BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ VĂN LANG

BÁO CÁO ĐỒ ÁN CUỐI KỲ

HỌC PHẦN: THỊ GIÁC MÁY TÍNH (COMPUTER VISION)

---------o0o---------

TẠO ẢNH PHONG CÁCH NGHỆ THUẬT BẰNG NEURAL STYLE TRANSFER

Sinh Viên Chủ Nhiệm: Trần Lê Hoàng Bảo – 2174801090010
Nguyễn Lê Minh Nhật – 2174801090015

Khoa: Kỹ Thuật Cơ – Điện Và Máy Tính

Giảng Viên Hướng Dẫn: TS. Nguyễn Quốc Dũng

Lớp: 71K27KHDL01

TP. HỒ CHÍ MINH, THÁNG 7 NĂM 2024

TRƯỜNG ĐẠI HỌC VĂN LANG
KHOA KỸ THUẬT CƠ – ĐIỆN VÀ MÁY TÍNH

BÁO CÁO ĐỒ ÁN CUỐI KỲ

TẠO ẢNH PHONG CÁCH NGHỆ THUẬT BẰNG NEURAL STYLE TRANSFER

Tên học phần:	Thị giác máy tính		
Mã học phần:	71DSCV40013	Số tín chỉ:	3
Mã nhóm lớp học phần:	233_71DSCV40013_01		

Sinh Viên Thực Hiện:	Trần Lê Hoàng Bảo – 2174801090010 Nguyễn Lê Minh Nhật – 2174801090015
Giảng viên hướng dẫn:	TS. Nguyễn Quốc Dũng
Ngành:	Khoa Học Dữ Liệu
Lớp:	71K27KHDL01

TP. HỒ CHÍ MINH, THÁNG 7 NĂM 2024

MỤC LỤC

LỜI CẢM ƠN

Trong quá trình thực hiện đồ án cuối kỳ với đề tài "Tạo Ảnh Phong Cách Nghệ Thuật Bằng Neural Style Transfer", nhóm em là Bảo và Nhật đã nhận được sự giúp đỡ, hướng dẫn và truyền đạt cho nhóm em nhiều kiến thức về mạng Nơ-ron và những kiến thức bổ ích về Thị Giác Máy Tính, Deep Learning, trong thời gian thực hiện Đồ án Tiểu luận cuối kỳ.

Trước hết, chúng em xin gửi lời cảm ơn sâu sắc đến TS. Nguyễn Quốc Dũng, giảng viên hướng dẫn Bộ môn Thị giác máy tính trong ngành Khoa Học Dữ Liệu – Trường Đại Học Văn Lang. Người đã chỉ dẫn, hỗ trợ và truyền đạt những kiến thức trong suốt quá trình học và thực hiện đồ án.

Chúng tôi cũng xin cảm ơn toàn thể các thầy cô trong Khoa Kỹ Thuật Cơ – Điện Và Máy Tính, Trường Đại Học Công Nghệ Văn Lang, đã cung cấp cho chúng tôi nền tảng kiến thức vững chắc và môi trường học tập lý tưởng.

TP. Hồ Chí Minh, tháng 7 năm 2024

SINH VIÊN THỰC HIỆN

Trần Lê Hoàng Bảo **Nguyễn Lê Minh Nhật**

MỞ ĐẦU

Trong thời đại hiện nay, công nghệ **thị giác máy tính** đang trở thành một trong những lĩnh vực nghiên cứu phát triển mạnh mẽ và có tác động sâu rộng đến nhiều lĩnh vực trong cuộc sống. Việc ứng dụng các kỹ thuật **học sâu (Deep Learning)** trong **thị giác máy tính** đã mở ra nhiều cơ hội mới và nâng cao hiệu quả của các ứng dụng như **nhận diện khuôn mặt**, **tự động lái xe**, và **phân tích hình ảnh y tế**.

Một trong những ứng dụng thú vị và sáng tạo nhất của thị giác máy tính là **tạo ảnh phong cách nghệ thuật bằng phương pháp Neural Style Transfer**. Đề tài này nhằm khám phá và ứng dụng **Neural Style Transfer** để tạo ra những bức tranh mang phong cách nghệ thuật đặc trưng của các bức tranh nổi tiếng. Bằng cách sử dụng một bức ảnh gốc và một bức tranh phong cách nghệ thuật, chúng ta có thể tạo ra một hình ảnh mới kết hợp nội dung của ảnh gốc với phong cách nghệ thuật của bức tranh.

Mục tiêu của đề tài này bao gồm:

- **Sử dụng Neural Style Transfer để tạo ra ảnh mới với phong cách nghệ thuật của một bức tranh khác**: Thực hiện các thí nghiệm sử dụng Neural Style Transfer trên nhiều bức tranh phong cách khác nhau để tạo ra các tác phẩm nghệ thuật mới.
- **Thử nghiệm với các bức tranh phong cách khác nhau và đánh giá kết quả**: Đánh giá chất lượng của các bức tranh được tạo ra bằng cách thay đổi các tham số trong mô hình và so sánh với các phương pháp khác.
- **Phân tích các yếu tố ảnh hưởng đến chất lượng ảnh tạo ra**: Nghiên cứu các yếu tố như lựa chọn tầng mạng, cách tính toán Gram matrix, và các tham số điều chỉnh khác ảnh hưởng đến chất lượng ảnh phong cách nghệ thuật được tạo ra.

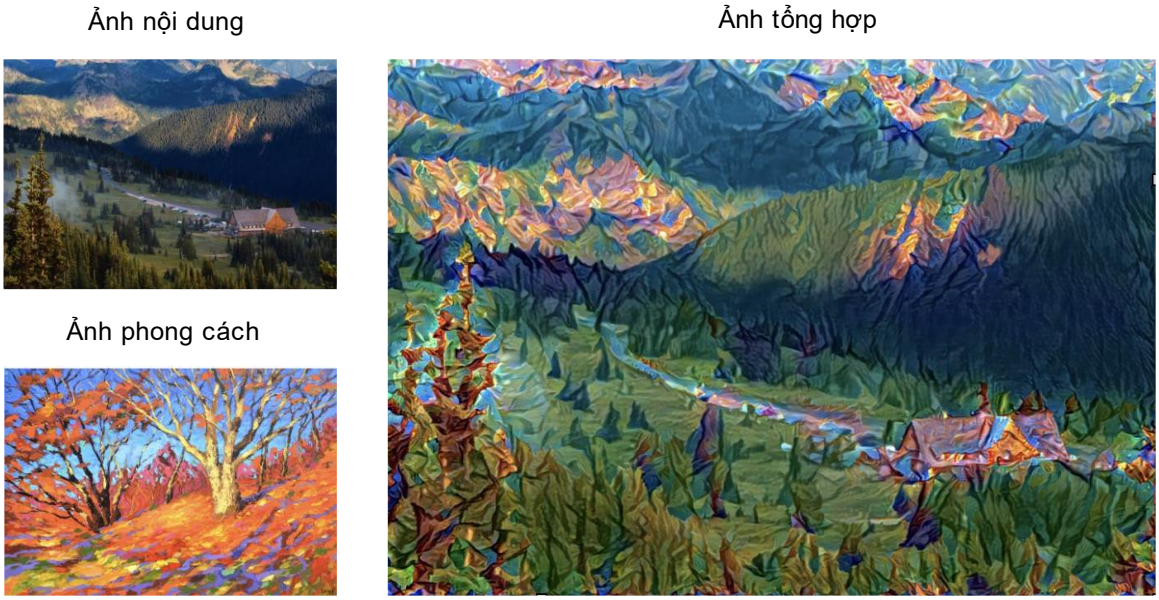
Bộ dữ liệu sử dụng trong đề tài này là **WikiArt**, bao gồm các bức tranh phong cách nghệ thuật từ **Wikipedia**. Bộ dữ liệu này cung cấp một nguồn tài nguyên phong phú và đa dạng để thử nghiệm và đánh giá phương pháp **Neural Style Transfer**.

Thông qua đề tài này, chúng tôi hy vọng sẽ góp phần làm sáng tỏ hơn về khả năng của **Neural Style Transfer** trong việc tạo ra các tác phẩm nghệ thuật độc đáo và mở ra các ứng dụng mới trong lĩnh vực thị giác máy tính và nghệ thuật số.

PHẦN I: GIỚI THIỆU VỀ ĐỀ TÀI

1.1. MỤC TIÊU VÀ Ý NGHĨA CỦA ĐỀ TÀI

Trong phần này, chúng ta sẽ thảo luận cách sử dụng mạng nơ-ron tích chập (CNN) để tự động áp dụng phong cách của ảnh này cho ảnh khác. Thao tác này được gọi là truyền tải phong cách ***(style transfer)*** [Gatys et al., 2016]. Ở đây ta sẽ cần hai ảnh đầu vào, một ảnh nội dung và một ảnh phong cách. Ta sẽ dùng mạng nơ-ron để biến đổi ảnh nội dung sao cho phong cách của nó giống như ảnh phong cách đã cho. Trong Hình 1, ảnh nội dung là một bức ảnh phong cảnh được tác giả chụp ở công viên quốc gia Mount Rainier, gần Seattle. Ảnh phong cách là một bức tranh sơn dầu vẽ cây gỗ sồi vào mùa thu. Ảnh kết hợp đầu ra giữ lại được hình dạng tổng thể của các vật trong ảnh nội dung, nhưng được áp dụng phong cách tranh sơn dầu của ảnh phong cách, nhờ đó khiến màu sắc tổng thể trở nên sống động hơn.



Hình 1: ảnh chúng ta tạo ra ảnh kết quả sẽ là bức ảnh nội dung và ảnh phong cách đầu vào cùng với ảnh kết hợp được tạo ra từ việc truyền tải phong cách - Nguồn: d2l.aivivn.com

Mục tiêu của đề tài là **ứng dụng Neural Style Transfer (NST)** để tạo ra các bức ảnh mang phong cách nghệ thuật từ các bức tranh nổi tiếng. Việc này không chỉ tạo ra các tác phẩm nghệ thuật mới mẻ mà còn có thể áp dụng trong nhiều lĩnh vực như thiết kế, quảng cáo, và thậm chí là trong công nghiệp game.

Ý nghĩa của đề tài:

- **Nghệ thuật và Thiết kế**: NST cho phép tạo ra các tác phẩm nghệ thuật số mới, mang đậm dấu ấn cá nhân và phong cách của các bức tranh nổi tiếng. Điều này mở ra cơ hội cho các nhà thiết kế và nghệ sĩ trong việc sáng tạo và biểu đạt cá nhân.
- **Quảng cáo và Truyền thông**: Trong lĩnh vực quảng cáo, các bức ảnh được chuyển đổi phong cách có thể tạo ra những hình ảnh độc đáo, thu hút sự chú ý và mang lại hiệu quả cao trong chiến dịch tiếp thị.
- **Giáo dục và Nghiên cứu**: NST cung cấp một công cụ hữu ích cho việc nghiên cứu và giảng dạy về thị giác máy tính, học sâu và nghệ thuật. Nó giúp sinh viên và nhà nghiên cứu hiểu rõ hơn về cách mạng nơ-ron hoạt động và cách chúng có thể được áp dụng vào thực tế.
- **Công nghiệp game**: Trong phát triển game, NST có thể được sử dụng để tạo ra các hiệu ứng hình ảnh độc đáo, giúp tăng cường trải nghiệm người chơi và tạo ra các thế giới game với phong cách hình ảnh đa dạng.
- **Tâm lý học và Trị liệu**: NST cũng có thể được áp dụng trong trị liệu tâm lý và nghệ thuật, giúp bệnh nhân biểu đạt cảm xúc và giảm stress thông qua việc sáng tạo nghệ thuật số.

Việc nghiên cứu và ứng dụng **Neural Style Transfer** không chỉ mang lại những giá trị thực tiễn mà còn góp phần thúc đẩy sự phát triển của lĩnh vực học sâu và thị giác máy tính, mở ra những hướng đi mới cho các ứng dụng công nghệ trong cuộc sống hàng ngày.

1.2. LĨNH VỰC ỨNG DỤNG CỦA ĐỀ TÀI

Neural Style Transfer (NST) có thể được ứng dụng trong nhiều lĩnh vực, bao gồm:

- **Nghệ thuật và Thiết kế Đồ họa**: Một nghệ sĩ số có thể sử dụng NST để tạo ra các bức tranh kỹ thuật số mới mang phong cách của các họa sĩ nổi tiếng như Van Gogh hay Picasso, từ đó tạo ra các sản phẩm nghệ thuật độc đáo mà không cần phải vẽ tay từ đầu.
- **Sản xuất Phim và Trò chơi Điện tử**: Trong bộ phim "Loving Vincent," các nhà sản xuất đã sử dụng công nghệ NST để biến các cảnh quay thành các bức tranh theo phong cách của Vincent Van Gogh, tạo nên một bộ phim hoàn toàn bằng tranh vẽ. Tương tự, trong trò chơi điện tử, NST có thể được sử dụng để

tạo ra các phong cảnh và nhân vật có phong cách nghệ thuật đặc trưng, nâng cao trải nghiệm người chơi.

- **Công nghệ Thực tế Ảo (VR) và Thực tế Tăng cường (AR)**: Trong một ứng dụng VR, người dùng có thể trải nghiệm đi qua một bảo tàng ảo nơi tất cả các tác phẩm nghệ thuật được tạo ra bằng cách sử dụng NST để chuyển đổi phong cách của các tác phẩm gốc. Trong AR, người dùng có thể áp dụng các phong cách nghệ thuật khác nhau lên môi trường xung quanh của họ trong thời gian thực.
- **Marketing và Quảng cáo**: Một công ty có thể sử dụng NST để tạo ra các quảng cáo trực quan bắt mắt bằng cách áp dụng các phong cách nghệ thuật khác nhau lên hình ảnh sản phẩm của họ. Chẳng hạn, một thương hiệu thời trang có thể tạo ra các bức ảnh chiến dịch quảng cáo với phong cách nghệ thuật của các bức tranh nổi tiếng để thu hút sự chú ý và tạo ấn tượng mạnh mẽ với khách hàng.

Neural Style Transfer thể hiện tiềm năng to lớn trong việc kết hợp nghệ thuật và công nghệ, mang lại sự đổi mới và sáng tạo trong nhiều lĩnh vực khác nhau.

1.3. VẤN ĐỀ NGHIÊN CỨU VÀ PHƯƠNG PHÁP GIẢI QUYẾT

Vấn đề nghiên cứu trong đề tài này là làm thế nào để chuyển đổi phong cách của một bức ảnh sang phong cách của một tác phẩm nghệ thuật mà vẫn giữ nguyên nội dung gốc của bức ảnh. Phương pháp giải quyết là sử dụng mạng nơron tích chập (Convolutional Neural Networks - CNN) và kỹ thuật Neural Style Transfer để phân tích và chuyển đổi phong cách giữa các ảnh

Phương pháp giải quyết bao gồm việc sử dụng các mạng nơ-ron tích chập **(Convolutional Neural Networks - CNNs)** và các kỹ thuật học sâu để thực hiện Neural Style Transfer. Các bước cơ bản của phương pháp này bao gồm:

- **Chọn mô hình CNN**: Sử dụng mô hình **VGG19** đã được tiền huấn luyện trên tập dữ liệu **WikiArt**, do khả năng nắm bắt tốt các đặc trưng không gian của hình ảnh.

- **Tính toán Gram Matrix**: Sử dụng Gram Matrix để biểu diễn các đặc trưng phong cách của ảnh nghệ thuật. Gram Matrix giúp nắm bắt sự tương quan giữa các đặc trưng, từ đó áp dụng phong cách lên ảnh gốc.
- **Định nghĩa hàm mất mát (Loss Function)**: Xây dựng hàm mất mát kết hợp giữa mất mát nội dung **(content loss)** và mất mát phong cách **(style loss)**. Mất mát nội dung giúp bảo toàn chi tiết của ảnh gốc, trong khi mất mát phong cách giúp áp dụng đặc trưng phong cách từ ảnh nghệ thuật.
- **Tối ưu hóa**: Sử dụng các thuật toán tối ưu hóa như L-BFGS hoặc Adam để điều chỉnh ảnh đầu ra sao cho tối thiểu hóa hàm mất mát.

Bằng cách áp dụng các bước trên, NST có thể tạo ra các bức ảnh với chất lượng cao, kết hợp hài hòa giữa nội dung gốc và phong cách nghệ thuật, mở ra nhiều ứng dụng sáng tạo trong nghệ thuật và công nghệ.

1.4. KHÁI NIỆM VỀ MẠNG NƠRON TÍCH CHẬP

Trong mạng nơron đa lớp truyền thống (Multi-layer Perceptron – MLP), mỗi nơron trong một lớp kết nối với tất cả các nơron trong lớp tiếp theo. Điều này dẫn đến sự gia tăng đáng kể khối lượng tính toán khi độ sâu của mô hình tăng lên.

Mạng nơron tích chập (Convolutional Neural Network - CNN) là một loại mạng nơron chuyên biệt trong việc xử lý dữ liệu có cấu trúc dạng lưới, chẳng hạn như hình ảnh. CNN được thiết kế để tự động và thích ứng học các đặc trưng không gian từ dữ liệu đầu vào thông qua các lớp tích chập.

CNN là một tập hợp các lớp tích chập (convolutional layers) chồng lên nhau, sử dụng các hàm kích hoạt phi tuyến (nonlinear activation functions) như ReLU và tanh để kích hoạt các trọng số trong các nơron. Sau mỗi lớp kích hoạt, các thông tin trừu tượng hơn được tạo ra cho các lớp tiếp theo.

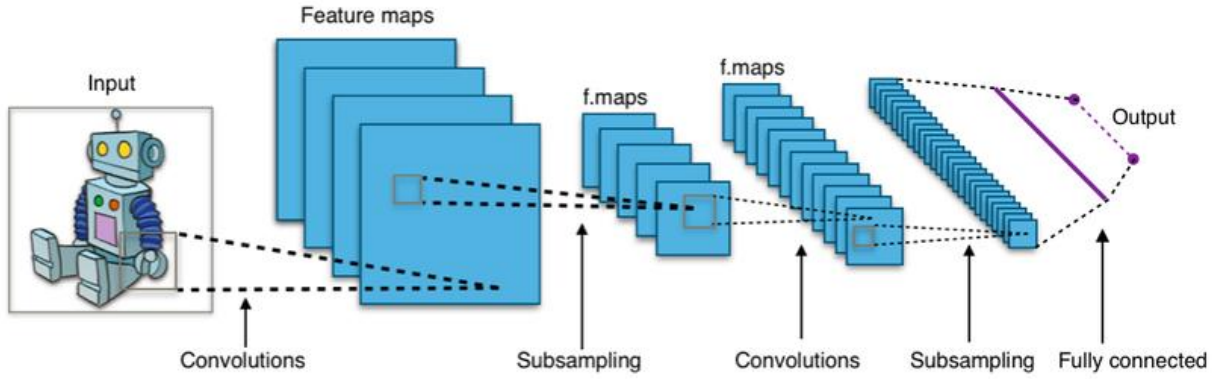
Trong mô hình mạng truyền ngược (feedforward neural network), mỗi nơron đầu vào kết nối đến mỗi nơron đầu ra trong các lớp tiếp theo. Mô hình này gọi là mạng kết nối đầy đủ (fully connected layer) hay mạng toàn vẹn (affine layer). Tuy nhiên, trong CNN, các lớp liên kết với nhau thông qua cơ chế convolution. Lớp tiếp theo là kết quả của quá trình convolution từ lớp trước đó, tạo ra các kết nối cục bộ. Nhờ

vậy, mỗi nơron ở lớp kế tiếp được sinh ra từ kết quả của các bộ lọc (filters) áp đặt lên các vùng ảnh cục bộ của nơron trước đó.

Mỗi lớp sử dụng các bộ lọc khác nhau, thông thường có hàng trăm đến hàng nghìn bộ lọc như vậy, và kết hợp kết quả của chúng lại. Ngoài ra, còn có các lớp khác như lớp gộp (pooling/subsampling layer) dùng để chắt lọc lại các thông tin hữu ích hơn, loại bỏ các thông tin nhiễu.

Trong quá trình huấn luyện mạng (training), CNN tự động học các giá trị qua các lớp bộ lọc dựa vào cách thức mà bạn thực hiện. Ví dụ, trong tác vụ phân loại ảnh, CNNs sẽ cố gắng tìm ra các thông số tối ưu cho các bộ lọc tương ứng theo thứ tự từ pixel thô (raw pixel) đến cạnh (edges), hình dạng (shapes), khuôn mặt (facial features), và các đặc trưng mức cao (high-level features). Lớp cuối cùng thường được dùng để phân loại ảnh.

CNN giải quyết các vấn đề trên dựa trên ba ý tưởng cơ bản: vùng tiếp nhận cục bộ (local receptive fields), trọng số chia sẻ (shared weights), và phương pháp lấy mẫu xuống (pooling). Nhìn chung, cấu trúc của CNN bao gồm các thành phần chính sau:



Hình 2: Kiến trúc mạng CNN cơ bản – Nguồn: topdev.vn

1.4.1. Lớp Tích Chập (Convolutional Layer)

Lớp tích chập là thành phần cốt lõi của CNN, được sử dụng để trích xuất các đặc trưng của hình ảnh thông qua việc áp dụng các bộ lọc (kernel) trên toàn bộ ảnh. Kết quả đầu ra của lớp này là các feature map, thể hiện các đặc trưng cụ thể của hình ảnh.

Cơ chế hoạt động

- **Bộ lọc (kernel):** Mỗi bộ lọc là một ma trận nhỏ có kích thước cố định (ví dụ: 3x3, 5x5). Bộ lọc quét qua dữ liệu đầu vào và tính toán tích chập giữa các giá trị của bộ lọc và các giá trị tương ứng trong dữ liệu đầu vào.
- **Bước nhảy (stride):** Bước nhảy xác định khoảng cách di chuyển của bộ lọc trên dữ liệu đầu vào. Ví dụ, stride=1 nghĩa là bộ lọc di chuyển từng pixel một, còn stride=2 nghĩa là bộ lọc di chuyển mỗi lần hai pixel.
- **Bổ sung (padding):** Bổ sung thêm các giá trị 0 xung quanh dữ liệu đầu vào để kiểm soát kích thước của bản đồ đặc trưng. Padding giúp duy trì kích thước của ảnh sau khi áp dụng bộ lọc, tránh mất mát thông tin biên.
- **Nhóm bộ lọc (filter group):** Nhiều bộ lọc có thể được sử dụng cùng lúc để trích xuất các đặc trưng khác nhau. Mỗi bộ lọc sẽ phát hiện ra các đặc trưng khác nhau của hình ảnh, tạo ra một loạt các feature map.

Quá trình tính toán

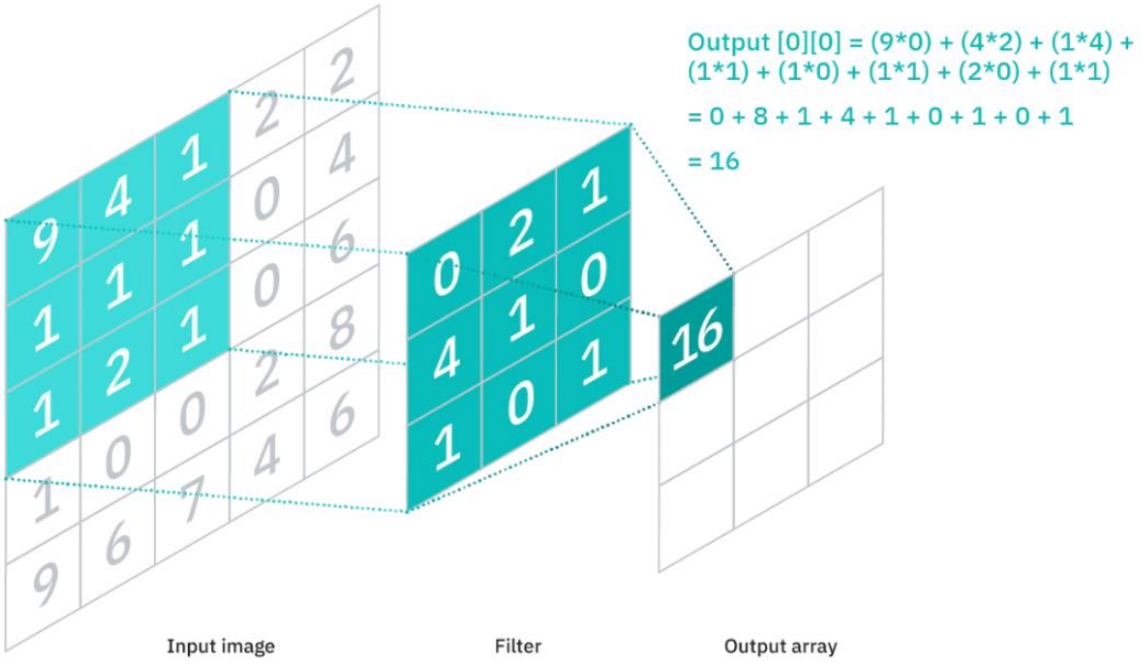
Quá trình convolutional kernel không chỉ được sử dụng trong CNNs mà còn là một yếu tố quan trọng của nhiều thuật toán thị giác máy tính khác. Đây là quá trình mà chúng ta lấy một ma trận nhỏ các số (gọi là kernel hoặc bộ lọc), đi qua ảnh và biến đổi nó dựa trên các giá trị từ bộ lọc.

Công thức tính toán feature map Y từ ảnh đầu vào X và kernel K như sau:

$$Y_{ij} = \sum_{m=0}^{c_1} \sum_{n=1}^{c_1} X_{(i+m-1)(j+n-1)} \cdot K_{mn}$$

Trong phương trình này:

- Y_{IJ}: ma trận đầu ra kết quả từ phép tích chập
- c_1: đại diện cho chiều rộng và chiều cao của kernel tích chập.
- X_{ij}: ma trận đầu vào, trong đó i và j là chỉ số chỉ vị trong ma trận đầu vào.
- K_{ij}: trọng số trong kernel tích chập tại vị trí i và j.



Hình 3: Sơ đồ toán tử tích chập.

Ưu điểm của lớp tích chập so với mạng nơ-ron truyền thống MLP

- **Trích xuất thông tin theo phân vùng không gian:** CNN có khả năng tự động học và trích xuất các đặc trưng cục bộ từ ảnh, giúp phát hiện các đặc trưng quan trọng mà không cần phải thiết kế bằng tay.
- **Giảm thiểu số lượng tham số cần học:** Thông qua việc chia sẻ trọng số giữa các bộ lọc, CNN giảm bớt khối lượng tính toán và số lượng tham số so với mạng nơ-ron truyền thống MLP, giúp mô hình dễ dàng huấn luyện và tổng quát hóa tốt hơn.
- **Khả năng xử lý dữ liệu không gian:** CNN được thiết kế đặc biệt để xử lý dữ liệu có cấu trúc không gian như ảnh, giúp mô hình nhận dạng và phân loại các đối tượng trong ảnh một cách hiệu quả.

1.4.2. Lớp Phi Tuyến (Non-Linear Activation Layer)

Lớp phi tuyến là một thành phần quan trọng trong kiến trúc của mạng nơron tích chập (CNN), đặc biệt khi áp dụng trong Tạo ảnh phong cách nghệ thuật bằng Neural Style Transfer. Việc sử dụng các hàm kích hoạt phi tuyến giúp mạng nơron có khả năng học các đặc trưng phức tạp và phi tuyến tính từ dữ liệu đầu vào, điều

này là cần thiết để mô hình có thể nắm bắt và tái tạo các đặc điểm nghệ thuật một cách hiệu quả.

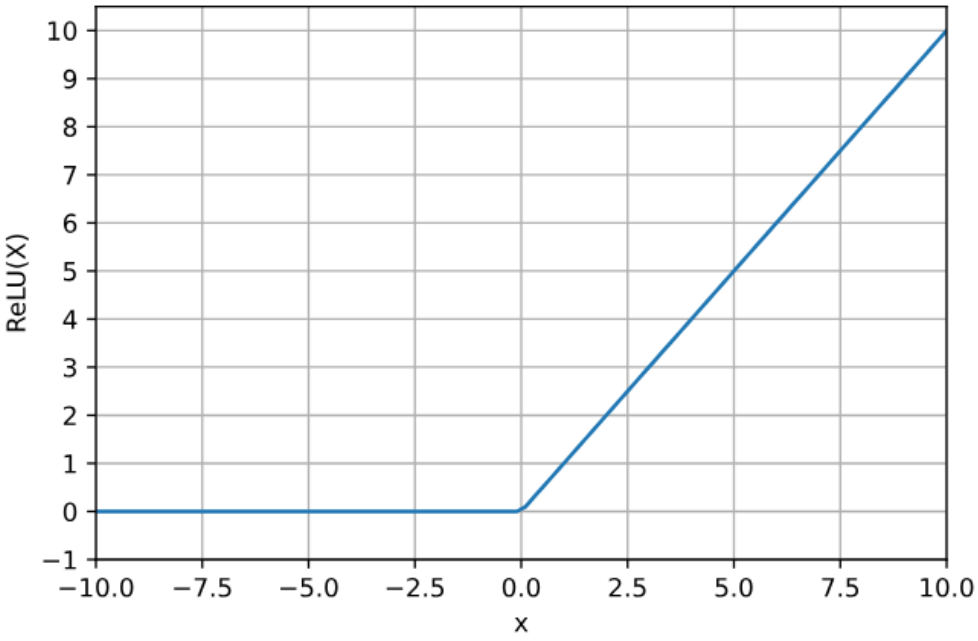
Vai trò của lớp phi tuyến trong Neural Style Transfer

Trong Neural Style Transfer, mục tiêu là chuyển đổi phong cách của một bức ảnh (ảnh phong cách) sang một bức ảnh khác (ảnh nội dung) trong khi vẫn giữ được các đặc trưng nội dung của bức ảnh gốc. Để đạt được điều này, cần phải sử dụng một mô hình học sâu có khả năng phân tích và kết hợp cả đặc trưng nội dung và phong cách. Lớp phi tuyến đóng vai trò quan trọng trong quá trình này thông qua các hàm kích hoạt như **ReLU và tanh**

Các hàm kích hoạt phổ biến

1. **ReLU (Rectified Linear Unit):**

Công thức: $f(x) = ReLu(x) = max(0, x)$



Hình 4: Hàm ReLu - Nguồn: vietanh.dev

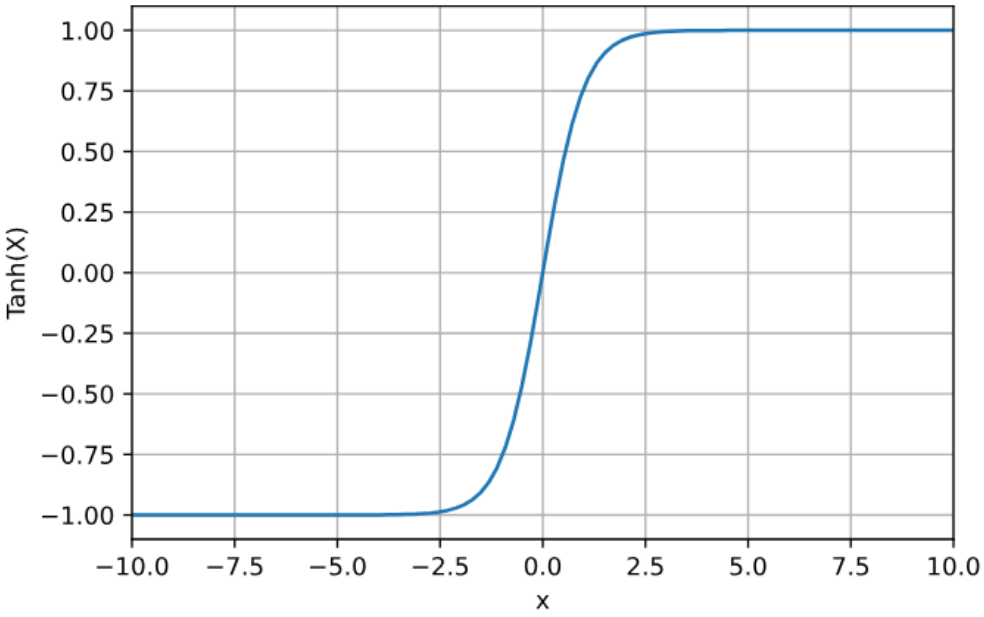
Tức là, giá trị đầu ra của ReLU là x nếu x lớn hơn 0, ngược lại là 0.

Đặc điểm: ReLU là một trong những hàm kích hoạt phổ biến nhất trong các mô hình học sâu hiện đại do tính đơn giản và hiệu quả tính toán. Nó giúp loại bỏ các giá trị âm, giữ lại các giá trị dương và giới thiệu tính phi tuyến vào mô hình, từ đó giúp mạng học được các đặc trưng phức tạp từ dữ liệu.

2. **Tanh (Hyperbolic Tangent):**

Công thức: $f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Đặc điểm: Hàm tanh chuyển đổi các giá trị đầu vào thành các giá trị nằm trong khoảng [-1, 1], giúp trung hòa các giá trị và tạo ra tính phi tuyến mạnh mẽ. Tanh thường được sử dụng trong các mô hình nơi cần cân bằng các giá trị âm và dương.



Hình 5: Hàm tanh - Nguồn: vietanh.dev

Hình dạng của đồ thị hàm tanh là một đường cong S, giống như hàm sigmoid, nhưng nằm trong khoảng giá trị mở rộng từ -1 đến 1. Khi đầu vào tiến gần đến âm vô cùng, hàm tanh tiệm cận -1, và khi đầu vào tiến gần đến dương vô cùng, hàm tanh tiệm cận 1. Điều này cho phép hàm tanh thực hiện một biến đổi toàn bộ phạm vi giá trị, giúp lan truyền gradient trong quá trình huấn luyện mạng neural trở nên ổn định hơn.

Hàm tanh thường được sử dụng trong các lớp kết nối đầy đủ và lớp ẩn của mạng nơ-ron

1.4.3. Lớp Gộp (Pooling/Subsampling Layer)

Lớp gộp là một thành phần quan trọng trong mạng nơron tích chập (CNN), có nhiệm vụ giảm kích thước không gian của dữ liệu, giảm số lượng tham số và tính

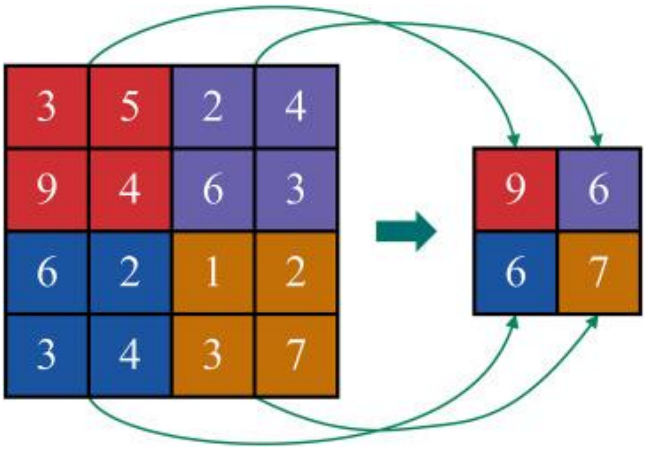
toán trong mô hình, đồng thời giữ lại các thông tin quan trọng. Các kỹ thuật gộp phổ biến bao gồm max pooling và average pooling, giúp mạng nơron học được các đặc trưng cục bộ của hình ảnh một cách hiệu quả và tăng khả năng khái quát hóa của mô hình.

Vai trò của lớp gộp trong Neural Style Transfer

Trong Neural Style Transfer, việc giảm kích thước không gian của dữ liệu giúp mô hình tập trung vào các đặc trưng quan trọng của ảnh, giảm thiểu số lượng tham số cần học và tăng tốc độ huấn luyện. Lớp gộp giúp mạng nơron xử lý các đặc trưng một cách khái quát, từ đó tạo ra các hình ảnh phong cách nghệ thuật với các đặc trưng được học một cách chính xác.

Các kỹ thuật gộp phổ biến

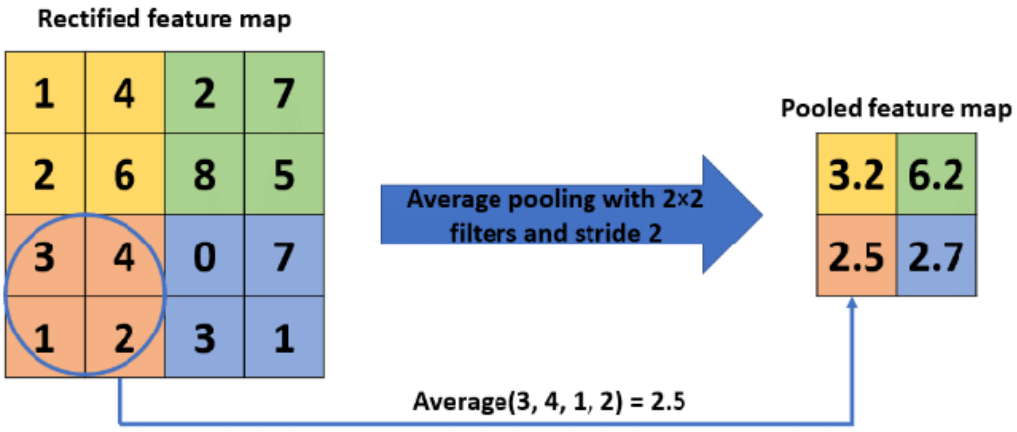
1. **Max Pooling:**
    - **Công thức:** Chọn giá trị lớn nhất trong mỗi vùng con của ảnh.
    - **Ví dụ:** Nếu sử dụng max pooling với kích thước cửa sổ 2x2 và stride 2, mỗi vùng 2x2 trong ảnh sẽ được giảm xuống còn một pixel, là giá trị lớn nhất trong vùng đó.
    - **Ưu điểm:** Giúp mạng nơron giữ lại các đặc trưng mạnh mẽ và loại bỏ các thông tin không quan trọng.



Hình 6: Nguyên tắc cơ bản của maxpooling

2. **Average Pooling:**
    - **Công thức:** Tính giá trị trung bình của tất cả các pixel trong mỗi vùng con của ảnh.

- **Ví dụ:** Nếu sử dụng average pooling với kích thước cửa sổ 2x2 và stride 2, mỗi vùng 2x2 trong ảnh sẽ được giảm xuống còn một pixel, là giá trị trung bình của vùng đó.
- **Ưu điểm:** Giúp mạng nơron giảm kích thước dữ liệu một cách mềm mại, giảm thiểu sự thay đổi của dữ liệu.



Hình 7: Nguyên tắc cơ bản của average pooling

Lợi ích của lớp gộp

- **Giảm kích thước dữ liệu:** Lớp gộp giúp giảm số lượng pixel trong ảnh, từ đó giảm số lượng tham số cần học và tăng tốc độ huấn luyện mô hình.
- **Giảm thiểu overfitting:** Bằng cách giảm kích thước không gian của dữ liệu, lớp gộp giúp giảm hiện tượng quá mức hóa (overfitting), giúp mô hình trở nên khái quát hóa tốt hơn.
- **Tăng khả năng khái quát hóa:** Lớp gộp giúp mạng nơron học được các đặc trưng cơ bản và khái quát của hình ảnh, giúp mô hình nhận diện các đối tượng và đặc trưng một cách chính xác hơn.

1.4.4. Lớp Kết Nối Đầy Đủ (Fully Connected Layer)

Lớp Kết Nối Đầy Đủ (Fully Connected Layer - FC) là một trong những lớp quan trọng trong mạng nơron, được sử dụng trong các giai đoạn cuối của mô hình để tổng hợp thông tin và đưa ra kết quả dự đoán. Lớp này kết nối toàn bộ các nơron từ lớp trước đó, thường là từ lớp tích chập hoặc lớp gộp cuối cùng, và có nhiệm vụ thực hiện tác vụ phân loại hoặc hồi quy tùy vào bài toán cụ thể.

Vai trò của lớp kết nối đầy đủ trong Neural Style Transfer

Trong Neural Style Transfer, lớp kết nối đầy đủ đóng vai trò quan trọng trong việc kết hợp và xử lý thông tin từ các lớp trước đó để tạo ra các đặc trưng phong cách nghệ thuật cuối cùng. Lớp FC giúp mạng nơron học và tổng hợp các đặc trưng phức tạp từ ảnh nội dung và ảnh phong cách, từ đó tạo ra các bức ảnh nghệ thuật mới với các đặc trưng được học.

Cấu trúc và hoạt động của lớp kết nối đầy đủ

1. **Kết nối toàn bộ:** Trong lớp FC, mọi nơron đầu vào đều được kết nối với mọi nơron đầu ra. Điều này giúp mô hình có khả năng học và tổng hợp thông tin từ toàn bộ ảnh đầu vào.
2. **Làm phẳng dữ liệu:** Trước khi đưa vào lớp kết nối đầy đủ, dữ liệu thường được làm phẳng (flatten) từ dạng ma trận thành dạng vector để phù hợp với kiến trúc của lớp này.
3. **Forward Propagation:** Trong quá trình lan truyền tiến (Forward Propagation), giá trị của mỗi nơron đầu ra được tính bằng công thức:

$$\mathbf{y = Wx + b}$$

Trong đó:

- **W** là ma trận trọng số
- **x** là vector đầu vào
- **b** là vector bù (bias)
- **y** là đầu ra của lớp

Trực quan hóa các kích thước

Giả sử kích thước minibatch là $m = 100$, thứ nguyên của dữ liệu là $28 \times 28 = 784$, và số lượng lớp là $c = 10$. Mặc dù **X** và **Y** là các ma trận do việc phân nhóm, nhưng theo quy ước, chúng thường được đặt tên biến bằng chữ thường, như thể chúng chỉ dành cho một ví dụ.

BackPropagation trong lớp FC

Để tính đạo hàm lỗi đối với đầu ra trong quá trình lan truyền ngược (BackPropagation), chúng ta cần:

- Đạo hàm của sai số đối với các tham số ($\partial E/\partial W$, $\partial E/\partial B$)

$$\frac{\partial \mathbf{E}}{\partial \mathbf{W}} = \boldsymbol{X}^{\boldsymbol{t}} \frac{\partial \mathbf{E}}{\partial \mathbf{Y}} \quad ; \quad \frac{\partial \mathbf{E}}{\partial \mathbf{B}} = \frac{\partial \mathbf{E}}{\partial \mathbf{Y}} ;$$

- Đạo hàm của sai số đối với đầu vào ($\partial E/\partial X$)

$$\frac{\partial \mathbf{E}}{\partial \mathbf{X}} = \frac{\partial \mathbf{E}}{\partial \mathbf{Y}} \boldsymbol{W}^{\boldsymbol{t}}$$

1.4.5. Lớp Dropout

Lớp Dropout là một kỹ thuật regularization hiệu quả nhằm giảm hiện tượng overfitting trong mạng nơron bằng cách "tắt" ngẫu nhiên một số nơron trong quá trình huấn luyện. Kỹ thuật này được áp dụng trên các lớp khác nhau như lớp kết nối đầy đủ, lớp tích chập và lớp lặp lại dày đặc, ngoại trừ lớp đầu ra.

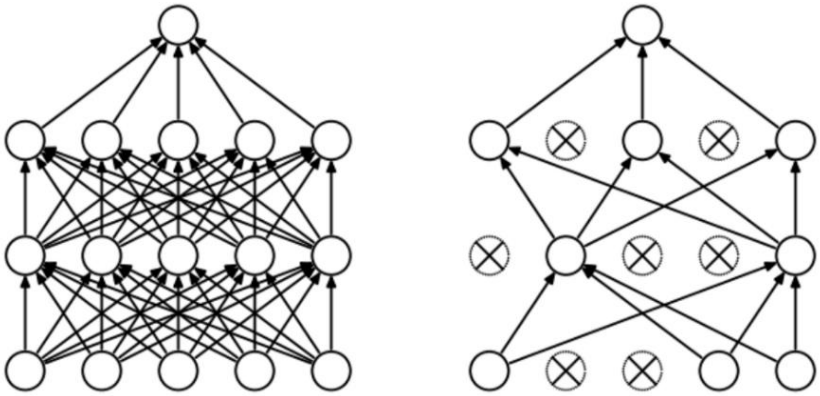
Nguyên lý hoạt động của Dropout

Trong quá trình huấn luyện, Dropout hoạt động bằng cách ngẫu nhiên loại bỏ một số nơron với xác suất ppp, đồng thời giữ lại các nơron còn lại với xác suất $1 - p$. Mỗi lớp có thể có một hệ số Dropout p khác nhau. Điều này giúp ngăn ngừa các nơron trở nên quá phụ thuộc vào nhau, từ đó tăng khả năng khái quát hóa của mô hình.

Xác Định Hệ Số p

1. **Sử dụng tập dữ liệu xác thực:** Trong quá trình huấn luyện, giá trị của tham số p nên được xác định bằng cách sử dụng một tập dữ liệu xác thực để tránh overfitting. Thông thường, giá trị p được chọn trong khoảng 0.2 đến 0.5 cho các lớp ẩn và khoảng 0.1 cho lớp đầu vào.
2. **Loại trừ holes trong dữ liệu đầu vào:** Để tránh sự không đồng đều trong dữ liệu đầu vào, giá trị p cho lớp đầu vào nên được thiết lập là 1.0, tương đương với việc không áp dụng Dropout ở đó.
3. **Tối ưu hóa Dropout:** Dropout hoạt động tốt khi kết hợp với các kỹ thuật khác như max-norm regularization, tỷ lệ học cao với việc giảm dần, và động lượng lớn. Thuật toán tối ưu hóa phổ biến nhất là Adam, nhờ khả năng tối ưu hóa hiệu quả và ổn định của nó.

Ví dụ về áp dụng Dropout



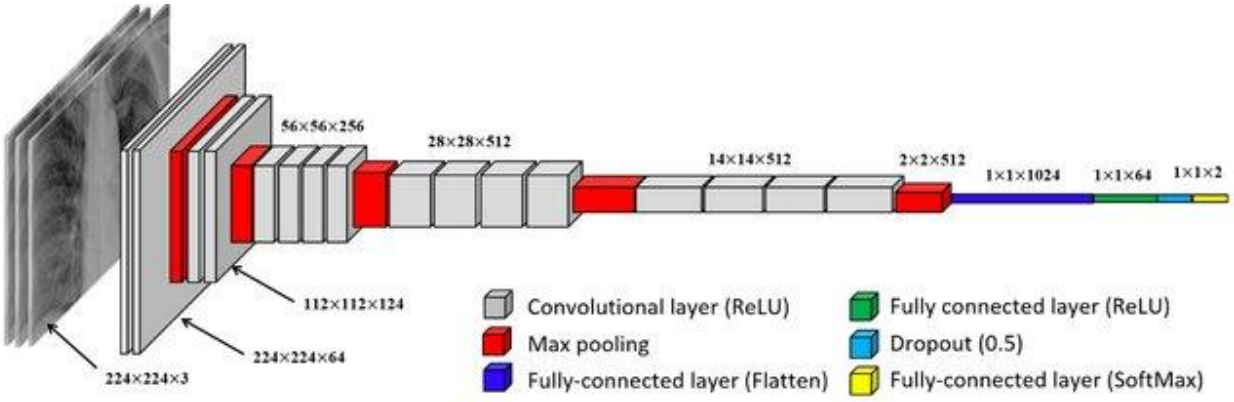
Hình 8: Phương pháp dropout có số lượng các liên kết mạng bị giảm so với trước đó làm mô hình ít phức tạp hơn. Đồng thời đây cũng là một dạng ensemble model giúp giảm thiểu được overfitting.

Hình 8 mô tả một ví dụ về mạng nơron tiêu chuẩn có hai lớp ẩn (bên trái) và một ví dụ về mạng được tạo ra bằng cách áp dụng Dropout (bên phải). Các đơn vị chéo đã bị loại bỏ trong quá trình huấn luyện.

PHẦN II: GIỚI THIỆU CÁC MẠNG NƠRON TÍCH CHẬP CỤ THỂ

2.1. MÔ HÌNH VGG19

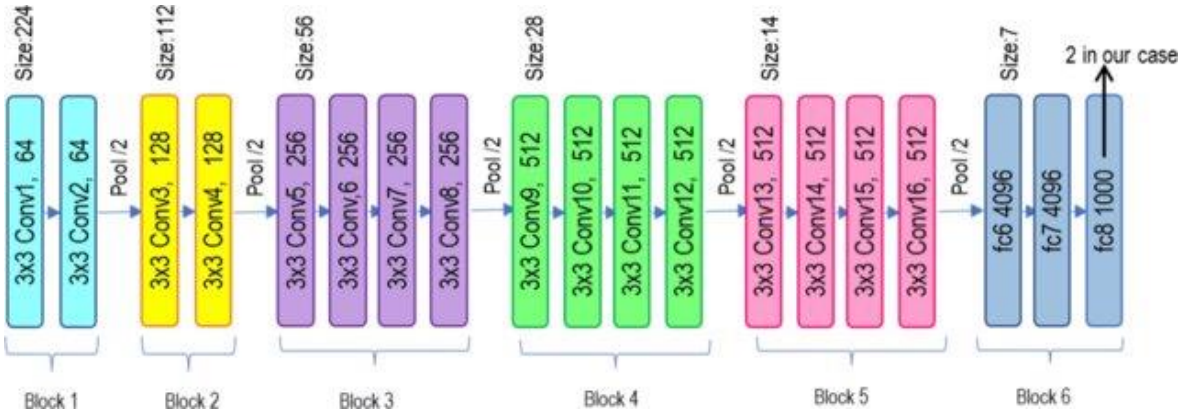
VGG19 là một kiến trúc mạng nơron tích chập (Convolutional Neural Network - CNN) được phát triển bởi Visual Geometry Group (VGG) tại Đại học Oxford. Mô hình này nổi bật với cấu trúc đơn giản nhưng hiệu quả trong việc trích xuất các đặc trưng từ ảnh, nhờ vào việc sử dụng các kernel kích thước nhỏ và một số lượng lớn các lớp tích chập.



Hình 9: Mô hình VGG-19 – Nguồn: researchgate.net

2.1.1. Kiến Trúc Của VGG19

VGG19 bao gồm tổng cộng 19 lớp, bao gồm 16 lớp tích chập (convolutional layers) và 3 lớp kết nối đầy đủ (fully connected layers). Mô hình này sử dụng các kernel kích thước nhỏ (3x3) trong các lớp tích chập và các lớp gộp (pooling layers) để giảm kích thước không gian của ảnh. Dưới đây là các thành phần chi tiết của kiến trúc VGG19:



Hình 10: Kiến trúc VGG-19. VGG-19 có 16 lớp tích chập được nhóm thành 5 khối. Sau mỗi khối, có một lớp Maxpool làm giảm kích thước của hình ảnh đầu vào đi 2 và tăng số lượng bộ lọc trong lớp tích chập đi 2. Kích thước của ba lớp dày đặc cuối cùng trong khối 6 lần lượt là 4096, 4096 và 1000. VGG phân loại hình ảnh đầu vào thành 1000 loại khác nhau. Vì có hai lớp đầu ra trong nghiên cứu này, nên kích thước của fc8 được đặt thành hai

Nguồn: researchgate.net

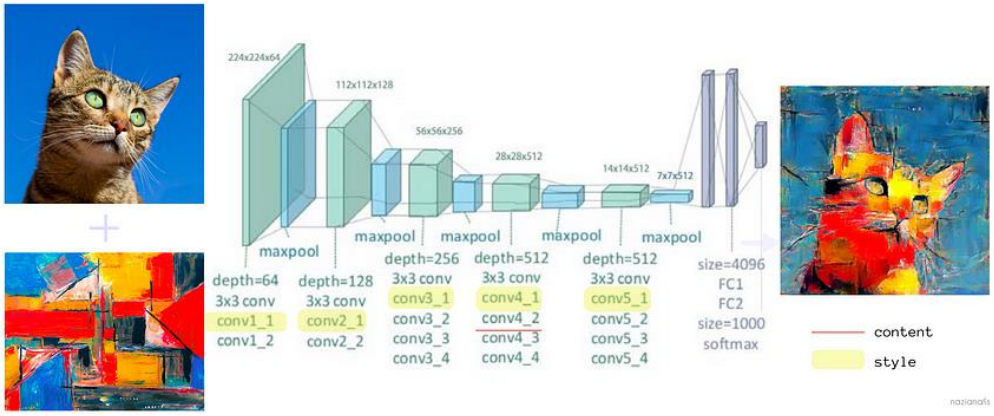
1. **Các lớp tích chập:**
    - VGG19 sử dụng các bộ lọc kích thước 3x3 với stride 1 và padding 1, giúp duy trì kích thước không gian của ảnh sau mỗi lớp tích chập. Điều này đảm bảo rằng kích thước của ảnh đầu vào không bị giảm đi quá nhiều sau mỗi lớp.
    - Số lượng bộ lọc trong mỗi lớp tăng dần từ 64, 128, 256 đến 512, cho phép mô hình học được các đặc trưng ngày càng phức tạp hơn.
2. **Các lớp gộp (Pooling layers):**
    - Các lớp gộp được sử dụng sau mỗi cụm lớp tích chập để giảm kích thước không gian của ảnh. Lớp gộp sử dụng max pooling với kích thước 2x2 và stride 2, giúp giảm số lượng tham số và tăng tốc độ tính toán.
    - Các lớp gộp trong VGG19 giúp giảm kích thước của ảnh đầu vào một cách hiệu quả, đồng thời giữ lại các đặc trưng quan trọng.
3. **Các lớp kết nối đầy đủ (Fully Connected Layers):**
    - Sau các lớp tích chập và gộp, dữ liệu được làm phẳng và đưa qua ba lớp kết nối đầy đủ, kết thúc bằng một lớp softmax cho phân loại. Các lớp này giúp tổng hợp thông tin từ các đặc trưng đã học được để đưa ra dự đoán cuối cùng.

- Lớp kết nối đầy đủ cuối cùng sử dụng hàm softmax để thực hiện phân loại đa lớp, đưa ra xác suất cho từng lớp trong tập dữ liệu.

2.1.2. Ứng Dụng VGG19 Trong Neural Style Transfer

Trong đề tài "Tạo ảnh phong cách nghệ thuật bằng Neural Style Transfer", VGG19 đóng vai trò quan trọng trong việc trích xuất các đặc trưng từ ảnh nội dung và ảnh phong cách. Dưới đây là cách mà VGG19 được sử dụng cụ thể:

1. **Trích xuất đặc trưng:**
   - Các lớp tích chập trong VGG19 giúp mô hình học được các đặc trưng về cấu trúc và phong cách của ảnh. Các lớp này bao gồm các kernel kích thước nhỏ giúp mô hình học được các đặc trưng từ pixel cơ bản đến các đặc trưng cao cấp như cạnh, hình dạng, và các bộ phận cơ thể.
2. **Sử dụng các lớp trung gian:**
   - Trong Neural Style Transfer, các lớp tích chập của VGG19 ở các tầng khác nhau được sử dụng để trích xuất đặc trưng từ ảnh nội dung và ảnh phong cách. Các lớp này giúp mô hình học được các đặc trưng cụ thể của mỗi ảnh, từ đó kết hợp chúng để tạo ra ảnh phong cách mới.
   - Đối với ảnh nội dung, chúng ta lấy ảnh này và đưa qua VGG19, sau đó lấy các kích hoạt (activations) của mạng tại một lớp tích chập muộn (ví dụ: conv4_2). Điều này giúp chúng ta giữ lại nội dung chính của ảnh.
   - Đối với ảnh phong cách, chúng ta lấy ảnh này và đưa qua cùng mạng, sau đó lấy các kích hoạt tại các lớp tích chập từ sớm đến trung bình (ví dụ: conv1_1, conv2_1, conv3_1, conv4_1, conv5_1). Các kích hoạt này được mã hóa vào ma trận Gram, biểu diễn phong cách của ảnh.



Hình 11: Kiến trúc VGG 19 trong tạo ảnh phong cách nghệ thuật bằng Neural Style Transfer – Nguồn: medium.com

3. **Tối ưu hóa việc tạo ảnh phong cách:**
    - Mục tiêu của Neural Style Transfer là tạo ra một ảnh mới có nội dung từ ảnh nội dung và phong cách từ ảnh phong cách. Để đạt được điều này, chúng ta tính toán các loại mất mát (loss) sau:
        - **Content loss**: Là khoảng cách L2 giữa ảnh nội dung và ảnh được tạo ra.
        - **Style loss**: Là tổng các khoảng cách L2 giữa các ma trận Gram của ảnh nội dung và ảnh phong cách, được trích xuất từ các lớp khác nhau của VGG19.
        - **Total variation loss**: Giúp duy trì tính liên tục không gian giữa các pixel của ảnh được tạo ra, giảm nhiễu và tăng tính mạch lạc trực quan.
        - **Total loss**: Là tổng của các loại mất mát trên, mỗi loại được nhân với trọng số tương ứng.
    - Sử dụng kỹ thuật tối ưu hóa lặp lại (ví dụ: L-BFGS), chúng ta sẽ dần dần giảm các mất mát này để đạt được kết quả mong muốn.

2.1.3. Ưu Điểm Của VGG19

1. **Đơn giản và hiệu quả:**

- VGG19 có cấu trúc đơn giản nhưng hiệu quả cao, dễ dàng triển khai và tùy chỉnh. Cấu trúc của nó giúp mô hình dễ hiểu và dễ dàng thực hiện các điều chỉnh cần thiết.

2. **Trích xuất đặc trưng tốt:**
    - Các lớp tích chập với kernel nhỏ cho phép mô hình học các đặc trưng từ cơ bản đến phức tạp. Điều này giúp VGG19 đạt được hiệu quả cao trong việc trích xuất các đặc trưng hình ảnh, từ các cạnh cơ bản đến các hình dạng phức tạp.
3. **Ứng dụng rộng rãi:**
    - VGG19 được sử dụng rộng rãi trong nhiều bài toán thị giác máy tính như phân loại ảnh, nhận diện đối tượng, và tạo ảnh phong cách nghệ thuật. Cấu trúc của VGG19 đã được chứng minh là hiệu quả trong nhiều thử nghiệm và ứng dụng thực tế.

2.2. MÔ HÌNH RESNET-50

ResNet (Residual Network) được giới thiệu đến công chúng vào năm 2015 và giành được vị trí thứ 1 trong cuộc thi ILSVRC 2015 với tỉ lệ lỗi top 5 chỉ 3.57%. Không những thế, nó còn đứng vị trí đầu tiên trong các cuộc thi ILSVRC và COCO 2015 về ImageNet Detection, ImageNet Localization, COCO Detection và COCO Segmentation. Hiện tại, có rất nhiều biến thể của kiến trúc ResNet với số lớp khác nhau như ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152, ...

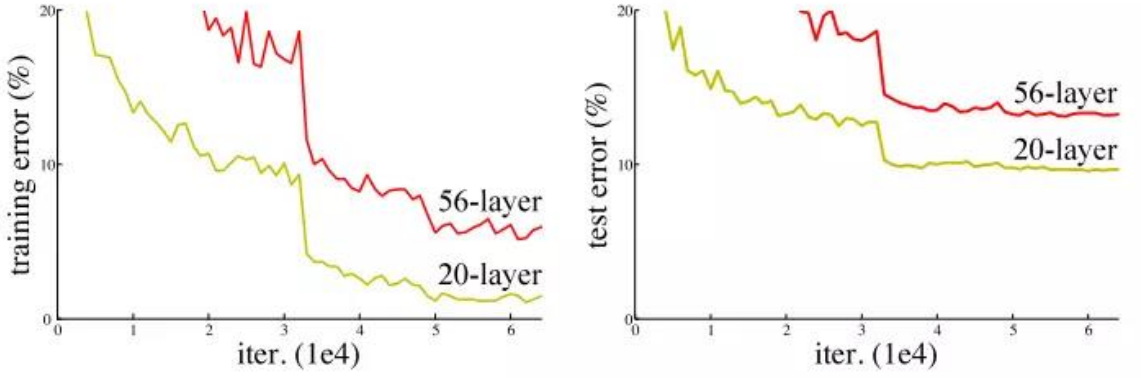
ResNet50 là một kiến trúc mạng nơron tích chập (Convolutional Neural Network - CNN) sâu với 50 lớp, được thiết kế để giải quyết vấn đề gradient biến mất và duy trì hiệu suất cao khi mô hình trở nên sâu hơn. ResNet50 sử dụng các khối residual, cho phép thông tin được truyền trực tiếp qua các lớp, giúp cải thiện quá trình huấn luyện và hiệu suất của mô hình.

Tại sao lại xuất hiện mạng ResNet

Mạng ResNet là một mạng CNN được thiết kế để làm việc với hàng trăm lớp. Một vấn đề xảy ra khi xây dựng mạng CNN với nhiều lớp là hiện tượng Vanishing Gradient dẫn đến quá trình học tập không hiệu quả.

Vanishing Gradient:

Trước hết, thuật toán Backpropagation là một kỹ thuật thường được sử dụng trong quá trình huấn luyện. Ý tưởng chung của thuật toán là đi từ output layer đến input layer và tính toán gradient của hàm cost tương ứng cho từng parameter (weight) của mạng. Gradient Descent sau đó được sử dụng để cập nhật các parameter đó.



Hình 12: Mối tương quan giữa độ sâu và hiệu suất mạng – Nguồn: trituenhantao.io

Toàn bộ quá trình trên sẽ được lặp đi lặp lại cho đến khi các parameter của mạng hội tụ. Thông thường, chúng ta sẽ có một hyperparameter (số Epoch - số lần mà training set được duyệt qua một lần và weights được cập nhật) để định nghĩa số lượng vòng lặp. Nếu số lượng vòng lặp quá nhỏ, mạng có thể sẽ không cho ra kết quả tốt, và ngược lại, thời gian huấn luyện sẽ lâu nếu số lượng vòng lặp quá lớn.

Tuy nhiên, trong thực tế, gradients thường có giá trị nhỏ dần khi đi xuống các layer thấp hơn, dẫn đến các cập nhật thực hiện bởi Gradient Descent không làm thay đổi nhiều weights của các layer đó, khiến chúng không thể hội tụ và mạng không thu được kết quả tốt. Hiện tượng này gọi là Vanishing Gradient.

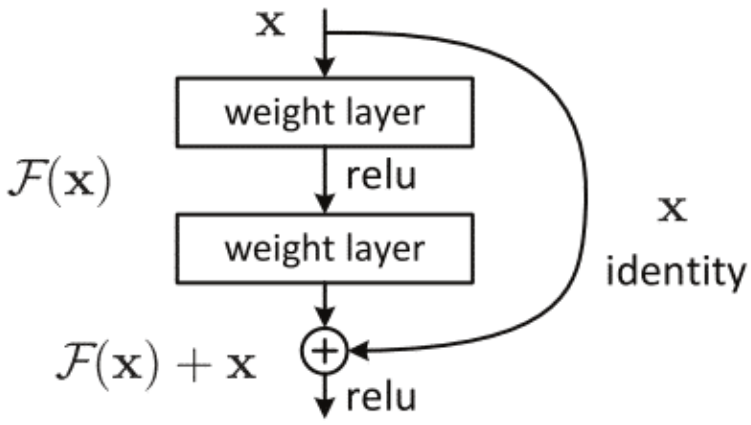
Mạng ResNet ra đời để giải quyết vấn đề này.

Skip connection - cách khắc phục của ResNet

Tác giả đã thực hiện residual mapping để copy thông tin từ các layer nông **shallow layer** trước đó đến các layer sâu hơn. Chúng ta giả sử output của shallow layer là $\boldsymbol{x}$. Trong quá trình forward của mạng nó được đưa qua một phép biến đổi tuyến tính $\boldsymbol{F(x)}$. Chúng ta giả sử output của phép biến đổi tuyền tính này là $\boldsymbol{H(x)}$. Một residual (phần dư) giữa deep layer và shallow layer là

$$\mathcal{F}(\mathbf{x}; W_i) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$$

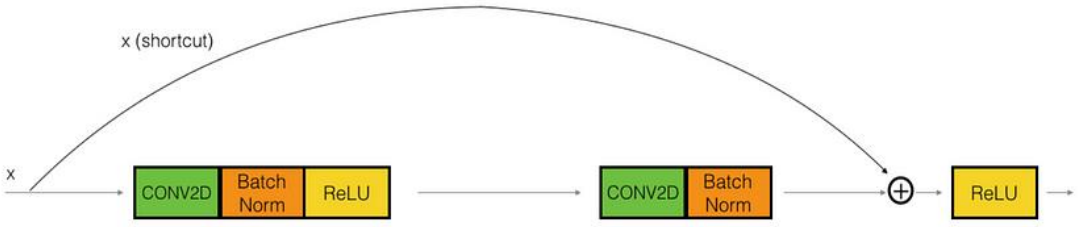
Trong đố W_i là các tham số của mô hình CNN với phép biến đổi F và nó được tối ưu trong quá trình huấn luyện.



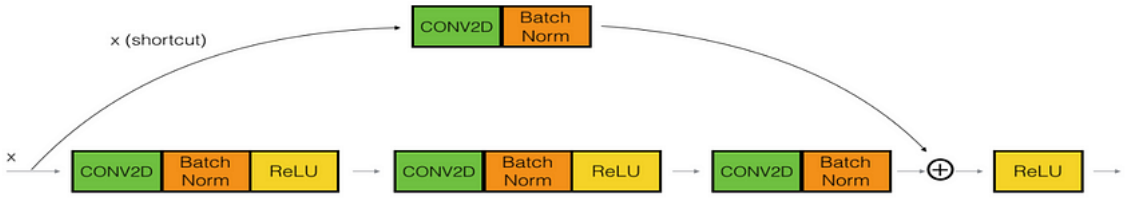
Hình 13: Minh họa một khối residual trong mạng ResNet, cụ thể là một "Residual Block" – Nguồn: viblo.asia

Việc thêm vào các residual block vào trong kiến trúc mạng deep learning có hai cách tuỳ thuộc vào từng trường hợp cụ thể.

- **identity mapping** trong trường hợp này residual mapping đơn giản là việc cộng trực tiếp x vào đầu ra của các stacked block $F(x)$. Đây là một cách sử dụng khá phổ biến trong thiết kế mạng ResNet nếu như input activation có cùng số chiều với output activation. Chúng ta có thể minh hoạ nó trong hình sau:



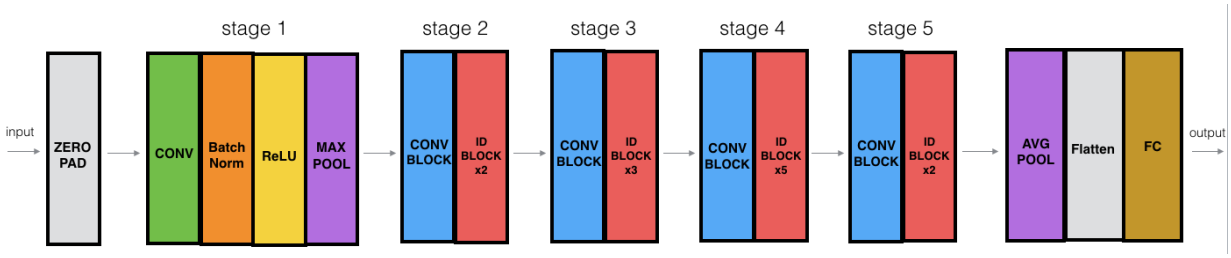
- **Convolutional block** một trường hợp khác là thay vì cộng trực tiếp giá trị của input activation chúng ta sẽ đưa qua một convolution transformation. Trường hợp này có thể được thực hiện trong trường hợp input activation và output activation có số chiều khác nhau. Lúc này đầu ra được xác định như sau $y = F(x;Wi) + \mathbf{Conv}(x)$. Chúng ta có thể xem hình minh hoạ dưới đây



2.2.1. Kiến Trúc Của ResNet50

ResNet50 là một mô hình mạng nơ-ron sâu với kiến trúc gồm 50 lớp, nổi bật với các khối residual giúp giảm thiểu vấn đề gradient biến mất và cải thiện khả năng học của mô hình. Dưới đây là mô tả chi tiết về kiến trúc của ResNet50:

Xây dựng mạng ResNet-50



Hình 14: Minh họa cấu trúc chi tiết của mô hình ResNet50 – Nguồn: viblo.asia

1. **Zero-padding:**
    - Input với kích thước (3x3).
2. **Stage 1:**
    - **Convolution (Conv1)** với 64 filters với kích thước (7x7), sử dụng stride (2x2).
    - **Batch Normalization**.
    - **MaxPooling** với kích thước (3x3).
3. **Stage 2:**
    - **Convolutional block** sử dụng 3 filters với kích thước 64x64x256, kernel (f=3), stride (s=1).
    - Có **2 Identity blocks** với filter size 64x64x256, kernel (f=3).
4. **Stage 3:**

- **Convolutional block** sử dụng 3 filters với kích thước 128x128x512, kernel (f=3), stride (s=2).
- Có **3 Identity blocks** với filter size 128x128x512, kernel (f=3).

5. **Stage 4:**
   - **Convolutional block** sử dụng 3 filters với kích thước 256x256x1024, kernel (f=3), stride (s=2).
   - Có **5 Identity blocks** với filter size 256x256x1024, kernel (f=3).
6. **Stage 5:**
   - **Convolutional block** sử dụng 3 filters với kích thước 512x512x2048, kernel (f=3), stride (s=2).
   - Có **2 Identity blocks** với filter size 512x512x2048, kernel (f=3).
7. **The 2D Average Pooling:**
   - Sử dụng lớp gộp trung bình với kích thước (2x2) để giảm kích thước không gian của đặc trưng.
8. **The Flatten.**
   - Chuyển đổi các tensor thành vector phẳng để đưa vào lớp fully connected
9. **Fully Connected (Dense):**
   - Sử dụng softmax activation để dự đoán đầu ra cuối cùng

2.2.2. Ưu Điểm Của ResNet50

1. **Khắc phục vấn đề gradient biến mất:**
   - Các khối residual giúp truyền thông tin qua các lớp, giảm thiểu vấn đề gradient biến mất và giúp mô hình học tốt hơn. Điều này cho phép ResNet50 đào tạo các mô hình sâu hơn mà không gặp phải các vấn đề về độ sâu của mạng.
2. **Hiệu suất cao:**
   - ResNet50 có khả năng tổng quát hóa tốt và đạt hiệu suất cao trên nhiều bài toán thị giác máy tính. Mô hình này đã được chứng minh là có hiệu

suất vượt trội trong nhiều cuộc thi và ứng dụng thực tế, từ phân loại ảnh đến nhận diện đối tượng.

3. **Ứng dụng rộng rãi:**
    - ResNet50 được sử dụng trong nhiều ứng dụng khác nhau như phân loại ảnh, nhận diện đối tượng, và các bài toán thị giác máy tính phức tạp. Kiến trúc của ResNet50 đã trở thành tiêu chuẩn trong nhiều nghiên cứu và ứng dụng thực tế.

2.2.3. Ứng Dụng ResNet50 Trong Neural Style Transfer

Trong đề tài "Tạo ảnh phong cách nghệ thuật bằng Neural Style Transfer", ResNet50 có thể được sử dụng để trích xuất các đặc trưng từ ảnh nội dung và ảnh phong cách. Các khối residual giúp mô hình học được các đặc trưng phức tạp và truyền tải chúng một cách hiệu quả, từ đó tạo ra các ảnh nghệ thuật với sự kết hợp hoàn hảo giữa nội dung và phong cách.

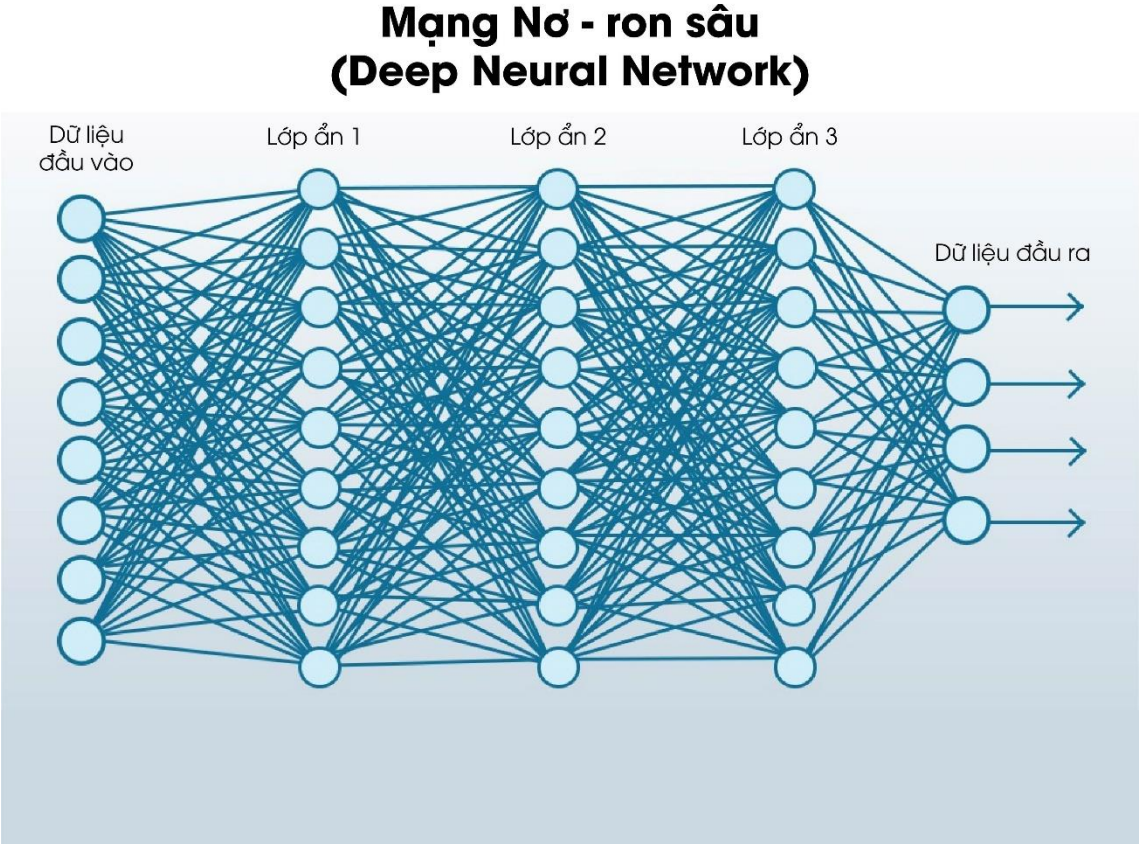
1. **Trích xuất đặc trưng:**
    - Các khối residual của ResNet50 giúp mô hình học được các đặc trưng phức tạp từ ảnh nội dung và phong cách. Các khối này cho phép thông tin được truyền qua các lớp mà không bị mất mát, giúp giữ lại các đặc trưng quan trọng của ảnh.
2. **Sử dụng các khối residual:**
    - Trong Neural Style Transfer, các khối residual của ResNet50 giúp mô hình học được các đặc trưng phức tạp và truyền tải chúng một cách hiệu quả. Điều này giúp tạo ra các ảnh phong cách mới với sự kết hợp giữa nội dung và phong cách, giữ được tính chất đặc trưng của cả hai ảnh.
3. **Tối ưu hóa việc tạo ảnh phong cách:**
    - Mục tiêu của Neural Style Transfer là tạo ra một ảnh mới có nội dung từ ảnh nội dung và phong cách từ ảnh phong cách. Để đạt được điều này, chúng ta tính toán các loại mất mát (loss) sau:
        - **Content loss:** Là khoảng cách L2 giữa ảnh nội dung và ảnh được tạo ra.

        - **Style loss:** Là tổng các khoảng cách L2 giữa các ma trận Gram của ảnh nội dung và ảnh phong cách, được trích xuất từ các lớp khác nhau của ResNet50.
        - **Total variation loss:** Giúp duy trì tính liên tục không gian giữa các pixel của ảnh được tạo ra, giảm nhiễu và tăng tính mạch lạc trực quan.
        - **Total loss:** Là tổng của các loại mất mát trên, mỗi loại được nhân với trọng số tương ứng.
    - Sử dụng kỹ thuật tối ưu hóa lặp lại (ví dụ: L-BFGS), chúng ta sẽ dần dần giảm các mất mát này để đạt được kết quả mong muốn.

PHẦN III: GIỚI THIỆU VỀ PHƯƠNG PHÁP HỌC SÂU

3.1. Khái Niệm Học Sâu (Deep Learning)

Học sâu là một nhánh tiên tiến của trí tuệ nhân tạo (AI) và học máy (Machine Learning) tập trung vào việc sử dụng các mạng nơ-ron nhân tạo (Artificial Neural Networks - ANNs) với nhiều lớp (layers) để học từ các dữ liệu lớn và phức tạp.



Hình 15: Cách học sâu (Deep learning) hoạt động – Nguồn: Internet

Các mạng nơ-ron nhân tạo này được cấu thành từ nhiều lớp ẩn (hidden layers) nằm giữa lớp đầu vào (input layer) và lớp đầu ra (output layer). Mỗi nơ-ron trong một lớp ẩn nhận đầu vào từ các nơ-ron của lớp trước đó, áp dụng một hàm kích hoạt (activation function), và truyền đầu ra tới các nơ-ron của lớp kế tiếp. Trong quá trình huấn luyện, mạng điều chỉnh các trọng số (weights) liên kết giữa các nơ-ron để giảm thiểu lỗi giữa dự đoán của mạng và giá trị thực tế thông qua một quá trình gọi là lan truyền ngược (backpropagation).

Học sâu đã đạt được nhiều thành tựu đáng kể trong các lĩnh vực như nhận dạng hình ảnh, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên, và chơi game. Những mô hình học sâu có khả năng học và trích xuất các đặc trưng từ dữ liệu mà không cần thiết phải thiết kế các đặc trưng đó một cách thủ công.

3.2. Giới Thiệu Các Phương Pháp Học Sâu Cụ Thể Được Sử Dụng

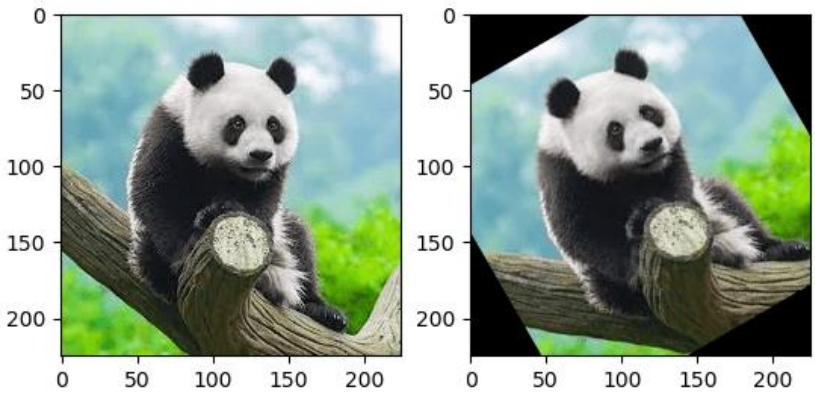
3.2.1. Data Augmentation

Data augmentation là một kỹ thuật được sử dụng để tăng cường tập dữ liệu huấn luyện bằng cách thêm vào các biến đổi hoặc biến động vào dữ liệu hiện có mà không làm tăng chi phí tính toán. Ý tưởng chính của data augmentation là tạo ra các phiên bản mới của dữ liệu huấn luyện bằng cách thực hiện các biến đổi nhỏ trên dữ liệu hiện có, như xoay, phóng to, thu nhỏ, lật ảnh, thay đổi màu sắc, cắt ảnh, vv.

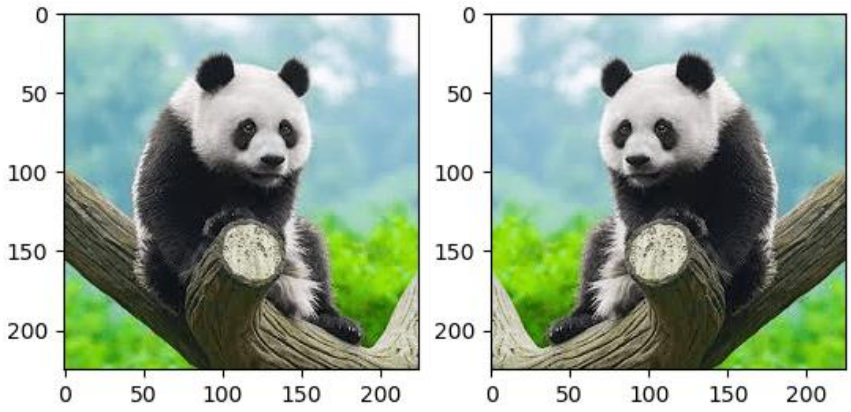
Ví dụ, giả sử chúng ta có một tập dữ liệu hình ảnh về các loài hoa, và mỗi hình ảnh chỉ hiển thị loài hoa từ một góc độ nhất định. Sử dụng kỹ thuật augmentation dữ liệu, chúng ta có thể tạo ra các phiên bản mới của các hình ảnh này bằng cách áp dụng các biến đổi như xoay ảnh một góc nhất định, phóng to hoặc thu nhỏ ảnh, lật ảnh theo chiều ngang hoặc dọc, thay đổi ánh sáng hoặc màu sắc, vv.

Cụ thể, một số biến đổi phổ biến được áp dụng trong data augmentation bao gồm:

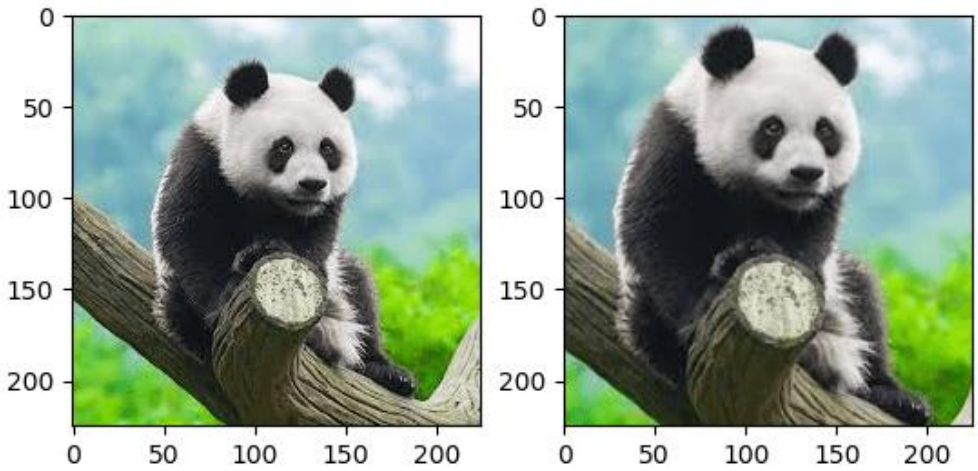
1. **Rotation:** Xoay ảnh: Ảnh được xoay một góc nhất định để tạo ra các phiên bản mới từ các góc độ khác nhau.



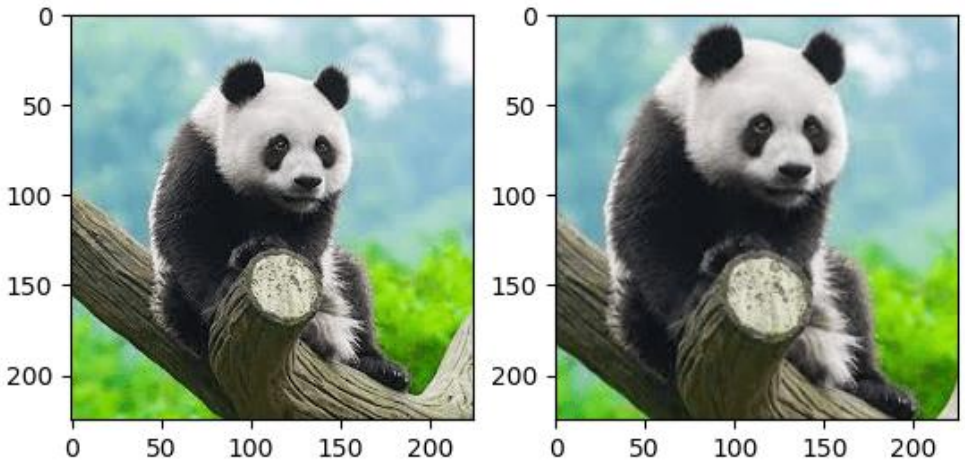
2. **Flip:** Lật ảnh: Ảnh được lật theo chiều ngang hoặc dọc để tạo ra các phiên bản lật ngược.



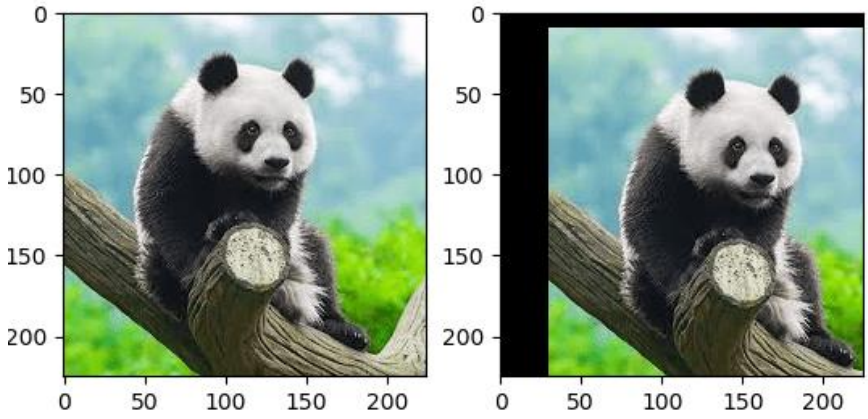
3. **Scale:** Phóng to và thu nhỏ: Ảnh được phóng to hoặc thu nhỏ với tỉ lệ khác nhau để tạo ra các phiên bản có kích thước khác nhau.



4. **Crop:** Cắt ảnh: Một phần của ảnh được cắt ra để tạo ra các phiên bản mới với các khung cảnh khác nhau.



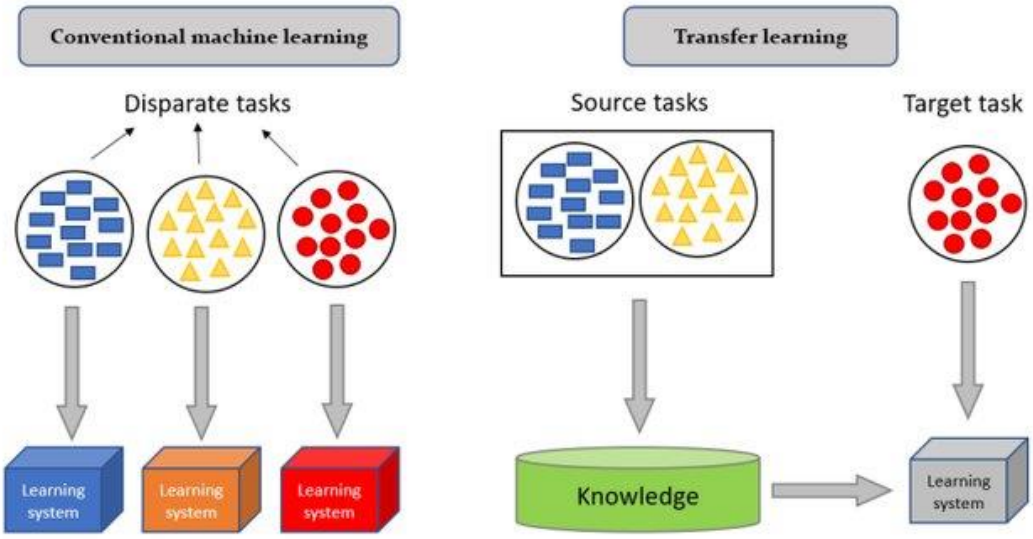
5. **Translation:** dịch chuyển ảnh theo chiều x, y.



3.2.2. Transfer Learning

Transfer Learning là một kỹ thuật trong học sâu (Deep Learning) cho phép sử dụng các mô hình đã được huấn luyện trên một nhiệm vụ cụ thể và sau đó tái sử dụng hoặc điều chỉnh để giải quyết một nhiệm vụ khác nhưng có liên quan. Mục đích của Transfer Learning là tận dụng các kiến thức và đặc trưng đã học từ nhiệm vụ ban đầu để cải thiện hiệu suất và giảm thời gian huấn luyện cho nhiệm vụ mới.

Transfer Learning đặc biệt hữu ích khi chúng ta có ít dữ liệu cho nhiệm vụ mới, vì việc huấn luyện một mô hình từ đầu đòi hỏi một lượng lớn dữ liệu và tài nguyên tính toán. Bằng cách sử dụng mô hình đã được huấn luyện trước, chúng ta có thể tiết kiệm đáng kể thời gian và tài nguyên.



Hình 16: Sơ đồ so sánh các quá trình học tập giữa Machine Learning thông thường và Transfer Learning – Nguồn: www.researchgate.net

Các Phương Pháp Transfer Learning

Có một số phương pháp để thực hiện Transfer Learning, bao gồm:

1. **Feature Extraction (Trích xuất đặc trưng):**
    - Sử dụng mô hình đã được huấn luyện để trích xuất các đặc trưng từ dữ liệu mới và sau đó huấn luyện một mô hình đơn giản (như SVM, Logistic Regression) trên các đặc trưng này.
    - Thường giữ nguyên các trọng số của các lớp convolutional và chỉ huấn luyện lại các lớp fully connected cuối cùng.
2. **Fine-Tuning (Tinh chỉnh):**
    - Sử dụng mô hình đã được huấn luyện và huấn luyện lại toàn bộ hoặc một phần của mô hình với dữ liệu mới.
    - Thường tinh chỉnh các lớp cuối cùng hoặc các lớp có trọng số gần lớp đầu ra nhất để phù hợp với nhiệm vụ mới.

Quy Trình Thực Hiện Transfer Learning

Dưới đây là quy trình cơ bản để thực hiện Transfer Learning với một mô hình đã được huấn luyện trước:

1. **Chọn Mô Hình Cơ Sở:**

- **Mô tả:** Chọn một mô hình đã được huấn luyện trước trên một bộ dữ liệu lớn và có nhiệm vụ tương tự (ví dụ: ResNet, VGG đã được huấn luyện trên ImageNet).
- **Minh họa:** Chọn mô hình ResNet50 đã được huấn luyện trên ImageNet để phân loại hình ảnh động vật.

2. **Chuẩn Bị Dữ Liệu:**
   - **Mô tả:** Chuẩn bị dữ liệu cho nhiệm vụ mới, bao gồm tiền xử lý (preprocessing) như thay đổi kích thước, chuẩn hóa và chia tập dữ liệu thành tập huấn luyện và tập kiểm tra.
   - **Minh họa:** Thu thập hình ảnh mèo và chó, thay đổi kích thước hình ảnh về 224x224 pixels, chuẩn hóa và chia dữ liệu thành tập huấn luyện và tập kiểm tra.
3. **Thay Thế và Tinh Chỉnh Lớp Cuối Cùng:**
   - **Mô tả:** Thay thế các lớp fully connected cuối cùng của mô hình gốc bằng các lớp mới phù hợp với nhiệm vụ mới.
   - **Minh họa:** Thay thế lớp cuối cùng của mô hình ResNet50 bằng một lớp fully connected mới với số lượng đầu ra tương ứng với số lượng loài động vật cần phân loại (mèo và chó).
4. **Huấn Luyện và Tinh Chỉnh Mô Hình:**
   - **Mô tả:** Huấn luyện các lớp mới và tinh chỉnh toàn bộ hoặc một phần của mô hình. Sử dụng kỹ thuật giảm tốc độ học (learning rate) để tránh làm hỏng các trọng số đã được học từ mô hình gốc.
   - **Minh họa:** Huấn luyện lớp fully connected mới và tinh chỉnh các lớp convolutional cuối cùng của mô hình ResNet50 với tốc độ học giảm dần.
5. **Đánh Giá và Tối Ưu:**
   - **Mô tả:** Đánh giá hiệu suất của mô hình trên tập kiểm tra.
   - **Minh họa:** Đo lường độ chính xác (accuracy) của mô hình trên tập kiểm tra hình ảnh mèo và chó và tối ưu mô hình bằng cách điều chỉnh các siêu tham số (hyperparameters) như tốc độ học, số lượng epochs, và số lượng lớp tinh chỉnh.

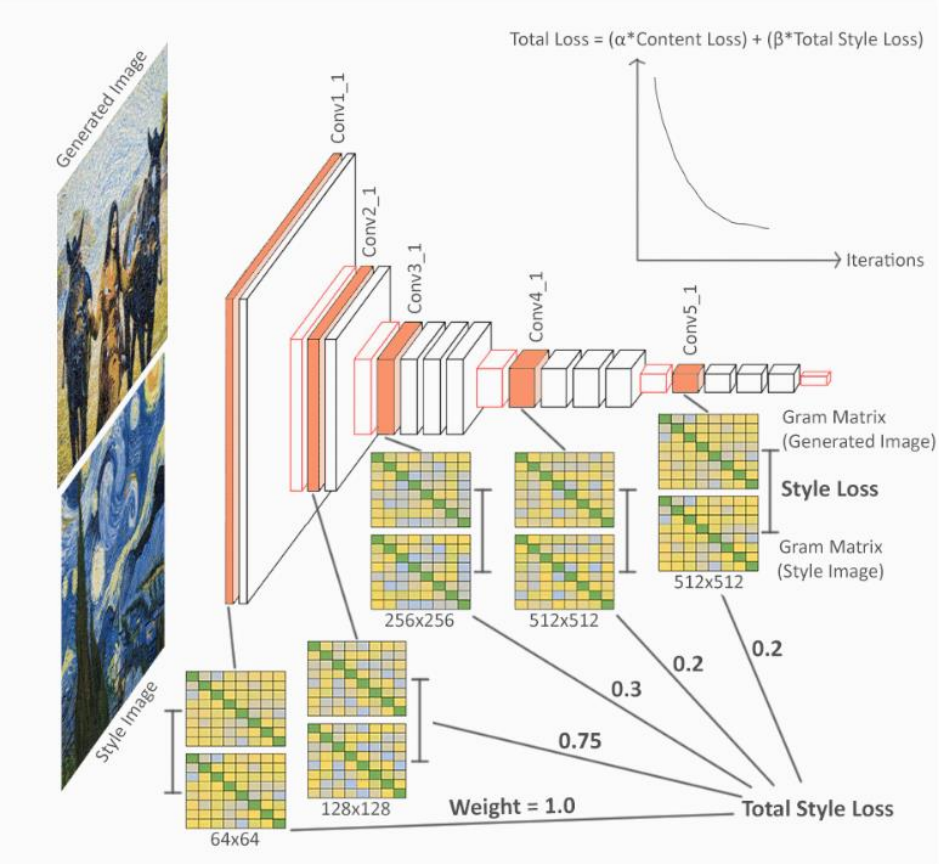
3.2.3. Neural Style Transfer

Neural Style Transfer (NST) là một kỹ thuật mạnh mẽ trong học sâu, cho phép kết hợp nội dung của một bức ảnh (Content Image) với phong cách của một bức ảnh khác (Style Image) để tạo ra một bức ảnh mới (Generated Image). Bức ảnh được tạo ra sẽ duy trì nội dung từ ảnh gốc và phong cách từ ảnh mẫu. NST sử dụng các mạng tích chập đã được huấn luyện trước như VGG19 để trích xuất các đặc trưng từ cả ảnh gốc và ảnh mẫu, sau đó tối ưu hóa ảnh đầu ra để đạt được sự kết hợp mong muốn.

Quy trình Thuật Toán NST

Quá trình thực hiện NST bao gồm ba bước chính:

1. **Xây dựng hàm chi phí nội dung $J_{content}(C, G)$**
2. **Xây dựng hàm chi phí phong cách $J_{style}(S, G)$**
3. **Kết hợp hai hàm chi phí để có hàm chi phí tổng $J(G) = \propto J_{content}(C, G) + \beta J_{style}(S, G)$**



1. Hàm Chi Phí Nội Dung $J_{content}(C, G)$

Hàm chi phí nội dung đo lường sự khác biệt về nội dung giữa ảnh gốc (Content Image, C) và ảnh được tạo (Generated Image, G). Để đo lường nội dung, thường chọn một lớp trong mạng tích chập đã huấn luyện, chẳng hạn như một lớp sâu trong mô hình VGG19.

Công thức tính hàm chi phí nội dung là:

$$J_{content}(C,G) = \frac{1}{4n_H n_W n_C} \sum_{h=1}^{n_H} \sum_{w=1}^{n_W} \sum_{c=1}^{n_C} (a_{C,h,w,c} - a_{G,h,w,c})^2$$

Trong đó:

- $\mathbf{a_C}$ **và** $\mathbf{a_G}$ là các đặc trưng của ảnh gốc và ảnh được tạo tại lớp tích chập đã chọn.
- $\mathbf{n_H}$**,** $\mathbf{n_W}$**,** $\mathbf{n_C}$ là chiều cao, chiều rộng và số lượng kênh của các đặc trưng.

2. Hàm Chi Phí Phong Cách $J_{style}(S, G)$

Hàm chi phí phong cách đo lường sự khác biệt về phong cách giữa ảnh mẫu (Style Image, S) và ảnh được tạo (Generated Image, G). Để đo lường phong cách, thường sử dụng ma trận Gram, biểu diễn sự tương quan giữa các đặc trưng ở mỗi lớp tích chập.

Công thức tính ma trận Gram là:

$$G^{(l)} = A^{(l)} (A^{(l)})^T$$

Hàm chi phí phong cách cho một lớp ***l*** là:

$$J_{style}^{(l)}(S,G) = \frac{1}{4(n_H^{(l)} n_W^{(l)} n_C^{(l)})^2} \sum_{i=1}^{n_C^{(l)}} \sum_{j=1}^{n_C^{(l)}} (G_{S,i,j}^{(l)} - G_{G,i,j}^{(l)})^2$$

Hàm chi phí phong cách tổng là:

$$J_{style}(S,G) = \sum_l w_l J_{style}^{(l)}(S,G)$$

Trong đó $\mathbf{w_l}$ là trọng số cho lớp **J**

3. Hàm Chi Phí Tổng J(G)

Hàm chi phí tổng kết hợp hàm chi phí nội dung và hàm chi phí phong cách. Mục tiêu của thuật toán là tối ưu hóa ảnh được tạo (Generated Image, G) để giảm hàm chi phí tổng này:

$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$

Trong đó:

- $\propto$ và $\boldsymbol{\beta}$ là các trọng số điều chỉnh mức độ ảnh hưởng của hàm chi phí nội dung và phong cách.

Quy Trình Thực Hiện Neural Style Transfer (NST)

1. **Chọn Mô Hình CNN**:
    - Sử dụng một mạng nơ-ron tích chập đã được huấn luyện trước, như ResNet-50 hoặc VGG19, để trích xuất các đặc trưng của nội dung và phong cách. Các mô hình này đã được huấn luyện trên các tập dữ liệu lớn như WikiArt, có khả năng nhận diện các đặc trưng cao cấp của hình ảnh.
2. **Chuẩn Bị Hình Ảnh**:
    - **Hình ảnh nội dung**: Bức ảnh bạn muốn giữ nguyên nội dung.
    - **Hình ảnh phong cách**: Bức ảnh nghệ thuật mà bạn muốn áp dụng phong cách lên hình ảnh nội dung.
    - **Hình ảnh đầu ra**: Bức ảnh khởi tạo, có thể là hình ảnh nội dung, hình ảnh phong cách, hoặc một hình ảnh ngẫu nhiên.
3. **Tính Toán Hàm Mất Mát**:
    - **Hàm mất mát nội dung**: Sử dụng các lớp convolutional của mô hình CNN để tính toán hàm mất mát nội dung, đo lường sự khác biệt giữa các đặc trưng nội dung của ảnh đầu ra và ảnh gốc.
    - **Hàm mất mát phong cách**: Tính toán ma trận Gram cho các lớp convolutional của ảnh phong cách và ảnh đầu ra, sau đó so sánh để đo lường sự khác biệt giữa các phong cách.

4. **Tối Ưu Hóa**:
    - Sử dụng các thuật toán tối ưu hóa như Gradient Descent để cập nhật hình ảnh đầu ra nhằm giảm thiểu hàm mất mát tổng hợp, bao gồm cả nội dung và phong cách.
5. **Điều Chỉnh Các Siêu Tham Số**:
    - **Tỷ lệ phong cách (Style Weight)**: Điều chỉnh mức độ ảnh hưởng của phong cách so với nội dung trong hình ảnh đầu ra.
    - **Tỷ lệ nội dung (Content Weight)**: Điều chỉnh mức độ ảnh hưởng của nội dung so với phong cách.

Quy trình này bao gồm việc khởi tạo ảnh được tạo, tính toán hàm chi phí nội dung và phong cách, và sử dụng các thuật toán tối ưu hóa để cải thiện ảnh đầu ra. Sau đó, điều chỉnh các siêu tham số sẽ giúp cân bằng giữa nội dung và phong cách để đạt được kết quả mong muốn

Ưu Điểm và Hạn Chế

Ưu Điểm:

- **Tạo ra hình ảnh nghệ thuật độc đáo:** NST cho phép tạo ra các tác phẩm nghệ thuật mới bằng cách kết hợp phong cách của một bức tranh với nội dung của hình ảnh khác.
- **Tính sáng tạo và linh hoạt:** Có thể áp dụng phong cách của nhiều bức tranh nổi tiếng lên các hình ảnh khác nhau.

Hạn Chế:

- **Khó khăn trong việc duy trì chất lượng phong cách:** Đôi khi phong cách có thể không được truyền tải rõ ràng hoặc bị pha loãng trong hình ảnh đầu ra.

Ứng Dụng Thực Tiễn

- **Nghệ thuật và thiết kế:** Sử dụng NST để tạo ra các hình ảnh nghệ thuật mới cho các ứng dụng như quảng cáo, thiết kế đồ họa, và trang trí nội thất.
- **Nghiên cứu:** NST được sử dụng trong nghiên cứu để hiểu và phát triển các mô hình học sâu trong lĩnh vực xử lý hình ảnh và thị giác máy tính.

PHẦN IV: NỘI DUNG NGHIÊN CỨU

4.1. QUÁ TRÌNH THU THẬP VÀ XỬ LÝ DỮ LIỆU

4.1.1. Dataset Và Phương Pháp Thu Thập Dataset

Giới Thiệu Về Bộ Dữ Liệu WikiArt

Bộ dữ liệu được sử dụng trong đề tài là WikiArt, bao gồm các bức tranh phong cách nghệ thuật từ Wikipedia. Bộ dữ liệu này được thu thập từ nhiều nguồn và đã được sử dụng trong nhiều nghiên cứu và dự án liên quan đến học sâu và thị giác máy tính. Bộ dữ liệu có sẵn tại Kaggle.

Thông Tin Chi Tiết Về Bộ Dữ Liệu

- **Nguồn Gốc:** Bộ dữ liệu ban đầu được sử dụng cho việc huấn luyện mô hình StyleGAN2-conditional tại archive.org bởi Peter Baylies.
- **Nguồn Ban Đầu:** Các bức ảnh gốc được lấy từ kho dữ liệu tại GitHub của dự án ArtGAN.
- **Giấy Phép Sử Dụng:** Bộ dữ liệu WikiArt chỉ được sử dụng cho mục đích nghiên cứu phi thương mại. Các hình ảnh trong bộ dữ liệu này được thu thập từ WikiArt.org.
- **Số Lượng:** Bộ dữ liệu chứa 80,020 hình ảnh độc đáo từ 1,119 nghệ sĩ khác nhau, thuộc 27 phong cách nghệ thuật.

Phân Tích Thống Kê Bộ Dữ Liệu

- **Nghệ Sĩ:** Vincent van Gogh chiếm 2%, Nicholas Roerich chiếm 2%, và các nghệ sĩ khác chiếm 95%.
- **Phong Cách Nghệ Thuật:** Impressionism chiếm 16%, Realism chiếm 13%, và các phong cách khác chiếm 71%.
- **Kích Thước Ảnh:** Bộ dữ liệu có tổng cộng 80,020 hình ảnh với kích thước khác nhau. Các hình ảnh được chia thành các nhóm dựa trên chiều rộng và chiều cao như sau:
    - Chiều rộng từ 1381.00 - 2099.65 pixel: 71,868 ảnh
    - Chiều cao từ 1381.00 - 2200.35 pixel: 75,205 ảnh

4.1.2. Các Bước Tiền Xử Lý Dữ Liệu

Chuyển Đổi Kích Thước Ảnh:

Tất cả các bức ảnh được chuyển đổi về cùng một kích thước chuẩn để đảm bảo tính nhất quán trong quá trình huấn luyện mô hình.

Chuẩn Hóa Hình Ảnh

Sau khi chuyển đổi kích thước, các hình ảnh được chuẩn hóa để đưa về khoảng giá trị phù hợp với mô hình. Quy trình chuẩn hóa bao gồm việc điều chỉnh giá trị pixel của hình ảnh sao cho nằm trong khoảng [0, 1] và áp dụng các phép chuẩn hóa dựa trên giá trị trung bình và độ lệch chuẩn của bộ dữ liệu huấn luyện mô hình (như được sử dụng trong mô hình VGG19).

```
def load_image(img_path, max_size=128, shape=None):
    ''' Load in and transform an image, making sure the image
       is <= 400 pixels in the x-y dims.'''
    if "http" in img_path:
        response = requests.get(img_path)
        image = Image.open(BytesIO(response.content)).convert('RGB')
    else:
        image = Image.open(img_path).convert('RGB')

    # large images will slow down processing
    if max(image.size) > max_size:
        size = max_size
    else:
        size = max(image.size)

    if shape is not None:
        size = shape

    in_transform = transforms.Compose([
                        transforms.Resize(size),
                        transforms.ToTensor(),
                        transforms.Normalize((0.485, 0.456, 0.406),
                                             (0.229, 0.224, 0.225))])

    # discard the transparent, alpha channel (that's the :3) and add the batch dimension
    image = in_transform(image)[:3,:,:].unsqueeze(0)

    return image
```

Tạo Tensor và Thay Đổi Kích Thước

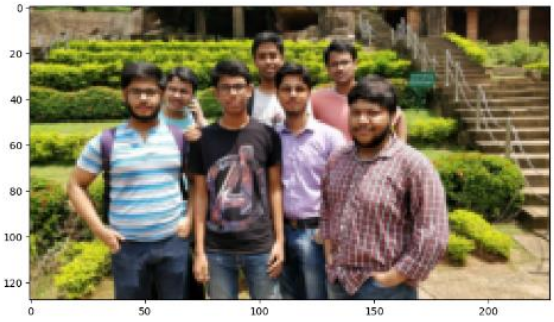
Các hình ảnh được chuyển đổi thành tensor để phù hợp với định dạng đầu vào của mô hình học sâu. Điều này bao gồm việc thêm một kích thước batch và loại bỏ các kênh alpha.

```python
# Function to load image and convert it to tensor
def load_image(image_path, max_size=400, shape=None):
    image = Image.open(image_path)
    if max_size is not None:
        size = max_size, int((max_size / float(image.size[0])) * float(image.size[1]))
        image.thumbnail(size, Image.LANCZOS)
    if shape is not None:
        image = image.resize(shape, Image.LANCZOS)
    image = transforms.ToTensor()(image).unsqueeze(0)
    return image

# Function to get all image files recursively from a directory
def get_all_image_files(directory):
    image_files = []
    for root, _, files in os.walk(directory):
        for file in files:
            if file.endswith('.jpg'):
                image_files.append(os.path.join(root, file))
    return image_files

# Paths to the content and style image directories
content_path = '/kaggle/input/images-for-style-transfer/Data/TestCases/'
style_path = '/kaggle/input/wikiart/'

# Get list of all image files in the directories
content_files = get_all_image_files(content_path)
style_files = get_all_image_files(style_path)

# Select random content and style images
content_image_path = random.choice(content_files)
style_image_path = random.choice(style_files)

# Load content and style images
content = load_image(content_image_path).to(device)
style = load_image(style_image_path, shape=content.shape[-2:]).to(device)

print(f"Content image path: {content_image_path}")
print(f"Style image path: {style_image_path}")
```

- **load_image**: Chức năng này tải hình ảnh từ đường dẫn, thay đổi kích thước nếu cần và chuyển đổi thành tensor.
- **get_all_image_files**: Hàm này duyệt qua thư mục và tìm tất cả các tệp hình ảnh với định dạng .jpg.
- **Chọn hình ảnh ngẫu nhiên**: Sử dụng random.choice để chọn ngẫu nhiên một hình ảnh từ danh sách các tệp hình ảnh nội dung và phong cách.
- **Tải hình ảnh và chuyển đổi thành tensor**: Hình ảnh được tải và chuyển đổi thành tensor, và sau đó được đưa vào thiết bị mà bạn đang sử dụng (CPU hoặc GPU).

Tải hình ảnh nội dung và hình ảnh phong cách.

```python
def im_convert(tensor):
    """ Hiển thị một tensor dưới dạng hình ảnh. """

    # Chuyển tensor sang CPU, tạo một bản sao để tránh việc sửa đổi bản gốc, và tách nó ra khỏi đồ thị tính toán
    image = tensor.to("cpu").clone().detach()

    # Chuyển tensor thành mảng NumPy và loại bỏ bất kỳ chiều kích thước dư thừa nào
    image = image.numpy().squeeze()

    # Hoán vị các chiều từ (C, H, W) sang (H, W, C) để phù hợp với định dạng hình ảnh
    image = image.transpose(1, 2, 0)

    # Khôi phục lại quá trình chuẩn hóa đã áp dụng trong quá trình tiền xử lý
    image = image * np.array((0.229, 0.224, 0.225)) + np.array((0.485, 0.456, 0.406))

    # Giới hạn giá trị của các pixel trong khoảng từ 0 đến 1
    image = image.clip(0, 1)

    return image
```



4.2. GIẢI THÍCH VIỆC ÁP DỤNG PHƯƠNG PHÁP HỌC SÂU

Giải Thích Nguyên Tắc Hoạt Động Của Các Phương Pháp Học Sâu Được Sử Dụng

Trong nghiên cứu này, phương pháp chính được sử dụng là Neural Style Transfer (NST) dựa trên mô hình VGG19 và ResNet50. Mô hình này đã được huấn luyện trên tập dữ liệu ImageNet và được sử dụng để trích xuất các đặc trưng nội dung và phong cách từ các hình ảnh.

Mô Hình VGG19: Được sử dụng để trích xuất đặc trưng từ các lớp convolutional. Các lớp này giúp xác định các đặc trưng nội dung và phong cách của hình ảnh.

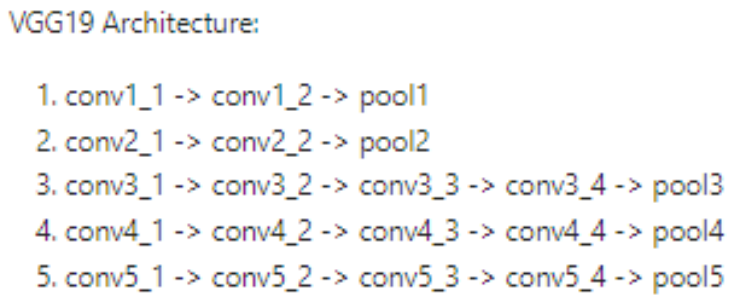
```python
# chuyển model sang gpu
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
# chúng ta chỉ lấy vgg19 fearture mà không dùng classifier vì đây không phải bài toán classifi
vgg = models.vgg19(pretrained=True).features

# dùng hàm freeze để các trọng số của model sẽ không được cập nhật
for param in vgg.parameters():
    param.requires_grad_(False)
vgg.to(device)
```

- **Ma Trận Gram**: Được sử dụng để đo lường sự tương đồng giữa các đặc trưng phong cách của hình ảnh. Ma trận Gram giúp tính toán mức độ tương đồng giữa các đặc trưng phong cách và so sánh với hình ảnh phong cách gốc.

```python
def get_features(image, model, layers=None):
    if layers is None:
        layers = {'0': 'conv1_1',
                  '5': 'conv2_1',
                  '10': 'conv3_1',
                  '19': 'conv4_1',
                  '21': 'conv4_2',  # content representation
                  '28': 'conv5_1'}
    features = {}
    x = image
    for name, layer in model._modules.items():
        x = layer(x)
        if name in layers:
            features[layers[name]] = x
    return features

def gram_matrix(tensor):
    _, d, h, w = tensor.size()
    tensor = tensor.view(d, h * w)
    gram = torch.mm(tensor, tensor.t())
    return gram

# Calculate features and gram matrices
content_features = get_features(content, vgg)
style_features = get_features(style, vgg)
style_grams = {layer: gram_matrix(style_features[layer]) for layer in style_features}
```

  - **Lớp conv1_1 (Lớp 0)**: Đặc trưng cấp thấp, như cạnh và góc.
    - Lớp đầu tiên của mạng, biểu diễn các đặc trưng cơ bản nhất như cạnh, đường thẳng
  - **Lớp conv2_1 (Lớp 5)**: Đặc trưng kết cấu phức tạp hơn.
    - Lớp đầu tiên của khối tích chập thứ hai, biểu diễn các đặc trưng phức tạp hơn như các hình dạng đơn giản.

- **Lớp conv3_1 (Lớp 10)**: Đặc trưng kết cấu và hình dạng.
    - Lớp đầu tiên của khối tích chập thứ ba, biểu diễn các đặc trưng phức tạp hơn nữa
- **Lớp conv4_1 (Lớp 19)**: Đặc trưng cấp cao hơn, gần với nội dung.
    - Lớp đầu tiên của khối tích chập thứ tư, biểu diễn các đặc trưng rất phức tạp và có ý nghĩa cao về mặt ngữ nghĩa
- **Lớp conv4_2 (Lớp 21) - Content Representation**: Đại diện cho nội dung.
    - Lớp này được chọn để biểu diễn nội dung của hình ảnh. Nó chứa thông tin chi tiết và có ý nghĩa cao về mặt nội dung, giúp giữ lại cấu trúc tổng quát của hình ảnh gốc.
- **Lớp conv5_1 (Lớp 28)**: Đặc trưng cấp cao nhất, nắm bắt cấu trúc và bố cục tổng thể.
    - Lớp đầu tiên của khối tích chập cuối cùng, biểu diễn các đặc trưng phức tạp nhất



Mô Hình ResNet50: ResNet50 là một kiến trúc mạng nơ-ron sâu được thiết kế với các lớp residual (còn gọi là lớp bỏ qua) giúp cải thiện khả năng huấn luyện của các mô hình rất sâu. Trong nghiên cứu này, ResNet50 được sử dụng để so sánh với VGG19 trong việc trích xuất các đặc trưng nội dung và phong cách từ hình ảnh. Việc sử dụng ResNet50 có thể mang lại các lợi ích như cải thiện độ chính xác và giảm thiểu vấn đề gradient biến mất, nhờ vào các khối residual giúp duy trì thông tin qua các lớp sâu hơn của mạng.

```python
# get the "features" portion of ResNet50 (we will not need the "classifier" portion)
resnet = models.resnet50(pretrained=True)
resnet_features = nn.Sequential(*list(resnet.children())[:-2])

# freeze all ResNet parameters since we're only optimizing the target image
for param in resnet_features.parameters():
    param.requires_grad_(False)
```

```
Downloading: "https://download.pytorch.org/models/resnet50-19c8e357.pth" to /root/.cache/torch/hub/checkpoints/resnet50-19c8e357.pth
100%  97.8M/97.8M [00:00<00:00, 214MB/s]
```

+ Code + Markdown

```python
# move the model to GPU, if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

resnet.to(device)
```

```python
def get_features(image, model, layers=None):
    if layers is None:
        layers = {
            '0': 'conv1',
            '1': 'layer1',
            '2': 'layer2',
            '3': 'layer3',
            '4': 'layer4'
        }
    features = {}
    x = image
    for name, layer in model._modules.items():
        x = layer(x)
        if name in layers:
            features[layers[name]] = x
        if name == 'avgpool':
            break
    return features
```

Gram Matrix

```python
def gram_matrix(tensor):
    _, d, h, w = tensor.size()
    tensor = tensor.view(d, h * w)
    gram = torch.mm(tensor, tensor.t())
    return gram
```

- Conv1: Lớp tích chập đầu tiên.
- Layer1 đến Layer4: Các nhóm lớp tích chập tiếp theo, mỗi nhóm chứa nhiều residual blocks.
- AvgPool: Lớp pooling trung bình toàn cục.
- FC (Fully Connected): Lớp kết nối đầy đủ cuối cùng (thường dùng cho phân loại).
- Khởi Tạo Từ Điển features:

    - Từ điển này sẽ chứa các đặc trưng được trích xuất từ các lớp đã chọn.
- Duyệt Qua Các Lớp Trong Mô Hình:
    - Sử dụng vòng lặp for name, layer in model._modules.items() để duyệt qua từng lớp trong mô hình.
    - x = layer(x): Áp dụng lớp hiện tại lên dữ liệu đầu vào x.
    - Lưu Đặc Trưng: Nếu tên lớp (name) nằm trong từ điển layers, lưu đầu ra của lớp đó vào từ điển features với khóa là tên đặc trưng tương ứng.
    - Dừng Khi Gặp Lớp avgpool: Dừng vòng lặp nếu lớp hiện tại là avgpool.
- Trả Về Từ Điển features:
    - Trả lại từ điển chứa các đặc trưng đã trích xuất.
- Kích Thước Tensor:
    - _, d, h, w = tensor.size(): Lấy kích thước của tensor, trong đó d là số kênh (channels), h là chiều cao, và w là chiều rộng.
- Chuyển Đổi Kích Thước:
    - tensor = tensor.view(d, h * w): Chuyển đổi tensor thành một ma trận 2D có kích thước (d, h * w). Điều này có nghĩa là chúng ta làm phẳng chiều không gian (height x width) và giữ nguyên số kênh.
- Tính Ma Trận Gram:
    - gram = torch.mm(tensor, tensor.t()): Tính tích của tensor và chuyển vị của nó. Ma trận Gram này biểu diễn các mối quan hệ tương quan giữa các kênh của tensor.

Mô Tả Cách Thức Áp Dụng Các Phương Pháp Học Sâu Vào Đề Tài

- **Tính Toán Đặc Trưng**: Chúng tôi sử dụng mô hình VGG19 và ResNet50 để trích xuất các đặc trưng nội dung và phong cách từ hình ảnh nội dung và phong cách. Cụ thể, các đặc trưng nội dung và phong cách được lấy từ các lớp convolutional của các mô hình này. Các đặc trưng này sau đó được sử

dụng để tính toán hàm mất mát, giúp đánh giá mức độ phù hợp giữa hình ảnh mục tiêu và hình ảnh gốc.

VGG19

```
# get content and style features only once before training
content_features = get_features(content, vgg)
style_features = get_features(style, vgg)
```

ResNet50

```
# get content and style features only once before training
content_features = get_features(content, resnet_features)
style_features = get_features(style, resnet_features)
```

- content_features và style_features:
    - Chúng ta sử dụng hàm get_features để trích xuất các đặc trưng từ hình ảnh nội dung (content) và hình ảnh phong cách (style) sử dụng mô hình VGG19 và ResNet50 đã được huấn luyện trước (vgg, resnet_features).
    - Hàm get_features: Hàm này đi qua từng lớp của mô hình VGG19 và ResNet50 và lưu lại đầu ra của các lớp mà chúng ta quan tâm vào một từ điển. Các lớp được chọn sẽ chứa các đặc trưng ở nhiều cấp độ khác nhau của hình ảnh.
- **Tính Toán Hàm Mất Mát**:
    - Hàm mất mát trong Neural Style Transfer bao gồm hai phần chính:
        - Mất Mát Nội Dung (Content Loss): Đo lường sự khác biệt giữa các đặc trưng nội dung của hình ảnh mục tiêu và hình ảnh gốc. Điều này giúp đảm bảo rằng hình ảnh đầu ra vẫn giữ được nội dung chính của hình ảnh gốc.
        - Mất Mát Phong Cách (Style Loss): Đo lường sự khác biệt giữa các ma trận Gram của các lớp phong cách của hình ảnh mục tiêu và hình ảnh phong cách. Ma trận Gram giúp tính toán mức độ tương đồng giữa các đặc trưng phong cách và so sánh với hình ảnh phong cách gốc.

```
# calculate the gram matrices for each layer of our style representation
style_grams = {layer: gram_matrix(style_features[layer]) for layer in style_features}
```

- **style_grams:**
    - Đây là một từ điển chứa ma trận Gram cho mỗi lớp phong cách.
    - **Ma trận Gram:** Được tính từ các đặc trưng phong cách của mỗi lớp, biểu diễn các mối quan hệ tương quan giữa các kênh (channels) trong một lớp.
    - **Hàm gram_matrix:** Hàm này nhận đầu vào là một tensor (đặc trưng của một lớp) và trả về ma trận Gram của nó.

- **Tối Ưu Hóa**: Thuật toán tối ưu hóa Gradient Descent được sử dụng để giảm thiểu tổng hàm mất mát. Trong quá trình tối ưu hóa, hình ảnh đầu ra (được khởi tạo là bản sao của hình ảnh nội dung) sẽ được cập nhật để cân bằng giữa việc giữ nội dung và áp dụng phong cách. Các tham số điều chỉnh sẽ giúp điều chỉnh ảnh hưởng của từng yếu tố trong hình ảnh đầu ra.

```
# create a third "target" image and prep it for change
# it is a good idea to start off with the target as a copy of our *content* image
# then iteratively change its style
target = content.clone().requires_grad_(True).to(device)
```

- **target:**
    - Đây là hình ảnh mà chúng ta sẽ thay đổi dần dần trong quá trình huấn luyện để nó mang phong cách của hình ảnh phong cách nhưng vẫn giữ lại nội dung của hình ảnh nội dung.
    - **Khởi tạo:** Ban đầu, hình ảnh "mục tiêu" được khởi tạo bằng cách sao chép từ hình ảnh nội dung (content.clone()).
    - **requires_grad_(True):** Đặt cờ này để cho phép tính toán gradient đối với target trong quá trình tối ưu hóa. Điều này có nghĩa là chúng ta sẽ thay đổi target trong quá trình huấn luyện để đạt được kết quả mong muốn.
    - **to(device):** Chuyển target sang thiết bị tính toán (CPU hoặc GPU) mà chương trình đang sử dụng.

Giải Thích Các Tham Số Điều Chỉnh Cho Các Phương Pháp Học Sâu

- **Tỷ Lệ Phong Cách (Style Weight)**: Tỷ lệ này điều chỉnh mức độ ảnh hưởng của phong cách so với nội dung trong hình ảnh đầu ra. Một giá trị lớn hơn sẽ làm nổi bật phong cách hơn trong hình ảnh đầu ra.

VGG19

```
style_weights = {'conv1_1': 1,
                 'conv2_1': 0.75,
                 'conv3_1': 0.2,
                 'conv4_1': 0.2,
                 'conv5_1': 0.2}
```

ResNet50

```
style_weights = {'conv1': 1,
                 'layer1': 0.75,
                 'layer2': 0.2,
                 'layer3': 0.2,
                 'layer4': 0.2}

content_layer = 'layer4'  # This is similar to conv4_2 in VGG19
```

- **Tỷ Lệ Nội Dung (Content Weight)**: Tỷ lệ này điều chỉnh mức độ ảnh hưởng của nội dung so với phong cách trong hình ảnh đầu ra. Một giá trị cao hơn giúp hình ảnh đầu ra giữ được nội dung gốc nhiều hơn.

```
content_weight = 1   # alpha
style_weight = 1e3   # beta
```

4.3. TRÌNH BÀY KẾT QUẢ THÍ NGHIỆM VÀ ĐÁNH GIÁ HIỆU SUẤT MÔ HÌNH

Nêu Rõ Các Chỉ Số Đánh Giá Hiệu Suất Mô Hình VGG19 Sử Dụng

- **Mất Mát Nội Dung**: Mất mát nội dung đo lường sự khác biệt giữa các đặc trưng nội dung của hình ảnh mục tiêu và hình ảnh gốc. Sự khác biệt này được tính bằng cách so sánh các đặc trưng nội dung tại lớp conv4_2 của mô hình VGG19.

```
# get the features from your target image
target_features = get_features(target, vgg)

# the content loss
content_loss = torch.mean((target_features['conv4_2'] - content_features['conv4_2'])**2)
```

- **Mất Mát Phong Cách**: Mất mát phong cách đo lường sự khác biệt giữa các ma trận Gram của hình ảnh mục tiêu và hình ảnh phong cách. Mỗi lớp trong mô hình VGG19 đóng góp vào việc tính toán mất mát phong cách. Mất mát phong cách được tính cho từng lớp và sau đó cộng dồn lại.

```
# the style loss
# initialize the style loss to 0
style_loss = 0
# then add to it for each layer's gram matrix loss
for layer in style_weights:
    # get the "target" style representation for the layer
    target_feature = target_features[layer]
    target_gram = gram_matrix(target_feature)
    _, d, h, w = target_feature.shape
    # get the "style" style representation
    style_gram = style_grams[layer]
    # the style loss for one layer, weighted appropriately
    layer_style_loss = style_weights[layer] * torch.mean((target_gram - style_gram)**2)
    # add to the style loss
    style_loss += layer_style_loss / (d * h * w)
```

- **Tổng Hàm Mất Mát**: Tổng hàm mất mát là sự kết hợp của mất mát nội dung và phong cách, với các trọng số tương ứng. Tổng hàm mất mát được sử dụng để tối ưu hóa hình ảnh đầu ra.

```
# calculate the *total* loss
total_loss = content_weight * content_loss + style_weight * style_loss
```

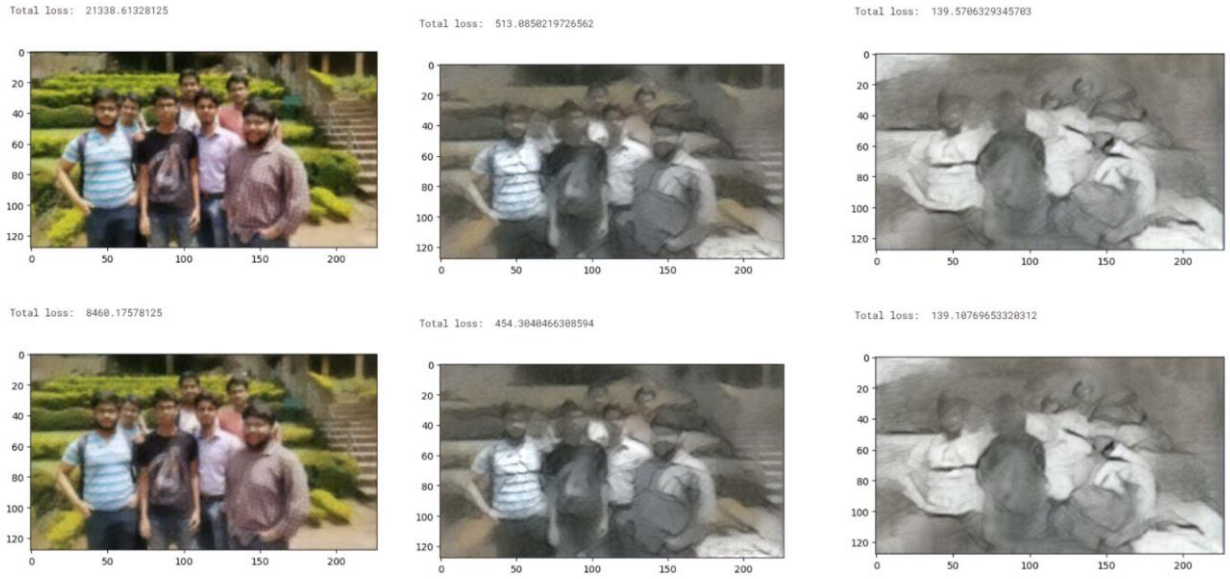
Trình Bày Kết Quả Thí Nghiệm Trên Tập Dữ Liệu Huấn Luyện Và Tập Dữ Liệu Kiểm Tra

Kết quả huấn luyện được theo dõi qua các vòng lặp của thuật toán tối ưu hóa. Hình ảnh đầu ra được hiển thị sau mỗi số lượng vòng lặp cụ thể, giúp đánh giá sự thay đổi của hình ảnh qua các bước huấn luyện.

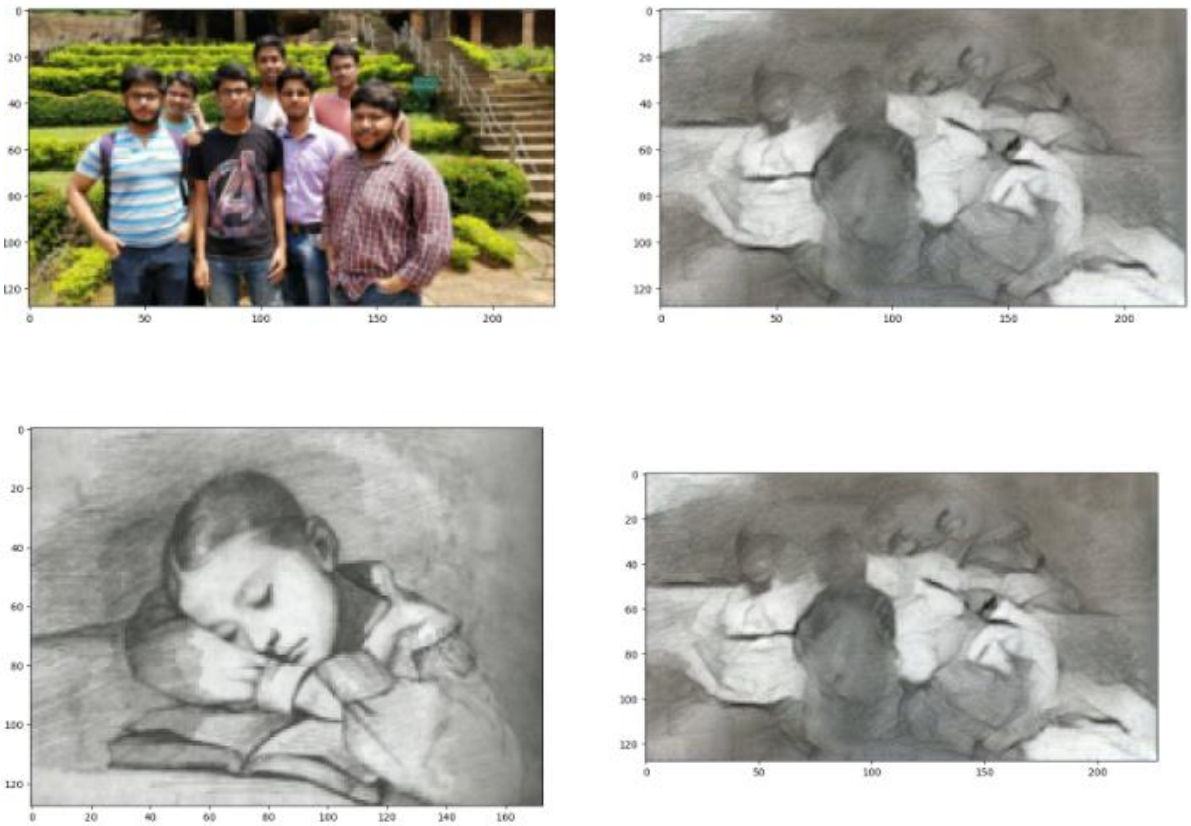
Trình Bày Kết Quả Thí Nghiệm Trên Tập Dữ Liệu Huấn Luyện Và Tập Dữ Liệu Kiểm Tra.

Kết quả huấn luyện được theo dõi qua các vòng lặp của thuật toán tối ưu hóa. Hình ảnh đầu ra được hiển thị sau mỗi số lượng vòng lặp cụ thể, giúp đánh giá sự thay đổi của hình ảnh qua các bước huấn luyện. Dưới đây là mã code để thực hiện điều này:

```python
# for displaying the target image, intermittently
show_every = 500

# iteration hyperparameters
optimizer = optim.Adam([target], lr=0.001)
steps = 20000  # decide how many iterations to update your image (20000)

for ii in range(1, steps+1):

    # get the features from your target image
    target_features = get_features(target, vgg)

    # the content loss
    content_loss = torch.mean((target_features['conv4_2'] - content_features['conv4_2'])**2)

    # the style loss
    # initialize the style loss to 0
    style_loss = 0
    # then add to it for each layer's gram matrix loss
    for layer in style_weights:
        # get the "target" style representation for the layer
        target_feature = target_features[layer]
        target_gram = gram_matrix(target_feature)
        _, d, h, w = target_feature.shape
        # get the "style" style representation
        style_gram = style_grams[layer]
        # the style loss for one layer, weighted appropriately
        layer_style_loss = style_weights[layer] * torch.mean((target_gram - style_gram)**2)
        # add to the style loss
        style_loss += layer_style_loss / (d * h * w)

    # calculate the *total* loss
    total_loss = content_weight * content_loss + style_weight * style_loss

    # update your target image
    optimizer.zero_grad()
    total_loss.backward()
    optimizer.step()

    # display intermediate images and print the loss
    if  ii % show_every == 0:
        print('Total loss: ', total_loss.item())
        plt.imshow(im_convert(target))
        plt.show()
```



Kiểm tra kết quả cuối cùng

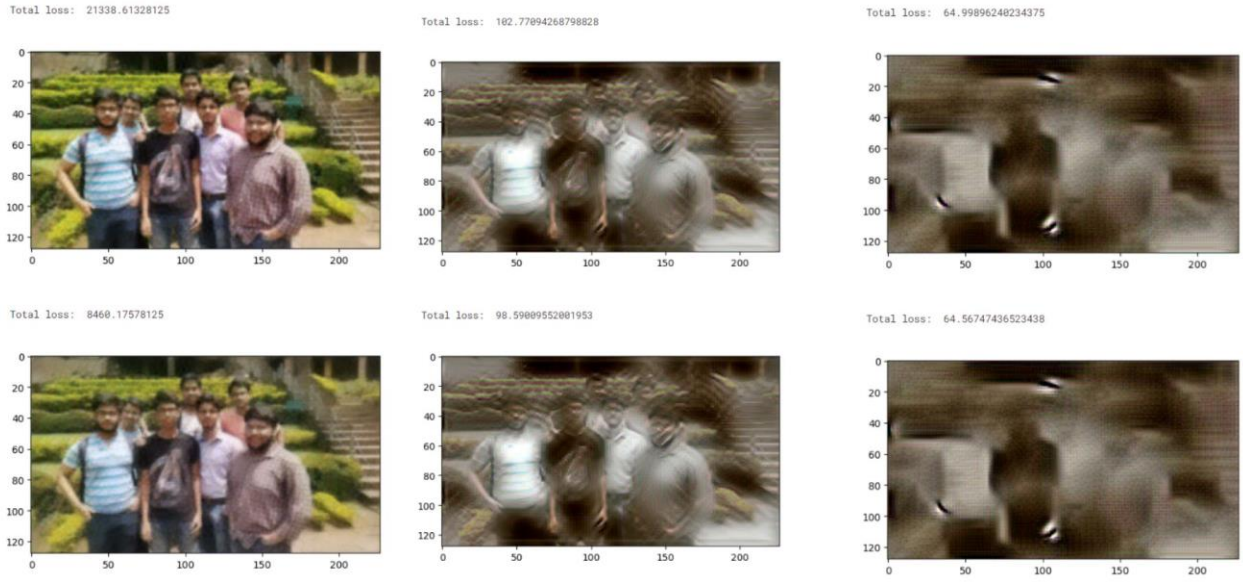


Nhận Xét Mô Hình VGG19:

1. **Quá Trình Giảm Loss:**
    - Hình ảnh đầu tiên (bên trái) có tổng loss khá cao (21338.61328125), nhưng sau một số bước huấn luyện, loss đã giảm đáng kể (xuống còn khoảng 139.10). Điều này cho thấy mô hình VGG19 có khả năng tối ưu hóa tốt qua từng epoch để giảm thiểu sự sai lệch giữa nội dung gốc và phong cách mong muốn.
2. **Kết Quả Tạo Ảnh:**
    - Trong hình ảnh bên phải (kiểm tra kết quả cuối cùng), các hình ảnh đã chuyển đổi rõ ràng từ ảnh gốc sang phong cách vẽ tay với những nét đặc trưng như đổ bóng và nét vẽ mềm mại. Điều này chứng tỏ mô hình VGG19 đã thành công trong việc tách biệt và áp dụng phong cách lên hình ảnh gốc một cách hiệu quả.
    - Tuy nhiên, vẫn còn một số chi tiết nhỏ bị mất hoặc không rõ nét trong quá trình chuyển đổi, có thể do việc tối ưu loss chưa đạt đến mức hoàn hảo.
3. **Chất Lượng và Độ Sắc Nét:**
    - Các hình ảnh sau quá trình transfer không hoàn toàn sắc nét như ảnh gốc, nhưng đã nắm bắt được tinh thần và phong cách nghệ thuật từ hình ảnh tham chiếu. Đây là kết quả khá phổ biến trong NST khi mô hình tập trung vào việc bảo toàn phong cách hơn là chi tiết nội dung.
4. **Đánh Giá Tổng Quan:**
    - Mô hình VGG19 đã chứng minh hiệu quả trong việc áp dụng Neural Style Transfer, đặc biệt khi giảm được loss qua nhiều bước và tạo ra hình ảnh có phong cách nghệ thuật rõ ràng. Tuy nhiên, để đạt kết quả tối ưu, có thể cần thêm các kỹ thuật tinh chỉnh hoặc thời gian huấn luyện lâu hơn để đạt được độ sắc nét và bảo toàn nội dung tốt hơn.

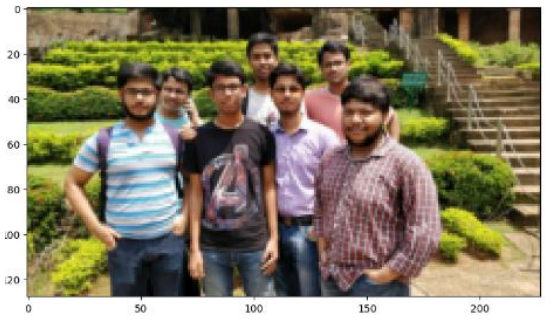
Nêu Rõ Các Chỉ Số Đánh Giá Hiệu Suất Mô Hình ResNet50 Sử Dụng

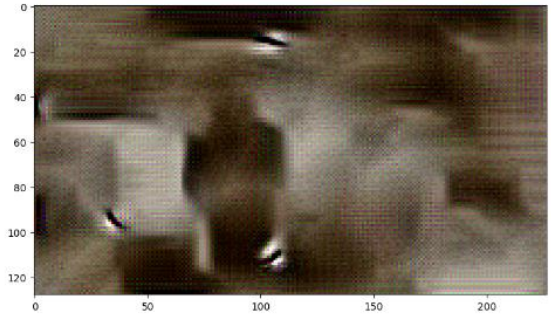
```python
show_every = 500

optimizer = optim.Adam([target], lr=0.001)
steps = 20000

for ii in range(1, steps+1):

    # get the features from your target image
    target_features = get_features(target, resnet_features)

    # the content loss
    content_loss = torch.mean((target_features[content_layer] - content_features[content_layer])**2)

    # the style loss
    style_loss = 0
    for layer in style_weights:
        target_feature = target_features[layer]
        target_gram = gram_matrix(target_feature)
        _, d, h, w = target_feature.shape
        style_gram = style_grams[layer]
        layer_style_loss = style_weights[layer] * torch.mean((target_gram - style_gram)**2)
        style_loss += layer_style_loss / (d * h * w)

    total_loss = content_weight * content_loss + style_weight * style_loss

    optimizer.zero_grad()
    total_loss.backward()
    optimizer.step()

    if ii % show_every == 0:
        print('Total loss: ', total_loss.item())
        plt.imshow(im_convert(target))
        plt.show()
```



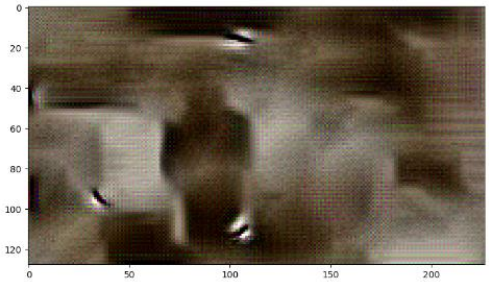
Kiểm tra kết quả cuối cùng

```
<matplotlib.image.AxesImage at 0x785284b8a7d0>
```





```
<matplotlib.image.AxesImage at 0x78528492f190>
```



Nhận Xét Mô Hình ResNet50:

1. **Quá Trình Giảm Loss:**
    - Cũng giống như VGG19, mô hình ResNet50 bắt đầu với tổng loss khá cao (21338.61328125) và sau đó giảm xuống chỉ còn khoảng 64.56. Điều này cho thấy ResNet50 cũng có khả năng tối ưu hóa tốt, nhưng tốc độ giảm loss có thể nhanh hơn so với VGG19.
2. **Kết Quả Tạo Ảnh:**
    - Hình ảnh sau quá trình transfer với ResNet50 có sự biến dạng mạnh hơn so với kết quả của VGG19. Đặc biệt, ảnh phong cách được tạo ra không rõ nét và có sự mờ nhòe đáng kể.

- Điều này có thể là do ResNet50 được thiết kế với kiến trúc phức tạp hơn, dẫn đến việc mô hình khó khăn hơn trong việc tách biệt rõ ràng giữa nội dung và phong cách.

3. **Chất Lượng và Độ Sắc Nét:**
   - Các hình ảnh được tạo ra từ mô hình ResNet50 thiếu sự rõ nét và chi tiết so với VGG19. Hình ảnh có xu hướng bị nhòe, và phong cách nghệ thuật không được thể hiện rõ ràng. Điều này cho thấy ResNet50 có thể không phù hợp bằng VGG19 cho bài toán Neural Style Transfer, ít nhất là trong trường hợp cụ thể này.
4. **Đánh Giá Tổng Quan:**
   - Mô hình ResNet50 mặc dù có khả năng giảm loss nhanh chóng, nhưng chất lượng ảnh sau khi transfer không đạt yêu cầu, với hình ảnh bị nhòe và phong cách không rõ nét. Điều này có thể là do ResNet50 không được tối ưu hóa tốt cho bài toán này hoặc cần có sự tinh chỉnh thêm về hyperparameters và kỹ thuật transfer.

KẾT QUẢ TỔNG QUAN:

1. **So Sánh Giữa VGG19 và ResNet50:**
   - **<u>Về Quá Trình Giảm Loss:</u>**
     - Cả hai mô hình VGG19 và ResNet50 đều cho thấy khả năng giảm loss đáng kể qua quá trình huấn luyện, từ mức rất cao xuống mức thấp hơn nhiều. Tuy nhiên, ResNet50 dường như giảm loss nhanh hơn VGG19.
   - **<u>Chất Lượng Hình Ảnh:</u>**
     - VGG19: Mô hình này đã tạo ra hình ảnh với phong cách nghệ thuật rõ ràng và giữ được nhiều chi tiết từ ảnh gốc. Phong cách được áp dụng một cách mềm mại và không làm mất đi quá nhiều nội dung quan trọng.
     - ResNet50: Trong khi ResNet50 giảm loss nhanh, kết quả hình ảnh cuối cùng lại có chất lượng kém hơn so với VGG19. Hình

ảnh bị nhòe và phong cách nghệ thuật không được thể hiện rõ ràng, làm mất đi nhiều chi tiết quan trọng.

- **Phù Hợp của Mô Hình:**
  - VGG19: Tỏ ra phù hợp hơn cho bài toán Neural Style Transfer nhờ vào khả năng bảo toàn nội dung và áp dụng phong cách một cách rõ ràng.
  - ResNet50: Với kiến trúc phức tạp hơn, ResNet50 có thể phù hợp cho các bài toán khác nhưng trong trường hợp này, nó không tạo ra kết quả tốt bằng VGG19.

2. **Đánh Giá Tổng Thể:**
   - Dựa trên kết quả thực nghiệm, mô hình VGG19 đã chứng tỏ khả năng vượt trội trong việc tạo ảnh phong cách nghệ thuật với chất lượng cao hơn so với ResNet50. Mặc dù ResNet50 có khả năng giảm loss nhanh, chất lượng hình ảnh không đạt yêu cầu, đặc biệt là khi so sánh với VGG19.
   - Do đó, trong ngữ cảnh của bài toán này, VGG19 là mô hình được khuyến nghị sử dụng để thực hiện Neural Style Transfer, nhờ vào sự cân bằng tốt giữa việc bảo toàn nội dung gốc và áp dụng phong cách nghệ thuật.

4.4. SỬ DỤNG MÔ HÌNH VGG19 Ở HÌNH ẢNH THỰC TẾ

Khi áp dụng mô hình VGG19 cho hình ảnh thực tế, như hình ảnh cá nhân hoặc các tác phẩm nghệ thuật cụ thể, một số điều chỉnh về tham số huấn luyện đã được thực hiện để tối ưu hóa quá trình tạo ảnh phong cách

```
# Load content and style images
content = load_image('/kaggle/input/hinh-van-lang/hinh vanlang.jpg').to(device)
style = load_image('/kaggle/input/vangoh/Van_Gogh_-_Starry_Night_-_Google_Art_Project.jpg').to(device)
```

- **Ảnh Trường Đại Học Văn Lang và Bức Tranh Starry Night của Van Gogh:**
  - Đối với ảnh này, mô hình đã được huấn luyện trong 20000 bước với learning rate (lr) = 0.001. Mục tiêu là để đảm bảo rằng các chi tiết nội

dung từ hình ảnh gốc (ảnh Văn Lang) được bảo toàn tốt nhất trong khi áp dụng phong cách nghệ thuật của bức tranh "Starry Night" của Van Gogh.

- **Áp Dụng cho Ảnh Cá Nhân:**
    - Khi chuyển sang làm việc với ảnh cá nhân, số bước huấn luyện được giảm xuống còn 5000 bước, và learning rate được tăng lên 0.003. Việc này giúp mô hình hội tụ nhanh hơn, đồng thời vẫn đảm bảo rằng phong cách nghệ thuật được áp dụng một cách hiệu quả mà không làm mất đi nhiều chi tiết của ảnh gốc.

```python
def im_convert(tensor):
    """ Display a tensor as an image. """

    image = tensor.to("cpu").clone().detach()
    image = image.numpy().squeeze()
    image = image.transpose(1,2,0)
    image = image * np.array((0.229, 0.224, 0.225)) + np.array((0.485, 0.456, 0.406))
    image = image.clip(0, 1)

    return image
```

Trực quan hóa kết quả đầu ra

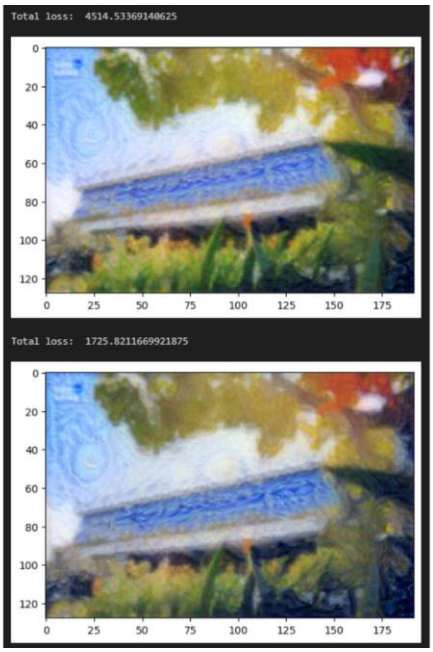
```python
# for displaying the target image, intermittently
show_every = 500

# iteration hyperparameters
optimizer = optim.Adam([target], lr=0.003)
steps = 5001  # decide how many iterations to update your image (5000)

for ii in range(1, steps+1):

    # get the features from your target image
    target_features = get_features(target, vgg)

    # the content loss
    content_loss = torch.mean((target_features['conv4_2'] - content_features['conv4_2'])**2)

    # the style loss
    # initialize the style loss to 0
    style_loss = 0
    # then add to it for each layer's gram matrix loss
    for layer in style_weights:
        # get the "target" style representation for the layer
        target_feature = target_features[layer]
        target_gram = gram_matrix(target_feature)
        _, d, h, w = target_feature.shape
        # get the "style" style representation
        style_gram = style_grams[layer]
        # the style loss for one layer, weighted appropriately
        layer_style_loss = style_weights[layer] * torch.mean((target_gram - style_gram)**2)
        # add to the style loss
        style_loss += layer_style_loss / (d * h * w)

    # calculate the *total* loss
    total_loss = content_weight * content_loss + style_weight * style_loss

    # update your target image
    optimizer.zero_grad()
    total_loss.backward()
    optimizer.step()

    # display intermediate images and print the loss
    if  ii % show_every == 0:
        print('Total loss: ', total_loss.item())
        plt.imshow(im_convert(target))
        plt.show()
```

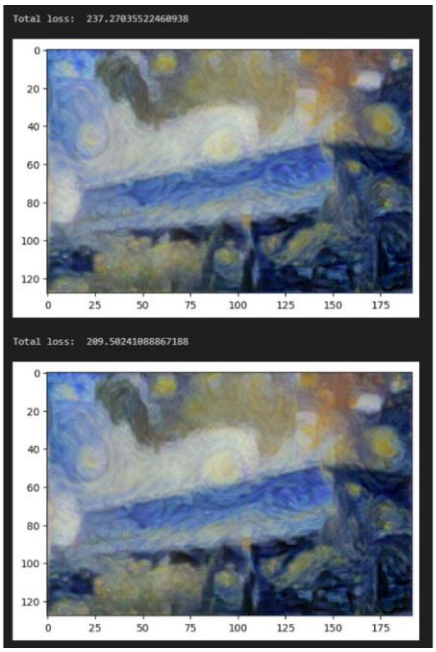
Chi Tiết Huấn Luyện:

- **Optimizer:**
    - Adam optimizer đã được sử dụng với learning rate = 0.003 cho ảnh cá nhân. Điều này giúp điều chỉnh mục tiêu (target image) trong quá trình tối ưu hóa, nhằm đạt được sự kết hợp hoàn hảo giữa nội dung và phong cách.
- **Loss Function:**
    - **Content Loss:** Được tính toán bằng sự khác biệt giữa các đặc trưng của ảnh mục tiêu (target features) và ảnh gốc (content features) tại lớp conv4_2 của VGG19.

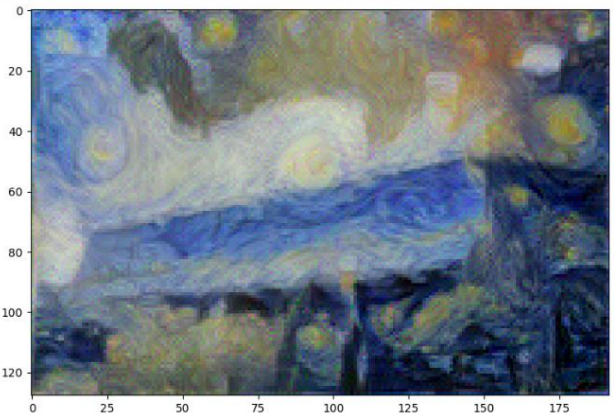
- **Style Loss:** Được tính dựa trên sự khác biệt giữa các Gram matrix của ảnh mục tiêu và ảnh phong cách tại các lớp khác nhau, mỗi lớp được gán một trọng số cụ thể (style weights).
- **Total Loss:** Là tổng hợp của content loss và style loss, được cân bằng bằng cách sử dụng content weight và style weight.

- **Hiển Thị Kết Quả:**
  - Trong quá trình huấn luyện, hình ảnh trung gian được hiển thị sau mỗi 500 bước để theo dõi tiến độ và hiệu quả của quá trình chuyển đổi phong cách.





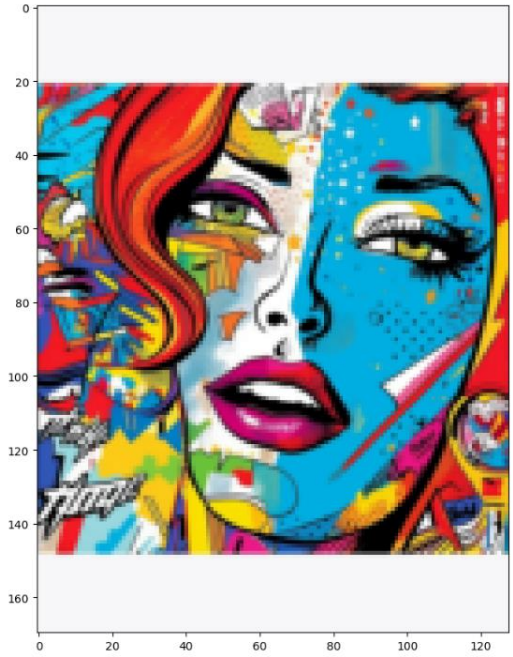
Kết quả hiển thị nội dung và hình ảnh cuối cùng, mục tiêu:



Chúng ta sẽ thử trải nghiệm 2 góc nhìn khác tại 2 vị trí đẹp của Trường Đại Học Văn Lang với 2 tấm hình nghệ thuật ngẫu nhiên

```
# Load content and style images
content = load_image('/kaggle/input/artstyle1/vanlang2.jpg').to(device)
style = load_image('/kaggle/input/artstyle1/comicpopart.jpg').to(device)
# helper function for un-normalizing an image
# and converting it from a Tensor image to a NumPy image for display
def im_convert(tensor):
    """ Display a tensor as an image. """

    image = tensor.to("cpu").clone().detach()
    image = image.numpy().squeeze()
    image = image.transpose(1,2,0)
    image = image * np.array((0.229, 0.224, 0.225)) + np.array((0.485, 0.456, 0.406))
    image = image.clip(0, 1)

    return image
# display the images
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))
# content and style ims side-by-side
ax1.imshow(im_convert(content))
ax2.imshow(im_convert(style))
```

```
# Load content and style images
content = load_image('/kaggle/input/artstyle1/vanlang3.jpg').to(device)
style = load_image('/kaggle/input/artstyle1/vangoh2.jpeg').to(device)
# helper function for un-normalizing an image
# and converting it from a Tensor image to a NumPy image for display
def im_convert(tensor):
    """ Display a tensor as an image. """

    image = tensor.to("cpu").clone().detach()
    image = image.numpy().squeeze()
    image = image.transpose(1,2,0)
    image = image * np.array((0.229, 0.224, 0.225)) + np.array((0.485, 0.456, 0.406))
    image = image.clip(0, 1)

    return image
# display the images
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))
# content and style ims side-by-side
ax1.imshow(im_convert(content))
ax2.imshow(im_convert(style))
```



Và Sau khi dùng VGG19 kết quả sẽ ra những tấm hình đẹp đẽ

TÀI LIỆU THAM KHẢO

[1]. Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). "Image Style Transfer Using Convolutional Neural Networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[Link to paper] (https://arxiv.org/abs/1508.06576)

[2]. Johnson, J., Alahi, A., & Li, F. F. (2016). "Perceptual Losses for Real-Time Style Transfer and Super-Resolution." European Conference on Computer Vision (ECCV).

[Link to paper] (https://arxiv.org/abs/1603.08155)

[3]. Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576v2.

[4]. Chollet, F. (2017). "Deep Learning with Python."Manning Publications.

[Link to book] (https://www.manning.com/books/deep-learning-with-python)

[5]. Liu, J., & Peng, H. (2020). "A Comprehensive Review of Neural Style Transfer: Algorithms and Applications." Computers & Graphics.

[Link to paper] (https://www.sciencedirect.com/science/article/abs/pii/S0097849320300778)

[6]. Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks." Proceedings of the IEEE International Conference on Computer Vision (ICCV).

[Link to paper](https://arxiv.org/abs/1703.10593)

[7]. Gahlan, M., & Sharma, T. (2024). NeuArt: Style transfer using VGG19 & RESNET50 and their comparison. International Journal of Research Publication and Reviews, 5(1), 2246-2251.

[Link to paper](https://ijrpr.com/uploads/V5ISSUE1/IJRPR21856.pdf)

[8]. https://github.com/nazianafis/Neural-Style-Transfer/tree/main

[9]. A Lightweight PyTorch Implementation of Neural Style Transfer - Nazia Nafis

[Link to paper] (https://medium.com/geekculture/a-lightweight-pytorch-implementation-of-neural-style-transfer-86603e5eb551)

[10]. Phạm Văn Toàn (2021). “Hiểu về Skip Connection - một kĩ thuật "nhỏ mà có võ" trong các kiến trúc Residual Networks”

[Link to paper] (https://viblo.asia/p/paper-explain-hieu-ve-skip-connection-mot-ki-thuat-nho-ma-co-vo-trong-cac-kien-truc-residual-networks-3Q75w7bQ5Wb)

[11]. Khoa Nguyễn (2018). “Sử dụng Deep Learning để vẽ tranh”

[Link to paper] (https://viblo.asia/p/su-dung-deep-learning-de-ve-tranh-maGK7j4L5j2)

[12]. ResNet – Mạng học sâu đúng nghĩa

[Link to paper] (https://trituenhantao.io/kien-thuc/resnet-mang-hoc-sau-dung-nghia/)

[13]. Afshine Amidi, Shervine Amidi. “Học sâu, Mạng neural tích chập cheatsheet”

[Link to paper] (https://stanford.edu/~shervine/l/vi/teaching/cs-230/cheatsheet-convolutional-neural-networks)

[14]. Phạm Hoàng Anh (2019). “Biến ảnh của bạn thành các tác phẩm hội hoạ với Neural Style Transfer”

[Link to paper] (https://viblo.asia/p/bien-anh-cua-ban-thanh-cac-tac-pham-hoi-hoa-voi-neural-style-transfer-WAyK8dXoKxX)

[15]. 김상현-sanghyun kim (2021). Neural Style Transfer

[Link to paper] (https://velog.io/@skhim520/Neural-Style-Transfer)

[16]. Agarwal, M. (2020, August). Real-time image style transfer using feed-forward networks. Towards Data Science

[Link to paper] (https://towardsdatascience.com/fast-and-restricted-style-transfer-bbfc383cccd6)