tài trợ từ sự ổn định AI. Mô hình khuếch tán ổn định được đào tạo trên ba tập hợp con của Laion-5b: Laion2b-en, Laion-High-Resolution và Laion-Aesthetic V2 5+.

Một phân tích của bên thứ ba về dữ liệu đào tạo của mô hình đã xác định rằng trong số một tập hợp nhỏ hơn 12 triệu hình ảnh được lấy từ bộ dữ liệu rộng hơn được sử dụng, khoảng 47% kích thước mẫu của hình ảnh đến từ 100 miền khác nhau, với Pinterest chiếm 8,5% của tập hợp con, tiếp theo là các trang web như WordPress, BlogSpot, Flickr, Deviantart và Wikimedia Commons.

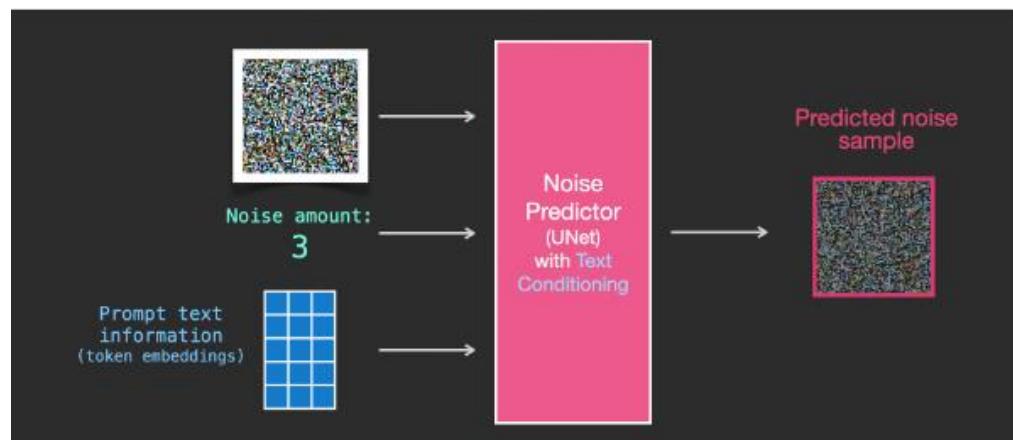


Hình 3.2. Kết quả tạo văn bản thành hình ảnh trên LAION-Aesthetic



Hình 3.3. Tạo hình ảnh có điều kiện bằng văn bản

Để biến văn bản thành một phần của quá trình tạo hình ảnh, phải điều chỉnh bộ dự đoán nhiễu để sử dụng văn bản làm đầu vào.



Hình 3.4. Điều chỉnh bộ dự đoán nhiễu

Tập dữ liệu bao gồm văn bản được mã hóa. Vì đang hoạt động trong latent space nên cả hình ảnh đầu vào và nhiễu dự đoán đều nằm trong không gian này.



Hình 3.5. Tập dữ liệu đầu vào

3.3. Công cụ sử dụng

- Google Colab: Sử dụng làm back end
- Python: Cung cấp thư viện Machine Learning
- Thư viện gradio: Xây dựng giao diện website
- Ngrok: Công cụ proxy ngược, mở các đường hầm an toàn từ các URL công cộng đến Localhost.
- Thư viện Pytorch: Framework machine learning mã nguồn mở.
- Stable Diffusion: Mã nguồn mở text to image.

3.3.1. Sử dụng Visual Studio Code với Github

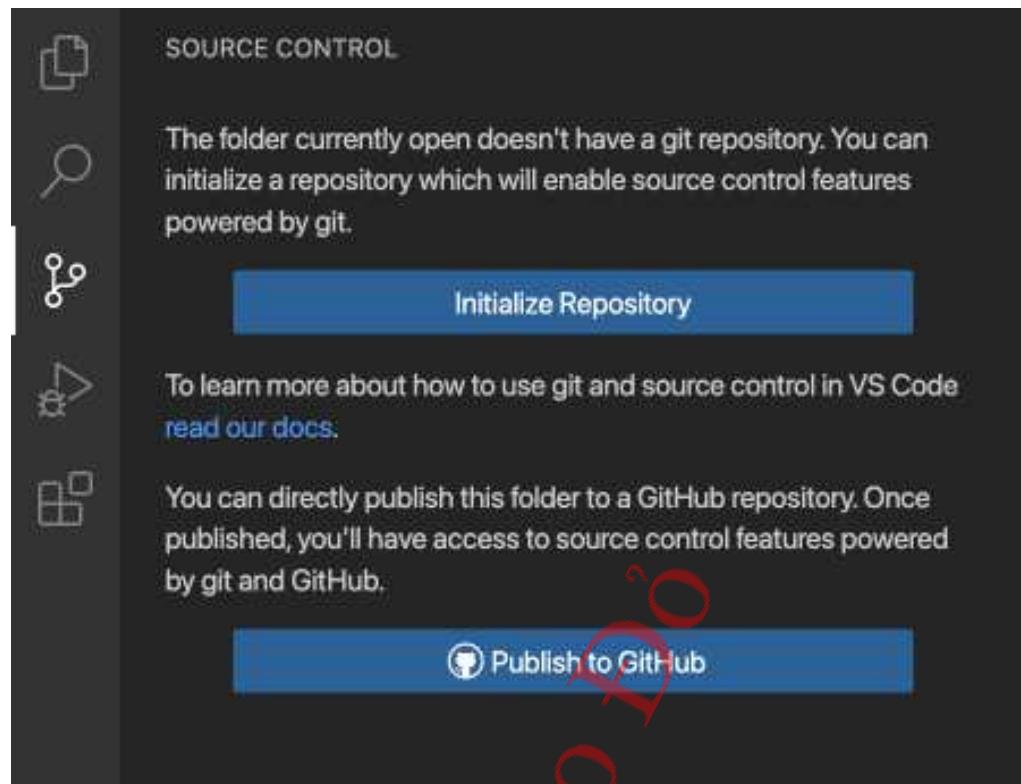
GitHub là một dịch vụ dựa trên đám mây để lưu trữ và chia sẻ mã nguồn. Sử dụng GitHub với Visual Studio Code cho phép chia sẻ mã nguồn của và cộng tác với những người khác ngay trong trình màn hình soạn thảo code. Có nhiều cách để tương tác với GitHub, chẳng hạn như thông qua trang web tại <https://github.com> hoặc giao diện dòng lệnh Git (CLI), nhưng trong VS Code, khả năng tích hợp GitHub phong phú được cung cấp bởi yêu cầu kéo GitHub và phần mở rộng các tác vụ tương tác tiện lợi.

Để sử dụng Git và GitHub trong VS Code, trước tiên cần có Git trên máy tính. Nếu thiếu Git, chế độ xem Kiểm soát nguồn sẽ hiển thị hướng dẫn về cách cài đặt nó.

Ngoài ra, bạn có thể đăng nhập vào VS Code bằng tài khoản GitHub trong menu Tài khoản ở phía dưới bên phải của thanh Hoạt động để bật các tính năng bổ sung như Đồng bộ hóa cài đặt, đồng thời sao chép và xuất bản các kho lưu trữ từ GitHub.

a) Khởi tạo kho lưu trữ trong thư mục cục bộ

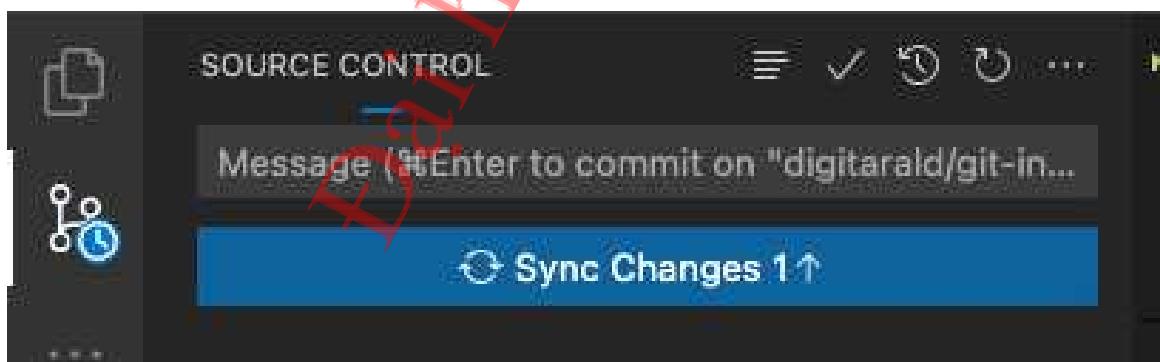
Để khởi tạo kho lưu trữ cục bộ mới, trong thư mục hiện có hoặc thư mục mới và mở trong VS Code. Trong chế độ xem kiểm soát nguồn, chọn “Initialize Repository”. Điều này tạo ra một kho lưu trữ Git mới trong thư mục hiện tại.



Hình 3.6. Khởi tạo mã nguồn cục bộ mới trên github

b) Thực hiện đẩy và di chuyển từ xa

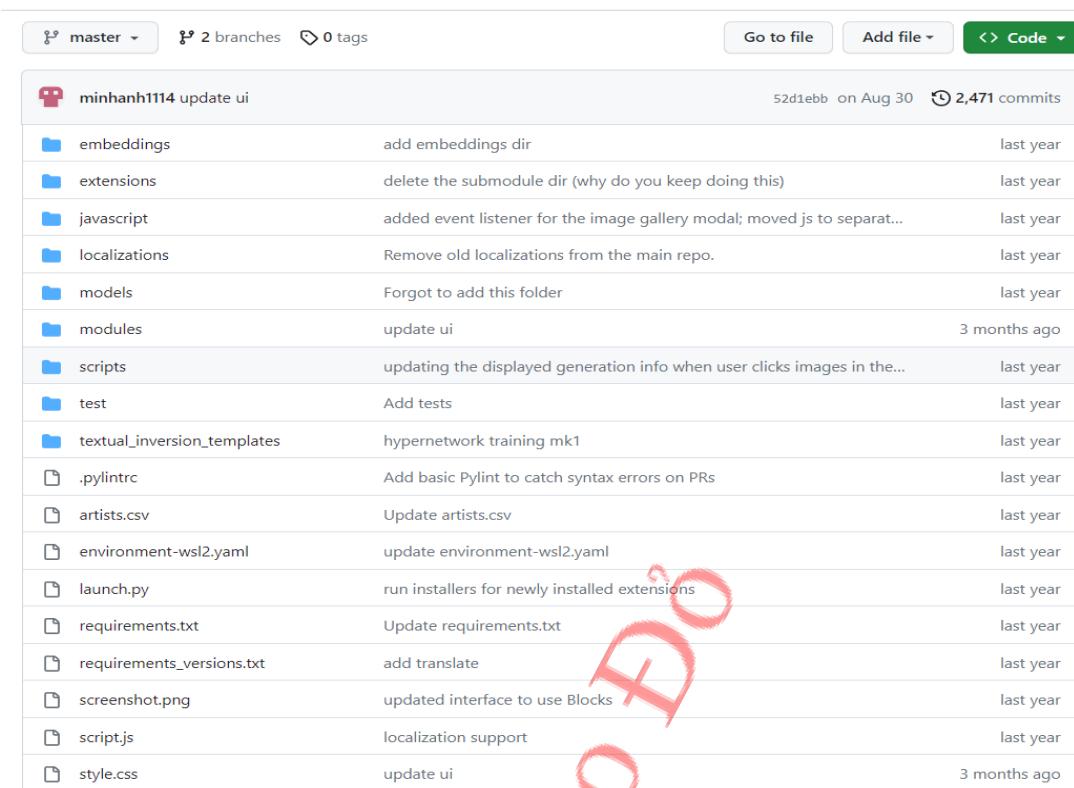
Khi thực hiện xác nhận với kho lưu trữ Git cục bộ của mình, có thể đẩy chúng sang kho lưu trữ từ xa. Nút đồng bộ hóa thay đổi cho biết có bao nhiêu cam kết sẽ được đẩy và di chuyển lên git. Chọn nút Đồng bộ hóa Thay đổi sẽ tải xuống (kéo) mọi cam kết từ xa mới và tải lên (đẩy) các cam kết cục bộ mới vào kho lưu trữ từ xa.



Hình 3.7. Đồng bộ và đẩy mã nguồn

c) Truy cập kho lưu trữ github

Sau khi thực hiện di chuyển code lên Github truy cập <https://github.com/> với tài khoản đã di chuyển code lên kiểm tra nhánh đã di chuyển thực hiện sao chép đường dẫn mã nguồn.



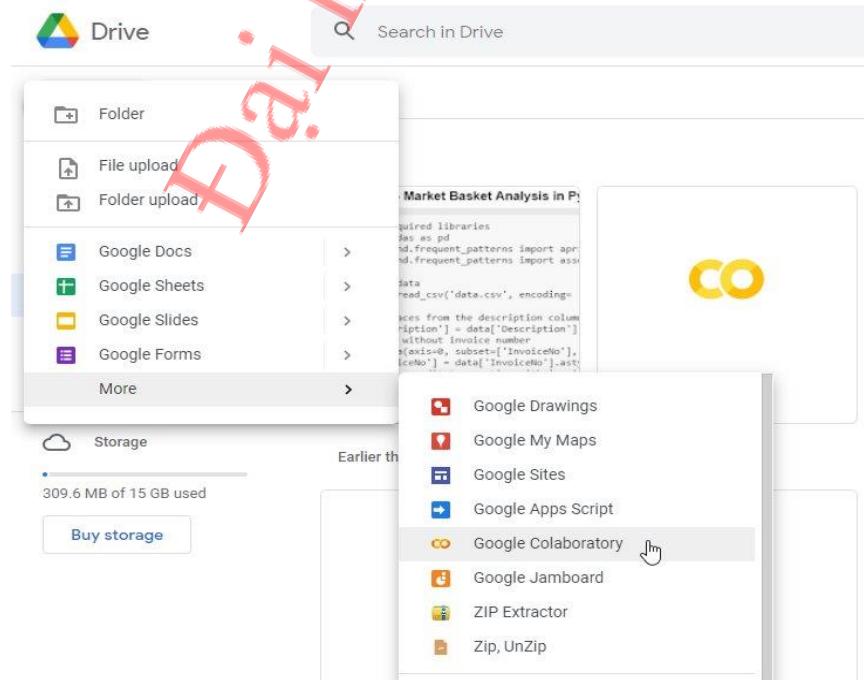
master	2 branches	0 tags	Go to file Add file <> Code
minhhanh1114 update ui		52d1ebb on Aug 30	2,471 commits
embeddings	add embeddings dir		last year
extensions	delete the submodule dir (why do you keep doing this)		last year
javascript	added event listener for the image gallery modal; moved js to separat...		last year
localizations	Remove old localizations from the main repo.		last year
models	Forgot to add this folder		last year
modules	update ui		3 months ago
scripts	updating the displayed generation info when user clicks images in the...		last year
test	Add tests		last year
textual_inversion_templates	hypernetwork training mk1		last year
.pylintrc	Add basic Pylint to catch syntax errors on PRs		last year
artists.csv	Update artists.csv		last year
environment-wsl2.yaml	update environment-wsl2.yaml		last year
launch.py	run installers for newly installed extensions		last year
requirements.txt	Update requirements.txt		last year
requirements_versions.txt	add translate		last year
screenshot.png	updated interface to use Blocks		last year
script.js	localization support		last year
style.css	update ui		3 months ago

Hình 3.8. Mã nguồn đã di chuyển lên github

3.3.2. Sử dụng Google Colab

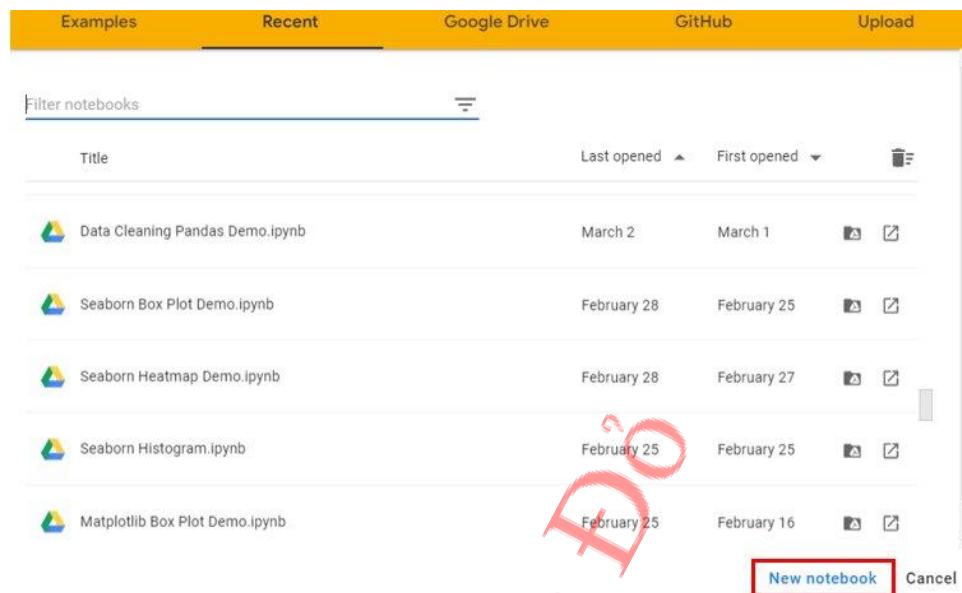
a) Mở sổ tay Google Colab:

Cách 1: Truy cập Google Drive và tạo tài khoản Google Drive. Nhấp vào nút “Mới” ở góc trên cùng bên trái của trang Google Drive, sau đó đi tới “Thêm > Google Collaboratory”.



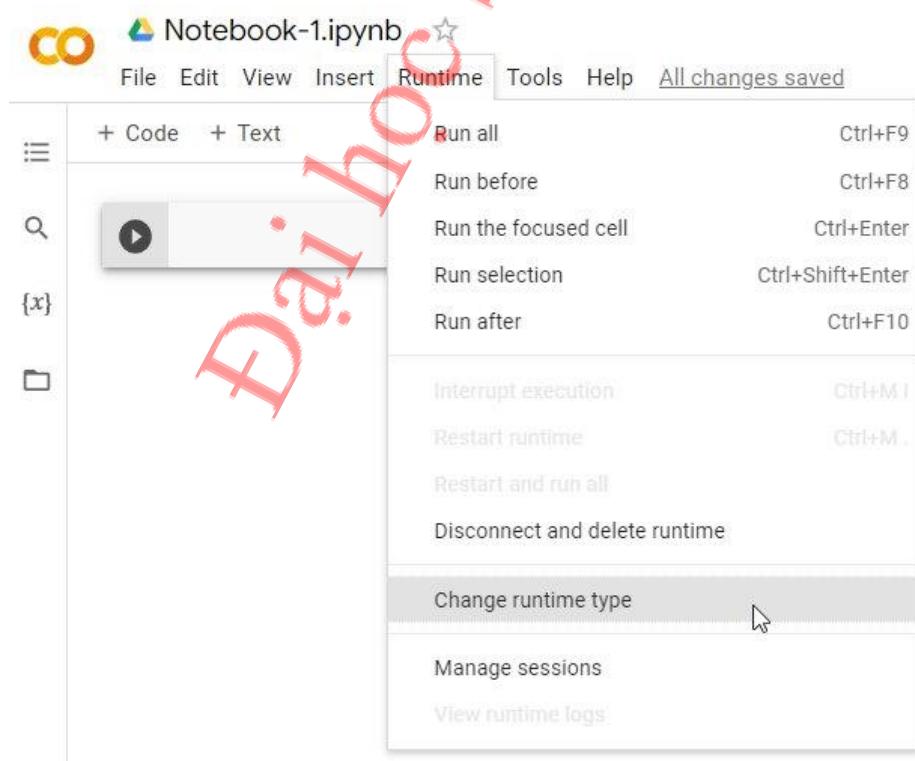
Hình 3.9. Tạo sổ tay mới trên colab (cách 1)

Cách 2: truy cập trực tiếp vào Google Colab theo đường dẫn: <https://colab.research.google.com/> sau đó màn hình sẽ xuất hiện hộp thoại. Sau đó nhấp vào “new notebook” như được hiển thị ở trên để tạo sổ ghi chép Colab mới.



Hình 3.10. Thay đổi kiểu phản ứng trên Google Colab

Nhấp vào “Runtime” trên menu trên cùng và chọn “Change runtime type”. Chọn sử dụng GPU



Hình 3.11. Thay đổi kiểu phản ứng

b) Thực thi mã trên Google Colab

Chọn từng ô mã -> Nhấn vào nút ► ở mỗi góc trên cùng bên trái để chạy mã.

If you're unfamiliar with Python 3, here are some of the most common changes from Python 2 to look out for.

- Print is a function**

```
print("Hello!")
Hello!
```

Without parentheses, printing will not work.

- Floating point division by default**

↳ 3 cells hidden

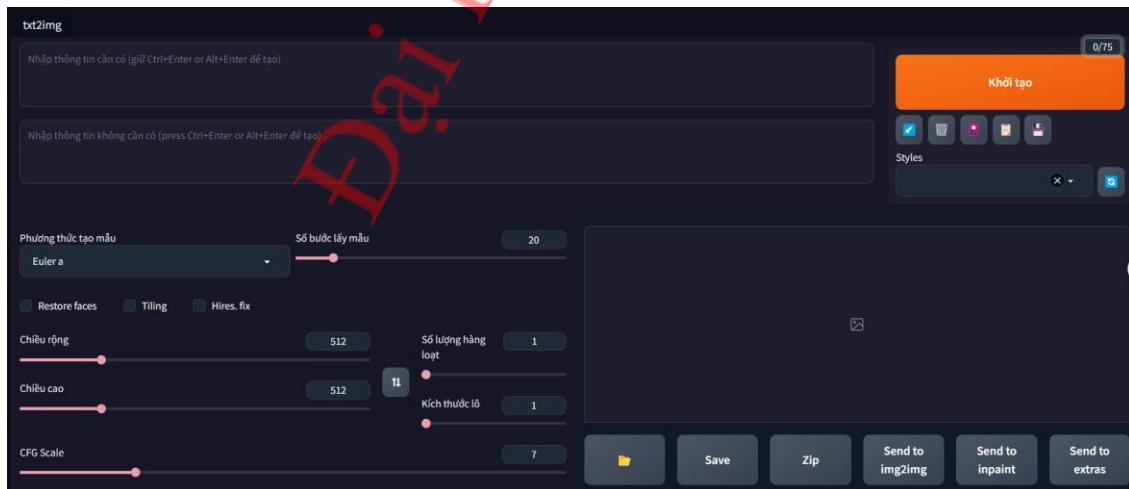
Hình 3.12. Thực thi mã trên colab

3.4. Thiết kế ứng dụng

3.4.1. Thiết kế giao diện

Giao diện là lớp cấp cao chính của Gradio và cho phép tạo GUI / demo dựa trên web xung quanh mô hình học máy (hoặc bất kỳ hàm Python nào) trong một vài dòng mã. Cần chỉ định ba tham số: (1) hàm để tạo GUI cho (2) các thành phần đầu vào mong muốn và (3) các thành phần đầu ra mong muốn. Các tham số bổ sung có thể được sử dụng để kiểm soát sự xuất hiện và hành vi của bản demo.

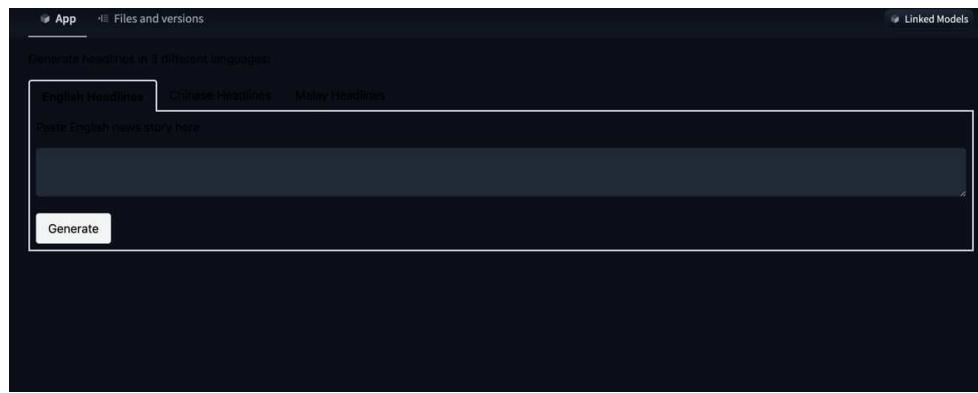
Thành phần trong giao diện gồm có: textbox, button, thanh progressbar.



Hình 3.13. Giao diện

- Thiết kế các textbox

Tạo TextBox và TextArea cho người dùng để nhập đầu và và hiển thị thông tin sử dụng hàm gradio.Textbox(...). Nhập mô tả hình ảnh đầu vào của mô hình.

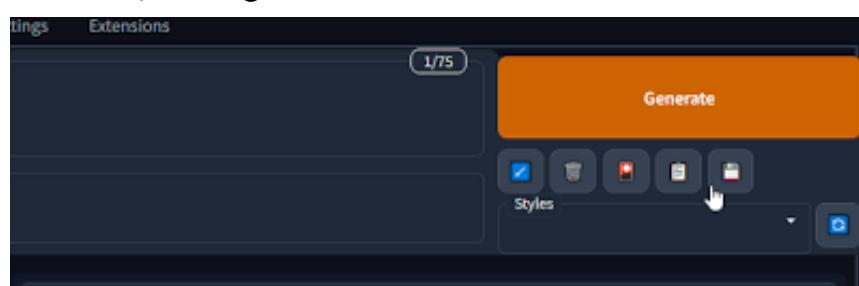


Hình 3.14. Giao diện textbox

```
with gr.Row(elem_id="toprow"):
    with gr.Column(scale=6):
        with gr.Row():
            with gr.Column(scale=80):
                with gr.Row():
                    prompt = gr.Textbox(label="Mô tả thông tin
hình ảnh", elem_id=f"{id_part}_prompt", show_label=False, lines=2,
placeholder="Nhập mô tả ảnh( Ctrl+Enter
or Alt+Enter để tạo)")
                )
            with gr.Row(visible=False):
                with gr.Column(scale=80):
                    with gr.Row():
                        negative_prompt = gr.Textbox(label="Negative
prompt", elem_id=f"{id_part}_neg_prompt", show_label=False, lines=2,
placeholder="Negative      prompt      (press
Ctrl+Enter or Alt+Enter to generate)")
```

- Thiết kế các nút bấm

Gradio hỗ trợ sử dụng để tạo một nút, có thể được gán các sự kiện nhấp chuột tùy ý. Nhãn (giá trị) của nút có thể được sử dụng làm đầu vào hoặc được đặt thông qua đầu ra của hàm. Tham số thứ nhất: Văn bản mặc định cho nút để hiển thị. Tham số thứ 2 elem_id thêm id chỉ định riêng cho thẻ button.



Hình 3.15. Giao diện nút bấm

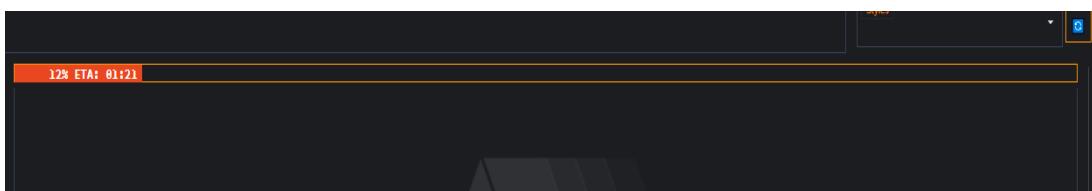
```

with gr.Column(scale=1):
    with gr.Row():
        skip = gr.Button('Skip', elem_id=f"{id_part}_skip")
        interrupt = gr.Button('Interrupt',
elem_id=f"{id_part}_interrupt")
        submit = gr.Button('Tạo ảnh',
elem_id=f"{id_part}_generate", variant='primary')
        skip.click(
            fn=lambda: shared.state.skip(),
            inputs=[],
            outputs[])
        interrupt.click(
            fn=lambda: shared.state.interrupt(),
            inputs=[],
            outputs[])
    with gr.Row(visible=False):
        with gr.Column(scale=1, elem_id="style_pos_col"):
            prompt_style = gr.Dropdown(label="Style 1",
elem_id=f"{id_part}_style_index", choices=[k for k, v in
shared.prompt_styles.styles.items()],
value=next(iter(shared.prompt_styles.styles.keys())))
            prompt_style.save_to_config = True
        with gr.Column(scale=1, elem_id="style_neg_col"):
            prompt_style2 = gr.Dropdown(label="Style 2",
elem_id=f"{id_part}_style2_index", choices=[k for k, v in
shared.prompt_styles.styles.items()],
value=next(iter(shared.prompt_styles.styles.keys())))
            prompt_style2.save_to_config = True

```

- Thiết kế progressbar

Thanh progressbar hiển thị quá trình xử lý tạo ra hình ảnh. Biểu thị phần trăm mà ứng dụng đang xử lý được.



Hình 3.16. Giao diện thanh progressbar

```

def setup_progressbar(progressbar, preview, id_part, textinfo=None):
    if textinfo is None:
        textinfo = gr.HTML(visible=False)

```

```

    check_progress      =         gr.Button('Check progress',
elem_id=f'{id_part}_check_progress', visible=False)
    check_progress.click(
        fn=lambda: check_progress_call(id_part),
        show_progress=False,
        inputs=[],
        outputs=[progressbar, preview, preview, textinfo],
    )
    check_progress_initial = gr.Button('Check progress (first)',
elem_id=f'{id_part}_check_progress_initial', visible=False)
    check_progress_initial.click(
        fn=lambda: check_progress_call_initial(id_part),
        show_progress=False,
        inputs=[],
        outputs=[progressbar, preview, preview, textinfo])

```

3.4.2. Thiết kế các chức năng chính

- Kết nối với ngrok

Ngrok để hiển thị các máy chủ web cục bộ, xây dựng tích hợp webhook, cho phép truy cập SSH, phát triển máy học trên website. Sử dụng hàm ngrok.connect() thực hiện tạo kết nối thông qua nó. Sau đó nó trả về Ngrok tunnel và đối tượng được trả về này có một tham chiếu đến URL công khai được tạo bởi Ngrok trong thuộc tính public_url.

```

def connect(token, port, region):
    account = None
    if token is None:
        token = 'None'
    else:
        if ':' in token:
            account = token.split(':')[1] + ':' + token.split(':')[-
1]
            token = token.split(':')[0]
    config = conf.PyngrokConfig(auth_token=token, region=region)
    try:
        if account is None:
            public_url = ngrok.connect(port, pyngrok_config=config,
bind_tls=True).public_url
        else:
            public_url = ngrok.connect(port, pyngrok_config=config,
bind_tls=True, auth=account).public_url

```

- Load các thành phần chính của mô hình

Mô hình được đào tạo trước bao gồm tất cả các thành phần cần thiết để thiết lập một diffusion pipeline hoàn chỉnh. Chúng được lưu trữ trong các thư mục sau:

- Text_encoder: khuếch tán ổn định sử dụng clip, nhưng các mô hình khuếch tán khác có thể sử dụng các bộ mã hóa khác như Bert.
- Tokenizer : Nó phải khớp với mô hình được sử dụng bởi mô hình text_encoder.
- Scheduler: Thuật toán lập lịch được sử dụng để tăng dần nhiễu vào hình ảnh trong quá trình đào tạo.

• UNET: Mô hình được sử dụng để tạo biểu diễn tiềm ẩn của đầu vào. VAE: Mô -đun Autoencoder sử dụng để giải mã các biểu diễn tiềm ẩn thành hình ảnh thật. Chúng ta có thể tải các thành phần bằng cách tham khảo thư mục chúng đã được lưu.

```
from transformers import CLIPTextModel, CLIPTokenizer
from diffusers import AutoencoderKL, UNet2DConditionModel,
PNDMSchedule
```

```
vae = AutoencoderKL.from_pretrained("CompVis/stable-diffusion-v1-4", subfolder="vae")
tokenizer = CLIPTokenizer.from_pretrained("openai/clip-vit-large-patch14")
text_encoder = CLIPTextModel.from_pretrained("openai/clip-vit-large-patch14")
unet = UNet2DConditionModel.from_pretrained("CompVis/stable-diffusion-v1-4", subfolder="unet")
```

- Chuyển các mô hình đến GPU

```
vae = vae.to(torch_device)
text_encoder = text_encoder.to(torch_device)
unet = unet.to(torch_device)
```

- Định nghĩa các tham số

Định nghĩa các thông số về chiều cao và rộng Stable Diffusion là 512px

Prompt : dữ liệu văn bản đầu vào

Num_inference_steps thành 100 để có được một hình chính xác hơn.

generator định nghĩa là 2 giúp tạo nhiễu latent.

```
prompt = ["a photograph of an astronaut riding a horse"]
```

```
height = 512
```

```
width = 512
```

```
num_inference_steps = 100
```

```
guidance_scale = 7.5
```

```
generator = torch.manual_seed(32)
```

```
batch_size = 1
```

- Lấy text input từ prompt

Sử dụng text input lấy được mã hóa đưa tất cả các mã nhúng vào mô hình

```
text_input      = tokenizer(prompt,           padding="max_length",
                           max_length=tokenizer.model_max_length,
                           truncation=True,
                           return_tensors="pt")

with torch.no_grad():
    text_embeddings = text_encoder(text_input.input_ids.to(torch_device))[0]
```

- Tạo nhiễu ngẫu nhiên

Dầu ra của latents.shape là 64×64 Mô hình sẽ chuyển đổi biểu diễn latent này thành hình ảnh 512×512 sau này.

Tiếp theo, khởi tạo scheduler với num_inference_steps đã chọn. Điều này sẽ tính toán các giá trị bước thời gian chính xác và thời gian chính xác được sử dụng trong quá trình khử nhiễu.

```
latents = torch.randn(
    (batch_size, unet.in_channels, height // 8, width // 8),
    generator=generator)
latents = latents.to(torch_device)
scheduler.set_timesteps(num_inference_steps)
```

- Vòng lặp khử nhiễu ảnh

Sử dụng VAE để giải mã các tọa độ được tạo trở lại vào hình ảnh.

Model unet thực hiện dự đoán nhiễu

```
from tqdm.auto import tqdm
from torch import autocast
for t in tqdm(scheduler.timesteps):
    latent_model_input = torch.cat([latents] * 2)
    latent_model_input =
    scheduler.scale_model_input(latent_model_input, t)
    with torch.no_grad():
        noise_pred = unet(latent_model_input, t,
encoder_hidden_states=text_embeddings).sample
        noise_pred_uncond, noise_pred_text = noise_pred.chunk(2)
        noise_pred = noise_pred_uncond + guidance_scale * (noise_pred_text
- noise_pred_uncond)
    latents = scheduler.step(noise_pred, t, latents).prev_sample
    latents = 1 / 0.18215 * latents

    with torch.no_grad():
        image = vae.decode(latents).sample
```

- Chuyển đổi hình ảnh

Chuyển đổi hình ảnh thành PIL để chúng ta có thể hiển thị hoặc lưu ảnh

```
image = (image / 2 + 0.5).clamp(0, 1)
image = image.detach().cpu().permute(0, 2, 3, 1).numpy()
images = (image * 255).round().astype("uint8")
pil_images = [Image.fromarray(image) for image in images]
pil_images[0]
```

- Chuyển đổi văn bản tiếng việt

Thư viện Translate trên Python hỗ trợ chuyển đổi ngôn ngữ nhập tiếng việt sang tiếng việt. Do mô hình sử dụng nội dung text dữ liệu mã hóa kèm hình ảnh là tiếng anh. Vì vậy chuyển đổi ngôn ngữ sẽ thuận tiện hơn cho người dùng bằng tiếng việt mà chương trình vẫn có thể sử dụng song ngôn ngữ.

Bước đầu tiên thêm thư viện chuyển đổi ngôn ngữ, sau đó dịch prompt từ người dùng nhập vào sang tiếng Anh. Tạo một đối tượng xử lý cho quá trình chuyển đổi văn bản thành hình ảnh tiếp theo đó chạy quá trình xử lý văn bản thành hình ảnh. Nội dung text chuyển đổi sẽ được chuyển sang tiếng anh.

```
from mtranslate import translate
def txt2img(id_task: str, prompt: str, negative_prompt: str,
prompt_styles, steps: int, sampler_index: int, restore_faces: bool,
tiling: bool, n_iter: int, batch_size: int, cfg_scale: float, seed: int,
subseed: int, subseed_strength: float, seed_resize_from_h: int,
seed_resize_from_w: int, seed_enable_extras: bool, height: int, width: int,
enable_hr: bool, denoising_strength: float, hr_scale: float,
hr_upscaler: str, hr_second_pass_steps: int, hr_resize_x: int,
hr_resize_y: int, hr_sampler_index: int, hr_prompt: str,
hr_negative_prompt, override_settings_texts, *args):
    override_settings = create_override_settings_dict(override_settings_texts)
    translated_text = translate(prompt, 'en')
    prompt=translated_text
    p = processing.StableDiffusionProcessingTxt2Img(
        sd_model=shared.sd_model,
        outpath_samples=opts.outdir_samples or
        opts.outdir_txt2img_samples,
        outpath_grids=opts.outdir_grids or opts.outdir_txt2img_grids,
        prompt=prompt,
        styles=prompt_styles,
        negative_prompt=negative_prompt,
        seed=seed,
        subseed=subseed,
```

```
subseed_strength=subseed_strength,
seed_resize_from_h=seed_resize_from_h,
seed_resize_from_w=seed_resize_from_w,
seed_enable_extras=seed_enable_extras,
sampler_name=sd_samplers.samplers[sampler_index].name,
batch_size=batch_size,
n_iter=n_iter,
steps=steps,
cfg_scale=cfg_scale,
width=width,
height=height,
restore_faces=restore_faces,
tiling=tiling,
enable_hr=enable_hr,
denoising_strength=denoising_strength if enable_hr else None,
hr_scale=hr_scale,
hr_upscaler=hr_upscaler,
hr_second_pass_steps=hr_second_pass_steps,
hr_resize_x=hr_resize_x,
hr_resize_y=hr_resize_y,
hr_sampler_name=sd_samplers.samplers_for_img[hr_sampler_index - 1].name if hr_sampler_index != 0 else None,
hr_prompt=hr_prompt,
hr_negative_prompt=hr_negative_prompt,
override_settings=override_settings,
)
p.scripts = modules.scripts.scripts_txt2img
p.script_args = args
if cmd_opts.enable_console_prompts:
    print(f"\ntxt2img: {prompt}", file=shared.progress_print_out)
processed = modules.scripts.scripts_txt2img.run(p, *args)
if processed is None:
    processed = processing.process_images(p)
p.close()
shared.total_tqdm.clear()
generation_info_js = processed.js()

if opts.samples_log_stdout:
    print(generation_info_js)
```

```

    if opts.do_not_show_images:
        processed.images = []
    return processed.images, generation_info_js,
plaintext_to_html(processed.info),
plaintext_to_html(processed.comments)

```

3.5. Thiết lập chương trình trên Google Colab

3.5.1. Thiết lập hệ thống

Sau khi thực hiện khởi tạo chương trình mới với Google Colab tại mục soạn thảo code đầu tiên thiết lập di chuyển đường dẫn máy chủ về thư mục content và thiết lập cái gói thông tin trên hệ thống Linux.

```
%cd /content
%env TF_CPP_MIN_LOG_LEVEL=1
!apt -y update -qq
```

3.5.2. Thiết lập môi trường

Cài đặt vào hệ thống các gói thư viện sử dụng apt và các gói Python sử dụng pip. Thiết lập môi trường với các phiên bản của nhiều thư viện khác nhau, để đảm bảo rằng quá trình thực hiện mã lệnh sau tránh cách xung đột tăng khả năng tương thích giữa các thư viện trong dự án.

```
!apt -y install -qq aria2 libcairo2-dev pkg-config python3-dev
!pip install -q torch==2.0.1+cu118 torchvision==0.15.2+cu118
torchaudio==2.0.2+cu118 torchtext==0.15.2 torchdata==0.6.1 --extra-index-url https://download.pytorch.org/whl/cu118 -U
!pip install -q xformers==0.0.20 triton==2.0.0 gradio_client==0.2.7-U
```

3.5.3. Tạo các thành phần cần thiết cho chương trình

Tải mã nguồn sử dụng url từ bản github đã tạo với nhánh số v2.4. Tiếp đó tải các file liên quan tới mô hình với phiên bản v2.1, các file là các checkpoint đã được đào tạo và file tiện ích.

```
!git clone -b v2.4 https://github.com/minhahanh1114/stable-diffusion-webui.git
```

```
!git clone https://huggingface.co/embed/negative /content/stable-diffusion-webui/embeddings/negative
```

```
!git clone https://huggingface.co/embed/lora /content/stable-diffusion-webui/models/Lora/positive
```

```
!aria2c --console-log-level=error -c -x 16 -s 16 -k 1M
https://huggingface.co/embed/upscale/resolve/main/4x-UltraSharp.pth -d /content/stable-diffusion-webui/models/ESRGAN -o 4x-UltraSharp.pth
```

```
!git clone https://github.com/deforum-art/deforum-for-automatic1111-webui /content/stable-diffusion-webui/extensions/deforum-for-automatic1111-webui
```

```
!git clone https://github.com/kohya-ss/sd-webui-additional-networks /content/stable-diffusion-webui/extensions/sd-webui-additional-networks
```

```
!git clone https://github.com/Mikubill/sd-webui-controlnet /content/stable-diffusion-webui/extensions/sd-webui-controlnet
```

```
!git clone https://github.com/fkunn1326/openpose-editor /content/stable-diffusion-webui/extensions/openpose-editor
```

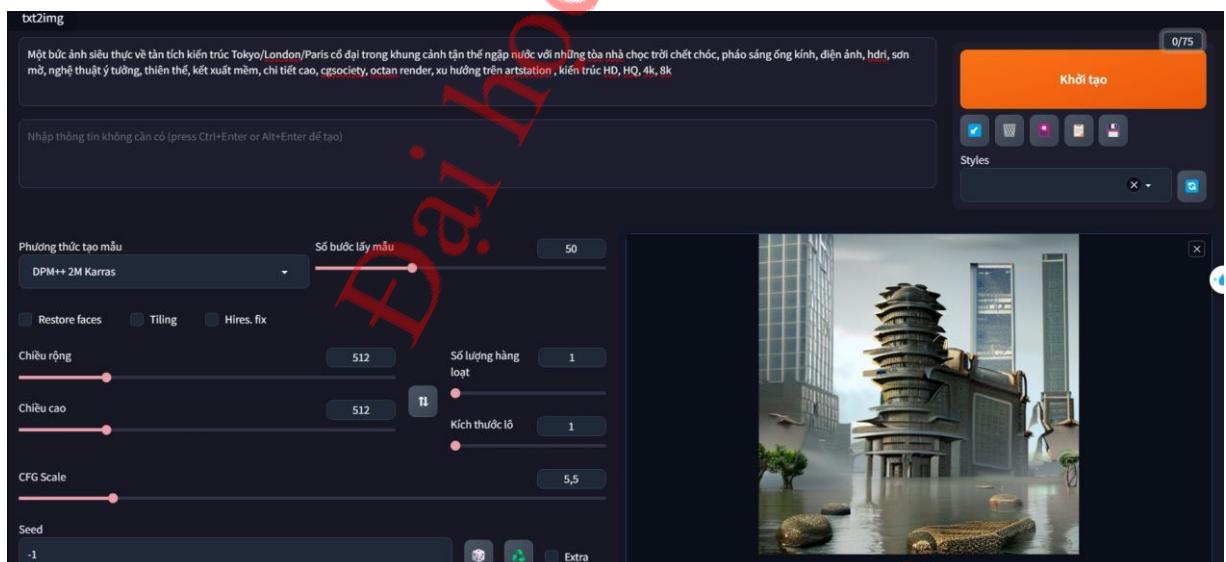
3.5.4. Thực thi chương trình

Sử dụng cấu trúc câu lệnh tham số để truyền các tham số vào trong hàm main chính của mã nguồn, thực thi mã nguồn.

```
!python launch.py --listen --xformers --enable-insecure-extension-access --theme dark --gradio-queue -multiple
```

3.6. Kết quả chương trình

Khi thực thi qua tất cả các đoạn code trên máy chủ Goole Colab sẽ trả về đường dẫn với địa chỉ tên miền, để truy cập vào chương trình website.

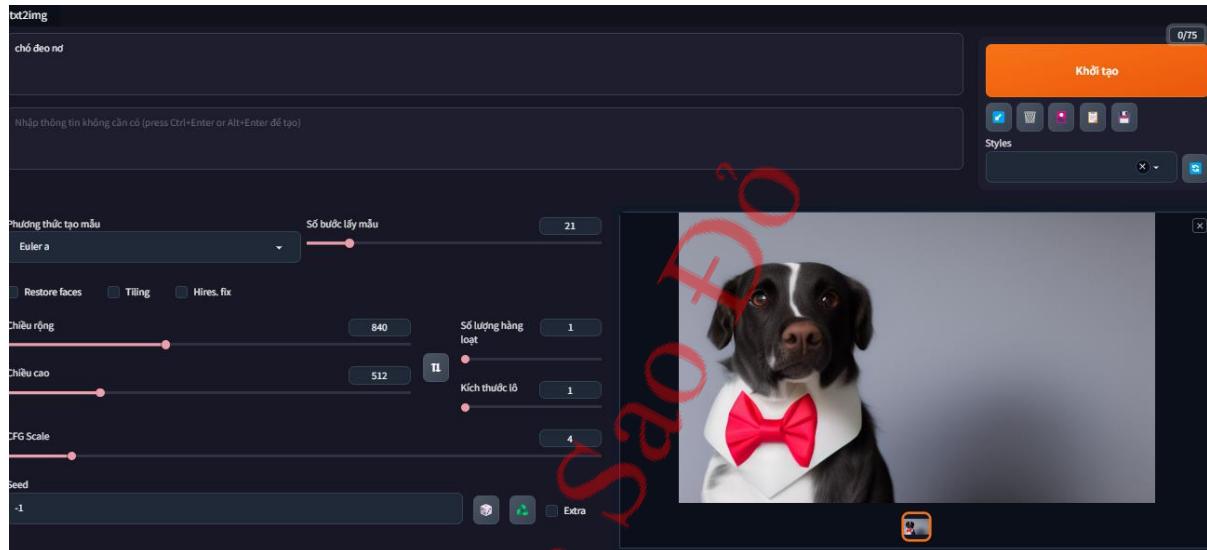


Hình 3.17. Giao diện chương trình với văn bản chi tiết

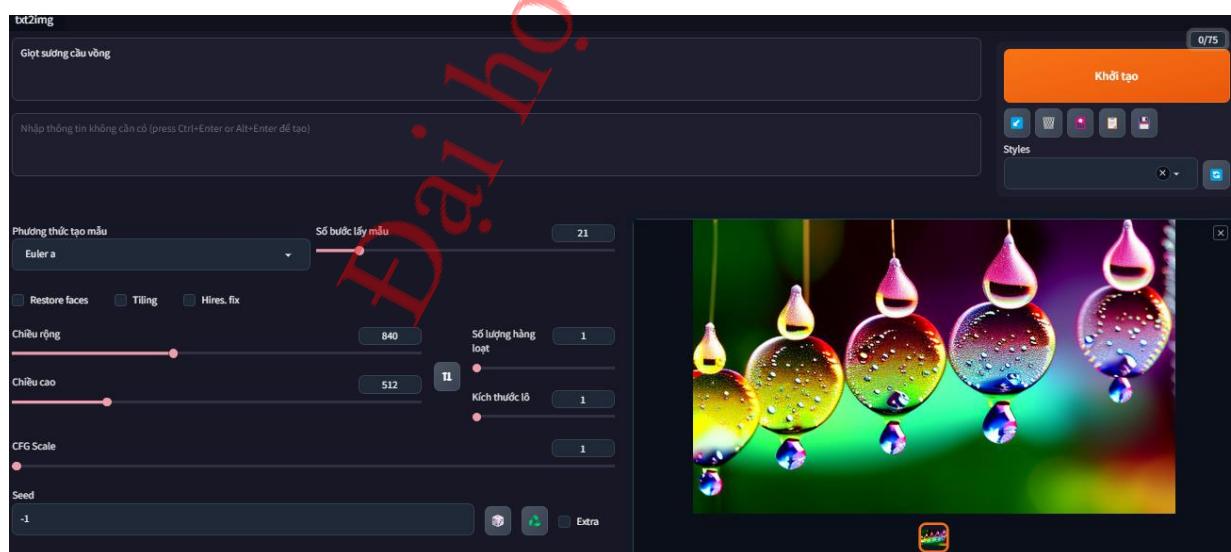
Hình 3.17 là kết quả thực thi của trình duyệt web với các đoạn văn bản cụ thể, gồm thông số như số bước lấy mẫu, hàm tạo mẫu, chiều cao, chiều rộng của hình ảnh, số lượng hình ảnh sau khi thực thi, độ sáng tạo của mô hình, lưu hình ảnh. Số lượng ảnh và số lượng lấy mẫu càng nhiều thì thời gian chờ đợi sẽ lâu hơn.

Số lượng lấy mẫu lớn đồng nghĩa với việc hình ảnh tạo ra sẽ được chân thực hơn. Dưới đây là hình với đoạn văn bản là tàn tích kiến trúc cổ đại của thành phố trong khung cảnh nước ngập, hình ảnh sắc nét số bước lấy mẫu 50 hàm lấy mẫu DPM++2M Karras.

Với sự trợ giúp của mô hình Chuyển văn bản thành hình ảnh, người dùng có thể nhanh chóng biến ý tưởng của mình thành tác phẩm nghệ thuật. Người dùng phải hoàn thiện lời nhắc của mình để nhận được kết quả tốt từ AI Khuếch tán Ổn định. Mức độ nhắc nhở cung cấp sẽ ảnh hưởng trực tiếp đến mức độ chi tiết và chất lượng của tác phẩm nghệ thuật.

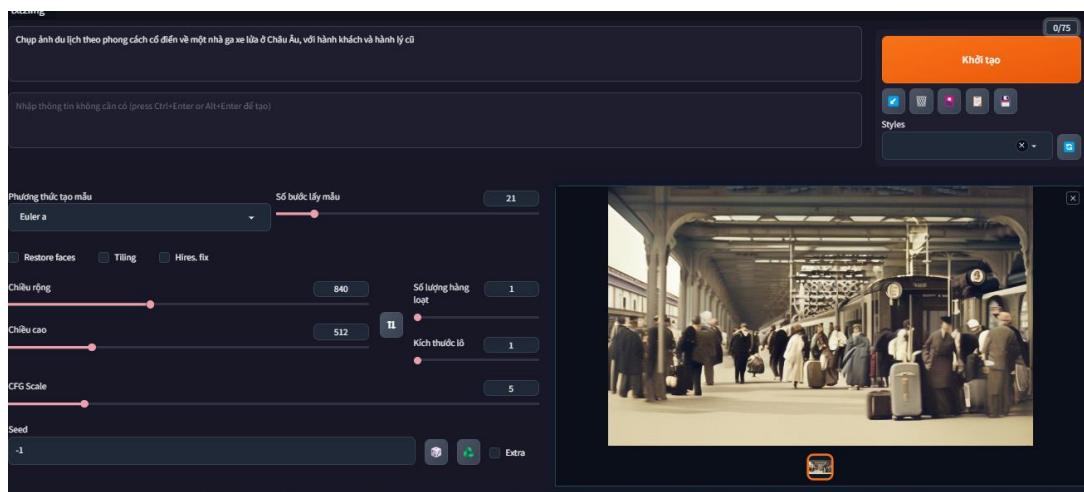


Hình 3.18. Chú chó đeo nơ



Hình 3.19. Giọt sương cầu vồng

Hơn nữa, có thể tùy chọn để nâng cao kết quả mong muốn bằng cách kết hợp các yếu tố bổ sung như kiểu dáng, tông màu, thiết lập ánh sáng đa dạng và nhiều thông số khác. Điều này cho phép tinh chỉnh và tùy chỉnh kết quả để phù hợp hơn với sở thích của người dùng.



Hình 3.20. Nhà ga xe lửa Châu Âu

Dai hoc Sao Do

KẾT LUẬN

1. Kết quả đạt được

Sau quá trình tìm hiểu và triển khai đồ án “Ứng dụng deeplearning xây dựng website chuyển văn bản thành hình ảnh” đã thực hiện được đầy đủ những yêu cầu đặt ra của nhiệm vụ đồ án tốt nghiệp, các kết quả có thể kể đến đó là:

- *Về mặt lý thuyết:* Đồ án đã tìm hiểu kỹ thuật về deeplearning sử dụng mô hình Stable Diffusion ứng dụng mô hình sử dụng Python để xây dựng website AI.
- *Về mặt thực tiễn:*
 - Phân tích và cách thức triển khai chương trình.
 - Lập trình, phát triển ứng dụng trên nền tảng website sử dụng một số từ khóa Stable-diffusion sẽ tạo ra những bức ảnh tuyệt vời, chưa bao giờ có ở bất kì đâu. Ngoài ra chúng ta có thể tạo vô số hình ảnh ra bất cứ nào cần tìm kiếm với những mô tả dài hơn, cụ thể hơn mà không cần tuân thủ bất cứ quy tắc nào.
 - Chuyển đổi văn bản thành ảnh giúp tạo ra nội dung động và sáng tạo hơn, thu hút sự chú ý của người xem, hỗ trợ người sáng tạo nội dung
 - Trang web chuyển đổi văn bản thành ảnh có thể giúp tự động hóa quy trình tạo nội dung, tiết kiệm thời gian và công sức so với việc tạo hình ảnh thủ công.
 - Sử dụng cho những người không có kỹ năng thiết kế đồ họa, việc sử dụng trang web chuyển đổi văn bản thành hình ảnh là một cách đơn giản và hiệu quả, tiết kiệm chi phí.

2. Khuyến nghị

Để hệ thống hoạt động hiệu quả hơn, cần thực hiện các khuyến nghị sau đây:

- Yêu cầu về GPU: Để tạo ra hình ảnh với tốc độ và chất lượng hình ảnh cao. Cần sử dụng GPU sử dụng nhiều VRAM. Khuyến nghị ít nhất 8GB, độ phân giải cao hơn yêu cầu 12GB trở lên.
 - Yêu cầu về Ram: Bộ nhớ hệ thống giúp cung cấp dữ liệu cho GPU, do đó, có ít nhất 16GB RAM có thể đảm bảo hiệu suất tối ưu. Nhiều RAM hơn, lên tới 32GB hoặc 64GB, có thể cải thiện tốc độ hơn nữa.
 - Ứng dụng có thể mở rộng để Stable Diffusion được sử dụng để cải thiện chất lượng ảnh, đặc biệt là trong các ứng dụng như siêu phân giải (super-resolution) và tái tạo hình ảnh. Trong lĩnh vực y sinh và sinh học, "Stable Diffusion" có thể được áp dụng để xử lý và phục hồi hình ảnh y tế, hình ảnh sinh học, giảm nhiễu trong ảnh chụp cắt lớp vi. Ứng dụng Inpainting khắc phục vùng thiếu sót, bị làm mờ, hoặc bị che khuất trong ảnh để tạo ra một bức ảnh hoàn chỉnh.

TÀI LIỆU THAM KHẢO

- [1] Robin Rombach, Andreas Blattmann, Dominik Lorenz and Patrick Esser (2015), *Björn Ommer: U-Net: Computer Vision and Pattern Recognition*, arXiv:2112.10752.
- [2] Howard Jeremy. Fastai (2019), *Dynamic U-Net*, <https://www.youtube.com/watch?v=0frKXR-2PBY>. Accessed 2 September 2019.
- [3] Underfitted (2019), *Generating Images From Text. Stable Diffusion* <https://www.youtube.com/watch?v=0frKXR-2PBY>. Accessed 2 September 2019.
- [4] Đại học Sao Đỏ (2022), *Mạng noron*
- [5] <https://github.com/CompVis/stable-diffusion>

Dai hoc Sao Do