|  |  |
| --- | --- |
|  |  |
| NGUYỄN HỒNG PHONG | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| ĐA, KLTN ĐẠI HỌC NGÀNH CÔNG NGHỆ THÔNG TIN |
|  |
| **XÂY DỰNG ỨNG DỤNG DI ĐỘNG CHUYỂN ĐỔI**  **ẢNH THẬT THÀNH ẢNH HOẠT HÌNH** |
|  |
|  |
| **CBHD: TS. *Trần Hùng Cường*** |
| CÔNG NGHỆ THÔNG TIN | **Sinh viên: Nguyễn Hồng Phong** |
| **Mã số sinh viên: *2019605743*** |
|  |
|  |
|  |
| Hà Nội – Năm 2023 |
|  |

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**TÊN ĐỀ TÀI:**

**Xây dựng ứng dụng di động chuyển đổi ảnh thật**

**thành ảnh hoạt hình**

|  |  |
| --- | --- |
| **CBHD:** | **TS. Trần Hùng Cường** |
| **Sinh viên:** | **Nguyễn Hồng Phong** |
| **Mã số sinh viên:**  **Lớp:** | **2019605743**  **2019DHCNTT05** |

Hà Nội – Năm 2023

MỤC LỤC

[MỤC LỤC I](#_Toc134217616)

[DANH MỤC HÌNH ẢNH I](#_Toc134217617)

[DANH MỤC TỪ VIẾT TẮT II](#_Toc134217618)

[LỜI CẢM ƠN III](#_Toc134217619)

[CHƯƠNG 1: TỔNG QUAN VỀ NỘI DUNG NGHIÊN CỨU 1](#_Toc134217620)

[1.1 Giới thiệu chung 1](#_Toc134217621)

[1.2 Lý do lựa chọn đề tài 1](#_Toc134217622)

[1.3 Đối tượng, mục tiêu và phạm vi nghiên cứu 2](#_Toc134217623)

[a. Mục tiêu nghiên cứu 2](#_Toc134217624)

[b. Đối tượng nghiên cứu 2](#_Toc134217625)

[c. Phạm vi nghiên cứu 2](#_Toc134217626)

[1.4 Phương pháp nghiên cứu 2](#_Toc134217627)

[1.5 Tình hình nghiên cứu 5](#_Toc134217628)

[1.6 Tính cấp thiết của đề tài 5](#_Toc134217629)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 6](#_Toc134217630)

[2.1 Giới thiệu về học sâu 6](#_Toc134217631)

[2.1.1 Lịch sử của học sâu 8](#_Toc134217632)

[2.1.2 Các thành phần chính: Dữ liệu, Mô hình và Thuật toán 12](#_Toc134217633)

[2.2 Giới thiệu về mạng Generative Adversarial Networks(GAN) 26](#_Toc134217634)

[2.2.1 GAN là gì? 26](#_Toc134217635)

[2.2.2 Mô hình phân biệt (Discriminative model) 27](#_Toc134217638)

[2.2.3 Mô hình sinh(Generative model) 28](#_Toc134217641)

[2.2.4 Kiến trúc mạng GAN 29](#_Toc134217642)

[2.2.5 Lan truyền ngược (Backpropagation) 30](#_Toc134217643)

[2.2.6 Độ dốc giảm dần của đạo hàm mạng sinh 32](#_Toc134217646)

[CHƯƠNG 3: THIẾT KẾ VÀ THỰC NGHIỆM 33](#_Toc134217647)

[3.1. Mô hình sinh ảnh hoạt hình từ ảnh thật 33](#_Toc134217648)

[3.1.1 Dữ liệu phục vụ cho bài toán 33](#_Toc134217649)

[3.1.2 Chuẩn bị dữ liệu cho huấn luyện 34](#_Toc134217650)

[3.1.3 Thiết kế mô hình 36](#_Toc134217651)

[3.1.4 Hàm loss của mô hình 40](#_Toc134217652)

[3.1.5 Huấn luyện mô hình 42](#_Toc134217653)

[3.1.6 Thử nghiệm 43](#_Toc134217654)

[3.1.7 Đánh giá mô hình 44](#_Toc134217655)

[3.2 Ứng dụng di động tạo ảnh hoạt hình từ ảnh thật 45](#_Toc134217656)

[3.2.1 Thiết kế các UseCase 45](#_Toc134217657)

[3.2.2 Thiết kế giao diện 51](#_Toc134217658)

[KẾT LUẬN 52](#_Toc134217659)

[TÀI LIỆU THAM KHẢO 53](#_Toc134217660)

DANH MỤC HÌNH ẢNH

[Hình 3. 1 Mô phỏng cách hoạt động của mạng GAN 36](#_Toc134217547)

[Hình 3. 2 Kiến trúc mô hình GAN sử dụng trong bài toán 36](#_Toc134217548)

[Hình 3. 3 Cách khối chính được sử dụng 38](#_Toc134217549)

[Hình 3. 4 Quá trình huấn luyện mô hình với colab 43](#_Toc134217550)

[Hình 3. 5 Thử nghiệm với Larava style 43](#_Toc134217551)

[Hình 3. 6 Thử nghiệm với Paprika style 44](#_Toc134217552)

[Hình 3. 7 VOPC các lớp use case chuyển đổi ảnh 46](#_Toc134217553)

[Hình 3. 8 Biểu đồ trình tự Use case Chuyển đổi ảnh 47](#_Toc134217554)

# DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| ANN | Artificial Neural Network | Mạng nơ-ron nhân tạo |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| CNN | Convolutional Neural Network | Mạng nơ-ron tích chập |
| CPU | Central Processing Unit | Bộ điều khiển trung tâm |
| DL | Deep Learning | Học sâu |
| ID | Identification | Nhận dạng |
| FC | Fully Connected | Kết nối đầy đủ |
| ML | Machine Learning | Học máy |
| MLP | Multiple Layer Perceptron | Mạng nơron truyền thẳng nhiều lớp |
| RELU | Rectified Linear Unit | Lớp chuyển đổi |
| SVM | Support Vector Machine | Máy vector hỗ trợ |
| GAN | Generative Adversarial Networks | Mạng nơ ron đối nghịch |

LỜI CẢM ƠN

Để báo cáo tốt nghiệp này đạt kết quả tốt đẹp, đầu tiên em xin chân thành cảm ơn trường Đại học Công nghiệp Hà Nội, thầy cô giáo khoa Công nghệ thông tin đã tạo điều kiện thuận lợi và truyền dạy kiến thức cho em trong thời gian qua để em hoàn thành đề tài một cách tốt nhất. Và hơn hết em xin bày tỏ lòng biết ơn sâu sắc nhất tới thầy giáo hướng dẫn **TS. Trần Hùng Cường** đã tận tình giúp đỡ, định hướng em rất nhiều trong suốt quá trình tìm hiểu nghiên cứu và hoàn thành thực tập tốt nghiệp.

Trong quá trình nghiên cứu đề tài, em đã cố gắng hoàn thành tốt nhất báo cáo thực tập tốt nghiệp. Tuy nhiên kiến thức chuyên ngành của bản thân còn nhiều hạn chế. Vì vậy trong báo cáo không tránh khỏi những thiếu sót, em rất mong nhận được sự đóng góp của tất cả các thầy cô giáo để đồ án của em được đầy đủ và hoàn chỉnh hơn.

Em xin chân thành cảm ơn!

# CHƯƠNG 1: TỔNG QUAN VỀ NỘI DUNG NGHIÊN CỨU

## Giới thiệu chung

Trí tuệ nhân tạo (AI) là lĩnh vực đang phát triển rất nhanh, đem lại nhiều ứng dụng đáng kinh ngạc trong nhiều lĩnh vực khác nhau như tài chính, y tế, giao thông vận tải, giáo dục, sản xuất và năng lượng. Một ứng dụng tiêu biểu là việc sử dụng mô hình mạng Generative Adversarial Networks (GAN) để chuyển đổi ảnh thật thành ảnh hoạt hình, giúp tiết kiệm thời gian và công sức so với việc vẽ thủ công.

## 1.2 Lý do lựa chọn đề tài

"Xây dựng ứng dụng di động chuyển đổi ảnh thật thành ảnh hoạt hình" là sự kết hợp giữa một đề tài nghiên cứu tiên tiến và một ứng dụng thực tiễn hấp dẫn. Các mô hình GAN (Generative Adversarial Networks) đã trở thành một trong những kỹ thuật tiên tiến nhất trong lĩnh vực Machine Learning, đặc biệt là trong việc tạo ra ảnh, video và âm thanh mới.

Trong lĩnh vực hoạt hình, việc tạo ra các bức ảnh và đoạn video động đã và đang trở thành một nhu cầu ngày càng tăng của ngành công nghiệp giải trí. Tuy nhiên, việc tạo ra các ảnh hoạt hình chất lượng cao vẫn là một thách thức lớn đối với các nhà sản xuất, đặc biệt là khi muốn tạo ra các đoạn video với nội dung phức tạp và động tác chuyển động phức hợp.

Để giải quyết thách thức này, sử dụng các mô hình GAN là một phương pháp tiếp cận hợp lý. Bằng cách sử dụng GAN, ta có thể huấn luyện mô hình để tạo ra các bức ảnh và đoạn video mới với chất lượng cao, đồng thời giảm thiểu thời gian và chi phí so với các phương pháp truyền thống.

Mặc dù đã có nhiều nghiên cứu về GAN và ứng dụng của nó trong sinh ảnh, tuy nhiên, vẫn còn rất nhiều thách thức cần giải quyết. Nghiên cứu sâu hơn về GAN và ứng dụng của nó trong hoạt hình sẽ giúp tăng hiểu biết về lĩnh vực này, đồng thời cũng giúp cho các ứng dụng sinh ảnh hoạt hình tốt hơn. Có thể tích hợp vào ngay trên thiết bị di động.

Ngoài ra, đề tài này còn có tính ứng dụng cao và thực tiễn trong cuộc sống. Việc tạo ra các ảnh hoạt hình có chất lượng cao có thể được áp dụng trong nhiều lĩnh vực, từ giải trí đến giáo dục.

## 1.3 Đối tượng, mục tiêu và phạm vi nghiên cứu

### a. Mục tiêu nghiên cứu

- Tìm hiểu và phân tích các thuật toán GAN phổ biến nhất hiện nay và áp dụng chúng vào việc sinh ảnh hoạt hình động chất lượng cao.

- Tập trung vào các kỹ thuật huấn luyện và tối ưu hóa mô hình GAN để đạt được hiệu suất tốt nhất trong việc sinh ảnh hoạt hình. Đồng thời, nghiên cứu cũng sẽ tập trung vào việc xác định các thách thức và hạn chế trong việc sử dụng GAN trong hoạt hình.

-Nghiên cứu cách áp dụng mô hình sinh ảnh hoạt hình chất lượng cao vào ứng dụng di động.

### b. Đối tượng nghiên cứu

-Bao gồm các mô hình GAN phổ biến nhất, bao gồm cả cấu trúc GAN cơ bản và các biến thể như Conditional GAN, CycleGAN…

### c. Phạm vi nghiên cứu

-Phạm vi sẽ giới hạn trong việc tạo ra các ảnh hoạt hình 2D, chứ không bao gồm các ứng dụng 3D.

-Nghiên cứu tích hợp vào ứng dụng di động Android.

## 1.4 Phương pháp nghiên cứu

* Phương pháp nghiên cứu về mặt lý thuyết:
  + Nghiên cứu và tiến hành thu thập các tài liệu có liên quan đến đề tài.
  + Tổng hợp các tài liệu đã thu thập và tiến hành phân tích.
  + Chọn lọc các tài liệu nghiên cứu để báo cáo đồ án tốt nghiệp.
* Phương pháp nghiên cứu trong thực nghiệm:
  + Tìm hiểu và phân tích các mô hình GAN phổ biến nhất hiện nay: Chúng tôi sẽ tìm hiểu về các thuật toán GAN phổ biến nhất hiện nay, bao gồm cả cấu trúc GAN cơ bản và các biến thể như Conditional GAN, CycleGAN... Chúng tôi sẽ phân tích các đặc điểm của mỗi thuật toán, ưu điểm và nhược điểm, cũng như các ứng dụng của chúng trong việc tạo ra ảnh hoạt hình động.
  + Xây dựng và huấn luyện các mô hình GAN: Chúng tôi sẽ xây dựng các mô hình GAN sử dụng các thuật toán đã được phân tích trong bước 1 và thực hiện quá trình huấn luyện trên các tập dữ liệu ảnh hoạt hình động phù hợp. Chúng tôi sẽ tập trung vào việc tối ưu hóa các tham số của mô hình để đạt được hiệu suất tốt nhất trong việc sinh ảnh hoạt hình.
  + Đánh giá và so sánh hiệu suất của các mô hình GAN: Chúng tôi sẽ đánh giá hiệu suất của các mô hình GAN đã huấn luyện bằng các chỉ số chất lượng ảnh như độ tương tự cấu trúc (Structural Similarity Index, SSIM), độ tương tự chung (Peak Signal-to-Noise Ratio, PSNR) để giúp đạt được mô hình có hiệu suất tốt nhất.
  + Tìm hiểu ngôn ngữ lập trình Arduino, công cụ lập trình Arduino IDE
  + Áp dụng công nghệ vào xây dựng ứng dụng.

Tổng hợp tất các kiến thức đã học hoàn thành báo cáo.

## 1.5 Tình hình nghiên cứu

-Hoạt hình là một loại hình nghệ thuật phổ biến trong cuộc sống hàng ngày của chúng ta. Hình thức nghệ thuật này được rộng rãi được sử dụng trong một số lĩnh vực bao gồm quảng cáo, phim và giáo dục trẻ em. Tuy nhiên, việc sản xuất phim hoạt hình chủ yếu dựa vào việc thực hiện thủ công.

- Tuy nhiên, việc tạo ảnh hoạt hình theo cách thủ công rất tốn công sức và liên quan đến nhiều những kỹ năng nghệ thuật. Đối với các nghệ sĩ, việc tạo ra các tác phẩm hoạt hình chất lượng cao đòi hỏi xem xét cẩn thận các đường nét, kết cấu, màu sắc và bóng, có nghĩa là rất khó và tốn thời gian để tạo ra các tác phẩm. Do đó, cần nghiên cứu các kỹ thuật có thể tự động chuyển đổi ảnh trong thế giới thực sang ảnh hoạt hình chất lượng cao.

## 1.6 Tính cấp thiết của đề tài

- Khắc phục những khuyết điểm, hạn chế của việc vẽ các tác phẩm truyện tranh thủ công.

- Giúp nghệ sĩ để tập trung vào công việc sáng tạo hơn và tiết kiệm thời gian, người dùng bình thường cũng có thể dễ dàng tạo ra ảnh hoạt hình của riêng họ.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1 Giới thiệu về học sâu

Cho tới tận gần đây, gần như tất cả mọi chương trình máy tính mà chúng ta tương tác hàng ngày đều được tạo ra bởi lập trình viên phần mềm từ những định đề cơ bản. Giả sử chúng ta muốn viết một ứng dụng quản lý hệ thống thương mại điện tử. Sau khi tụm lại quanh chiếc bảng trắng để suy nghĩ về vấn đề một cách cặn kẽ, chúng ta có thể phác thảo một giải pháp vận hành được, phần nào sẽ nhìn giống như sau: (i) người dùng tương tác với ứng dụng thông qua một giao diện chạy trên trình duyệt web hoặc ứng dụng trên điện thoại; (ii) ứng dụng tương tác với một hệ thống cơ sở dữ liệu thương mại để theo dõi trạng thái của từng người dùng và duy trì hồ sơ lịch sử các giao dịch; và (iii) (cũng là cốt lõi của ứng dụng) các logic nghiệp vụ (hay cũng có thể nói bộ não) mô tả cách thức xử lí cụ thể của ứng dụng trong từng tình huống có thể xảy ra.

Để xây dựng bộ não của ứng dụng này, ta phải xem xét tất cả mọi trường hợp mà chúng ta cho rằng sẽ gặp phải, qua đó đặt ra những quy tắc thích hợp. Ví dụ, mỗi lần người dùng nhấn để thêm một món đồ vào giỏ hàng, ta thêm một trường vào bảng giỏ hàng trong cơ sở dữ liệu, liên kết ID của người dùng với ID của món hàng được yêu cầu. Mặc dù hầu như rất ít lập trình viên có thể làm đúng hết trong lần đầu tiên, (sẽ cần vài lần chạy kiểm tra để xử lý hết được những trường hợp hiểm hóc), hầu như phần lớn ta có thể lập trình được từ những định đề cơ bản và tự tin chạy ứng dụng trước khi được dùng bởi một khách hàng thực sự nào. Khả năng phát triển những sản phầm và hệ thống tự động từ những định đề cơ bản, thường là trong những điều kiện mới lạ, là một kỳ công trong suy luận và nhận thức của con người. Và khi bạn có thể tạo ra một giải pháp hoạt động được trong mọi tình huống, bạn không nên sử dụng học máy.

May mắn thay cho cộng đồng đang tăng trưởng của các nhà khoa học về học máy, nhiều tác vụ mà chúng ta muốn tự động hoá không dễ dàng bị khuất phục bởi sự tài tình của con người. Thử tưởng tượng bạn đang quây quần bên tấm bảng trắng với những bộ não thông minh nhất mà bạn biết, nhưng lần này bạn đang đương đầu với một trong những vấn đề dưới đây:

* Viết một chương trình dự báo thời tiết ngày mai, cho biết trước thông tin địa lý, hình ảnh vệ tinh, và một chuỗi dữ liệu thời tiết trong quá khứ.
* Viết một chương trình lấy đầu vào là một câu hỏi, được diễn đạt không theo khuôn mẫu nào, và trả lời nó một cách chính xác.
* Viết một chương trình hiển thị ra cho người dùng những sản phẩm mà họ có khả năng cao sẽ thích, nhưng lại ít có khả năng gặp được khi duyệt qua môt cách tự nhiên.

Trong mỗi trường hợp trên, cho dù có là lập trình viên thượng thừa cũng không thể lập trình lên được từ con số không. Có nhiều lý do khác nhau. Đôi khi chương trình mà chúng ta cần lại đi theo một khuôn mẫu thay đổi theo thời gian và chương trình của chúng ta cần phải thích ứng với điều đó. Trong trường hợp khác, mối quan hệ (giả dụ như giữa các điểm ảnh và các hạng mục trừu tượng) có thể là quá phức tạp, yêu cầu hàng ngàn hàng triệu phép tính vượt ngoài khả năng thấu hiểu của nhận thức chúng ta (mặc dù mắt của chúng ta có thể xử lý tác vụ này một cách dễ dàng). Học máy (Machine Learning - ML) là lĩnh vực nghiên cứu những kỹ thuật tiên tiến mà có thể học từ kinh nghiệm. Khi thuật toán ML tích luỹ thêm nhiều kinh nghiệm, thường là dưới dạng dữ liệu quan sát hoặc tương tác với môi trường, chất lượng của nó sẽ tăng lên. Tương phản với hệ thống thương mại điện tử tất định của chúng ta, khi mà nó luôn tuân theo cùng logic nghiệp vụ đã có, mặc cho đã tích luỹ thêm bao nhiêu kinh nghiệm, tận cho tới khi lập trình viên tự học và quyết định rằng đã tới lúc cập nhật phần mềm này. Trong cuốn sách này, chúng tôi sẽ dạy cho bạn về những điều căn bản nhất trong học máy, và tập trung đặc biệt vào học sâu, một tập hợp hùng mạnh những kỹ thuật đang thúc đẩy sự đổi mới ở nhiều lĩnh vực khác nhau như thị giác máy tính, xử lý ngôn ngữ tự nhiên, chăm sóc y tế và nghiên cứu cấu trúc gen.

### 2.1.1 Lịch sử của học sâu



*Machine Learning đang dần trở nên phổ biến trong cuộc sống ngày nay.*

* 1950 - Nhà bác học Alan Turing đã tạo ra "Turing Test (phép thử Turing)" để xác định xem liệu một máy tính có trí thông minh thực sự hay không. Để vượt qua bài kiểm tra đó, một máy tính phải có khả năng đánh lừa một con người tin rằng nó cũng là con người.
* 1952 - Arthur Samuel đã viết ra chương trình học máy (computer learning) đầu tiên. Chương trình này là trò chơi cờ đam, và hãng máy tính IBM đã cải tiến trò chơi này để nó có thể tự học và tổ chức những nước đi trong chiến lược để giành chiến thắng.
* 1957 - Frank Rosenblatt đã thiết kế mạng nơron (neural network) đầu tiên cho máy tính, trong đó mô phỏng quá trình suy nghĩ của bộ não con người.
* 1967 - Thuật toán "nearest neighbor" đã được viết, cho phép các máy tính bắt đầu sử dụng những mẫu nhận dạng (pattern recognition) rất cơ bản. Nó được sử dụng để vẽ ra lộ trình cho một người bán hàng có thể bắt đầu đi từ một thành phố ngẫu nhiên nhưng đảm bảo anh ta sẽ đi qua tất cả các thành phố khác theo một quãng đường ngắn nhất.
* 1979 - Sinh viên tại trường đại học Stanford đã phát minh ra giỏ hàng "Stanford Cart" có thể điều hướng để tránh các chướng ngại vật trong một căn phòng.
* 1981 - Gerald Dejong giới thiệu về khái niệm Explanation Based Learning (EBL), trong đó một máy tính phân tích dữ liệu huấn luyện và tạo ra một quy tắc chung để nó có thể làm theo bằng cách loại bỏ đi những dữ liệu không quan trọng.
* 1985 - Terry Sejnowski đã phát minh ra NetTalk, nó có thể học cách phát âm các từ giống như cách một đứa trẻ tập nói.
* 1990s - Machine Learning đã dịch chuyển từ cách tiếp cận hướng kiến thức (knowledge-driven) sang cách tiếp cận hướng dữ liệu (data-driven). Các nhà khoa học bắt đầu tạo ra các chương trình cho máy tính để phân tích một lượng lớn dữ liệu và rút ra các kết luận - hay là "học" từ các kết quả đó.
* 1997 - Deep Blue của hãng IBM đã đánh bại nhà vô địch cờ vua thế giới.
* 2006 - Geoffrey Hinton đã đưa ra một thuật ngữ "deep learning" để giải thích các thuật toán mới cho phép máy tính "nhìn thấy" và phân biệt các đối tượng và văn bản trong các hình ảnh và video.
* 2010 - Microsoft Kinect có thể theo dõi 20 hành vi của con người ở một tốc độ 30 lần mỗi giây, cho phép con người tương tác với máy tính thông qua các hành động và cử chỉ.
* 2011 - Máy tính Watson của hãng IBM đã đánh bại các đối thủ là con người tại Jeopardy.
* 2011 - Google Brain đã được phát triển, và mạng deep nơron (deep neural network) của nó có thể học để phát hiện và phân loại nhiều đối tượng theo cách mà một con mèo thực hiện.
* 2012 - X Lab của Google phát triển một thuật toán machine learning có khả năng tự động duyệt qua các video trên YouTube để xác định xem video nào có chứa những con mèo.
* 2014 - Facebook phát triển DeepFace, một phần mềm thuật toán có thể nhận dạng hoặc xác minh các cá nhân dựa vào hình ảnh ở mức độ giống như con người có thể.
* 2015 - Amazon ra mắt nền tảng machine learning riêng của mình.
* 2015 - Microsoft tạo ra Distributed Machine Learning Toolkit, trong đó cho phép phân phối hiệu quả các vấn đề machine learning trên nhiều máy tính.
* 2015 - Hơn 3.000 nhà nghiên cứu AI và Robotics, được sự ủng hộ bởi những nhà khoa học nổi tiếng như Stephen Hawking, Elon Musk và Steve Wozniak (và nhiều người khác), đã ký vào một bức thư ngỏ để cảnh báo về sự nguy hiểm của vũ khí tự động trong việc lựa chọn và tham gia vào các mục tiêu mà không có sự can thiệp của con người.
* 2016 - Thuật toán trí tuệ nhân tạo của Google đã đánh bại nhà vô địch trò chơi Cờ Vây, được cho là trò chơi phức tạp nhất thế giới (khó hơn trò chơi cờ vua rất nhiều). Thuật toán AlphaGo được phát triển bởi Google DeepMind đã giành chiến thắng 4/5 trước nhà vô địch Cờ Vây.

### 2.1.2 Các thành phần chính: Dữ liệu, Mô hình và Thuật toán

#### 2.1.2.1 Dữ liệu

Ta không thể làm khoa học dữ liệu mà không có dữ liệu. Chúng ta sẽ tốn rất nhiều giấy mực để cân nhắc chính xác những gì cấu thành nên dữ liệu, nhưng bây giờ chúng ta sẽ rẽ sang khía cạnh thực tế và tập trung vào các thuộc tính quan trọng cần quan tâm. Thông thường, chúng ta quan tâm đến một bộ mẫu (còn được gọi là điểm dữ liệu, ví dụ hoặc trường hợp). Để làm việc với dữ liệu một cách hữu ích, chúng ta thường cần có một cách biễu diễn chúng phù hợp dưới dạng số. Mỗi ví dụ thường bao gồm một bộ thuộc tính số gọi là đặc trưng.

Nếu chúng ta đang làm việc với dữ liệu hình ảnh, mỗi bức ảnh riêng lẻ có thể tạo thành một mẫu được biểu diễn bởi một danh sách các giá trị số theo thứ tự tương ứng với độ sáng của từng pixel. Một bức ảnh màu có kích thước 200×200 sẽ bao gồm 200×200×3=120000 giá trị số, tương ứng với độ sáng của các kênh màu đỏ, xanh lá cây và xanh dương cho từng vị trí trong không gian. Trong một tác vụ truyền thống hơn, chúng ta có thể cố gắng dự đoán xem một bệnh nhân liệu có cơ hội sống sót hay không, dựa trên bộ đặc trưng tiêu chuẩn cho trước như tuổi, các triệu chứng quan trọng, thông số chẩn đoán, .v.v.

Tuy nhiên, không phải tất cả dữ liệu có thể được dễ dàng biểu diễn dưới dạng vector có độ dài cố định. Đôi khi ta có thể mong đợi hình ảnh từ kính hiển vi đến từ thiết bị tiêu chuẩn, nhưng ta không thể mong đợi hình ảnh được khai thác từ Internet sẽ hiển thị với cùng độ phân giải hoặc tỉ lệ được. Đối với hình ảnh, ta có thể tính đến việc cắt xén nhằm đưa chúng về kích thước tiêu chuẩn, nhưng chiến lược này chỉ đưa ta đến đấy mà thôi. Và ta có nguy cơ sẽ mất đi thông tin trong các phần bị cắt bỏ. Hơn nữa, dữ liệu văn bản không thích hợp với cách biểu diễn dưới dạng vector có độ dài cố định. Một lợi thế lớn của học sâu so với các phương pháp truyền thống đó là các mô hình học sâu hiện đại có thể xử lý dữ liệu có độ dài biến đổi một cách uyển chuyển hơn.

Nhìn chung, chúng ta có càng nhiều dữ liệu thì công việc sẽ càng dễ dàng hơn. Khi ta có nhiều dữ liệu hơn, ta có thể huấn luyện ra những mô hình mạnh mẽ hơn và ít phụ thuộc hơn vào các giả định được hình thành từ trước. Việc chuyển từ dữ liệu nhỏ sang dữ liệu lớn là một đóng góp chính cho sự thành công của học sâu hiện đại. Để cho rõ hơn, nhiều mô hình thú vị nhất trong học sâu có thể không hoạt động nếu như không có bộ dữ liệu lớn. Một số người vẫn áp dụng học sâu với số dữ liệu ít ỏi mà mình có được, nhưng trong trường hợp này nó không tốt hơn các cách tiếp cận truyền thống.

Cuối cùng, có nhiều dữ liệu và xử lý dữ liệu một cách khéo léo thôi thì chưa đủ. Ta cần những dữ liệu đúng. Nếu dữ liệu mang đầy lỗi, hoặc nếu các đặc trưng được chọn lại không dự đoán được số lượng mục tiêu cần quan tâm, việc học sẽ thất bại.

#### 2.1.2.2 Mô hình

Phần lớn học máy đều liên quan đến việc biến đổi dữ liệu theo một cách nào đó. Có thể ta muốn xây dựng một hệ thống nhận ảnh đầu vào và dự đoán mức độ cười của khuôn mặt trong ảnh. Hoặc đó cũng có thể là một hệ thống nhận vào dữ liệu đo đạc từ cảm biến và dự đoán độ bình thường hay bất thường của chúng. Ở đây chúng ta gọi mô hình là một hệ thống tính toán nhận đầu vào là một dạng dữ liệu và sau đó trả về kết quả dự đoán, có thể ở một dạng dữ liệu khác. Cụ thể, ta quan tâm tới các mô hình thống kê mà ta có thể ước lượng được từ dữ liệu. Dù các mô hình đơn giản hoàn toàn có thể giải quyết các bài toán đơn giản phù hợp, những bài toán được để tâm tới trong cuốn sách này sẽ đẩy các phương pháp cổ điển tới giới hạn của chúng. Điểm khác biệt chính của học sâu so với các phương pháp cổ điển là các mô hình mạnh mẽ mà nó nhắm vào. Những mô hình đó bao gồm rất nhiều phép biến đổi dữ liệu liên tiếp, được liên kết với nhau từ trên xuống dưới, và đó cũng là ý nghĩa của cái tên “học sâu”. Trong quá trình thảo luận về các mạng nơ-ron sâu, ta cũng sẽ nhắc tới các phương pháp truyền thống.

#### 2.1.2.3 Hàm mục tiêu

Như chúng tôi đã giới thiệu học máy là việc “học từ kinh nghiệm”. Học ở đây tức là việc tiến bộ ở một tác vụ nào đó theo thời gian. Nhưng ai biết được như thế nào là tiến bộ? Thử tưởng tượng ta đang đề xuất cập nhật mô hình, nhưng một số người có thể có bất đồng về việc bản cập nhật này có giúp cải thiện mô hình hay không.

Để có thể phát triển một mô hình toán học chính quy cho học máy, chúng ta cần những phép đo chính quy xem mô hình đang tốt (hoặc tệ) như thế nào. Trong học máy, hay rộng hơn là lĩnh vực tối ưu hoá, ta gọi chúng là các hàm mục tiêu (objective function). Theo quy ước, ta thường định nghĩa các hàm tối ưu sao cho giá trị càng thấp thì mô hình càng tốt. Nhưng đó cũng chỉ là một quy ước ngầm. Bạn có thể lấy một hàm ff sao cho giá trị càng cao thì càng tốt, sau đó đặt một hàm tương đương f′=−ff′=−f, có giá trị càng thấp thì mô hình càng tốt. Chính vì ta mong muốn hàm có giá trị thấp, nó còn được gọi là hàm mất mát (loss function) và hàm chi phí (cost function).

Khi muốn dự đoán một giá trị số, hàm mục tiêu phổ biến nhất là hàm bình phương sai số (y−y^)2(y−y^)2. Với bài toán phân loại, mục tiêu phổ biến nhất là cực tiểu hóa tỉ lệ lỗi, tức tỉ lệ mẫu mà dự đoán của mô hình lệch với nhãn thực tế. Một vài hàm mục tiêu (ví dụ như bình phương sai số) khá dễ tối ưu hóa. Các hàm khác (như tỉ lệ lỗi) lại khó tối ưu hóa trực tiếp, có thể do các hàm này không khả vi hoặc những vấn đề khác. Trong những trường hợp như vậy, ta thường tối ưu hóa một hàm mục tiêu thay thế (surrogate objective).

Thông thường, hàm mất mát được định nghĩa theo các tham số mô hình và phụ thuộc vào tập dữ liệu. Những giá trị tham số mô hình tốt nhất được học bằng cách cực tiểu hóa hàm mất mát trên một tập huấn luyện bao gồm các mẫu được thu thập cho việc huấn luyện. Tuy nhiên, mô hình hoạt động tốt trên tập huấn luyện không có nghĩa là nó sẽ hoạt động tốt trên dữ liệu kiểm tra (mà mô hình chưa nhìn thấy). Bởi vậy, ta thường chia dữ liệu sẵn có thành hai phần: dữ liệu huấn luyện (để khớp các tham số mô hình) và dữ liệu kiểm tra (được giữ lại cho việc đánh giá). Sau đó ta quan sát hai đại lượng:

* Lỗi huấn luyện: Lỗi trên dữ liệu được dùng để huấn luyện mô hình. Bạn có thể coi nó như điểm của một sinh viên trên bài thi thử để chuẩn bị cho bài thi thật. Ngay cả khi kết quả thi thử khả quan, không thể đảm bảo rằng bài thi thật sẽ đạt kết quả tốt.
* Lỗi kiểm tra: Đây là lỗi trên tập kiểm tra (không dùng để huấn luyện mô hình). Đại lượng này có thể chênh lệch đáng kể so với lỗi huấn luyện. Khi một mô hình hoạt động tốt trên tập huấn luyện nhưng lại không có khả năng tổng quát hóa trên dữ liệu chưa gặp, ta nói rằng mô hình bị quá khớp (overfit). Theo ngôn ngữ thường ngày, đây là hiện tượng “học lệch tủ” khi kết quả bài thi thật rất kém mặc dù có kết quả cao trong bài thi thử.

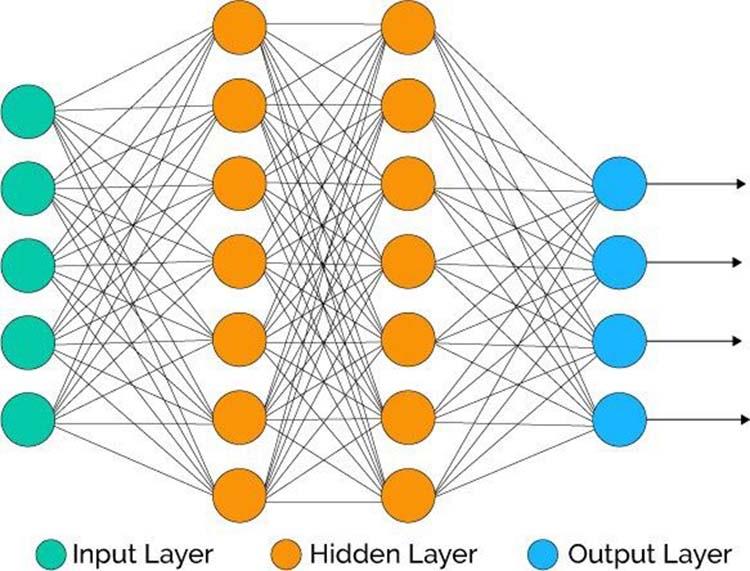
#### 2.1.2.4 Các thuật toán tối ưu

Một khi ta có dữ liệu, một mô hình và một hàm mục tiêu rõ ràng, ta cần một thuật toán có khả năng tìm kiếm các tham số khả dĩ tốt nhất để cực tiểu hóa hàm mất mát. Các thuật toán tối ưu phổ biến nhất cho mạng nơ-ron đều theo một hướng tiếp cận gọi là hạ gradient. Một cách ngắn gọn, tại mỗi bước và với mỗi tham số, ta kiểm tra xem hàm mất mát thay đổi như thế nào nếu ta thay đổi tham số đó bởi một lượng nhỏ. Sau đó các tham số này được cập nhật theo hướng làm giảm hàm mất mát.

#### 2.1.2.5 Mạng Nơ ron

Người đặt nền móng cho mạng nơron nhân tạo là hai nhà thần kinh học McCulloch và Pitts (1943) đã sử dụng logic hình thức để mô hình hóa cho mạng nơ ron. Mạng nơ ron nhân tạo được sử dụng rộng rãi trong thực tế do chúng có một số đặc điểm chính sau:

* Khả năng tạo ra ánh xạ phi tuyến giữa một tập dữ liệu quá khứ với dữ liệu tương lai trong chuỗi dữ liệu (data time series). Đặc trưng này được khai thác để xây dựng các mô hình dự đoán dữ liệu.
* Khả năng xấp xỉ hàm số. Cho phép sử dụng mạng nơ ron để mô hình hóa cho một hàm phi tuyến liên tục bất kỳ với độ chính xác tùy ý.
* Khả năng học và khái quát dữ liệu từ mẫu.



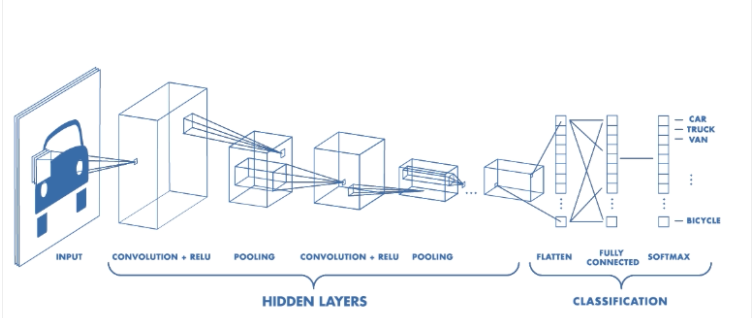
Hình ảnh 2.9: Hình ảnh cấu trúc mạng noron

Đối với đề tài của em, em sẽ sử dụng mạng nơ ron để nhận dạng được ảnh các ký tự đã được cắt từ hình ảnh biển số xe và sau đó sẽ chuyển sang dạng chuỗi ký tự, chính là biển số xe cần nhận dạng.

#### Mạng Nơ ron tích chập (Convolutional Neural Network)

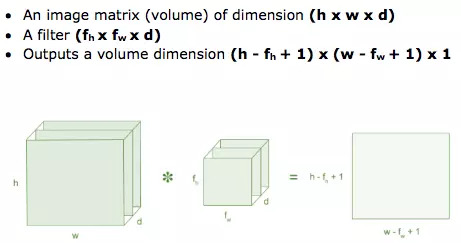
Mạng nơ ron tích chập (Convolutional Neural Network) là một trong những mô hình Học Sâu (Deep Learning) tiên tiến. Nó giúp chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao.

* Cấu trúc mạng nơ ron tích chập bao gồm các lớp cơ bản như sau: convolution layer + nonlinear layer, pooling layer và fully connected layer. Các lớp này liên kết với nhau theo một thứ tự nhất định.



Hình ảnh 2.10: Cấu trúc mạng nơ ron tích chập

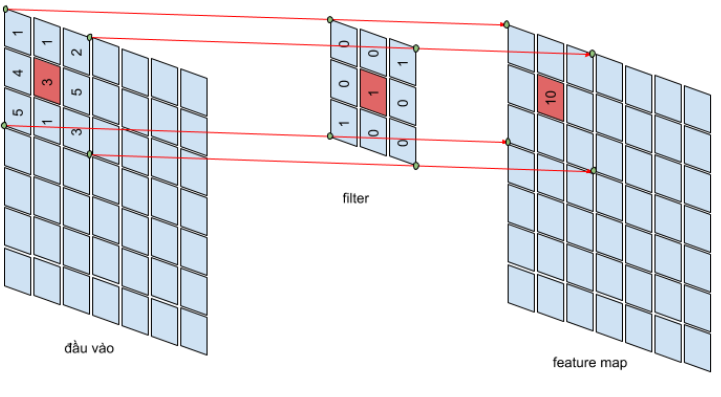
* Convolution layer là lớp quan trọng nhất và cũng là lớp đầu tiên của mô hình mạng nơ ron tích chập. Lớp này có chức năng chính là phát hiện các đặc trưng có tính hiệu quả. Trong tầng này có 3 đối tượng chính: Ma trận đầu vào, bộ filter và feature map. Convolution layer nhận đầu vào là một ma trận nhiều chiều và một bộ lọc filter cần phải học. Bộ filter này sẽ trượt qua từng vị trí trên ảnh để nhân chập giữa bộ filter và phần tương ứng của bức ảnh. Ma trận cho ra bởi quá trình trên được gọi là feature map.



Hình ảnh 2.11: Ảnh mô phỏng công thức tích chập

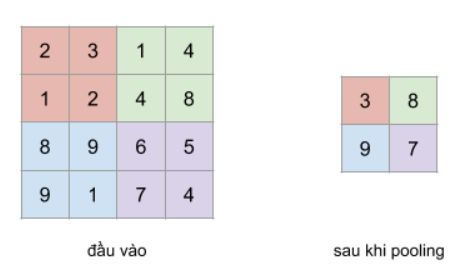
* Convolution layer có chức năng chính là phát hiện đặc trưng cụ thể của bức ảnh. Những đặc trưng này bao gồm đặc trưng cơ bản như góc, cạnh hay những đặc trưng phức tạp hơn như texture của ảnh. Vì bộ filter này quét qua toàn bộ bức ảnh nên những đặc trưng này có thể nằm ở vị trí bất kỳ trong bức ảnh, cho dù ảnh bị xoay trái/phải thì những đặc trưng này vẫn bị phát hiện.
* Kích thước của bộ filter là tham số quan trọng nhất của convolution layer. Bộ filter có thể có nhiều kích thước như 3x3, 5x5, 7x7…. Kích thước filter phổ biến nhất là 3x3 vì nó có những đặc trưng như sau:
* Rút trích được những đặc trưng mang tính cục bộ cao
* Phát hiện được các đặc trưng nhỏ hơn
* Giảm kích thước ảnh chậm hơn, cho phép mạng sâu hơn

Feature map chính là ma trận đặc trưng sau khi nhân chập ma trận đầu vào với filter:



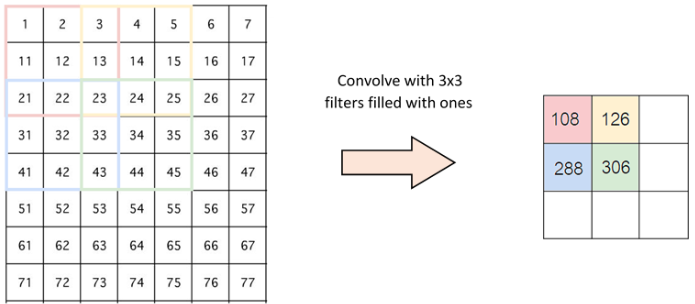
Hình ảnh 2.12: Hình ảnh nhân tích chập

* Một layer nữa cần quan tâm đến chính là Pooling layer. Một số loại pooling layer phổ biến như là max-pooling, average pooling, với chức năng chính là giảm chiều của layer trước đó. Với pooling có kích thước 2x2, ta cần phải trượt filter này trên những vùng ảnh có kích thước tương ứng rồi sau đó tính max hay average cho vùng ảnh đó.



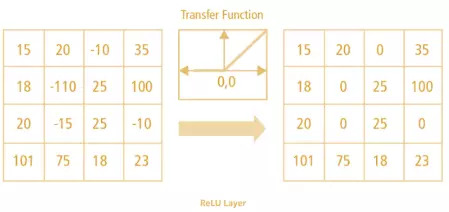
Hình ảnh 2.13: Ảnh mô phỏng bước max-pooling

* Stride là số pixel thay đổi trên ma trận đầu vào. Khi stride là 1 thì ta di chuyển các kernel 1 pixel. Khi stride là 2 thì ta di chuyển các kernel đi 2 pixel và tiếp tục như vậy.



Hình ảnh 2.14: Ảnh mô phỏng Bước nhảy - Stride

* Đường viền – Padding: Đôi khi kernel không phù hợp với hình ảnh đầu vào. Ta có 2 lựa chọn: Chèn thêm các số 0 vào 4 đường biên của hình ảnh (padding); Cắt bớt hình ảnh tại những điểm không phù hợp với kernel.
* Hàm phi tuyến – ReLU: ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là: ƒ (x) = max (0, x). ReLU giới thiệu tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm. Có 1 số hàm phi tuyến khác như tanh, sigmoid cũng có thể được sử dụng thay cho ReLU. Hầu hết người ta thường dùng ReLU vì nó có hiệu suất tốt.

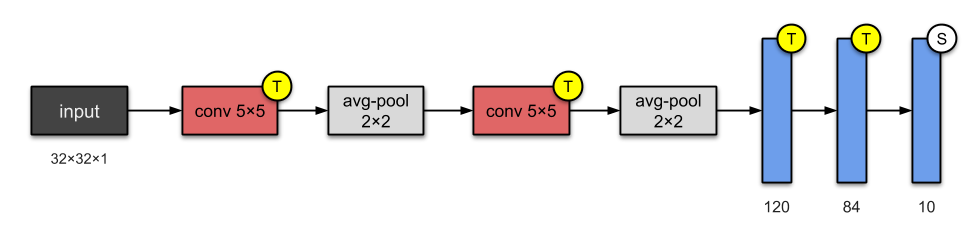


Hình ảnh 2.15: Ảnh mô phỏng hàm phi tuyết ReLU

* Layer cuối cùng của mô hình mạng nơ ron tích chập là fully connected layer. Tầng này có chức năng chuyển ma trận đặc trưng ở tầng trước đó thành vector chứa xác suất các đối tượng cần được dự đoán. Đối với bài toán nhận dạng biển số xe, sẽ có 31 lớp tương ứng được đánh nhãn từ 0-9 A-Z.

#### Các kiến trúc mạng Nơ ron tích chập tiêu biểu

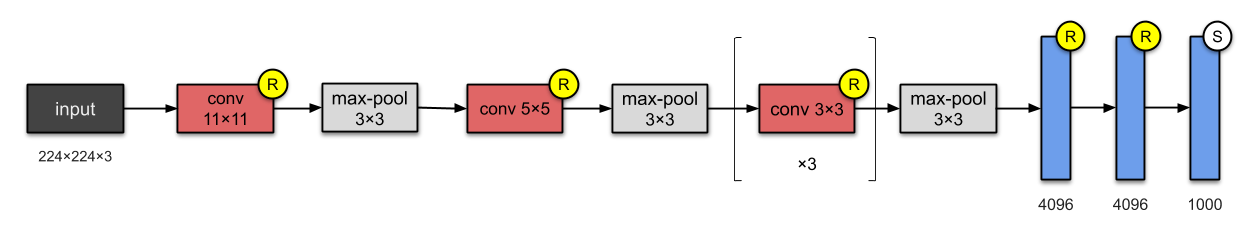
1. LetNet-5



Hình ảnh 2.16: Kiến trúc LetNet-5

* LetNet-5 là kiến trúc đầu tiên áp dụng mạng tích chập 2 chiều của giáo sư Yan Lecun, cha đẻ của kiến trúc CNN. Model ban đầu khá đơn giản và chỉ bao gồm 2 convolutional layers + 3 fully-connected layers. Mặc dù đơn giản nhưng nó có kết quả tốt hơn so với các thuật toán machine learning truyền thống khác trong phân loại chữ số viết tay như SVM, KNN.
* Trong kiến trúc mạng nơ ron đầu tiên, để giảm chiều dữ liệu, Yan Lecun sử dụng Sub-Sampling Layer là một Average-Pooling Layer (các layer nhằm mục đích giảm chiều dữ liệu mà không thay đổi đặc trưng, chúng ta còn gọi là Sub-Sampling Layer). Kiến trúc này khó hội tụ nên ngày nay chúng được thay thế bằng Max-Pooling.
* Đầu vào của mạng LeNet có kích thước nhỏ (chỉ 32x32) và ít layers nên số lượng tham số của nó chỉ khoảng 60 nghìn.

1. AlexNet

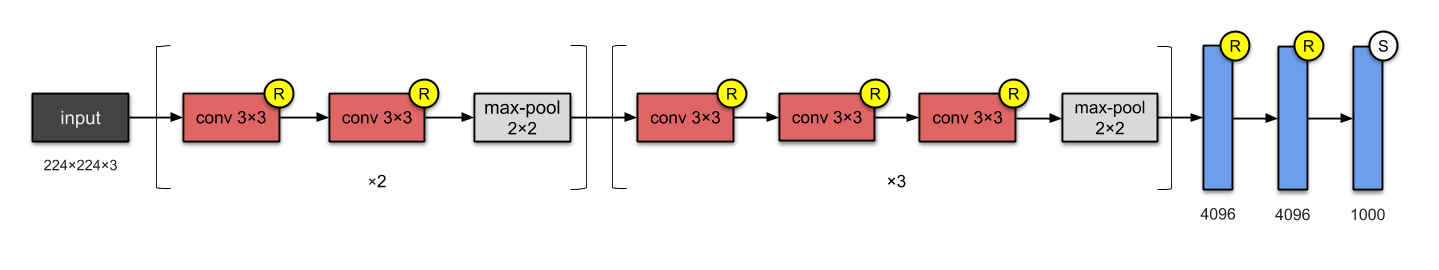


Hình ảnh 2.17: Kiến trúc AlexNet

AlexNet là mạng CNN được giới thiệu vào năm 2012 bởi Alex Krizhevsky và dành chiến thắng trong cuộc thi ImageNet với cách biệt khá lớn so với vị trí thứ hai. Lần đầu tiên AlexNet đã phá vỡ định kiến trước đó cho rằng các đặc trưng được học từ mô hình sẽ không tốt bằng các đặc trưng được tạo thủ công (thông qua các thuật toán SUFT, HOG, SHIFT). Ý tưởng của AlexNet dựa trên LetNet của Yan Lecun và cải tiến ở các điểm:

* Tăng kích thước đầu vào và độ sâu của mạng.
* Sử dụng các bộ lọc (*kernel* hoặc *filter*) với kích thước giảm dần qua các layers để phù hợp với kích thước của đặc trưng chung và đặc trưng riêng.
* Sử dụng local normalization để chuẩn hóa các layer giúp cho quá trình hội tụ nhanh hơn.

1. VGG-16



Hình ảnh 2.18: Kiến trúc VGG-16

* Với VGG-16, quan điểm về một mạng nơ ron sâu hơn sẽ giúp ích cho cải thiện độ chính xác của mô hình tốt hơn. Về kiến trúc thì VGG-16 vẫn dữ các đặc điểm của AlexNet nhưng có những cải tiến:
* Kiến trúc VGG-16 sâu hơn, bao gồm 13 layers tích chập 2 chiều (thay vì 5 so với AlexNet) và 3 layers fully connected.
* Lần đầu tiên trong VGG-16 chúng ta xuất hiện khái niệm về khối tích chập (block). Đây là những kiến trúc gồm một tập hợp các layers CNN được lặp lại giống nhau. Kiến trúc khối đã khởi nguồn cho một dạng kiến trúc hình mẫu rất thường gặp ở các mạng CNN kể từ đó.
* VGG-16 cũng kế thừa lại hàm activation ReLU ở AlexNet.
* VGG-16 cũng là kiến trúc đầu tiên thay đổi thứ tự của các block khi xếp nhiều layers CNN + max pooling thay vì xen kẽ chỉ một layer CNN + max pooling
* VGG-16 chỉ sử dụng các bộ lọc kích thước nhỏ 3x3 thay vì nhiều kích thước bộ lọc như AlexNet. Kích thước bộ lọc nhỏ sẽ giúp giảm số lượng tham số cho mô hình và mang lại hiệu quả tính toán hơn. VD: Nếu sử dụng 2 bộ lọc kích thước 3 x 3 trên một feature map (là output của một layer CNN) có độ sâu là 3 thì ta sẽ cần n\_filters x kernel\_size x kernel\_size x n\_channels = 2 x 3 x 3 x 3 = 54 tham số. Nhưng nếu sử dụng 1 bộ lọc kích thước 5 x 5 sẽ cần 5 x 5 x 3 = 75 tham số. 2 bộ lọc 3 x 3 vẫn mang lại hiệu quả hơn so với 1 bộ lọc 5 x 5.
* Mạng VGG-16 sâu hơn so với AlexNet và số lượng tham số của nó lên tới 138 triệu tham số. Đây là một trong những mạng mà có số lượng tham số lớn nhất. Kết quả của nó hiện đang xếp thứ 2 trên bộ dữ liệu ImageNet validation ở thời điểm công khai. Ngoài ra còn một phiên bản nữa của VGG-16 là VGG-19 tăng cường thêm 3 layers về độ sâu.
* Bắt đầu từ VGG-16, một hình mẫu chung cho các mạng CNN với các tác vụ học có giám sát trong xử lý ảnh đã bắt đầu hình thành đó là các mạng trở nên sâu hơn và sử dụng các block dạng [Conv2D\*n + Max Pooling].

## **Giới thiệu về mạng Generative Adversarial Networks(GAN)**

### **GAN là gì?**

A collage of two people

Description automatically generated with medium confidenceMạng GAN (Generative Adversarial Network) là một trong những mô hình deep learning phổ biến nhất hiện nay, được sử dụng để tạo ra các nội dung mới, chẳng hạn như ảnh, âm thanh hoặc văn bản. GAN thuộc nhóm generative model. Generative là tính từ nghĩa là khả năng sinh ra, model nghĩa là mô hình. Vậy hiểu đơn giản generative model nghĩa là mô hình có khả năng sinh ra dữ liệu. Hay nói cách khác, GAN là mô hình có khả năng sinh ra dữ liệu mới. Ví dụ như những ảnh mặt người ở dưới bạn thấy là do GAN sinh ra, không phải mặt người thật. Dữ liệu sinh ra nhìn như thật nhưng không phải thật. Mô hình GAN bao gồm hai phần chính: một mạng sinh và một mạng phân biệt. Mạng sinh được huấn luyện để tạo ra các dữ liệu mới từ nhiễu ngẫu nhiên, trong khi mạng phân biệt được huấn luyện để phân biệt giữa các dữ liệu được tạo bởi mạng sinh và các dữ liệu thật sự.

### **Mô hình phân biệt (Discriminative model)**

Chart, box and whisker chart

Description automatically generated

Trước đây, hầu hết các mạng đều thuộc dạng Discriminative model, tức trải qua quá trình huấn luyện, model đạt được khả năng định vị được vị trí của một điểm dữ liệu trong phân bố dữ liệu (điển hình là bài toán phân loại). Ví dụ: một đứa trẻ được cho xem 1000 ảnh có mèo và 1000 ảnh không mèo (ảnh kèm nhãn có-không). Trải qua quá trình quan sát và học hỏi, đứa trẻ đó sẽ có khả năng phân biệt những ảnh mới xem ảnh nào có, ảnh nào không có mèo. Đó là điển hình cho một "discriminative model". Các bài toán classify, regression, image semantic segmentation, object detection ... bản chất đều liên quan tới "discriminative model".

Chart

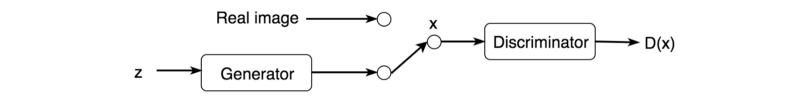
Description automatically generated

### Chart, box and whisker chart Description automatically generatedMô hình sinh(Generative model)

Cũng với ví dụ trên, trải qua sau quá trình học liệu đứa bé có thể tự hình dung ra hình ảnh một con mèo mới nào đó. Việc sinh ra ảnh mới đó là việc của "generative model".

### Kiến trúc mạng GAN

*Note: Để thống nhất và dễ hiểu, ta sẽ lấy kiểu dữ liệu là ảnh để mô tả thuật toán, các dạng dữ liệu khác như âm thanh, tín hiệu đều tương tự.*



GAN được kết hợp từ 2 model: generator - G và discriminator - D. GAN giống như 1 trò chơi minimax, trò cảnh sát tội phạm: tội phạm G tạo ra tiền giả, cảnh sát D học cách phân biệt thật giả. Cảnh sát càng cố gắng phân biệt tiền thật-giả thì tội phạm lại dựa vào feedback của cảnh sát để cải thiện khả năng tạo tiền giả của mình, cố gắng khiến cảnh sát phân biệt nhầm.

Diagram

Description automatically generated

Thuật toán như sau:

B1: Từ một nhiễu z bất kì, G sinh ra fake-image G(z) có kích thước như ảnh thật (ảnh thật là x). Tại lần sinh đầu tiên, G(z) hoàn toàn là ảnh nhiễu, không có bất kì nội dung gì đặc biệt

B2: x và G(z) cùng được đưa vào D kèm nhãn đúng sai. Train D để học khả năng phân biệt ảnh thật, ảnh giả.

B3: Đưa G(z) vào D, dựa vào feedback của D trả về, G sẽ cải thiện khả năng fake của mình.

B4: Quá trình trên sẽ lặp đi lặp lại như vậy, D dần cải thiện khả năng phân biệt, G dần cải thiện khả năng fake. Đến khi nào D không thể phân biệt được ảnh nào là ảnh do G tạo ra, ảnh nào là x, khi đó quá trình dừng lại.

Input cho G là một nhiễu z, được sinh ngẫu nhiên từ một phân phối xác suất (phổ biến nhất là Gaussian). Kiến trúc GAN phổ biến là DCGAN - Deep Convolution GAN: cả G và D là các mạng Convolution nhiều lớp, sâu như hình dưới đây:

Diagram

Description automatically generated with low confidence

### Lan truyền ngược (Backpropagation)

Để train được D, input gồm cả G(z) và x kèm nhãn. Như vậy, mục tiêu của D là maximinze:

Text

Description automatically generated with medium confidence

Để train được G, ta dựa vào D(G(z)). Bước này nhằm mục đích update các weight của G sao cho G(z) có thể đánh lừa được D, khiến D đoán nhầm nhãn của G(z) là y = 1. G cố gắng minimize:

Text

Description automatically generated with medium confidence

Tổng quát lại, D, G là kết quả của quá trình:



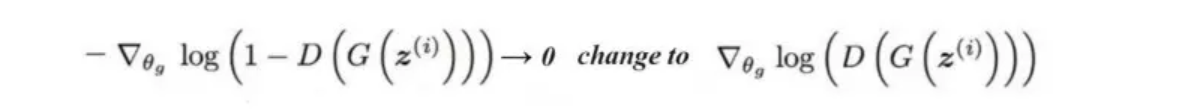
Dưới đây là pseudo code minh họa thuật toán. Trong 2 bài viết tiếp theo mình sẽ viết bài hướng dẫn code 2 ứng dụng GAN đơn giản: sinh ảnh chữ viết tay (dựa trên tập mnist) và sinh ra khuôn mặt nhân vật Anime:

Text

Description automatically generated

### Độ dốc giảm dần của đạo hàm mạng sinh

Trong quá trình train, ta dễ gặp phải vấn đề khi gradient cho G. Thường tại những bước đầu, D rất dễ dàng nhận dạng ảnh fake do G tạo ra. Điều đó khiến cho V = -log(1 - D(G(z))) có giá trị xấp xỉ 0. Điều này gây ra hiện tượng gradient vanishing khiến model khó train, khó hội tụ. Để cải thiện, ta thay đổi công thức 1 chút:



Như vậy, qua bài này ta đã biết thêm những khái niệm về GAN, cách hoạt động và khả năng kì diệu của nó. Từ GAN, người ta đã phát triển hàng trăm thuật toán khác nhau trong nhiều lĩnh vực lí thú.

# CHƯƠNG 3: THIẾT KẾ VÀ THỰC NGHIỆM

## 3.1. Mô hình sinh ảnh hoạt hình từ ảnh thật

### 3.1.1 Dữ liệu phục vụ cho bài toán

Tập dữ liệu train\_photo chứa hơn 6000 ảnh thật dùng cho tác vụ sinh ảnh hoạt hình. Bài toán sử dụng các bộ dữ liệu đại diện cho một phong cách hoạt hình khác nhau.

A picture containing table

Description automatically generated

Và trong đó sẽ có 2 thư mục con là style và smooth

A picture containing graphical user interface

Description automatically generated

Thư mục ảnh style chứa các ảnh mẫu về phong cách anime mà mô hình sẽ học. Trong quá trình huấn luyện, mô hình sẽ học cách tái tạo phong cách đó trên các ảnh đầu vào. Thư mục này có thể chứa các ảnh anime được chọn bởi người dùng hoặc được tải xuống từ các nguồn khác nhau.

Thư mục ảnh smooth chứa các ảnh mẫu để giúp mô hình học cách tạo ra các đường cong và kết cấu mịn trên các bề mặt trong ảnh đầu vào. Các ảnh trong thư mục này có thể là các ảnh vật thể, cảnh quan, hoặc bất kỳ loại ảnh nào có đặc tính cần học để tạo ra các phần mịn và các đường cong mềm mại trong ảnh anime đầu ra.

### 3.1.2 Chuẩn bị dữ liệu cho huấn luyện

\*Chuẩn bị ảnh style

Ta cần chuẩn bị một video của phim hoạt hình đó, và thay đổi các tham số trong file video\_to\_images.py.

Text

Description automatically generated

Ý nghĩa các tham số:

--video-path: đường dẫn tới video hoạt hình

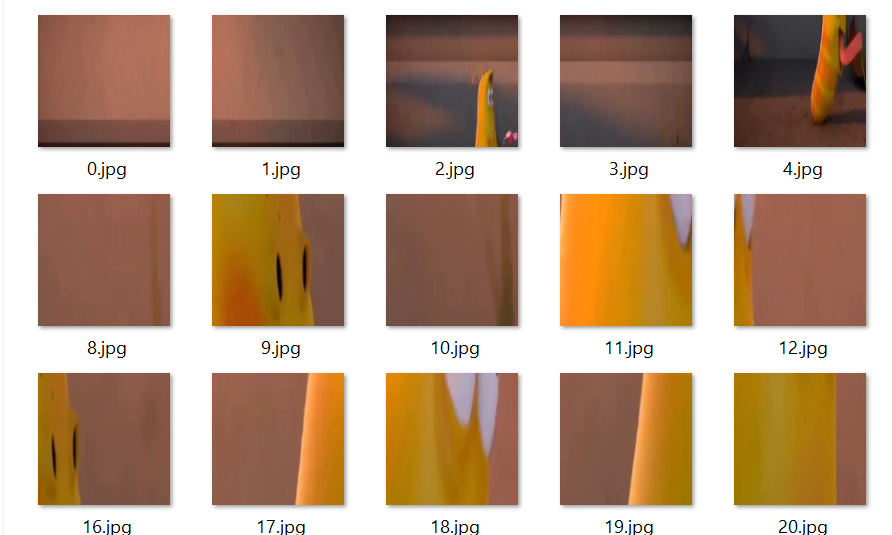
--save-path: đường dẫn thư mục lưu ảnh được cắt ra

--start: thời gian bắt đầu cắt video

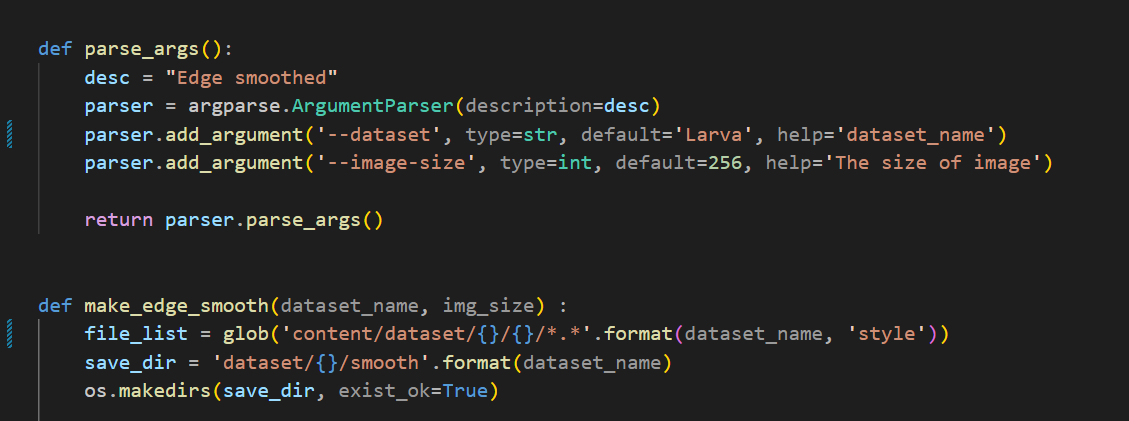
--end: thời gian kết thúc của cắt video

--image-size: kích thước của ảnh trả về

🡺 Kết quả trả về là thư mục style chứa các ảnh 255x255 của phong cách hoạt hình đó



\*Chuẩn bị ảnh smooth

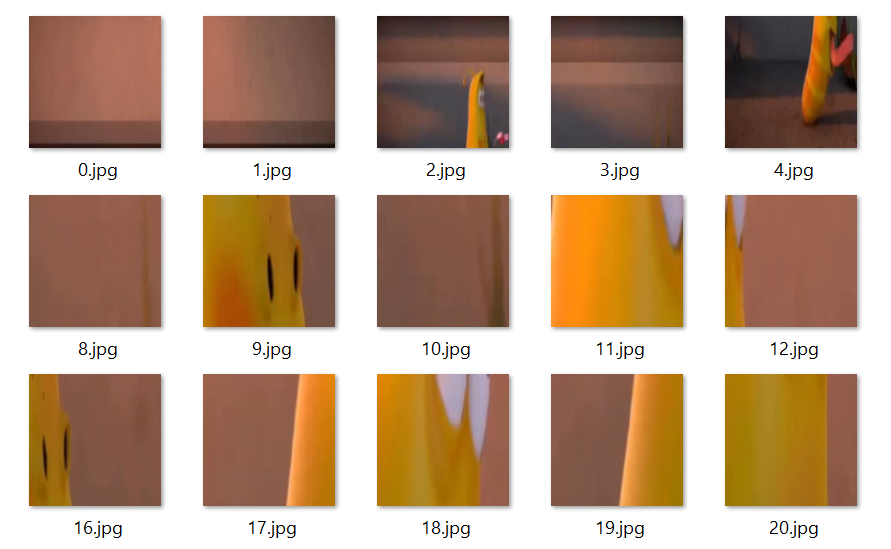


Tương tự ta cũng cần thay đổi các tham số cho phù hợp

--dataset: tên của phong cách hoạt hình cần tạo ra ảnh smooth

--image-size: kích thước của ảnh trả về

🡺Kết quả: các ảnh đã được chuyển đổi và lưu trong thư mục smooth



### 3.1.3 Thiết kế mô hình

-Phương pháp sử dụng: Mạng GAN

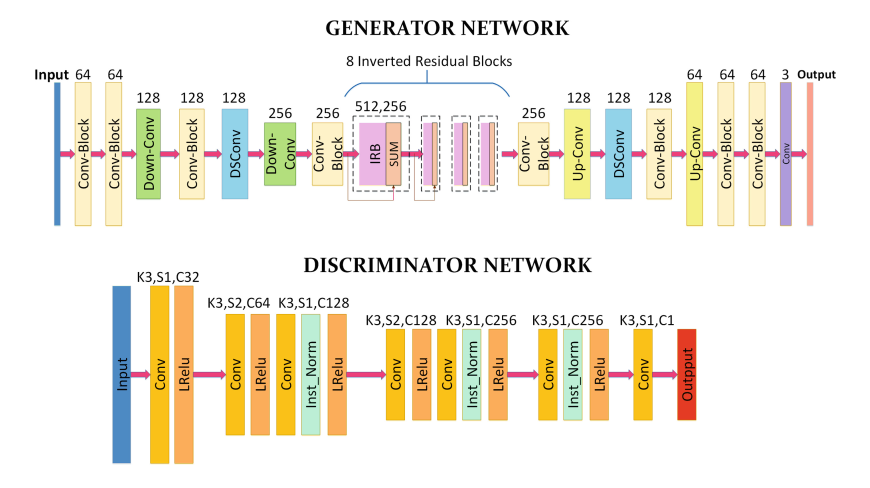
Diagram

Description automatically generated

Hình 3. 1 Mô phỏng cách hoạt động của mạng GAN

* Khối sinh: cố gắng đánh lừa người phân biệt đối xử bằng cách tạo ra những hình ảnh giống như thật
* Khối phân biệt: cố gắng phân biệt giữa hình ảnh thật và giả

Cấu trúc của 2 mô hình sinh và phân biệt



Hình 3. 2 Kiến trúc mô hình GAN sử dụng trong bài toán

#### 3.1.3.1 Cách hoạt động chi tiết của mạng sinh(Generator)

Mạng generator được xem như một mạng mã hóa giải mã đối xứng, chủ yếu được tạo thành từ các lớp tích chập tiêu chuẩn, các lớp tích chập phân rã độ sâu (depthwise separable convolutions), các khối lặp nghịch đảo (inverted residual blocks) và các mô-đun giảm và tăng mẫu. Cuối cùng của generator, lớp tích chập cuối cùng sử dụng các kernel tích chập 1x1 và không sử dụng lớp chuẩn hoá và được theo sau bởi hàm kích hoạt phi tuyến tanh.

Trong đó, mô hình được xây dựng từ các khối chính sau:

* + ConvBlock: Khối convolution được sử dụng để xây dựng các lớp tích chập trong mô hình. Mỗi khối ConvBlock bao gồm các lớp tích chập 2D, theo sau là một lớp batch normalization và một hàm kích hoạt LeakyReLU.
* DownConv và UpConv: Đây là các khối tích chập được sử dụng để giảm và tăng kích thước của đầu vào (downsampling và upsampling). DownConv bao gồm một lớp tích chập 2D và một lớp MaxPooling2D, trong khi UpConv bao gồm một lớp tích chập 2D và một lớp UpSampling2D.
* SeparableConv2D: Đây là một loại lớp tích chập khác, gọi là Separable Convolution, được sử dụng để giảm số lượng tham số của mô hình. Thay vì sử dụng một lớp tích chập 2D thông thường, SeparableConv2D sử dụng hai lớp tích chập 2D lần lượt để tích chập với hàng và cột của đầu vào.
* InvertedResBlock: Đây là một khối residual block đảo ngược được sử dụng để học các đặc trưng phức tạp của ảnh. Mỗi khối bao gồm các lớp tích chập và batch normalization theo một cấu trúc đặc biệt, kết hợp với kỹ thuật skip connection để giảm sự mất mát thông tin trong quá trình huấn luyện.
  + Cấu trúc chi tiết của các khối đã sử dụng:

Diagram

Description automatically generated

Hình 3. 3 Cách khối chính được sử dụng

Tóm lại, Generator trong mô hình được xây dựng từ các khối ConvBlock, DownConv, SeparableConv2D, UpConv, InvertedResBlock và một lớp Conv2D cuối cùng với hàm kích hoạt Tanh để đưa ra đầu ra. Quá trình lan truyền thuận của mô hình bao gồm xử lý đầu vào thông qua các khối encode\_blocks và res\_blocks, sau đó truyền qua các khối decode\_blocks để tạo ra đầu ra. Mỗi khối trong mô hình được khởi tạo bằng các trọng số ngẫu nhiên và được điều chỉnh trong quá trình huấn luyện để tối ưu hóa mục tiêu của mô hình.

#### 3.1.3.2 Cách hoạt động chi tiết của mạng phân biệt(Discriminator)

Mạng discriminator có kiến trúc tương tự như kiến trúc mạng discriminator được sử dụng trong các kiến trúc GAN khác. Ở đây ta sử dụng VGG19.

Chart

Description automatically generated

Hình 3. 4 Cấu trúc mạng VGG19

Kiến trúc của mạng VGG19 bao gồm 19 layer, với các layer được chia thành 2 phần chính: convolutional layers và fully connected layers.

* Phần convolutional layers của mạng VGG19 có thể được phân thành 5 khối, trong đó mỗi khối bao gồm một số lượng các convolutional layer và max pooling layer. Các convolutional layer trong các khối đầu tiên được thiết kế để phát hiện các đặc trưng cơ bản của hình ảnh như cạnh, góc và dạng. Các convolutional layer ở các khối sau được thiết kế để phát hiện các đặc trưng phức tạp hơn.
* Phần fully connected layers của mạng VGG19 bao gồm 3 layer fully connected (hay còn gọi là dense layer) và 2 layer softmax. Các fully connected layer này được sử dụng để phân loại hình ảnh dựa trên các đặc trưng đã được trích xuất ở phần convolutional layers.

Cách hoạt động của mạng VGG19 là chuyển đổi hình ảnh đầu vào thành một chuỗi các đặc trưng ở phần convolutional layers thông qua các lớp convolutional và max pooling. Sau đó, các đặc trưng này được truyền qua các fully connected layer để phân loại hình ảnh. Đối với các bài toán khác nhau, ta có thể sử dụng các phần khác nhau của mạng VGG19 để trích xuất các đặc trưng của hình ảnh. Ví dụ, các layer convolutional của mạng VGG19 có thể được sử dụng để trích xuất các đặc trưng của hình ảnh để sử dụng trong các bài toán khác như phân đoạn ảnh (image segmentation), nhận dạng đối tượng (object recognition), hay tạo ra các hình ảnh mới bằng các thuật toán sinh ảnh (image generation).

#### 3.1.3.3 Đánh giá kiến trúc của mô hình

Để tăng hiệu quả của AnimeGAN, các mô-đun giảm và tăng mẫu được đề xuất để giảm và tăng kích thước của các bản đồ đặc trưng. Ngoài ra, các khối lặp nghịch đảo (inverted residual blocks) được sử dụng để giảm số lượng tham số của mạng generator. Chúng bao gồm các lớp tích chập pointwise, tích chập depthwise và lớp chuẩn hoá.

Tổng quan lại, kiến trúc mạng ở đây đơn giản nhưng hiệu quả để tạo ra hình ảnh hoạt hình từ các bức ảnh thực tế. Nó kết hợp nhiều kỹ thuật hiện đại của machine learning như tích chập, phân rã độ sâu, khối lặp nghịch đảo và mô-đun giảm và tăng mẫu để đạt được chất lượng hình ảnh tốt và tiết kiệm tài nguyên tính toán.

### 3.1.4 Hàm loss của mô hình

Ở đây ta sử dụng hàm mất mát least squares loss function trong LSGAN để cải thiện chất lượng hình ảnh được tạo ra bởi mô mình và ổn định quá trình huấn luyện toàn bộ mạng.

LSGAN (Least Squares GAN) là một loại mô hình GAN được giới thiệu bởi Xudong Mao, et al. vào năm 2016. LSGAN được phát triển nhằm giải quyết vấn đề gradient vanishing khi sử dụng BCE loss function trong mô hình GAN.

Trong mô hình LSGAN, bộ phân biệt (discriminator) được huấn luyện để phân biệt giữa ảnh thật và ảnh giả. Điểm khác biệt của LSGAN so với các mô hình GAN khác đó là LSGAN sử dụng mean squared error (MSE) để tính toán sai số giữa đầu ra của bộ phân biệt và nhãn thực tế.

Việc sử dụng MSE thay vì BCE trong LSGAN giúp tránh được vấn đề gradient vanishing và tăng tính ổn định của quá trình huấn luyện. Kết quả thực nghiệm cho thấy rằng LSGAN thường cho ra kết quả tốt hơn so với các loại GAN khác, đặc biệt là đối với các bài toán sinh ảnh có độ phức tạp cao.

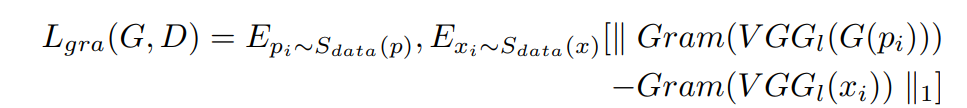
* Hàm mất mát L(G, D) được sử dụng trong AnimeGAN bao gồm bốn thành phần chính được trọng số hóa bởi các hệ số ωadv, ωcon, ωgra và ωcol.



* Hàm mất mát Lcon(G, D) được sử dụng để giữ lại nội dung của ảnh gốc.



* Hàm mất mát Lgra(G, D) giúp tạo ra các đường viền, nét vẽ của hình ảnh anime.



* Hàm mất mát Lcol(G, D) đảm bảo rằng màu sắc của hình ảnh tạo ra giống với màu sắc của ảnh gốc.

Text

Description automatically generated with medium confidence

Text, letter

Description automatically generated

### 3.1.5 Huấn luyện mô hình

Text

Description automatically generated

Ta truyền các tham số để thực hiển huấn luyện mô hình theo phong cách hoạt hình chỉ định:

--dataset: phong cách hoạt hình cẩn chuyển đổi

--epochs: số lần huấn luyện mô hình

--batch-size: số ảnh huấn luyện mỗi batch

--gan-loss: ở đây ta sử dụng LSGAN

--checkpoint-dir: đường dẫn tới thư mục lưu lại trọng số của mô hình

--lr-g: hệ số tốc độ học cho mạng sinh

--lr-d: hệ số tốc độ học cho mạng phân biệt

--wadvg: loss adv mạng sinh

--wadvd: loss adv mạng phân biệt

--wgra: loss Gram

--wcon: loss Content

--wcol: loss color

Ta thực hiện huấn luyện với google Colab:

Text

Description automatically generated

Hình 3. 4 Quá trình huấn luyện mô hình với colab

### 3.1.6 Thử nghiệm

a. Thử nghiệm với Larva style:

A picture containing painting, table, art, landscape

Description automatically generated

Hình 3. 5 Thử nghiệm với Larava style

Thử nghiệm với Paprika style:

A picture containing outdoor, building, sky, cloud

Description automatically generated

Hình 3. 6 Thử nghiệm với Paprika style

### 3.1.7 Đánh giá mô hình

Thời gian thực thi trung bình trên tập 100 ảnh:

* ảnh 1080x720: 1.14s
* ảnh 600x400: 0.32s

Chỉ số FID trung bình trên 100 ảnh:

* ảnh 1080x720: 136.12
* ảnh 600x400: 133.59

## 3.2 Ứng dụng di động tạo ảnh hoạt hình từ ảnh thật

### 3.2.1 Thiết kế các UseCase

#### 3.2.1.1 Use case chuyển đổi ảnh sang phong cách hoạt hình

a. Mô tả use case

**1. Tên use case:** Chuyển đổi ảnh

**2. Mô tả vắn tắt**: Use case cho phép người dùng chọn ảnh cần chuyển đổi, và sinh ảnh hoạt hình theo phong cách được chỉ định.

**3. Luồng sự kiện**

\* Luồng cơ bản:

1. Use case bắt đầu khi người quản trị kích chọn một phong cách hoạt hình bất kì tại giao diện trang chủ. Hệ thống sẽ lấy thông tin về trọng số của mô hình đảm nhiệm việc chuyển đổi phong cách hoạt hình đó. Đồng thời chuyển qua giao diện chọn ảnh cần biến đổi.

2. Người dùng chọn ảnh từ thiết bị cần chuyển đổi. Hệ thống sẽ biến đổi ảnh đó theo phong cách hoạt hình đã được chỉ định và hiển thị hai hình ảnh trước và sau khi biến đổi lên màn hình.

3. Người dùng nhấn nút “Lưu vào thiết bị”. Hệ thống sẽ lưu lại ảnh kết quả vào thư mục chỉ định trong thiết bị. Use case kết thúc.

\* Luồng rẽ nhánh:

Tại bất kì thời điểm nào, nếu hệ thống không kết nối được với cơ sở dữ liệu thì sẽ hiển thị thông báo lỗi. Use case kết thúc.

1. **Yêu cầu đặc biệt**

Không có.

**5. Tiền điều kiện**

Không có.

**6. Hậu điều kiện**

Không có.

**7. Mở rộng**

Không có.

b. Biểu đồ VOPC các lớp phân tích Use case Chuyển đổi ảnh



Hình 3. 7 VOPC các lớp use case chuyển đổi ảnh

c. Biểu đồ trình tự Use case Chuyển đổi ảnh



Hình 3. 8 Biểu đồ trình tự Use case Chuyển đổi ảnh

#### 3.2.1.2 Use case chuyển đổi video sang phong cách hoạt hình

a. Mô tả use case

**1. Tên use case:** Chuyển đổi video

**2. Mô tả vắn tắt**: Use case cho phép người dùng chọn video cần chuyển đổi, thời gian bắt dầu và kết thúc trong video.Hệ thống sinh video hoạt hình theo phong cách được chỉ định.

**3. Luồng sự kiện**

1)Luồng cơ bản:

1. Use case bắt đầu khi người quản trị kích chọn một phong cách hoạt hình bất kì tại giao diện trang chủ. Hệ thống sẽ lấy thông tin về trọng số của mô hình đảm nhiệm việc chuyển đổi phong cách hoạt hình đó. Đồng thời chuyển qua giao diện chọn video cần biến đổi.

2. Người dùng chọn video từ thiết bị cần chuyển đổi. Hệ thống sẽ biến đổi ảnh đó theo phong cách hoạt hình đã được chỉ định và hiển thị hai video trước và sau khi biến đổi lên màn hình.

3. Người dùng nhấn nút “Lưu vào thiết bị”. Hệ thống sẽ lưu lại ảnh kết quả vào thư mục chỉ định trong thiết bị. Use case kết thúc.

2)Luồng rẽ nhánh:

Tại bất kì thời điểm nào, nếu hệ thống không kết nối được với cơ sở dữ liệu thì sẽ hiển thị thông báo lỗi. Use case kết thúc.

**4. Yêu cầu đặc biệt**

Không có.

**5. Tiền điều kiện**

Không có.

**6. Hậu điều kiện**

Không có

**7. Mở rộng**

Không có.

A picture containing text, diagram, font, line

Description automatically generated

Hình 3. 7 VOPC các lớp use case chuyển đổi ảnh

c. Biểu đồ trình tự Use case Chuyển đổi video



Hình 3.8 Biểu đồ trình tự Use case Chuyển đổi video

### 3.2.2 Thiết kế giao diện

a. Giao diện trang chủ

b. Giao diện chuyển đổi ảnh

c. Giao diện chuyển đổi video

d. Giao diện xem kết quả

KẾT LUẬN

Sau khoảng thời gian nghiêm túc nghiên cứu, cùng với sự hướng dẫn tận tình của Tiến sĩ Trần Hùng Cường, em đã hoàn thành đề tài “Xây dựng ứng dụng di động chuyển đổi ảnh thật thành ảnh hoạt hình”.

Trong quá trình tìm hiểu và nghiên cứu đã giúp em nắm chắc các kiến thức cơ bản về xử lý ảnh, mạng noron tích chập, xây dựng giao diện phần mềm, tìm hiểu về các thư viện hỗ trợ...

Bên cạnh đó, em cũng còn tồn tại một số hạn chế chưa thực hiện được như:

* Khó khăn trong quá trình tiếp cận các công nghệ hỗ trợ.
* Mất nhiều thời gian trong quá trình thu thập và phân loại dữ liệu.
* Khó khăn trong việc quyết định các tham số khi xây dựng mô hình

Hướng phát triển của bài toán trong tương lai:

* Tối ưu hóa mô hình để đạt được kết quả đánh giá cao hơn.
* Tối ưu hóa tốc độ xử lý của chương trình sinh ảnh hoạt hình.
* Thêm một số chức năng mở rộng cho ứng dụng di động

Do thời gian hoàn thành đề tài có hạn và kinh nghiệp nghiệp vụ chưa sâu nên đề tài của em còn nhiều thiếu sót về chuyên môn. Em mong nhận được những ý kiến, góp ý, đánh giá của quý thầy cô để đề tài của em thêm hoàn thiện. Em xin chân thành cảm ơn quý thầy cô.

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | G. S. A. D. E. G. O. Syed Zain Masood, License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks, 2016. |
| [2] | R. M. B. T. N. Francisco Delmar Kurpiel, CONVOLUTIONAL NEURAL NETWORKS FOR LICENSE PLATE DETECTION IN IMAGES, Parana, Brazil, 2018. |
| [3] | T. H. QUAN, Vietnamese Plate Dataset. |
| [4] | Aston Zhang, Zachary C.Lipton, Mu Li, and Alexander J.Smola(2021). Dive into Deep Learning: Dive into Deep Learning Release 0.16.6. |
| [5] | C. C. C. e. a. 2. IOP, License Plate Recognition via Deep Convolutional Neural Network, 2019. |